UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMA DE INFORMAÇÃO

ANDREAZO SILVA SOUZA

UM PROTÓTIPO DE SISTEMA DE BUSCA DE CONTEÚDO PARA APLICAÇÕES COM TECNOLOGIAS IMERSIVAS

QUIXADÁ

2019

ANDREAZO SILVA SOUZA

UM PROTÓTIPO DE SISTEMA DE BUSCA DE CONTEÚDO PARA APLICAÇÕES COM TECNOLOGIAS IMERSIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistema de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistema de Informação.

Orientador: Prof. Dr. Cristiano B. de Oliveira

QUIXADÁ

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S713p Souza, Andreazo Silva.
Um protótipo de sistema de busca de conteúdo para aplicações / Andreazo Silva Souza. – 2019.
44 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2019.
Orientação: Prof. Dr. Cristiano Bacelar de Oliveira.

1. Android(Programa de Computador). 2. Serviços da Web. 3. Ambiente Virtual. I. Título.
CDD 005

ANDREAZO SILVA SOUZA

UM PROTÓTIPO DE SISTEMA DE BUSCA DE CONTEÚDO PARA APLICAÇÕES COM TECNOLOGIAS IMERSIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistema de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistema de Informação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Cristiano B. de Oliveira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jefferson de Carvalho Silva
Universidade Federal do Ceará (UFC)

Prof. Dr. João Vilnei de Oliveira Filho
Universidade Federal do Ceará (UFC)

Aos meus pais, Maria Nanci da Silva e José de Souza.

AGRADECIMENTOS

Agradeço aos meus pais que sempre me incentivaram e investiram no meu futuro.

Agradeço ao Professor Cristiano Bacelar de Oliveira, pela paciência e disponibilidade por me orientar durante o projeto.

Aos meus amigos e colegas de graduação pela parceria durantes esses 4 anos.

A todos que direta e indiretamente fizeram parte da minha graduação.

“O sucesso é ir de fracasso em fracasso sem perder entusiasmo.”

(Winston Churchill)

RESUMO

Este trabalho apresenta o protótipo de um sistema capaz de integrar a busca por recursos multimídias, como imagens, áudio e modelos em 3D, para aplicações que utilizam Realidade Aumentada e Realidade Virtual em ambientes virtuais. O sistema proposto é composto por uma aplicação móvel, responsável por interagir com o usuário, e por um Web Service que atua como provedor e centralizador dos conteúdos de multimídias disponíveis na internet. O trabalho mostra, ainda, testes preliminares com uma placa Arduíno, operando como um sistema de controle, o qual atua como controlador do ambiente físico a partir das configurações feitas pelo o usuário.

Palavras-chave: Android. Web Service. Ambientes virtuais.

ABSTRACT

This work presents the prototype of a system capable of integrating the search for multimedia resources, such as images, audio and 3D models, for applications that use Augmented Reality and Virtual Reality in virtual environments. The proposed system consists of a mobile application, responsible for interacting with the user, and a Web Service that acts as a provider and centralizer of multimedia content available on the Internet. The work also shows preliminary tests with an Arduino board, operating as a control system, which acts as controller of the physical environment from the settings made by the user.

Keywords: Android.Web Service. Virtual Environment.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
APK	*Android Package*
ART	*Android Runtime*
AV	Ambiente Virtual
CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
OHA	*Open Handset Alliance*
REST	*Representational State Transfe*
SDK	Software Development Kit
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

À medida que caminhamos para a próxima década, estamos às vésperas de uma nova transformação com o uso de tecnologias imersivas, como Realidade Virtual (RV) e Realidade Aumentada (RA). Tais tecnologias nos permitem interagir com os outros e experimentar o mundo como nunca antes (KIRNER; KIRNER, 2011), uma vez que permitem ao usuário atuar de forma multisensorial, por meio não só da visão, mas, também, dos demais sentidos (KIRNER; KIRNER, 2011).

RV e RA, bem como suas variações, representam técnicas de interface computacional que levam em conta o espaço tridimensional. Antes do seu surgimento, as interfaces computacionais estavam restritas ao espaço bidimensional da tela do monitor, viabilizando aplicações multimídia com textos, imagens, sons, vídeos e animações. Apesar das diferenças dimensionais, tanto a multimídia quanto a RV e RA têm elementos comuns, como interações multisensoriais e processamento em tempo real (KIRNER; KIRNER, 2011).

Como exemplo de aplicação, podemos citar a utilização da RA para o *design* de interiores, onde o *designer* pode modificar o ambiente com modelos 3D de vários móveis e objetos de decoração virtuais. Tais objetos podem ser adicionados, removidos e reorganizados até que os usuários fiquem satisfeitos com o resultado por meio da personalização do ambiente de acordo com seus critérios, permitindo que o usuário observe em tempo real aspectos físicos como reflexos, opacidade e volumetria.

Outro exemplo de aplicações nesse contexto são as Caverna digitais, ou CAVE's (*Cave Automatic Virtual Environment*). Uma CAVE é uma sala onde são projetados gráficos em 3 dimensões em suas paredes, de forma a poderem ser visualizados pelos usuários utilizando dispositivos avançados de interação. Assim, os usuários podem explorar e interagir com objetos ou pessoas virtuais (dentre outras possibilidades) para terem um ambiente virtual, desta forma mergulhando em um mundo virtual (ELISEU, 2017).

Em geral, aplicações como as citadas conectam-se a *Web Services* que fornecem o conteúdo para o seu funcionamento. Um *Web Service* é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes via Web. Com esta tecnologia, é possível que novas aplicações interajam com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis através da *Web* (MYERSON, 2015). Os *Web Services* permitem conectar qualquer tipo de sistema, independentemente da plataforma (Windows, Linux, etc.) ou das linguagens de programação utilizadas.

Muitos dos *Web Services* que fornecem conteúdos para aplicação de Realidade imersiva provém informações e funcionalidades muito específicas, apenas fornecendo informações de configuração ou mídias como imagens, sons e textos. Desenvolvedores que desejem criar aplicações para utilizarem esses serviços possuirão acesso a uma faixa limitada de recursos. Caso o desenvolvedor queria usar esses conteúdos em conjunto, terá que realizar acessos a *Web Services* específicos que forneçam cada conteúdo.

Ainda neste contexto, percebe-se que inúmeros projetos são desenvolvidos com dispositivos móveis, em especial para aqueles que usam o sistema Android[1]. O Android controla a maior fatia do mercado de dispositivos móveis (BELIZARIO, 2018) e possui parte do seu código-fonte livre, o que permite um maior nível de customização e liberdade para o desenvolvimento de sistemas, quando comparado aos seus principais concorrentes.

Assim, dado o exposto, este trabalho apresenta o protótipo de um sistema, constituído por uma aplicação móvel para Android e um *Web Service*, que centraliza a busca de conteúdo multimídia (imagens, áudio e modelos 3D) em repositórios *online*, funcionando como uma interface para a criação de ambientes imersivos. Além disso, esse sistema inclui a possibilidade de conexão a uma plataforma de controle do ambiente físico, através de uma placa Arduíno (BANZI; SHILOH, 2015), podendo, assim controlar equipamentos como projetores, caixas de som, ar-condicionados, etc.

Este trabalho está organizado da seguinte maneira: no capítulo 2 é mostrado o objetivo geral e os específicos. No capítulo 3 são apresentados os conceitos principais. O capítulo 4 apresenta os trabalhos relacionados e suas respectivas contribuições para este trabalho, enquanto que no capítulo 5 é apresentado o desenvolvimento do sistema. Por fim, o capítulo 6 mostra os testes realizados com o protótipo desenvolvido.

[1] <https://www.android.com/intl/pt-BR_br/gms/>

2 OBJETIVOS

2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema capaz de integrar a busca por recursos multimídias, como imagens, áudio e modelos em 3D, para aplicações que utilizam tecnologias imersivas em ambientes virtuais. O sistema desenvolvido é composto por uma aplicação móvel para a plataforma Android e um Web Service.

2.2 Objetivos específicos

São objetivos específicos deste trabalho:

- Desenvolver uma aplicação que sirva de interface para a criação de ambientes imersivos
- Desenvolver uma interface de uso comum a várias plataformas.
- Permitir a busca de conteúdo baseado em contextos fornecidos pelos usuários.
- Fornecer uma interface de comunicação com ambientes reais.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos principais abordados neste trabalho. Os assuntos tratados incluem Ambiente virtual, Realidade Virtual, Aumentada e Web Service.

