Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Sistema de Computação Móvel para Gerenciamento de Informações Relacionadas a Objetos do Espaço Físico

Ildeir de Oliveira Eler Junior

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

Vicente José Peixoto de Amorim

COORIENTAÇÃO:

Sérgio Evangelista Silva

Julho, 2019

João Monlevade–MG

Ildeir de Oliveira Eler Junior

Sistema de Computação Móvel para Gerenciamento de Informações Relacionadas a Objetos do Espaço Físico

Orientador: Vicente José Peixoto de Amorim

Coorientador: Sérgio Evangelista Silva

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2019

E392s Eler Junior, Ildeir de Oliveira .
Sistema de computação móvel para gerenciamento de informações relacionadas a objetos do espaço físico [manuscrito] / Ildeir de Oliveira Eler Junior. - 2019.

103f.: il.: color; grafs; tabs.

Orientador: Prof. MSc. Vicente José Peixoto de Amorim.
Coorientador: Prof. Dr. Sérgio Evangelista Silva.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Computação em nuvem. 2. Computação móvel. 3. Aplicativos móveis. 4. Sistemas de informação gerencial. I. Amorim, Vicente José Peixoto de. II. Silva, Sérgio Evangelista. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 004.775

Catalogação: ficha.sisbin@ufop.edu.br

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Colegiado do Curso de Engenharia de Computação
Campus João Monlevade

Curso Engenharia de Computação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Sistema de Computação Móvel para Gerenciamento de Informações Relacionadas a Objetos do Espaço Físico

Ildeir de Oliveira Eler Junior

Monografia apresentada ao Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CEA496 – Trabalho de Conclusão de Curso II, do curso de Bacharelado em Engenharia de Computação, e aprovada pela Banca Examinadora abaixo assinada:

Prof. Me. Vicente José Peixoto de Amorim
Orientador
Departamento de Computação e Sistemas/DECSI – UFOP

Prof. Dr. Sérgio Evangelista Silva
Coorientador
Departamento de Engenharia de Produção/DEPRO – UFOP

Prof. Me. Igor Muzetti Pereira
Prof. Convidado
Departamento de Computação e Sistemas/DECSI – UFOP

Profa. Dra. Gilda Aparecida Assis
Profa. Convidada
Departamento de Computação e Sistemas/DECSI – UFOP

João Monlevade, 11 de julho de 2019

Este trabalho é dedicado à Deus e à minha família.

Agradecimentos

Agradeço primeiramente à Deus, o Criador e mantenedor de tudo o que existe. Agradeço a Ele por estar comigo em todos os momentos, seja nas horas de calmaria, seja nos momentos difíceis me ajudando a passar por eles.

Agradeço à minha família por me ajudar e me apoiar em toda a jornada universitária, bem como em todas as áreas da minha vida. Agradeço também por me compreender em todos os meus momentos de ausência e indisponibilidade durante este período.

Agradeço ao meu orientador Vicente José Peixoto Amorim e meu co-orientador Sérgio Evangelista Silva por todos os conhecimentos passados e por me direcionarem nos caminhos corretos para a realização deste trabalho.

Agradeço também a todos os amigos que fiz até então, inclusive os da universidade, pelo companheirismo e ajuda nas mais diversas áreas da vida.

"Eu segurei muitas coisas em minhas mãos, e eu perdi tudo; mas tudo que eu coloquei nas mãos de Deus eu ainda possuo."

