Revista Perquirere • 15(3):164-177, set./dez. 2018

Arquitetura e desenvolvimento de aplicação mobile de *streaming* de vídeo para EaD

Architecture and development of mobile application of video streaming for distance education

Douglas Henrique Pereira Tiago
Graduando em Sistemas de Informação pelo Centro Universitário de Patos de Minas (UNIPAM).
e-mail: douglas_hptiago@hotmail.com

Eduardo Henrique Silva
Professor Orientador. Especialista em Engenharia de Software e Professor no Curso de Sistemas de Informação no Centro Universitário de Patos de Minas – UNIPAM.
e-mail: eduardohs@unipam.edu.br

RESUMO: O presente artigo apresenta soluções para se disponibilizar *streaming* de vídeo em aplicações *mobile* e aborda assuntos e questões que devem ser levados em consideração ao se desenvolver uma aplicação voltada para o contexto educacional. A aplicação possibilita que, depois de autenticado, o aluno tenha acesso às aulas previamente gravadas e disponibilizadas em *streaming* de vídeo. O sistema foi desenvolvido utilizando-se o *Xamarin Forms*, o Visual Studio 2015 e o *Google Drive*. O resultado final mostrou-se satisfatório, atendeu aos requisitos previamente levantados e possibilita a reaplicação em diversas outras aplicações *mobile*.
PALAVRAS-CHAVE: *Video Streaming*. *Video on Demand*. *Mobile. Ensino a Distância.*

ABSTRACT: This article presents solutions to make streaming videos available in mobile applications and questions that should be taken into account when developing an application focused on the educational context. The application allows that after authenticated the student has access to the lessons previously recorded and made available in video streaming. The system was developed using Xamarin Forms, Visual Studio 2015 and Google Drive. The final result was satisfactory, met the requirements previously raised and allow reapplication in the development of other mobile applications.
KEYWORDS: *Video Streaming*. *Video on Demand*. *Mobile. Distance learning.*

---

1. INTRODUÇÃO

Pesquisas voltadas para a transmissão de som e imagem possibilitaram o desenvolvimento de uma das tecnologias mais utilizadas atualmente, o *streaming* de mídia. Com a infraestrutura que se tem atualmente, *streamings* de alta qualidade de som e imagem são possíveis e possibilitaram que diversas aplicações e serviços surgissem, como, por exemplo, *Youtube*, *Netflix*, *Spotify*, etc. (ÁVILA, 2008).

Em um contexto educacional, o *streaming* de mídia vem sendo utilizado por diversas instituições de ensino, que, além de fornecer cursos presenciais, atualmente estão adotando uma nova modalidade de ensino, a Educação a distância, ou EaD (OLIVEIRA, 2013). A EaD é uma modalidade de ensino em que não só professores e alunos estão separados fisicamente, mas a proposta de ensino é diferenciada. O aluno não fica limitado às restrições de tempo e de espaço, pressupostos da educação presencial (OLIVEIRA, 2013).

O objetivo deste artigo é desenvolver uma aplicação *mobile* de *streaming* de *video on demand* para a Educação a Distância (EaD) do Centro Universitário de Patos de Minas (UNIPAM), que possibilitasse que os alunos tivessem acesso às aulas previamente gravadas e disponibilizadas pelos professores na aplicação, assim como descrever e documentar de forma sucinta toda a análise envolvendo a criação da arquitetura da aplicação, servindo assim como modelo para outras aplicações.

Servidores de *streaming* são caros. Assim, com este artigo, foram buscadas soluções mais viáveis. Buscou-se também conhecer e utilizar padrões, técnicas e boas práticas de desenvolvimento de *software* para desenvolver uma aplicação *mobile* com *layout* moderno e intuitivo, assim como com a qualidade e o desempenho esperados.

2. REVISÃO DE LITERATURA

Nesta seção são apresentados os conceitos referentes ao *streaming* de vídeo, *VoD* (*video on demand*), as soluções encontradas para o *streaming* de vídeo na aplicação e outras definições importantes para o desenvolvimento do artigo.

2.1. *DOWNLOAD*

Quando se faz o *download* de um arquivo, ele é salvo no computador ou celular (normalmente em uma pasta temporária) e, posteriormente, é executado, possibilitando assim sua visualização. Isso tem algumas vantagens (como acesso mais rápido a diferentes partes do arquivo), mas tem a grande desvantagem de ter que se esperar o arquivo inteiro ser baixado antes de vê-lo. Se o arquivo for pequeno, isso pode não incomodar muito, mas caso o arquivo seja grande, pode ser muito desagradável (MEDIA COLLEGE, 2017).