3.1 Ambientes virtuais

De acordo com (SEMENTILLE; JRF, 2000), um Ambiente Virtual (AV) é um ambiente no qual há a imersão do usuário em um ambiente tridimensional simulado. Um Ambiente Virtual também pode ser entendido como um sistema de *software* que cria a ilusão de um mundo que não existe na realidade. Isto requer a combinação de entrada (interação do usuário), computação (simulação de processos) e saída (estímulos multisensoriais).

Em um AV, o usuário deve poder navegar em três dimensões, ou seja, com seis graus de liberdade. Cada grau se aplica a uma direção ou rotação do movimento. Na verdade, conferem ao usuário a possibilidade de se movimentar em seis direções simultâneas: translação em torno dos três eixos cartesianos (x, y e z) e rotação em torno de cada um (SEMENTILLE; JRF, 2000).

Os objetos no Ambiente Virtual podem corresponder a objetos que existem no mundo real. Quando isso acontece, espera-se que esses objetos reajam como os objetos reais, sendo capazes de responder às ações dos usuários e às ações de outros objetos. Portanto, devem existir meios de especificar esses comportamentos e associá-los aos objetos no modelo. Alguns objetos podem ter comportamentos autônomos, como flutuar, por exemplo, o que envolve interações entre os objetos (SEMENTILLE; JRF, 2000).

É comum o uso de tecnologias imersivas na criação de ambientes virtuais. Exemplos destas tecnologias são a Realidade Virtual (RV) e a Realidade Aumentada (RA).

3.1.1 Realidade Virtual e Realidade Aumentada

Realidade Virtual é uma tecnologia de interface capaz envolver os sentidos de um usuário, por meio de um ambiente virtual, criado a partir de um sistema computacional, ao induzir efeitos visuais, sonoros e até táteis. A Realidade Virtual permite a imersão completa em um ambiente simulado, com ou sem interação do usuário (TORI *et al.*, 2006) através, geralmente, de estímulos visuais e auditivos. Para aumentar a sensação de imersão, muitas aplicações requerem o uso de dispositivos que limitam a percepção de estímulos externos por parte do usuário. Exemplos disso são aplicações que usam óculos ou *headsets* próprios, ou que confinam

o usuário em cabines que o isolam do contato externo. A Figura 1 mostra um exemplo de ambiente virtual visto no monitor.

Figura 1 – Exemplo de ambiente virtual em uma aplicação de RV.

Fonte: (KIRNER; KIRNER, 2011)

Ainda segundo Kirner e Kirner (2011), a Realidade Aumentada é uma tecnologia que permite, ao usuário, ver, ouvir, sentir e interagir com informações e elementos virtuais inseridos no seu ambiente físico, através de algum dispositivo tecnológico. Ela pode ser usada por qualquer área do conhecimento, uma vez que se baseia na inserção de textos, imagens e objetos virtuais tridimensionais no ambiente físico com o qual o usuário interage (TORI *et al.*, 2006). Por isso, ela está sendo usada em algumas áreas: medicina, arquitetura e engenharia, educação e treinamento, entretenimento, marketing, comércio eletrônico, museus, etc.

Em todos os casos, o usuário vê um cenário real e elementos complementares, consistindo de informações simbólicas e textuais, que podem ser animados e sonorizados, para amplificar sua capacidade de visualização e interação com o ambiente, no qual está inserido (KIRNER; KIRNER, 2011). Na Figura 2 temos um exemplo de uso de Realidade Aumentada.

3.2 Android

O Android é um sistema operacional, baseado em Linux e desenvolvido pelo Google em conjunto com a *Open Handset Alliance* (OHA) (DANIELSSON, 2016). Esse sistema operacional foi inicialmente pensado para ser uma plataforma para câmeras digitais quando o

Figura 2 – Objetos virtuais sendo projetado sobre uma mesa física.

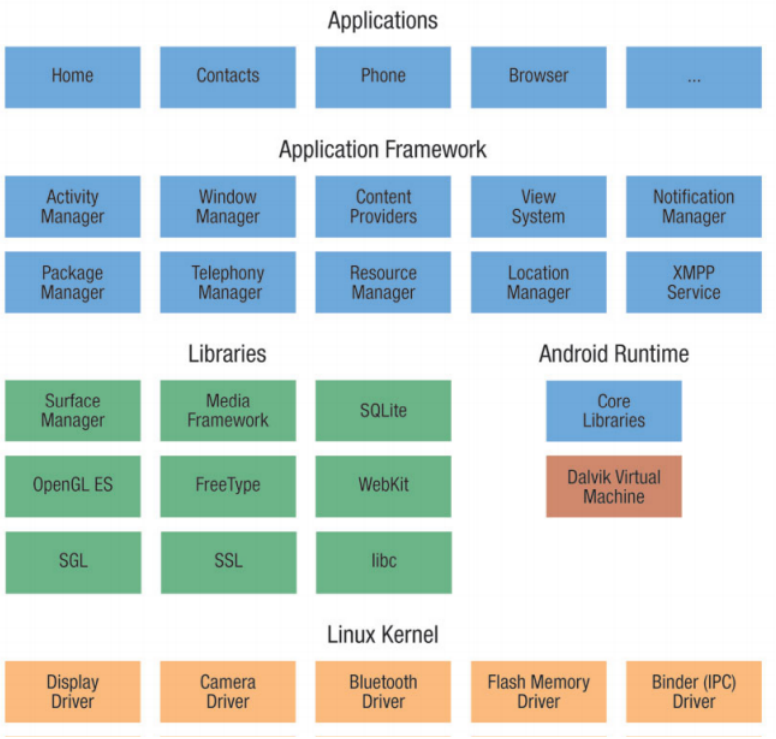
Fonte: (KIRNER; KIRNER, 2011)

desenvolvimento da Android Inc. começou em 2005. No entanto, o Google comprou a empresa e três anos depois, o primeiro smartphone rodando Android foi vendido.

A Figura 3 mostra a arquitetura do sistema operacional Android. Na camada de *kernel* existem programas que abstraem toda a complexidade do acesso aos recursos de *hardware* pelos demais programas. Acima do *kernel* Linux, estão as bibliotecas nativas e o módulo *Runtime*. As bibliotecas nativas são compostas por funções que podem ser usadas pelos demais aplicativos, permitindo que tenham acesso aos recursos que o Android oferece. Elas são escritas em C ou C++ e são chamadas através de uma interface Java que permite acessar diferentes variedades de dados ou áudio no telefone (DANIELSSON, 2016). O *Android Runtime* (ART) é o ambiente de tempo de execução dos aplicativos no Android. O ART melhora a eficiência geral da execução e reduz o consumo de energia. Já a camada *Application framework* oferece abstrações das bibliotecas nativas e acesso a recursos próprios dos dispositivos. Além disso, fornece a *Application Programming Interface* (API) e outros serviços na forma de classes Java. Um bloco importante do camada de estrutura de aplicativo é o Gerenciador de Atividades que lida com o ciclo de vida dos aplicativos (DANIELSSON, 2016). Na parte superior da arquitetura

fica a camada de aplicativo. Essa é a camada em que os aplicativos estão localizados. Essas aplicações podem ser escritas em Java ou Kotiln[1] e, posteriormente, são compiladas em código de máquina para serem instaladas. Quando um projeto Android é criado, o desenvolvedor pode compilar o código usando o Android Software Development Kit (SDK). Isso resultará em um aplicativo no formato *Android Package* (APK) que consiste em um pacote do Android que pode ser instalado e executado em qualquer dispositivo real ou virtual.

Figura 3 – Arquitetura do Android.