— Martin Luther King

Resumo

As informações de muitos objetos presentes no espaço físico precisam, de alguma forma, serem armazenadas. Entretanto, guardá-las em meios convencionais, como por exemplo o papel, pode ser algo difícil. Pode até mesmo ser perigoso, uma vez que existe um grande risco de perda por causa da concentração dos dados em um só lugar. No contexto acadêmico existem algumas soluções que visam resolver este problema através de sistemas computacionais, utilizando os conceitos de Computação Móvel e Computação em Nuvem. Neste trabalho é apresentado um sistema computacional móvel para gerenciamento de informações de objetos presentes no espaço físico. É utilizada a arquitetura cliente/servidor com um aplicativo móvel e uma interface *web* (clientes) para ter acesso às informações, e um servidor que as guarda e disponibiliza para quem quiser acessá-las. Este sistema foi feito tomando como base os objetos de patrimônio da Universidade Federal de Ouro Preto (UFOP). O aplicativo móvel faz a leitura de um código impresso (QR Code ou código de barras), e a partir dele recupera informações do servidor referentes a um objeto. Além disso, foi desenvolvido também um servidor de banco de dados, que realiza toda a manipulação dos dados conforme a necessidade do usuário.

Palavras-chaves: Computação em nuvem. Computação móvel. Gerenciamento. Objetos de patrimônio.

Abstract

The information of many objects present in physical space need, somehow, to be stored. However, store them in conventional ways, like paper for example, may be something difficult. May be even dangerous, once there is a big loss risk because of data concentration in only one place. In the academic context, there are some solutions that aims solve this problem through computational systems, using the Mobile Computing and Cloud Computing concepts. In this work is shown a mobile computational system for information management of objects presents in physycal space. It utilizes client/server architecture with a mobile application and a web interface (clients) for having information's access, and a server that store them and provides them to who wants access them. This system was made based on the property objects of Universidade Federal de Ouro Preto (UFOP). The mobile application reads a printed code (QR Code or Barcode), and based on it retrieves object's informations from server. Furthermore, also it was developed a database server, that accomplish all the data manipulation according to the user's need.

Key-words: Cloud computing. Mobile computing. Management. Property objects.

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

API *Application Programming Interface*

CSS *Cascading Style Sheets*

ER Entidade-Relacionamento

FAB *Floating Action Button*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

IC Iniciação Científica

IDE *Integrated Development Environment*

IoSp *Internet of Spaces*

IoT *Internet of Things*

JSON *JavaScript Object Notation*

JVM *Java Virtual Machine*

MDL *Material Design Lite*

PC *Personal Computer*

PDA *Personal Digital Assistant*

PHP *Hypertext Preprocessor*

QRCode *Quick Response Code*

REST *Representational State Transfer*

RFID *Radio Frequency Identification*

SGBD Sistema de Gerenciamento de Banco de Dados

SOAP *Single Object Access Protocol*

SQL *Structured Query Language*

TI Tecnologia da Informação

UFOP Universidade Federal de Ouro Preto

UPC *Universal Product Code*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

WAMP Servidor que integra: *Windows, Apache, MySQL, PHP*

WEEE *Waste Electrical and Electronic Equipment*

XML *eXtensible Markup Language*

Sumário

1 Introdução

A Tecnologia da Informação (TI) está em um constante processo de evolução. Desde o momento em que deu seus primeiros passos até os dias atuais, houve um grande avanço no que diz respeito a todas as áreas da vida humana (ROSSETTI; MORALES, 2007). Tal avanço vem do grande impacto social e econômico que gerou, visto que esta tecnologia mudou e vem mudando bastante a vida de várias pessoas.

Recentemente, nota-se uma grande evolução e popularização de dispositivos móveis (FIGUEIREDO; NAKAMURA, 2003), como os *smartphones*. Esses pequenos aparelhos foram projetados para serem verdadeiros computadores de bolso. Com eles, as pessoas podem realizar diversas tarefas, como por exemplo, navegar na *Internet*, fazer vídeo-conferências, checar *e-mails*, entre outras. Atividades que só eram possíveis em computadores de mesa, hoje podem ser realizadas na palma de nossas mãos.

Entretanto, muito além do que simplesmente realizar tais atividades, os *smartphones* possibilitam fazê-las de forma móvel, ou seja, não importando o lugar onde o usuário esteja (FIGUEIREDO; NAKAMURA, 2003). Esses aparelhos já vêm equipados com uma série de módulos de comunicação, como por exemplo, o *Wi-Fi*, *Bluetooth*, além de telefonia celular. Também vale ressaltar que os *smartphones* fazem uso de baterias, que proporcionam energia por um longo período de tempo sem a necessidade de alimentação externa. Todos esses fatores permitem a mobilidade do aparelho e a execução de tarefas em rede.