2.2. *STREAMING*

O *streaming* de mídia funciona um pouco diferente: o usuário final pode começar a assistir o arquivo quase assim que ele começa a baixar. Dessa forma, o arquivo é enviado para o usuário em um fluxo (mais ou menos) constante, e o usuário observa como ele chega. A vantagem com este método é que nenhuma espera está envolvida. A transmissão de mídia tem vantagens adicionais, como ser capaz de transmitir eventos ao vivo (MEDIA COLLEGE, 2017).

De acordo com Santos (2010), ao contrário da internet, em que o *download* é ainda a forma mais popular de distribuição de conteúdo, nos aparelhos móveis, a distribuição de conteúdo multimídia está mudando rapidamente para o *streaming*, pois este apresenta algumas vantagens quando comparado com a reprodução local após *download*:

- Os dados não são gravados permanentemente para o cliente final (ideal para equipamentos móveis em que a dimensão da memória é mais reduzida);
- O usuário não necessita de esperar que todo o conteúdo seja recebido para iniciar a reprodução;
- Possibilita a reprodução de programas diretamente;
- Uma grande vantagem para os fornecedores de serviços é o fato de não ser possível ao usuário final reencaminhar ou enviar os conteúdos para outros utilizadores, e por isso, os conteúdos podem ser cobrados a cada vez que são consumidos.

2.3. *VOD (VIDEO ON DEMAND)*

Segundo Paes (2007), no VoD os vídeos são armazenados num servidor e ficam disponíveis para serem acessados e transmitidos de acordo com os pedidos dos clientes. O cliente que pede o vídeo pode assistir qualquer parte dele no tempo que desejar.

2.4. SOLUÇÕES ENCONTRADAS

2.4.1. API Google Drive

Essa API possibilita consultar e fazer o *download* e o *upload* de arquivos do *Google Drive*. Ela fornece duas formas de trabalho, através do *Android* e via *HTTP* (GOOGLE DRIVE, 2017).

2.4.2. API Youtube

A API do *Youtube* possibilita a visualização de qualquer vídeo que esteja disponível no serviço, precisando-se saber apenas qual é o seu identificador (*key*),

e também fornece um *player* para o *Android* e métodos que fornecem diversas informações do vídeo, como progresso, descrição, título e checagem da disponibilidade da API. Através dos métodos que são disponibilizados, faz-se possível a criação de *players* customizados. O *player* está disponível apenas para a plataforma *Android*, sendo que no *iOS* é fornecida uma *library* que trabalha a visualização dos vídeos em formato *Embedded* (YOUTUBE, 2017).

2.4.3. Plugin Video Player

É um plugin voltado para o desenvolvimento no *Xamarin Forms*. Ele possibilita a execução de vídeos informando-se URLs válidas (com a extensão do arquivo no final do URL) e possibilita fazer *streaming* de sites populares como o *Youtube* e o *Vimeo*, sendo necessária a criação de adaptadores na aplicação, adaptadores estes que também são disponibilizados através de um *sample* disponibilizado no próprio site do *plugin* (FISHER, 2017).

2.4.4. WebView em conjunto com *Embedded Vídeos*

O *WebView* é um componente bastante utilizado no desenvolvimento de aplicativos, o qual é capaz de exibir páginas da *web* sem que seja necessário abrir outro programa separadamente. O *Embed* é um tipo de tag *HTML* para mídia, usada para incorporar arquivos multimídia de áudio e vídeo. Alguns sites disponibilizam seus vídeos dessa forma, possibilitando a exibição dos vídeos em sites e aplicativos. Tudo o que se deve saber é o URL a ser utilizado para exibir o vídeo em conjunto com a sua *key* (XAMARIN, 2016). Exemplos de URLs disponibilizados pelos sites *Youtube*, *Vimeo* e *Google drive*:

- *Youtube*: https://www.youtube.com/embed/ + *key* do vídeo;
- *Vimeo*: https://player.vimeo.com/video/ + *key* do vídeo;
- *Google Drive*: https://drive.google.com/file/d/ + *key* do vídeo + /preview.