Fonte: (DANIELSSON, 2016)

3.3 Desenvolvimento Multiplataforma

Nos últimos anos, várias técnicas e estruturas surgiram a fim de proporcionar uma solução para criar um aplicativo multiplataforma. O objetivo dessa abordagem é desenvolver apenas um serviço que pode ser acessado ou implantado em diferentes sistemas operacionais (AXELSSON; CARLSTRÖM, 2016). Uma técnica popular para isso é criar um site responsivo, um aplicativo na Web acessível por dispositivos móveis por meio de seu navegador. Usando recursos em *Hypertext Markup Language* (HTML) 5 e *Cascading Style Sheets* (CSS) 3 junta-

[1] https://kotlinlang.org/

mente com a biblioteca *front-end Bootstrap*[2], a aplicação web terá um interface que pode ser usada em um computador ou um dispositivo móvel. Isso resulta em um aplicativo da Web com uma interface como um site comum quando acessado por um computador, mas semelhante a um aplicativo quando acessado por um dispositivo móvel.

Outra alternativa é usar um *framework* híbrido para criação de um aplicativo que possa ser usado em qualquer sistema operacional. Essa abordagem é uma combinação do uso de web e desenvolvimento nativo, desde que o aplicativo seja construído usando técnicas da Web, mas executado, renderizado e exibido como um aplicativo nativo usando um *WebView* (AXELSSON; CARLSTRÖM, 2016). As capacidades do dispositivo são expostas por uma camada de abstração como API JavaScript, que permite que o aplicativo híbrido acesse funcionalidades e recursos do dispositivo. Contudo, mesmo que o custo de desenvolvimento de aplicativos híbridos seja significativamente menor do que os nativos, os híbridos não podem fornecer a mesma experiência de usuário nativa e, portanto, eles não foram sucesso na substituição do desenvolvimento nativo (AXELSSON; CARLSTRÖM, 2016).

3.3.1 React Native

Em 2015, o Facebook apresentou o *framework React Native*, cujo objetivo principal é permitir que o desenvolvedor crie aplicações para diferentes plataformas escrevendo um único código-fonte, sendo, portanto, uma alternativa para o desenvolvimento de aplicações móveis. Quando React Native foi lançado, havia apenas suporte para iOS, mas hoje já possui suporte para o Android (AXELSSON; CARLSTRÖM, 2016). Em vez de ser executado no navegador e renderizar "divs"e textos, o *React* executa nativamente em uma instância incorporada do *JavaScriptCore*[3] dentro dos aplicativos e renderiza componentes específicos de cada plataforma.

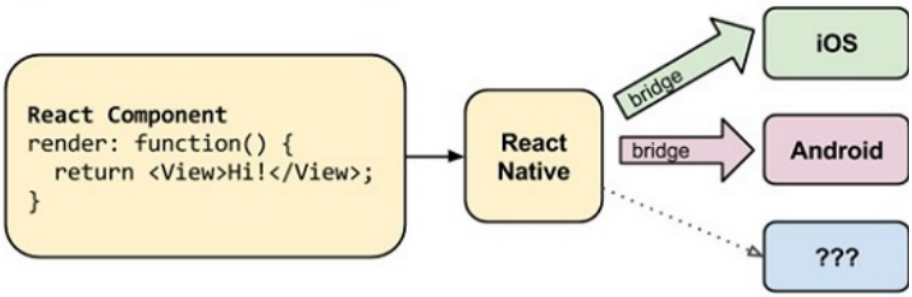
Na Figura 4 é mostrada a arquitetura do *React Native*. Nessa representação é possível notar que uma aplicação feita em React Native não usa WebViews e que é possível construir aplicativos com uma capacidade de resposta nativa. Além disso, em vez de criar uma base de código para todas as plataformas, o mesmo código é escrito e implantado para diversas plataformas e apenas algumas seções como elementos gráficos e os componentes específicos da plataforma são gravados separadamente.

Ao contrário de outras abordagens de desenvolvimento móvel multiplataforma, o

[2] https://getbootstrap.com/

[3] https://developer.apple.com/documentation/javascriptcore

Figura 4 – Arquitetura React Native.



Fonte: (DANIELSSON, 2016)

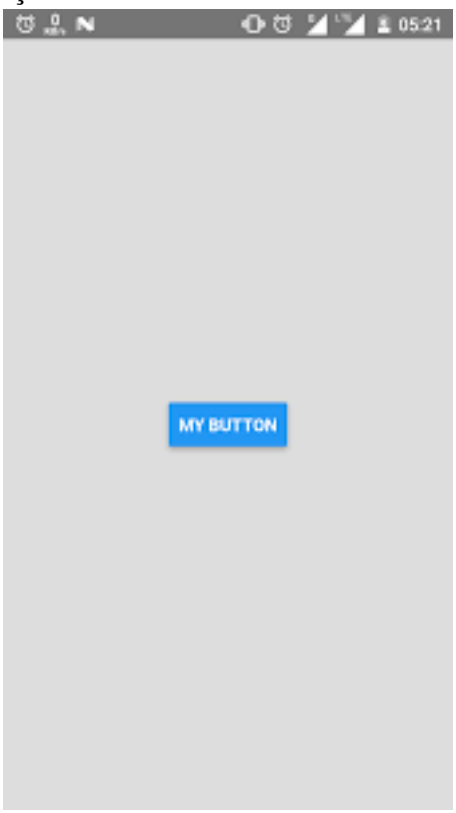
React Native não é uma solução híbrida e não depende de renderização dentro de WebViews ou tenta imitar HTML e CSS, como o Phonegap[4] e o Ionic[5]. O *layout* é executado em uma *thread* separada, sendo o segmento principal capaz de executar animações (AXELSSON; CARLSTRÖM, 2016).

Para exemplificar como o *React Native* torna o desenvolvimento de aplicativo mais fácil e rápido, consideremos uma aplicação simples executando na plataforma Android, mostrada na Figura 5. Essa aplicação possui um botão na cor azul no centro da tela. Na Figura 6 é mostrado o trecho do código dessa aplicação que representa a criação de um botão feito em código nativo Java.

Para a criação do botão, primeiramente, devemos criar um arquivo de interface baseado em *Extensible Markup Language* (XML), um arquivo que contém toda a definição de estilização da tela. Depois de criado, deve-se converter esse arquivo para um objeto da classe *Button* em Java. A função *findViewById* realiza esse processo. Após realizada essa etapa, passa-se a manipular esse objeto, adicionando mais ações. Para criar a mesma aplicação usando *React Native* utilizamos o código mostrado na Figura 7. É notório observar que, diferente do código nativo, basta realizar a importação do componente Button e já utilizá-lo onde for necessário. No *React Native* não é necessária a criação de qualquer arquivo XML para estilização de tela, uma vez que o componente *Button*, no processo final de geração do aplicativo, é convertido para um código Java Android semelhante ao apresentado na Figura 6.

[4] <https://phonegap.com/>
[5] <https://ionicframework.com/>

Figura 5 – Exemplo de aplicação Android



Fonte: Adaptada Autor.

Figura 6 – Exemplo de código em Java Android.

```
public class MyActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.content_layout_id);

        final Button button = findViewById(R.id.button_id);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Code here executes on main thread after user presses button
            }
        });
    }
}
```

Fonte: Adaptada Autor.

3.4 Web Service

O objetivo de um *Web Service* é realizar comunicação entre aplicações através do uso de padrões Web. A arquitetura de *Web Service* identifica os elementos globais da rede de serviços Web que são necessários para assegurar a interoperabilidade entre vários *Web Services*.

Figura 7 – Exemplo de código em React Native.

```
import { Button } from 'react-native';
...

<Button
  onPress={onPressLearnMore}
  title="Learn More"
  color="#841584"
  accessibilityLabel="Learn more about this purple button"
/>
```

Fonte: Adaptada Autor.

O funcionamento de um *Web Service* presume que deva existir, no mínimo, dois agentes trocando informações (um solicitante do serviço e um provedor do serviço). Existem também agentes de descobrimento de Web Services disponíveis na Web, que funcionam como verdadeiros catálogos de serviços disponíveis (ZAVALIK, 2004).