Atualmente, as informações consumidas pelas aplicações dos *smartphones* ficam armazenadas em grandes servidores de dados. Esses servidores possuem sua localização desconhecida pelos usuários e podem ser acessados através de aplicativos móveis, a fim de distribuírem dados armazenados previamente (TAURION, 2009). A isto dá-se o nome de Computação em Nuvem. Neste conceito, os dados são armazenados temporariamente nas memórias internas dos dispositivos. Mas eles são persistidos em grandes centros de dados (*datacenters*), para que possam ser acessados remotamente através da *Internet*.

1.1 O problema de pesquisa

Com este trabalho é pretendido apresentar uma solução para o problema de gerenciamento de bens de patrimônio de um local público. Segundo (VIECELLI, 2013), muitos funcionários ou servidores de uma determinada empresa se acham no direito de manipular e deslocar bens de patrimônio conforme sua vontade. E isto, consequentemente, acaba gerando certa dificuldade no gerenciamento destes bens, fazendo com que funcionários vão a campo para ver qual a situação que eles estão submetidos.

Todavia, a forma como isto é feito atualmente se dá através de meios impressos. Isto, por sua vez, implica em um acúmulo de grandes volumes de papéis. Por esta causa, a organização dos dados se torna mais difícil e, além disso, há uma maior dificuldade de alteração dos mesmos, uma vez que eles ficam gravados nos papéis. Vale observar também que desta maneira há uma centralização das informações em um ou poucos lugares específicos, o que os deixam suscetíveis a acidentes ou desastres naturais. Esses são alguns dos empecilhos encontrados no processo atual de gerenciamento.

Com a implementação de um sistema computacional, o trabalho de verificar o estado dos bens de patrimônio seria muito mais rápido e eficiente. Além disso, seria também muito mais flexível, tendo em vista a possibilidade de alterar dados no sistema em tempo real.

1.2 Objetivos

Este trabalho possui o seguinte objetivo geral:

- Desenvolver e implementar um sistema computacional móvel para acessar e gerenciar informações referentes a objetos presentes no espaço físico.

Especificamente neste trabalho, o sistema é desenvolvido com base nos objetos de patrimônio da Universidade Federal de Ouro Preto (UFOP). Para se alcançar o objetivo geral, é preciso completar os seguintes objetivos específicos:

- Desenvolver e implementar um aplicativo móvel para *smartphones* com sistema Android;
- Desenvolver e implementar um servidor de banco de dados para armazenar e manipular os dados referentes aos objetos de patrimônio;
- Desenvolver e implementar uma interface *web* para o servidor.

1.3 Organização do trabalho

O restante deste trabalho é organizado como se segue. O Capítulo 2 apresenta uma revisão bibliográfica dos principais conceitos teóricos envolvidos na realização deste trabalho. O Capítulo 3 apresenta de forma breve alguns trabalhos relacionados. O Capítulo 4 apresenta primeiramente uma breve descrição de cada material e tecnologia utilizados na implementação do sistema, e logo após descreve a metodologia seguida para se alcançar os objetivos propostos. O Capítulo 5 lista os resultados do desenvolvimento do sistema como um todo, mostrando capturas de tela tanto do aplicativo quanto da interface *web*. E por

fim, o Capítulo 6 apresenta algumas considerações finais, contribuições e alguns aspectos que podem ser trabalhados futuramente para a melhoria do sistema.

2 Conceitos Básicos

Este capítulo apresenta uma revisão da literatura, abordando todos os conceitos teóricos relevantes para a realização deste trabalho.