2.5. PADRÕES UTILIZADOS

2.5.1. Padrão *Model, View e Controler – MVC*

Esse padrão segue a seguinte premissa: dividir a aplicação em três ou mais partes/camadas, sendo estas *Model, View* e *Controller*. De forma sucinta, a camada *Model* é responsável por gerenciar um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado. A *View* apresenta as informações para o usuário através de uma combinação de gráficos e textos, e o *Controller* é o responsável por interpretar as entradas dos dados, enviar para o modelo (*Model*) e/ou para a janela de visualização (*View*) para efetuar a alteração apropriada (MEDEIROS, 2013).

2.5.2. *Factory Method*

O padrão *Factory Method* encapsula a criação de objetos deixando as subclasses decidirem quais objetos criar. Dessa forma, o código de criação fica em um objeto ou método, evitando assim a duplicação. Além disso, tem-se um local único para fazer manutenção. O padrão faz com que se tenha um código flexível e extensível para o futuro (MEDEIROS, 2012).

3. METODOLOGIA

Para o desenvolvimento do artigo e da aplicação, foi adotada a metodologia ágil *Kanban*. Segundo Beck *et al*. (2001), as metodologias ágeis basicamente vêm com a seguinte premissa:

- Indivíduos e interação entre eles mais que processos e ferramentas;
- *Software* em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

As decisões que foram tomadas, ao se definir a solução ideal para o *streaming* de vídeo na aplicação, ocorreram da seguinte forma: inicialmente foi realizado um estudo buscando encontrar soluções viáveis financeiramente para se fazer *streaming* em aplicações mobile, assim como, alguns executáveis dessas soluções foram criados. Posteriormente uma análise foi feita, a fim de se decidir qual delas melhor se adequaria à aplicação. Com a arquitetura de *streaming* definida, foi necessário pensar na arquitetura do aplicativo e na integração com ela; desta forma, foi iniciado um estudo a fim de se conhecer e utilizar os padrões de desenvolvimento. Posteriormente foi realizado o levantamento dos requisitos que compõem a aplicação e foram criados diversos protótipos do aplicativo com base neles.

Fazendo-se uso de todos os conhecimentos até então levantados, foi realizado o desenvolvimento do aplicativo. Inicialmente ele seria desenvolvido no *Android Studio*, porém, decidiu-se por criá-lo no Visual Studio com o *Xamarin Forms*, por possibilitar um desenvolvimento *cross platform*, para o *Android* e o *iOS*. A aplicação de testes no decorrer do desenvolvimento assegurou que o aplicativo não apresentasse erros e atendesse aos requisitos levantados.

4. DESENVOLVIMENTO E RESULTADOS

Nesta seção são apresentados os processos realizados e os resultados obtidos com o desenvolvimento do sistema.

4.1. ANÁLISE E ESCOLHA DA MELHOR SOLUÇÃO PARA A APLICAÇÃO

A análise foi feita com base nos objetivos do artigo. A Tabela 1 mostra sucintamente as informações resultantes da criação dos executáveis e pesquisas realizadas.

TABELA 1. Análise das Soluções encontradas

Solução	Qualidade	Desempenho	Disponibilidade	Plataforma	Diferencial	Limitadores
API Youtube	Sim	Sim	Não. Os Termos de uso da API não garante a disponibilidade da mesma.	*Android* e *iOS*	Possibilita a criação de um *player* customizado no *Android*.	Apenas vídeos do *Youtube.*
API Google Drive	Indefinido	Indefinido	Indefinido	*Android* e *HTTP*	Possibilita uma integração muito útil com o *Google Drive.*	Nada official voltado para a aplicação da API no *Xamarin Forms* e pouco material de pesquisa disponível na *Web*.
Plugin Video Player	Sim	Sim	Sim	*Xamarin Forms* (*Android* e *iOS*)	Possibilita a execução de videos de fontes do *Youtube* e *Vimeo*.	Não possibilita a execução de videos do *Google Drive*. Requisito solicitado pela instituição.
Web-View	Sim	Sim	Sim	*Xamarin Forms* (*Android* e *iOS*)	Inúmeras possíbilidades. Exibição de vídeos e documentos de fontes diversas utilizando o formato *Embed*.	Para ver o vídeo é preciso clicar duas vezes no vídeo para que ele seja executado.

Fonte 1: Elaborado pelos autores, 2017

Todas as soluções possuem meios de impossibilitar o acesso aos endereços (URLs) dos vídeos, impossibilitando também o seu compartilhamento indesejado, caso esse seja um dos requisitos solicitados pela instituição.