3.4.1 Padrão de comunicação REST

Umas das formas de comunicação através do uso do protocolo *Hypertext Transfer Protocol* (HTTP) é o padrão *Representational State Transfe* (REST). Trata-se de um estilo arquitetural que consiste em um conjunto coordenado de restrições aplicado a um sistema de hipermídia distribuído (FIELDING; TAYLOR, 2000). A seguir são descritas as restrições do REST, conforme definidas por Fielding e Taylor (2000):

- Modelo cliente-servidor: O modelo cliente-servidor usado para melhorar a portabilidade separa as responsabilidades entre as máquinas que fazem requisições e aquelas que as respondem. Por exemplo, no caso do cliente, não saber a forma que os dados são armazenados no servidor, enquanto este desconhece a interface do usuário ou os estados do cliente.
- Stateless (sem estado) é a restrição que impõe que cada requisição do cliente para o servidor precisa conter toda informação necessária para que ela seja entendida. Ou seja, o servidor não mantém estado, ou sessão, do cliente.
- Cache é usado para melhorar a eficiência da rede, é a restrição que requer que os dados contidos numa resposta de uma requisição seja implícita ou explicitamente rotulada como

cacheáveis ou não.

- Sistema em camadas, usado para melhorar a escalabilidade e reforçar políticas de segurança, é a restrição que previne um cliente de distinguir se está diretamente conectado ao servidor final ou a um servidor intermediário.
- Interface uniforme: Interface uniforme, usado para simplificar e desacoplar os componentes da arquitetura, permitindo que eles evoluam de forma independente. Essa é a restrição que distingue REST dos outros estilos arquiteturais, e é composta dos seguintes princípios:
  - Identificação de recursos: Qualquer informação que possa ser nomeada, pode também ser um recurso. Cada recurso tem um identificador, que no caso de web service é um *Uniform Resource Locator* (URL) (Uniform Resource Locator, Localizador Uniforme de Recurso). Alguns recursos são estáticos, outros são dinâmicos. Por exemplo, um web service fictício sobre música chamado webmusic poderia possuir os seguintes URLs: <https://webmusic.com/albums/madonna/hard_candy> (recurso estático) e <https://webmusic.com/albums/madonna/latest> (recurso dinâmico). O primeiro representa o álbum Hard Candy da Madonna, já o segundo, o último álbum lançado pela cantora. Em 2009, os dois URLs apresentariam a mesma informação, sendo que à partir de 23 de março de 2012, o URL <https://webmusic.com/albums/madonna/latest> passaria a apresentar informações do álbum MDNA, enquanto que <https://webmusic.com/albums/madonna/hard_candy>, continuaria apresentando o álbum Hard Candy. Apesar de ambos URLs apresentarem a mesma informação em um determinado momento, eles representam recursos diferentes, e, por isso, devem possuir identificadores distintos.
  - Uma representação é uma sequência de bytes mais os metadados que descrevem esta sequência. Um cliente executa ações sobre um recurso através de representações, afim de obter seu estado atual, ou modificá-lo transferindo uma representação do estado desejado.
  - Mensagens auto-descritivas: Cada mensagem possui informações suficientes para descrever como processar a mensagem.
  - Através de Hypermedia as the engine of application state (HATEOAS), Hipermídia como o motor do estado da aplicação, clientes executam transições de estado (navegação entre recursos e manipulação dos mesmos) somente através de ações que são dinamicamente identificadas em hipermídia pelo servidor. Ou seja, exceto por alguns

> simples pontos de entrada para a aplicação, um cliente não assume que nenhuma ação está disponível para nenhum recurso além daquelas descritas em representações anteriormente recebidas do servidor.

O núcleo da abordagem REST consiste na percepção de que, apesar do termo transporte em seu nome, o HTTP consiste em uma API e não um simples protocolo de transporte. Desta forma, o HTTP possui métodos bem definidos que correspondem as operações CRUD (*Create*, *Read*, *Update*, *Delete*).

No presente trabalho, é seguida a arquitetura de comunicação REST, para o desenvolvimento do *Web Service* proposto, enquanto que a aplicação cliente foi desenvolvida usando o *React Native*.

4 TRABALHOS RELACIONADOS

Esta seção mostra alguns trabalhos que utilizam ambientes virtuais. Foram selecionados trabalhos que lidam com tecnologias de RV e RA, uma vez que o sistema apresentado na Seção 5.1 foca na busca por imagens e modelos 3D, que são mídias tipicamente utilizadas em aplicações de RA e RV. Ao final da seção, o Quadro 1 mostrada uma comparação entre esses trabalhos e o sistema proposto.

O trabalho de Gugenheimer *et al.* (2016) apresenta uma cadeira que rotaciona de acordo com ações que acontecem no ambiente que é mostrado em um óculos de Realidade Virtual. Essa cadeira é equipada com dois motores rotacionais pequenos que são acoplados na parte traseira e na parte frontal. Esses motores são conectados a um sistema que possui como componente principal um Arduíno[1] que faz o controle desses motores. As ações dentro do ambiente virtual são diversas. Cada evento aciona o Arduíno que se encontra conectado ao óculos de Realidade Virtual. A partir dessas ações, o Arduíno rotaciona a cadeira à direita ou à esquerda, como, também, realizar pequenos chacoalhos, a fim de envolver mais o usuário nessa experiência.

O trabalho apresentado em (FERNANDES *et al.*, 2015) mostra uma nova ferramenta de RA para Automação Residencial, cuja finalidade é comandar ações, como ligar/desligar luz, por exemplo, para pessoas com deficiência física, com o objetivo de auxiliar a executar atividades do cotidiano. Nesse sistema, o usuário faz o controle de botões por meio de gestos faciais, em um aplicativo de RA para Android. Esse aplicativo está conectado a instrumentos de automação residencial, proporcionando maior interatividade durante o seu uso.

Em (SAFITRI *et al.*, 2017) é apresentado um aplicativo de Realidade Aumentada para plataforma Android para utilização pelos turistas que visitam a Indonésia. A Indonésia é um dos países onde o número de turistas brasileiros tem aumentado, no entanto, a falta de informação sobre esse pais é bastante significativa. Para contornar essa falta de conhecimento foi criada uma aplicação que mostra pontos de interesse na Indonésia, sendo que os locais são mostrados com imagens 360º e sons, além de ser permitida a manipulação de objetos 3D. Assim, o turista pode pré-visualizar o ponto de interesse, despertando sua curiosidade em visitar o local.

O Quadro 1 resume a relação entre este trabalho e os trabalhos mostrados. Em suma, o trabalho de Gugenheimer *et al.* (2016) apresenta a utilização de um sistema controlador que recebe informação do que está acontecendo dentro ambiente virtual e depois traduz para que

[1] https://www.arduino.cc/

dispositivo embarcado possa atuar na cadeira. O trabalho de Fernandes *et al.* (2015) utiliza aplicações para a plataforma Android, como interface de manipulação de ambientes físicos e, por fim, o trabalho de Safitri *et al.* (2017) utiliza tecnologias imersivas aliadas à plataforma Android. Contudo, um ponto de divergência em relação a este último trabalho é que toda a experiência do usuário, como visualização do ambiente e manipulação dos objeto 3D, é executado na aplicação cliente, enquanto que no presente trabalho a aplicação móvel é responsável por realizar a interface de busca de recurso e toda a experiência do usuário será controlada por um sistema de controle externo.

Quadro 1 – Comparação entre os trabalhos relacionados e o proposto

	(GUGENHEIMER *et al.*, 2016)	**(FERNANDES *et al.*, 2015)**	**(SAFITRI *et al.*, 2017)**	**Trabalho proposto**
Fez uso de Ambiente Virtual	Sim	Não	Não	Sim
Fez uso de aplicação servidora para prover conteúdo	Sim	Sim	Não	Sim
Fez uso de aplicação móvel	Não	Sim	Sim	Sim
Fez uso de aplicação de controle	Sim	Sim	Não	Sim