2.1 Computação móvel

Nos primórdios da história da computação eletrônica, por volta dos anos 40 (BROOKSHEAR, 2013), os computadores ocupavam um espaço físico grande e possuíam pouca capacidade de processamento e memória. Isso se dava, principalmente, pelo fato desses fazerem uso de componentes como válvulas e relês, que por si só já são volumosos e não têm uma resposta rápida. Entretanto, com a evolução da tecnologia, novas descobertas foram feitas. Entre elas, está a invenção dos transistores. Estes pequenos componentes marcaram o início da era dos circuitos integrados, o que significou a possibilidade de miniaturização dos computadores até então existentes (PERLES, 2007).

Com o advento dos circuitos integrados, os computadores deixaram de ocupar salas inteiras e passaram a ocupar mesas domésticas. Os chamados *Personal Computer*s (PCs) se popularizaram bastante e representam a forma mais conhecida de computação que se vê atualmente. Com eles, se tornou possível realizar tarefas cotidianas de maneira fácil, como por exemplo fazer cálculos, escrever textos, guardar arquivos, etc. Logo mais, com o advento da *Internet* por volta dos anos 80 (ABREU, 2009), tornou-se possível o compartilhamento de conteúdo através dos *web sites*, e também a comunicação através de *e-mails* e *chats*.

Conforme a tecnologia da informação foi avançando cada vez mais, os circuitos dos computadores foram diminuindo ainda mais o seu tamanho e aumentando sua capacidade de processamento e memória. Chegou a tal ponto que começaram a surgir os primeiros dispositivos móveis, sendo eles os *laptops*, *palmtops*, *Personal Digital Assistant*s (PDAs) e também os telefones celulares. Estes dispositivos já eram capazes de realizar tarefas do mesmo modo que nos computadores pessoais de mesa, com a diferença de poder fazê-las sem precisar estar em uma estação de trabalho fixa. Por esta razão, começou a surgir o conceito de Computação Móvel.

Segundo (FIGUEIREDO; NAKAMURA, 2003), a Computação Móvel representa um novo paradigma computacional que possibilita usuários obterem acesso a serviços independentemente da localização, podendo até mesmo, estar em movimento. Ainda segundo o autor, um dispositivo para ser considerado móvel deve cumprir os seguintes requisitos:

- Ser capaz de realizar processamento;
- Poder trocar informações via rede; e
- Poder ser facilmente transportado pelo usuário.

Já segundo (MATEUS; LOUREIRO, 1998), a "computação móvel representa um novo paradigma computacional". Ele também se baseia na questão de poder se locomover, mas não somente isso. O autor também ressalta que a computação móvel diz respeito à mudança de localização, ou seja, mobilidade. O autor também define um sistema distribuído com computadores móveis. Este consiste em uma parte tradicional fixa de infra-estrutura estática que está interligada a uma parte móvel em uma área onde existe comunicação sem fio entre os elementos computacionais móveis.

Neste trabalho, o conceito de Computação Móvel se aplica na utilização dos *smartphones* e *tablets*. Pois estes dispositivos possuem capacidade de realizar processamento e de serem movidos juntamente com o usuário para qualquer local. Uma vez que o aplicativo móvel desenvolvido neste trabalho executará nestes dispositivos, logo, tem-se a utilização deste conceito aqui.

2.2 Identificação visual

Grande parte dos objetos hoje em dia são visualmente identificáveis. Geralmente são marcados com desenhos, textos, cores diferentes ou até mesmo com números, de forma a facilitar a identificação pelos seres humanos. Além destas características, vale ressaltar também que as logomarcas e símbolos empresariais contribuem grandemente para que um objeto seja facilmente reconhecido, podendo até mesmo ser um fator que revele o valor e qualidade que este objeto possui.

Já para um sistema computacional, objetos são identificáveis através de números e códigos. Isso se dá principalmente pela necessidade de passar alguma informação importante sobre eles às pessoas, a fim de facilitar a interação entre objeto e usuário. Uma outra razão bastante comum para a identificação visual é a necessidade de gerenciar tais objetos, dependendo do contexto e local em que estão inseridos. Um bom exemplo disso são produtos de supermercado. Eles são marcados com códigos identificadores únicos, para que assim o estoque de produtos possa ser melhor controlado.