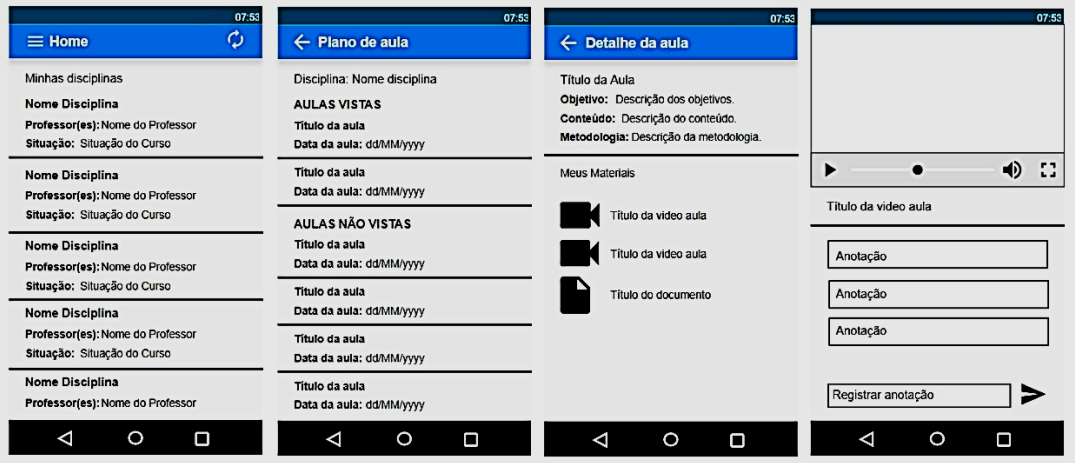
Com exceção do *Plugin Video Player*, todas as soluções são gratuitas para sua utilização. Para se utilizar o *Plugin Video Player*, deve-se desembolsar apenas $45,00 dólares.

A *Web View*, em conjunto com os *Embedded Videos*, mostrou-se a melhor solução dentre as encontradas, pois possibilitava não só a exibição de *Embedded Videos*, mas de arquivos variados no formato *Embed*, através do *Google Drive* e também de outras fontes, o que para uma instituição de ensino voltada para ensino a distância é bem interessante.

4.2. PROTOTIPAÇÃO

Com base nos dados que deveriam ser consumidos e exibidos no aplicativo, protótipos das páginas foram criados, e através deles se definem quais funções devem ser implementadas e como os dados serão exibidos na aplicação através delas. Alguns dos protótipos criados podem ser vistos na Figura 1.

FIGURA 1. Protótipos das páginas da aplicação



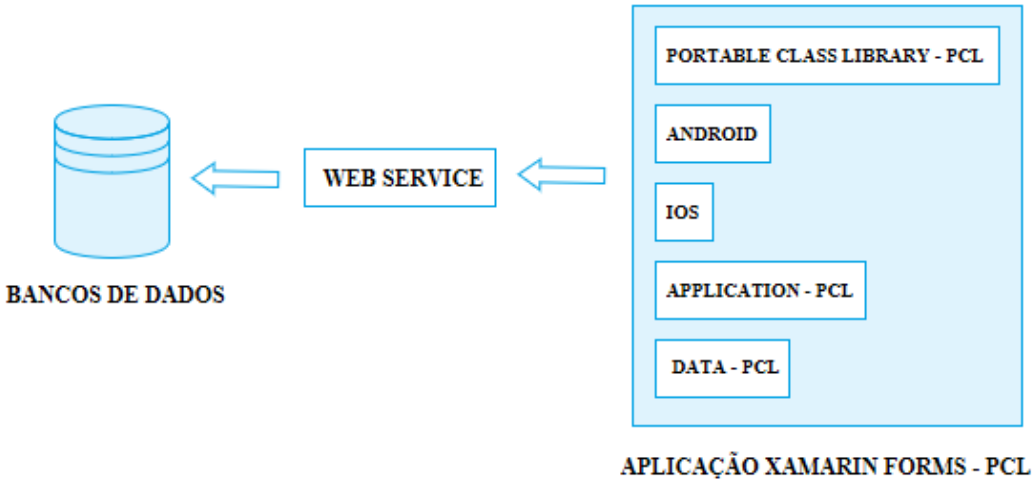
Fonte: Elaborado pelos autores, 2017

4.3. ARQUITETURA DA APLICAÇÃO

De forma sucinta e conforme pode-se ver na Figura 2 a arquitetura da aplicação, que possui bancos de dados que serão consumidos por um *Web Service*, o qual será utilizado pela aplicação *Xamarin Forms - PCL*. A aplicação *Xamarin Forms* possui 5 camadas, sendo elas: PCL principal, PCL *Application*, PCL *Data*, *Android* e *iOS*.