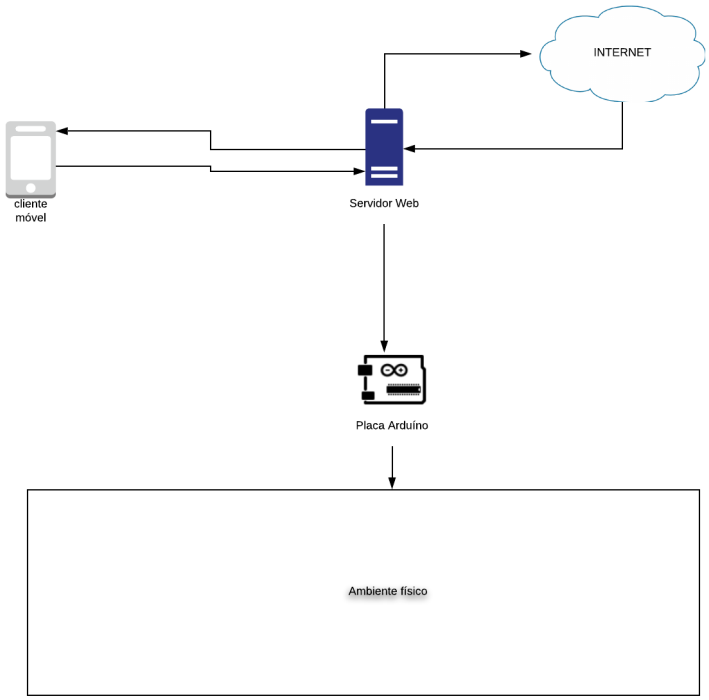
Fonte: elaborado pelo autor

5 METODOLOGIA

5.1 Módulos do sistema

O sistema proposto é dividido em três partes principais: Servidor Web, aplicação móvel e o sistema de controle, sendo que apenas as duas primeiras foram desenvolvidas. Na Figura 8 são representados os módulos do sistema e as interações entre eles. A aplicação cliente é responsável por interagir com usuário como também exibir imagens, músicas e objetos em três dimensões. O Web Service é responsável pela a busca e centralização de informações em diferentes tipos de provedores de recursos na internet. Já o sistema de controle é responsável por interagir com os objetos físicos, como lâmpadas, ar-condicionados e projetores.

Figura 8 – Arquitetura do sistema.

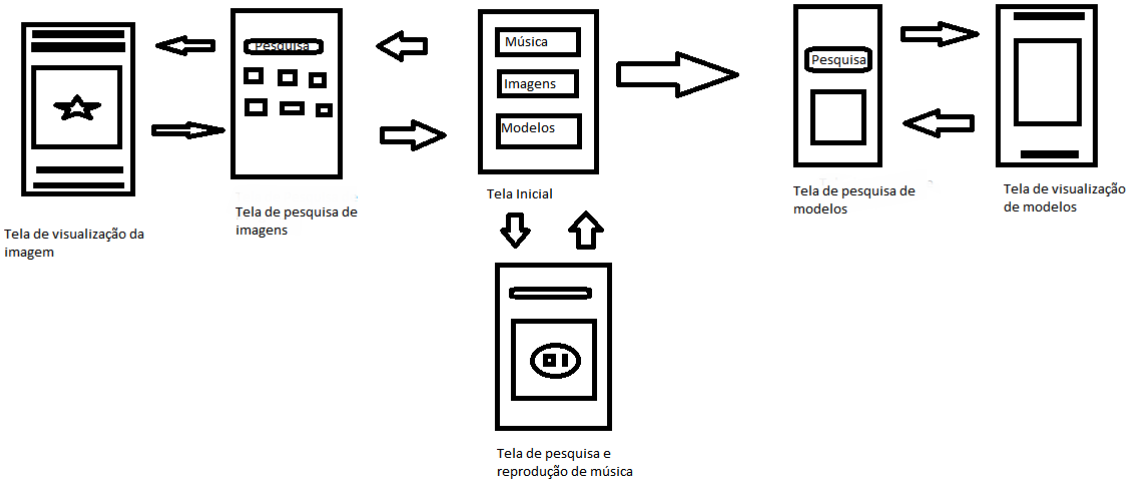


Fonte: Autor.

5.2 Aplicação móvel

A aplicação móvel é um aplicativo para plataforma Android. O aplicativo contém 6 telas. A partir da tela inicial é possível navegar para três dessas, as quais dão acesso às demais, conforme mostrado na Figura 9. A seguir cada tela será explicada individualmente.

Figura 9 – Navegação entre as telas da aplicação cliente.



Fonte: Autor

5.2.1 Telas de pesquisa e visualização das imagens

A tela de pesquisa de imagens contém um campo de texto no qual o usuário pode escrever o nome da imagem que queira procurar. Ao lado deste campo há um botão para realizar a pesquisa. Logo abaixo do botão e do campo de texto, tem-se uma lista que irá conter as imagens resultantes da pesquisa. Além disso, o usuário pode selecionar cada imagem individualmente, clicando sobre ela, de forma que logo em seguida seja aberto uma tela de visualização modal com essa imagem.

5.2.2 Tela de pesquisa das músicas

Diferente da tela de pesquisa de imagens, que possui um campo de busca, a tela de pesuisa de músicas possui um componente chamado "*dropdow*"que contém uma lista de opções preenchida com os gêneros musicais (MPB, Rock, etc.) que podem ser pesquisados. Depois de selecionado o gênero, o resultado da pesquisa é apresentado em formato de lista, sendo que cada elemento contém uma foto do compositor e o nome da música. Para reproduzi-la, basta clicar sobre a imagem enquanto que para pausar basta clicar novamente sobre a imagem.

5.2.3 Tela de pesquisa dos modelos 3D

Assim como na tela de imagem, a tela de pesquisa de modelos possui um campo de busca e botão para pesquisar. Depois que o usuário informa o nome do modelo, é apresentada uma lista contendo imagens, nome e descrição dos modelos. Para visualizar individualmente, basta clicar sobre a foto, que em seguida a tela da lista é substituída pela tela que exibe o modelo selecionado.

5.3 Web Service

O Web Service tem um papel importante, pois é a partir dele que aplicação móvel consegue realizar toda a busca de informação dos recursos para mostrar ao usuário. O Web Service atua como uma plataforma de centralização e busca de recurso na internet. Quando a aplicação cliente realiza uma requisição ao Web Service, ele realiza uma busca, em outro serviço na internet, pela informação solicitada pelo usuário. O resultado dessa pesquisa é filtrado para que sejam repassados para aplicação móvel somente os dados que ela necessita. Todas as requisições retornadas para aplicação móvel são em formato *JavaScript Object Notation* (JSON).

5.4 Sistema de controle

Com as configurações vindas do Web Service, o sistema de controle realiza as modificações no ambiente. O sistema de controle pode conectá-se aos aparelhos como ar-condicionado, projetores, luzes e caixa de som. Essa conexão pode ser através de uma placa de prototipação como Arduíno ou outra qualquer. Esse sistema de controle é encarregado de moldar o ambiente físico segundo alguns parâmetros fornecido pelo o Web Service. Um exemplo de atuação desse sistema é quando o usuário escolhe uma imagem e, dependendo da intensidade de cor que ela apresente, as luzes do ambiente irão acompanhar essa intensidade. Outro exemplo, é na escolha da música, que poderia ser reproduzida no ambiente.

Apesar de não ter sido desenvolvido um sistema de controle do ambiente físico, a Seção 6.3 mostra um exemplo funcional desse sistema utilizando o Arduíno, que é uma plataforma de prototipagem eletrônica baseada em software e hardware. Os Arduínos são capazes de ler entradas (por meio de sensores) e transformar em saída (por meio de atuadores) (BANZI; SHILOH, 2015).

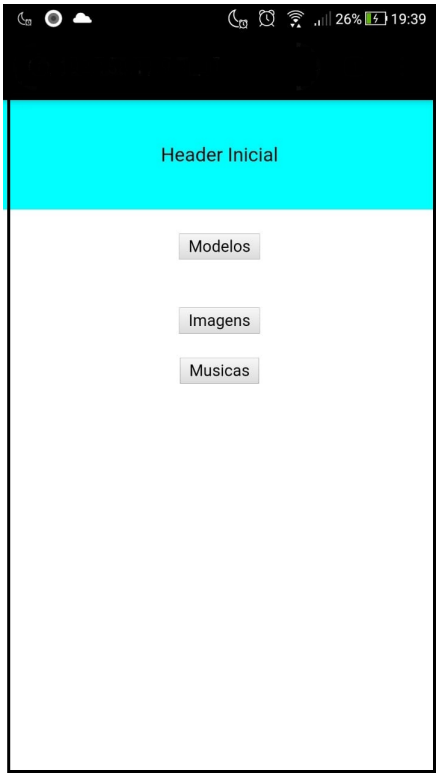
5.5 Desenvolvimento

5.5.1 Implementação da aplicação móvel

A aplicação móvel foi desenvolvida utilizando o *framework React Native*. Além do exposto na seção 3.3.1, outro fator para a escolha desse *framework* é que seus objetos são no formato JSON, que são os mesmos recebidos pelo *Web Service*, o que evita a necessidade de converter os objetos, facilitando o desenvolvimento da aplicação.