É interessante notar que estes códigos servem para serem lidos por máquinas, a fim de interpretar a identificação escrita no produto e guardá-la dentro de um computador para fins diversos. Para este objetivo, existem dois formatos de códigos bastante populares atualmente, que são o código de barras e o QRCode.

Figura 1 – Exemplo de código de barras

Fonte: (MILIES, 2006, p. 1)

2.2.1 Código de barras

Este tipo de código foi criado por George J. Laurer, funcionário na empresa IBM e teve a patente aceita em 1973 com o nome de *Universal Product Code* (UPC). Mesmo tendo recebido diversas outras versões com o passar do tempo, o UPC é, em sua essência, o mesmo código de barras mais popular e mais conhecido atualmente (MILIES, 2006).

O código de barras possui algumas vantagens. Ele pode ser lido pela máquina tanto em sentido normal como de cabeça pra baixo. Além disso, este tipo de código foi desenvolvido de forma que o número inscrito se auto-valide. Por exemplo, se um caixa informa manualmente o código do produto e digita um algarismo errado, o computador emite um sinal de erro dizendo que o código é inválido. Isso evita de o cliente pagar por um outro produto por causa do erro do caixa. Além da vantagem de poder ser lido por uma máquina, o código de barras também pode ser lido facilmente pelo ser humano, uma vez que o código é representado tanto em forma de barras como em forma de algarismos.

2.2.2 QRCode

O QRCode é um tipo de código mais recente. Foi criado no ano de 1994 pela empresa Denso, uma das maiores companhias do grupo Toyota. Antes dele, só existiam o código de barras e outros mais derivados desse, como o Code 39, Code 49 e o Code 16K. Porém, esses eram bem limitados, pois só podiam guardar até uma certa quantidade de dígitos (aproximadamente 100), e também não conseguiam codificar símbolos estrangeiros, que não pertencessem à língua inglesa. Por causa deste cenário, a Denso então decidiu desenvolver o que hoje é o QRCode, inicialmente para ser usado no controle de produção de peças automotivas (SOON, 2008).

O QRCode consiste em uma figura bi-dimensional com capacidade para armazenar até 7000 dígitos, podendo incluir símbolos estrangeiros, como os do Japão, por exemplo. Ele permite ser lido em todas as direções (360º) de forma eficiente. Além disso, ele pode

Figura 2 – Exemplo de QRCode

Fonte: (MASALHA; HIRZALLAH et al., 2014, p. 2)

ser lido quando as superfícies não são favoráveis, distorcendo de alguma forma os símbolos nele contidos. E também, mesmo se algo ocultar parte do QRCode, ele ainda pode ser interpretado sem maiores problemas pelo detector.

2.2.3 Identificação visual e computação móvel

As formas de identificação visual descritas acima tornaram possível uma maior integração entre o mundo real e o mundo virtual. Os computadores móveis atualmente estão evoluídos a ponto de serem capazes de ler e interpretar esses códigos através de uma câmera. Em decorrência disso, estes códigos (mais especificamente o QRCode), já estão presentes em grande parte de itens e objetos públicos, justamente para serem lidos pelos cidadãos, oferecendo a eles um ambiente de informações mais unificado e transparente.

(SILVA et al., 2018) propõem um novo paradigma de gestão da informação, chamado *Internet of Spaces* (IoSp), ou então, *Internet* dos Espaços. Este modelo se baseia nos conceitos de computação móvel e computação em nuvem, utilizando-se dos mesmos princípios e modos de funcionamento. O princípio deste paradigma consiste principalmente em "registrar e obter informações de objetos e espaços físicos diversos, a partir do uso da computação móvel e da computação em nuvem". Este modelo pode ser considerado como um conceito de *Internet of Things* (IoT), porém aplicado à espaços ao invés de objetos, significando assim que estes espaços interagem com o mundo virtual.