Em uma aplicação *Xamarin Forms*, a camada *Portable Class Library – PCL*, possibilita que se criem páginas em *XAML* que são interpretadas pelo *Android* e pelo *iOS*, possibilitando assim a criação de páginas e funções para ambas as plataformas. Ela também possibilita a criação de funções específicas para cada uma delas quando necessário.

FIGURA 2. Arquitetura da aplicação.



Fonte: Elaborado pelos autores, 2017

Requisições ao *Web Service* são feitas, e os dados resultantes dessas requisições são armazenados em um banco de dados interno do aplicativo, posteriormente consultas ao banco de dados interno são feitas e os dados são exibidos nas páginas. Dessa forma, garante-se o acesso aos dados, mesmo que não se esteja conectado à rede, e o usuário atualiza os dados quando desejar.

4.3.1. Bancos de dados

A instituição possui mais de um banco de dados e eles foram utilizados ao se criar o *web service*.

4.3.2. *Web Service*

O *web service* foi criado no *Visual Studio*, e em sua criação foi utilizado o padrão MVC, responsável por receber as requisições que serão feitas pelo aplicativo e com base nelas realizar as consultas aos bancos de dados da instituição e retornar o resultado proveniente dessas consultas.

Após a criação do *web service*, testes foram feitos através do *Postman* – extensão para o *Google Chrome* que permite fazer requisições do tipo POST – para verificar se o *service* estava de fato funcionando conforme o esperado.

Tanto as requisições ao *service* quanto as respostas geradas por ele trabalham com o *JavaScript Object Notation* (JSON), notação bastante utilizada para intercâmbio de dados. Para visualizar melhor o resultado obtido através do *Postman* foi utilizado o site http://jsonviewer.stack.hu/, que disponibiliza uma visualização melhor das respostas.

4.3.3. Aplicação *Xamarin Forms – PCL*

O aplicativo foi desenvolvido no *Visual Studio* com o *Xamarin Forms* através das PCLs e foi projetado utilizando-se os padrões de projeto MVC e *Factory Method*. Basicamente as camadas da aplicação podem ser descritas, de acordo com suas responsabilidades, da seguinte forma: a camada PCL principal possui todas as páginas do aplicativo. A parte visual das páginas é desenvolvida em *XAML* e cada *XAML* possui uma classe em c# para que se desenvolva todos as funções presentes nas páginas. A camada *Application* possui todas as regras de negócio da aplicação, ela possui todos as funções responsáveis por conectar a aplicação ao *web service*, consultar e manipular os dados do banco de dados *SQLite* da aplicação e possui também a *FactoryApplication*, classe criada com base no padrão *Factory Method*, que é a responsável por realizar a instanciação das classes presentes na camada *Application*.

A Figura 3 mostra o código responsável por criar uma instância de uma das classes da *Application*, no caso a *MaterialApplication*:

FIGURA 3: Código presente na *FactoryApplication*

```
namespace PortalProject.Appli.Factory
{
    18 references
    public class FactoryApplication
    {
        private static FactoryApplication factoryApplication;

        15 references
        public static FactoryApplication GetInstance()
        {
            if (factoryApplication == null)
                factoryApplication = new FactoryApplication();
            return factoryApplication;
        }

        1 reference
        public IMaterialApplication getMaterialApplication()
        {
            return new MaterialApplication();
        }
```

Fonte: Elaborado pelos autores, 2017

A camada *Data* possui todas as classes modelo do banco de dados interno (*SQLite)* e uma classe genérica com o CRUD básico da aplicação. O banco de dados *SQLite* deve ser criado de forma diferente em cada plataforma– *Android* e *iOS* – e em código nativo. O *Xamarin Forms* possibilita que isso seja feito através do *DependencyService*.