A tela inicial é mostrada na Figura 10. A partir dela que podemos realizar a navegação para outras telas. Foram utilizados três componentes de botão para realizar as transições entre as telas.

Figura 10 – Tela inicial



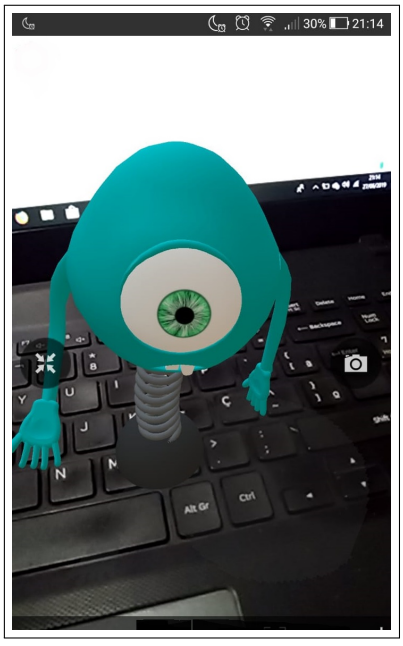
Fonte: Autor

Depois da tela inicial, foi desenvolvida a tela de imagens, composta por um componente de caixa de texto juntamente com um botão. Após concluir a pesquisa, o resultado é mostrado logo abaixo desses dois componentes. Para a visualização das imagens foi utilizada uma biblioteca do próprio React-native de imagens, que permite visualizar, ampliar e dimi-

nuir a imagem. Para a tela de música, foi utilizada a biblioteca de manipulação de música do React-native-audio-player[1], que permite a reprodução da música pela URL ou pelo arquivo de áudio. Todo o controle de reprodução, pausa e entre outras funcionalidades já está incluso nessa biblioteca.

Para o desenvolvimento da tela de modelos em 3D, foi utilizada a biblioteca Threejs[2]. O problema com essa biblioteca está no fato de ela não ser totalmente integrada com o React Native, exigindo uma implementação manual da visualização dos modelos. Contudo, em relação a outras bibliotecas disponíveis para plataforma React Native, a Threejs é que possui maior suporte da comunidade e documentação bastante clara. Para que um modelo seja renderizado na tela é necessário informar o link do arquivo no formato OBJ(Object File Wavefront 3D), que é um arquivo que contém toda a descrição do objeto como texturas, relevos, altura, largura entre outros. Na Figura 11 é mostrado um exemplo de visualização de objeto virtual com a biblioteca Threejs.

Figura 11 – Exemplo de visualização de um objeto virtual com a biblioteca Threejs.



Fonte: elaborado pelo autor.

[1] <https://github.com/GetStream/react-native-audio-player>
[2] <https://threejs.org>

5.5.2 Implementação do Web Service

O Web Service é responsável tanto pelo o gerenciamento do conteúdo, como pela sua centralização. Para o desenvolvimento do Web Service foi utilizada a linguagem Java[3], por meio do framework Spring Boot[4]. O Spring Boot utiliza a arquitetura REST com o protocolo de comunicação HTTP para a transferência de informações. O Spring Boot foi escolhido pela facilidade e rapidez no desenvolvimento de um servidor. Com essa plataforma é possível adicionar módulos que facilitam o desenvolvimento de aplicações escaláveis. Um dos módulos utilizados foi o *Feign*[5] que facilita o gerenciamento de requisições HTTP e realização requisições para outros sistema na internet via REST.

O Web Service funciona como Middleware, como forma de centralização de recurso. O Middleware age como uma "camada", capaz de fazer a mediação entre várias tecnologias de software, de modo que as informações (de diferentes fontes) são movidas ao mesmo tempo que suas diferenças de protocolos, plataformas, arquiteturas, ambientes e sistemas operacionais não interferem no processo. O processo de centralização atua na busca dos conteúdos de multimídias. Para cada recurso: imagem , música e modelos, a busca foi feito em diferentes API disponíveis na internet. Para a busca de imagem foi escolhido Unsplash, já para as músicas foi escolhido Spotify e, por fim, para os modelos foi escolhido Google Poly. Para a escolha dessas API foram levados, em consideração alguns critérios. Esses critérios podem ser visto nos quadros 2, 3 e 4.

Quadro 2 – Comparação entre API de Modelos

	Plataformas de suporte	**Base de dados**	**Base de Usuário**	**Velocidade de resposta requisiçãoo**
Google Poly	Mobile (Android) ,Unity, Web	40 milhões de modelos plano free	Não Informado	0.00553s
Sketchfab	Web	150 mil modelos plano free	Não Informado	0.00353s

Fonte: elaborado pelo autor.

[3] <https://openjdk.java.net/>
[4] <https://spring.io/>
[5] <https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-feign.html>

Quadro 3 – Comparação entre API de Imagem

	Plataformas de suporte	**Base de dados**	**Base de Usuário**	**Velocidade de resposta requisiçãoo**
unsplash	Modelo Rest	956 mihõess de imagens grátis	133.2 milhões de usuários	0.00203s
api.imgur	Modelo Rest	100 mihõess de imagens grátis	50 milhões de usuários	0.00103s

Fonte: elaborado pelo autor.

Quadro 4 – Comparação entre API de Música

	Plataformas de suporte	**Base de dados**	**Base de Usuário**	**Velocidade de resposta requisiçãoo**
Spotify	Web e Mobile (Android e IOS)	Não informado	174 milhões	0.00384s
musixmatch	Mobile (Android e IOS)	não informado	14 milhões	0.00353s

Fonte: elaborado pelo autor.

5.5.2.1 Interface de comunicação

Para realizar a comunicação entre o Web Service e a aplicação móvel foi definido o uso do JSON. O JSON é um formato leve de intercâmbio de dados baseado na linguagem JavaScript. O conteúdo de um documento JSON encontra-se em formato de texto mas usa convenções que são familiares às usadas em linguagens como C, C++,Java, Perl,Python, entre outras(IMHOF *et al.*, 2017).

5.5.2.2 Serviços

Quando uma requisição é feita ao Web Service o tratamento é feito utilizando o roteamento provido pelo próprio Spring Boot, onde é informado o tipo de requisição HTTP e a função que irá respondê-la. O acesso aos serviços do Web Service não requer nenhum tipo de autenticação, todos os URI's disponibilizadas são de acesso público.

Os serviços oferecido pela o Web Service são:

- **Buscar imagens**: Busca uma lista de imagens com base no nome informado pelo usuário.

- **Buscar uma imagem**: Busca uma imagem com base na lista de imagens já procuradas pela usuário.
- **Buscar músicas**: Busca uma lista de músicas com base no nome informado pelo usuário.
- **Buscar uma música**: Busca uma imagem com base na lista de músicas já procuradas pela usuário.
- **Buscar modelos**: Busca uma lista de modelos 3D com base no nome informado pelo usuário.
- **Criar ambiente virtual**: Cria uma configuração de ambiente virtual a ser enviada ao sistema de controle para projeção no ambiente físico.

Na Figura 12 são mostrados os *endpoints* que foram criados para disponibilizar os serviços.

Figura 12 – Lista dos serviços criados

Método HTTP	Endpoint's		
	ImageController	MusicControler	ModelControler
POST	-	-	-
GET	/image/{q}	/musicByName/{name} /music/{id}	/models/{keyword}
PUT	-		
DELETE	-		

Fonte: Autor

5.5.2.3 Pesquisa das imagens

Quando o usuário da aplicação cliente realiza uma pesquisa digitando no campo texto o nome de alguma imagem, essa informação é enviada para a URI "/api/image/{q}", na qual a letra "q"entre chaves representa o nome da imagem. Depois de enviada, ela é capturada pelo o Web Service que realiza o encaminhamento para pesquisa na API de imagem UNLASH, que a transforma em um objeto em formato JSON. Com o retorno da requisição vindo da API de imagem, o Web Service trata essa informação que também se encontra em formato JSON. O tratamento realizado é a busca das imagens que estão em formato codificado base64, logo em seguida essas imagens são enviadas para a aplicação móvel para serem mostradas ao usuário. Na Figura 13 é mostrada a requisição de resposta para tela de pesquisa de imagens.