2.3 Computação em nuvem

Este termo vem sendo amplamente utilizado para se referir ao processamento realizado em uma estação computacional remota. Neste paradigma, as aplicações tendem a assumir um papel mais passivo no processamento de dados, delegando esta função a um conjunto de máquinas servidoras. Há várias definições diferentes na literatura para este conceito.

Segundo (TAURION, 2009), a computação em nuvem pode ser usada para descrever um ambiente computacional baseado em uma imensa rede de máquinas servidoras, podendo elas serem virtuais ou físicas. O autor ainda sugere algumas características deste paradigma que podem ser reunidas:

- A computação em nuvem cria uma ilusão da disponibilidade de recursos infinitos acessáveis sob demanda;
- A computação em nuvem elimina a necessidade de adquirir e provisionar recursos antecipadamente;
- A computação em nuvem oferece elasticidade, permitindo-se que as empresas usem os recursos na quantidade que forem necessários;
- O pagamento de serviços em nuvem é pela quantidade de recursos utilizados.

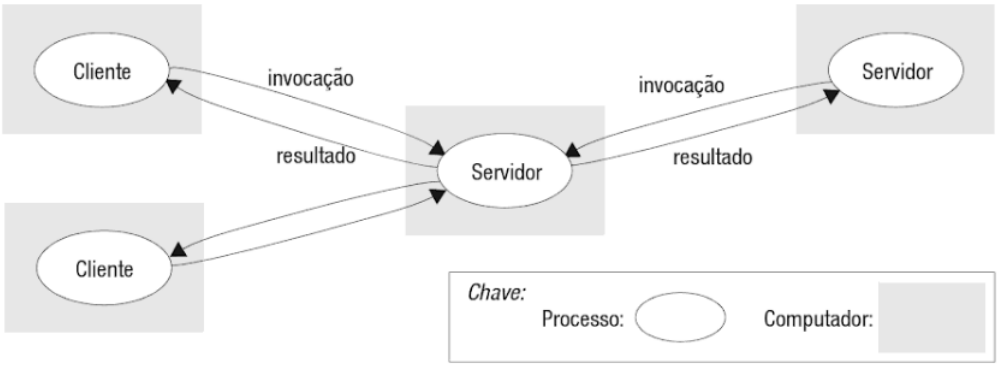
Já segundo (NIST, 2018), a computação em nuvem é um modelo para acessar convenientemente e sob demanda um conjunto de recursos computacionais compartilhados configuráveis (como por exemplo redes, servidores, armazenamento, aplicações e serviços), que podem ser rapidamente adquiridos e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços.

Para (RUSCHEL; ZANOTTO; MOTA, 2010), "a computação em nuvem é a ideia de utilizarmos, em qualquer lugar e independente da plataforma, os mais variados tipos de aplicações através da internet com a mesma facilidade de tê-las instaladas em nossos próprios computadores." Além de proporem esta definição, os autores fazem uma interessante analogia. Eles comparam a computação em nuvem aos serviços de utilidade pública, como água, eletricidade e telefone. Estes possuem toda uma infra-estrutura própria para entregar aos usuários seus serviços a qualquer hora e em qualquer lugar, conforme o pagamento é realizado. Da mesma forma, sistemas de computação em nuvem contam com uma infra-estrutura própria capaz de processar requisições e retorná-las ao usuário conforme este realiza o pagamento de acordo com os serviços contratados.

2.4 Arquitetura cliente/servidor

Este tipo de arquitetura já é bastante consolidada em sistemas distribuídos. Segundo (COULOURIS et al., 2013), ela é a mais citada neste assunto, a mais importante e continua sendo amplamente empregada em diversos sistemas computacionais. Os autores definem o funcionamento da arquitetura da seguinte forma: "os processos clientes interagem com processos servidores, localizados possivelmente em distintos computadores hospedeiros, para acessar os recursos compartilháveis que estes gerenciam". A Figura 3 mostra como é

Figura 3 – Esquema básico da arquitetura cliente/servidor



Fonte: (COULOURIS et al., 2013, p. 47)

organizada a estrutura básica deste tipo de arquitetura. Os retângulos cinzas representam os computadores como máquinas físicas, e as elipses representam os processos, podendo ser clientes ou servidores. De acordo com a figura, cada processo executa em uma máquina diferente. As setas de invocação significam requisições feitas a um determinado processo, e as setas de resultado representam as mensagens contendo o resultado dessas requisições. Vale ressaltar também que as mensagens são trocadas somente entre processos.