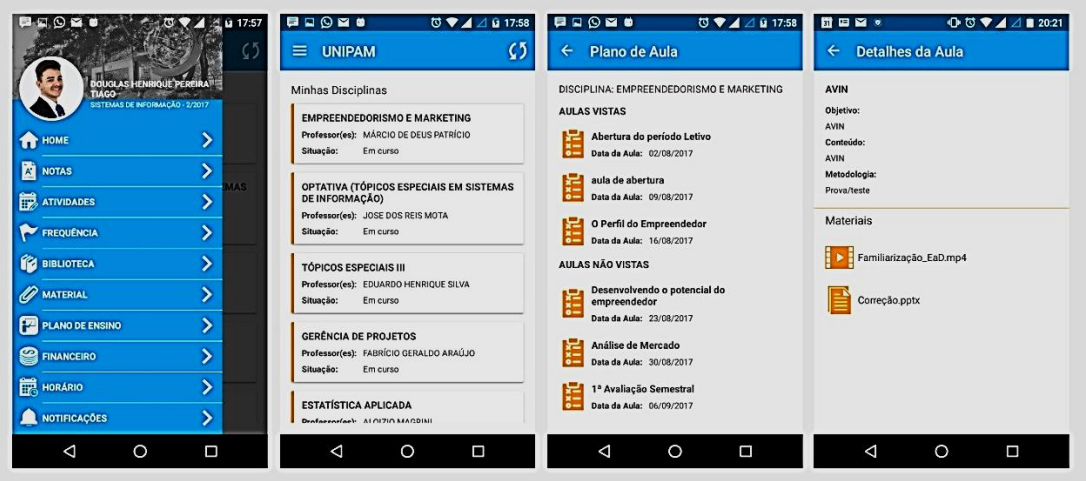
As camadas *Android* e *iOS* possuem apenas as classes de dependência, que

devem ser desenvolvidas em cada plataforma, uma página destinada a exibir a logo da instituição ao iniciar o aplicativo (*splash screen*) e as imagens utilizadas nas páginas da aplicação. A aplicação possui um total de sete páginas, sendo elas:

- Página de *login*: basicamente faz a autenticação do usuário na aplicação.
- Página de configuração: possibilita que o usuário escolha qual curso e período letivo que ele deseja acessar.
- Página inicial: lista e exibe as disciplinas referentes ao curso e período solicitado. Página de menu: possibilita que o usuário navegue pelas páginas da aplicação.
- Página de plano de aula: lista e exibe o plano de aula referente a determinada disciplina do curso.
- Página da aula: exibe detalhadamente os dados referentes a determinada aula e lista os materiais disponibilizados pelo professor.
- Página de material: exibe o material disponibilizado pelo professor na aula.

A Figura 5 mostra algumas das páginas do aplicativo:

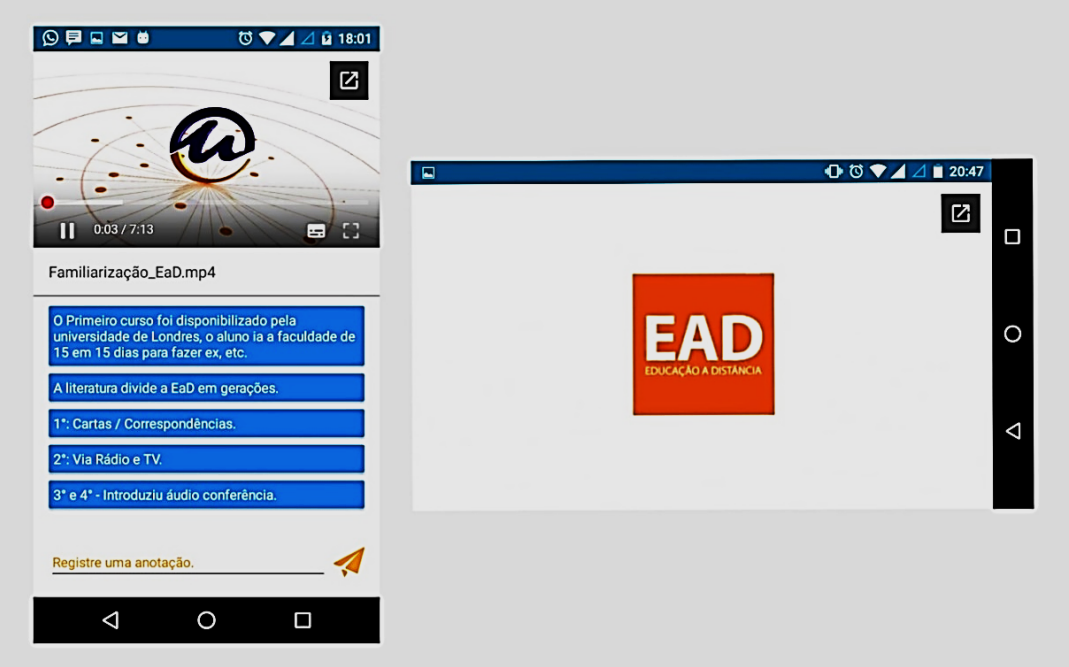
FIGURA 5. Imagens da versão 1.0 do aplicativo