Figura 13 – Requisição de retorno para a tela de imagens

```
{
  "id":"Código único da imagem",
  "urlImagem":"Link para imagem",
  "nome":"Nome da imagem",
  "descricao":"Descrição da imagem"
}
```

Fonte: Autor

5.5.2.4 Pesquisa dos modelos

A pesquisa de modelo em 3D se realiza da mesma forma que o processo de pesquisa das imagens. Depois de informado o tipo de modelo o qual se quer procurar, essa informação é enviada para a uri "/api/models/{keyword}"na qual a palavra *keyword* é substituída pelo critério informado pelo o usuário. A partir disso, o Web Service captura essa informação na URI e a repassa para a API POLY do Google para que seja feita a pesquisa dentro da sua base de dados.

O retorno da API é processado para filtrar apenas os campos de interesse à aplicação, como o nome do modelo, descrição e formatos. Os demais campos são desconsiderados. Esse tratamento é realizado para diminuir a quantidade de tráfego de informação entre o Web Service e a aplicação móvel, tornando mais rápidas as pesquisas. Os campos repassados para a aplicação móvel contém as informações dos modelos pesquisados em formato OBJ[6], formato que a biblioteca de visualização de objetos em 3D consegue decodificar. Na Figura 14 é mostrada a requisição de resposta para tela de modelos.

Figura 14 – Requisição de retorno para a tela de modelos.

```
{
  "id":"Código único do modelo",
  "urlImagem":"Link da imagem do modelo",
  "nome":"Nome do modelo",
  "modelo":"Link do arquivo do modelo"
}
```

Fonte: elaborado pelo autor.

[6] https://pt.wikipedia.org/wiki/OBJ

5.5.2.5 Pesquisa das músicas

Na pesquisa, o usuário informa o gênero escolhido e a partir disso a aplicação móvel envia uma requisição para a URI "/api/musica/{name}"na qual *name* é o gênero buscado. Com essa informação, o Web Service realiza uma pesquisa na API de música Spotify, e retorna uma coleção de músicas envolvendo o gênero informado, então essa informação é retornada em formato JSON, contendo *id*, nome e descrição da música, para aplicação móvel exibir para o usuário. Caso o usuário queira reproduzir, basta selecionar e logo em seguida aplicação móvel realiza novamente uma requisição para o Web Service passando o *id* da música para a URI "/api/music/{id}"uma vez capturada essa informação pelo Web Service, o *id* é repassado para a API do Spotify. O retorno desse processo é um JSON contendo um link para arquivo que contém o áudio da música solicitada. Por fim, esse link é repassado para a aplicação móvel a fim de ser reproduzida para o usuário. Na Figura 15 é mostrada a requisição de resposta para tela de música.

Figura 15 – Requisição de retorno para a tela de música.

```
{
  "id":"Código único da música",
  "url":"Link para a música",
  "nome":"Nome da música"
}
```

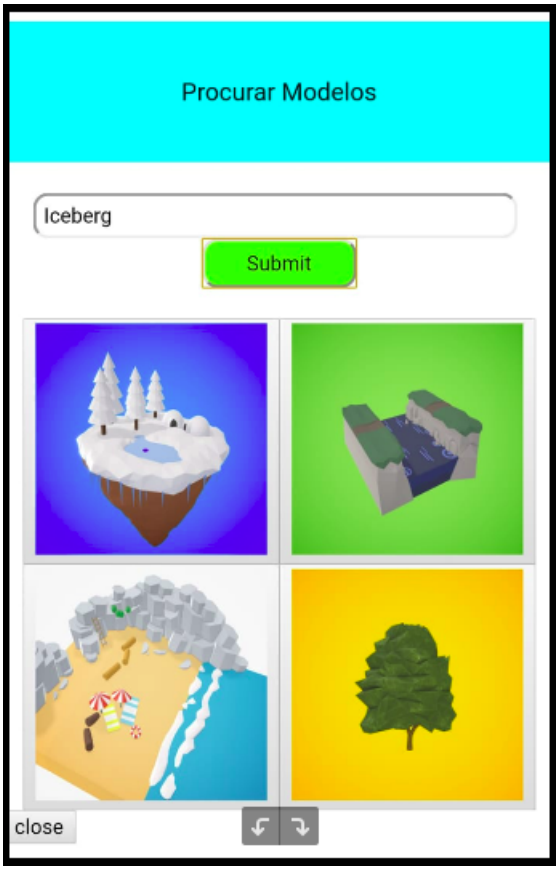
Fonte: elaborado pelo autor.

6 TESTES DO PROTÓTIPO

6.1 Aplicação cliente

A Figura 16 mostra a tela de pesquisa de imagens. Foram utilizados vários critérios de buscas para testar a pesquisa. Logo na parte inferior é mostrado o resultado da pesquisa, contendo várias imagens relacionadas aos critérios informados.

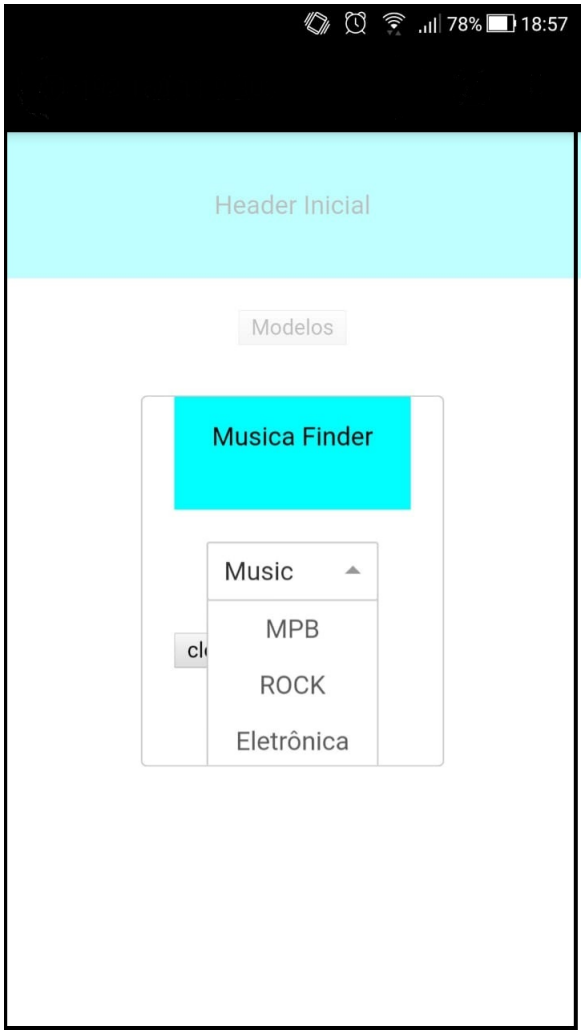
Figura 16 – Tela de pesquisa de imagens



Fonte: Autor

Na Figura 17 é exibida a tela de pesquisa de músicas. Nessa tela é possível observa o componente dropdown com alguns gêneros musicais já definidos. Depois de escolhido o gênero essa informação é enviada para o Web Service que retorna uma lista de músicas relacionada a essa busca. Já na Figura 18 é mostrada a tela de pesquisa e exibição dos modelos 3D.