Fonte: Elaborado pelos autores, 2017

A Figura 6 mostra a página de material, onde foi desenvolvido o *streaming* de vídeo da aplicação. Ela possui métodos que redimensionam a *WebView* de acordo com as proporções da tela, de forma que o vídeo ou o documento exibido se adapte à tela conforme ela é rotacionada. Note que anotações podem ser feitas conforme se assiste à vídeoaula.

FIGURA 6. Página de material, onde foi implementado o *streaming* de vídeo na aplicação



Fonte: Elaborado pelos autores, 2017

Conforme visto anteriormente, os vídeos e documentos são disponibilizados na aplicação através de URLs válidas que utilizam o formato *Embedded*. A Figura 7 mostra uma parte do código presente na página material, que é responsável por checar a conexão com a internet e atribuir uma dessas URLs a *WebView*.

FIGURA 4. Código presente na página material

```
var minhaConexao = Plugin.Connectivity.CrossConnectivity.Current.IsConnected;
if (minhaConexao.Equals(true))
{
    wvMaterial.IsVisible = true;
    imgConexaoInternet.IsVisible = false;
    wvMaterial.Source = "https://drive.google.com/file/d/" + myMaterial.UrlReduzido + "/preview";
}
```

Fonte. Elaborado pelos autores, 2017

4.4. ANÁLISE DE CUSTOS

Os custos com servidores de *streaming* são elevados. O *Wowza Media Server* é um dos servidores de *streaming* mais utilizados atualmente. No site do Wowza foram encontrados alguns cenários que proporcionaram um entendimento da quantidade de instâncias do servidor necessárias para a aplicação. O cenário ideal

para essa aplicação é de duas a cinco instâncias, o que possibilita que se tenham até 15.000 acessos ao mesmo tempo ao servidor de *streaming*. A Tabela 2 mostra os custos com esse servidor de *streaming* em comparação com a solução encontrada para a aplicação.

Tabela 2: Custos com Servidores de Streaming

Considerações	***Wowza Media Server***	***Google Drive***
Valor em dólares	65$ a 95$	0
Ideal para a aplicação: 2 a 5 instâncias em dólares	130$ a 450$	0
2 a 5 instâncias em reais	R$ 405,60 a R$ 1482,00	0

Fonte: Elaborado pelos autores, 2017.

Utilizando-se o *Web View* com *Embbed Videos* do *Google Drive* não há custo algum com servidores de *streaming*. O único custo que se pode ter é com a expansão do armazenamento do *Google Drive,* caso necessário. A Tabela 3 mostra os valores que se teriam mensalmente com a expansão desse armazenamento.

TABELA 3. Custos com armazenamento *Google Drive*

Quantidade	**Valor**
15 GB	Gratuito.
1 TB	R$ 35,00 mensais
10 TB	R$ 350,00 mensais

Fonte: Elaborado pelos autores, 2017.

5. CONSIDERAÇÕES FINAIS

Os principais objetivos do artigo foram encontrar uma forma viável financeiramente de se criar um aplicativo de *streaming* de vídeo para o EaD, assegurar que esses vídeos fossem vistos e disponibilizados apenas pelo aplicativo, que o *streaming* fosse de qualidade e que a arquitetura da aplicação fosse criada com base em padrões utilizados atualmente na criação de sistemas.

A principal dificuldade enfrentada no decorrer do artigo foi encontrar a solução ideal para se disponibilizar *streaming* de vídeo em um aplicativo e que ela fosse viável financeiramente. Todas as soluções financeiramente viáveis encontradas são externas e não há controle sobre elas. A maneira para contornar uma possível indisponibilidade foi achar uma solução que possibilitasse o acesso a vídeos de fontes diferentes, dessa forma, caso uma delas esteja indisponível, basta utilizar a outra.

O artigo deixou claro quais as vantagens e desvantagens das soluções encontradas. Por mais que a solução utilizada na aplicação tenha atendido aos seus objetivos, pode ser que não seja a ideal para outras.

Como o aplicativo foi desenvolvido no *Xamarin Forms*, buscou-se encontrar soluções viáveis para o desenvolvimento nele. Caso se busquem outras soluções para *streaming* de vídeo, seria interessante um estudo mais aprofundado dos servidores de *streaming*, que são pagos, bem como o estudo de outras APIs e possíveis outras soluções que não são voltadas para o *Xamarin Forms* e, sim, às plataformas específicas.

Futuramente pretende-se tornar a integração com o *Google Drive* maior. Não se obteve sucesso ao utilizar a REST API do *Google Drive* no *Xamarin Forms*, e novos estudos são necessários a fim de se aperfeiçoar essa integração. O aplicativo se encontra atualmente na *Google Play* e *Apple Store* e pode ser baixado e utilizado pelos alunos do Centro Universitário de Patos de Minas (UNIPAM).

REFERÊNCIAS

FISHER, Adam. *A Xamarin Forms control to render the native video player on every platform.* 2017. Disponível em: <https://components.xamarin.com/view/video-player>. Acesso em: 21 ago. 2017.

AVILA, Renato Nogueira Perez. *Streaming*: aprenda a criar e instalar sua rádio ou TV na internet. Rio de Janeiro: Ciencia Moderna, 2008.

CIRIACO, Douglas. *O que é API?* 2009. Disponível em: <https://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm>. Acesso em: 05 mar. 2017.

BECK, Kent et al. Manifesto para Desenvolvimento Ágil de Software. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 25 set. 2017.

GOOGLE DRIVE, *Guia de Preços.* 2017. Disponível em: < https://www.google.com/intl/pt-BR_ALL/drive/pricing/>. Acesso em: 04 out. 2017.

GOOGLE DRIVE. *Google Drive APIs.* 2017. Disponível em: <https://developers.google.com/drive/>. Acesso em: 10 nov. 2017.

MEDEIROS, Higor. Introdução ao Padrão MVC. 2013. Disponível em: <http://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>. Acesso em: 30 ago. 2017.

MEDEIROS, Higor. *Padrão de Projeto Factory Method em Java*. 2012. Disponível em:

<http://www.devmedia.com.br/padrao-de-projeto-factory-method-em-java/26348>. Acesso em: 30 ago. 2017.

MEDIA COLLEGE. *Introduction - How to Create Streaming Video*. 2017. Disponível em: <http://www.mediacollege.com/video/streaming/overview.html>. Acesso em: 23 fev. 2017.

OLIVEIRA, Débora Silva de. *O uso do vídeo em ead: desafios no processo de ensino aprendizagem.* 2013. 15 f. Tese (Doutorado) - Curso de Psicologia, Faculdade, Cesuca, Cachoeirinha, 2013. Disponível em: <ojs.cesuca.edu.br/index.php/cesucavirtual/article/download/422/207>. Acesso em: 12 fev. 2017.

PAES, Lucas Medaber Jambo Alves. *Video Peer-to-Peer.* 2007. Disponível em: <http://www.gta.ufrj.br/grad/07_2/lucas_paes/index.html>. Acesso em: 02 mar. 2017.

SANTOS, José Fernando Pereira dos. *Mobile Streaming*: qualidade de experiência. 2010. 67 f. Dissertação (Mestrado) - Curso de Comunicações Multimédia, Faculdade, Faculdade de Engenharia da Universidade do Porto, Porto, 2010. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/57907/1/000143374.pdf>. Acesso em: 06 fev. 2017.

WOWZA MEDIA SYSTEMS. *Plans and Pricing*. 2017. Disponível em: <https://www.wowza.com/pricing>. Acesso em: 04 out. 2017.

XAMARIN. *WebView:* Present local or network web content and documents. 2016. Disponível em: <https://developer.xamarin.com/guides/xamarin-forms/user-interface/webview/>. Acesso em: 23 jul. 2017.

YOUTUBE. *Add YouTube functionality to your sites and apps*. 2017. Disponível em: <https://developers.google.com/youtube/>. Acesso em: 10 nov. 2017.