Figura 17 – Tela de pesquisa de músicas



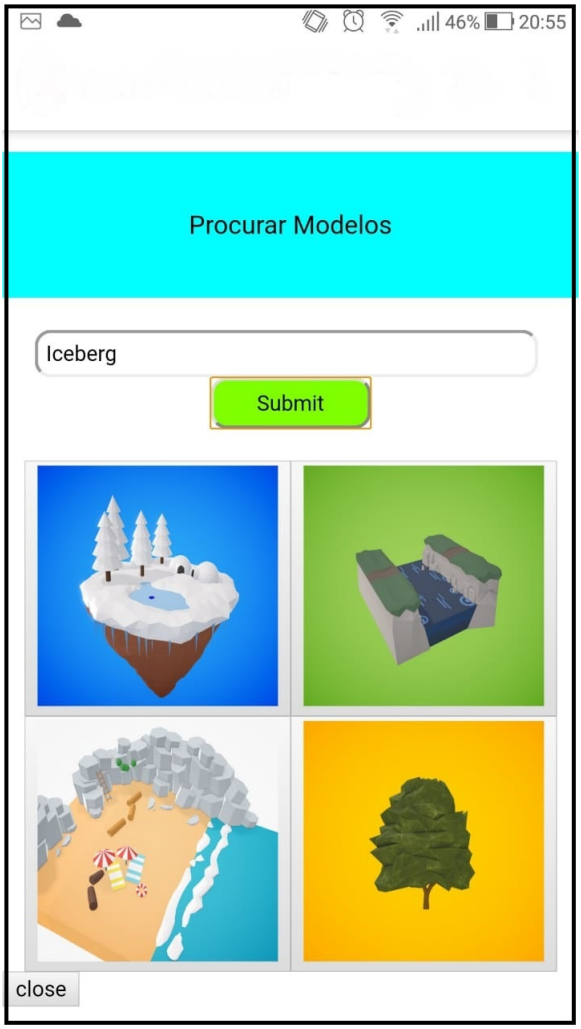
Fonte: Autor

6.2 Uso do Web Service

Primeiramente para utilizar o Web Service foram realizadas várias requisições via o aplicativo postman[1]. O postman é um aplicativo de teste de sistema com arquitetura REST, no qual é possível realizar requisições para os endpoint's da API. Essas requisições servem para testar como o sistema vai se comportar com determinada entrada de dados. Além disso, com os testes dos endpoint's, foi possível testar como o Web Service se comportaria na questão do tratamento das informações vindas dos outros serviços que se encontram conectados.

[1] https://www.getpostman.com/

Figura 18 – Tela de pesquisa de modelo



Fonte: Autor

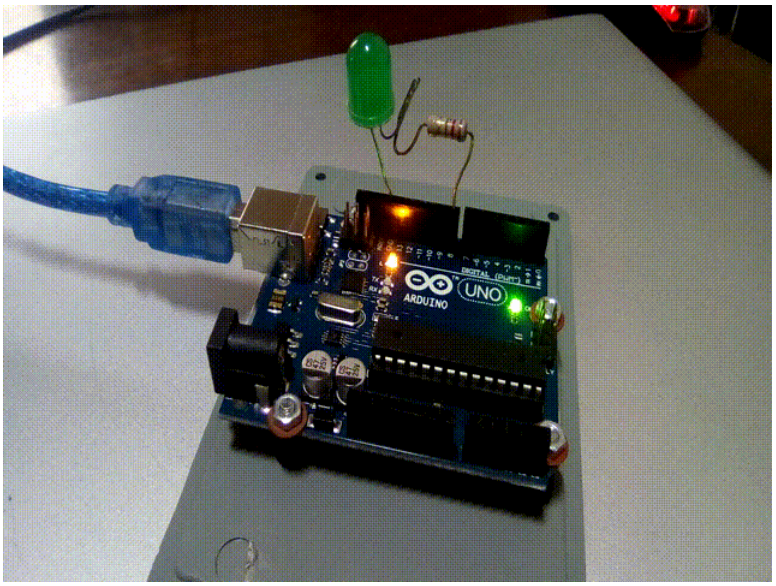
6.3 Exemplo de sistema de controle

Como forma de exemplificar o comportamento de um sistema de controle, foi feito um teste com a placa Arduino Uno (Figura 19). Esse teste foi realizado com um led para simular o controle de iluminação do ambiente. Com esse led, foram testadas várias intensidades de brilho para simular uma situação real em que o usuário escolhe uma imagem que contém a cor verde como predominante. Contudo não foi possível realizar mais testes com outros equipamentos, como controle de ar-condicionado e projetores, dada a indisponibilidade de equipamentos necessários para tal atividade. A Figura 20 mostra o uso do Arduíno como uma interface de controle.

Figura 19 – Estrutura de uma placa Arduino Uno

Fonte: (SILVA *et al.*, 2015)

Figura 20 – Teste com Arduíno e Led



Fonte: Autor

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou o protótipo de uma aplicação móvel que serve de interface para configuração de uma ambiente físico imersivo para o usuário. Além disso, apresentou o desenvolvimento de um Web Service que serve como centralizador de conteúdo entre diferentes fontes disponíveis na internet, além de um teste de sistema de controle mínimo do ambiente físico.

Dado o estágio atual de desenvolvimento, ainda há vários pontos a serem melhorados no projeto. A aplicação móvel é passível de adição de mais funcionalidades que melhorem a experiência do usuário. Uma dessas funcionalidades, é tornar a aplicação mais colaborativa. Os usuários poderiam compartilhar as configurações de ambientes. Já no sistema de controle, devem ser realizados testes de integração com mais equipamentos. Por falta de recursos e equipamentos não foi possível realizar testes na integração com aparelhos eletrônicos no ambiente.

Um trabalho futuro importante seria a implementação de um sistema para controle do ambiente. Esse sistema seria independente, com isso , seria responsável por modificar o ambiente de acordo com parâmetros repassados a ele. Esses parâmetros, em formato JSON, ajudariam na configuração e controle de aparelhos dispostos no local, como ar-condicionado, projetores ou lâmpadas.

REFERÊNCIAS

AXELSSON, O.; CARLSTRÖM, F. **Evaluation targeting react native in comparison to native mobile development**. [s.l.: s.n]. 2016.

BANZI, M.; SHILOH, M. **Make-Getting started with Arduino:** the open source electronics prototyping platform, make: Books. O'Reilly, United States, 2015.

BELIZARIO, J. **Android completa 10 anos com 88 por cento de participação no mercado de smartphones**. [S.l.]: TecInfo, 2018. Disponível em: https://www.tudocelular.com/mercado/noticias/n131097/10-anos-de-android-historia-sistema-.html. Acesso em: 28 abr.2019.

DANIELSSON, W. **React Native application development:** a comparison between native android and react native. [s.l.: s.n]. 2016.

ELISEU, S. R. T. D. N. **O mundo como uma CAVE-estratégias narrativas em realidade aumentada**. [s.l.: s.n]. 2017.

FERNANDES, F. G.; BARBOSA, J. L. M.; CARDOSO, A. Aplicação para auxílio às pessoas com deficiência física utilizando automação residencial e realidade aumentada. In: **CEEL–XIV Conferência de Estudos em Engenharia Elétrica, Uberlândia–MG**. [S.l.: s.n.], 2015.

FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7.

GUGENHEIMER, J.; WOLF, D.; HAAS, G.; KREBS, S.; RUKZIO, E. Swivrchair: A motorized swivel chair to nudge users' orientation for 360 degree storytelling in virtual reality. In: ACM. **Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems**. [S.l.], 2016. p. 1996–2000.

IMHOF, R.; FROZZ, A. A.; MELLO, R. dos S. **Um Survey sobre Extração de Esquemas de Documentos JSON**. [s.l.: s.n]. 2017.

KIRNER, C.; KIRNER, T. G. **Evolução e tendências da Realidade Virtual e da Realidade Aumentada**. [s.l.: s.n]. 2011.

MYERSON, J. M. **Web services architectures**. [s.l.: s.n]. 2015.

SAFITRI, R.; YUSRA, D. S.; HERMAWAN, D.; RIPMIATIN, E.; PRADANI, W. Mobile tourism application using augmented reality. In: IEEE. **2017 5th International Conference on Cyber and IT Service Management (CITSM)**. [S.l.], 2017. p. 1–6.

SEMENTILLE, A. B.; JRF, K. C., kubo, mm. ambientes virtuais distribuídos usando corba: Um estudo de caso. In: **III Workshop de Realidade Virtual–WRV**. [S.l.: s.n.], 2000. v. 3, p. 2000.

SILVA, J. T.; SILVA, J. T.; LIMA, G. F. Controle e monitoramento de nível utilizando plataforma open source arduino. **Revista INNOVER——ISSN: 2448-4105**, v. 1, n. 4, p. 85–92, 2015.

TORI, R.; KIRNER, C.; SISCOUTTO, R. A. **Fundamentos e tecnologia de realidade virtual e aumentada**. [S.l.]: Editora SBC, 2006.

ZAVALIK, C. **Integração de Sistemas de Informação através de Web services**.[s.l.: s.n]. 2004.