Este é um tipo de arquitetura em que a carga computacional fica centralizada no processo servidor, o qual fica sempre a disposição para receber requisições dos vários clientes, processá-las e retorná-las com os devidos resultados. A principal vantagem da arquitetura cliente/servidor é a sua simplicidade de implementação, uma vez que não é preciso se preocupar em processar a carga de maneira igual em todos os dispositivos de um sistema. Em contrapartida, este tipo de arquitetura pode facilmente ocasionar um gargalo no sistema, dependendo da quantidade de demanda que chega ao servidor para ser processada. Se esta demanda for relativamente grande, o servidor fica sobrecarregado e não consegue processar todas as requisições de forma correta.

2.5 *Web services*

Os *web services* (ou serviços *web*) são conjuntos de métodos disponibilizados na *Internet* para serem usados por diversas aplicações. Segundo diz (BAX; LEAL, 2001): "Os serviços *web*, aplicações que comunicam pela *Internet*, implementadas de forma flexível e fracamente acoplada, estão tendo um papel crescente nas interações entre empresas por meio eletrônico". Isso dá uma ideia de que estes métodos disponibilizados na *Internet* não dependem muito de uma plataforma ou linguagem específicas, sendo implementados de forma que qualquer aplicação consiga usar os recursos e obter uma resposta.

(LOPES; RAMALHO, 2004) descreve os *web services* como sendo aplicações

modulares, auto-descritivas, acessíveis através de URL, independentes das plataformas de desenvolvimento e que permitem a interação entre aplicações sem intervenção humana. Esta definição dá um ideia de que não necessariamente estes métodos precisam ser acessados por usuários humanos, mas também podem ser acessados por outros serviços e sistemas.

Os *web services* trabalham de forma bem semelhante à arquitetura cliente/servidor, baseando-se no modelo de requisições e respostas. Para isso, existem basicamente dois tipos de protocolo usados nestes serviços: o *Single Object Access Protocol* (SOAP) e o *Representational State Transfer* (REST).

- SOAP: Neste protocolo, é preciso definir toda a arquitetura responsável pela comunicação entre as partes, isto é, como os objetos serão transmitidos, como as interfaces serão implementadas, e quais metadados serão enviados (NUNES; DAVID, 2005). Tudo isto deve ser criado do início de forma a se encaixar da melhor maneira nos requisitos funcionais de um *web service* SOAP. Neste protocolo é utilizada a linguagem XML para definir toda essa estrutura de comunicação entre clientes e serviços.

- REST: Neste protocolo existem algumas partes da arquitetura que não precisam ser reimplementadas, pois os *web services* REST utilizam do modelo arquitetural HTTP para estabelecerem a comunicação entre os clientes e seus serviços. Neste caso, os recursos de um *web service* REST são acessados através de identificadores únicos de recursos, as URIs. É também utilizada a interface uniforme da *web*, que consiste em métodos *Hypertext Transfer Protocol* (HTTP) como o GET, POST, PUT e DELETE (NUNES; DAVID, 2005). A principal vantagem deste modelo é prover uma interface uniforme de acesso, já que essa é a interface HTTP, utilizada por todas as páginas *web*.

Neste trabalho, o modelo REST foi implementado no servidor para estabelecer um padrão simplificado de comunicação entre os clientes (aplicativo e interface *web*) e o servidor.

3 Trabalhos Relacionados

O trabalho de (DIAS; ZAVAN; OLIVEIRA, 2018) consistiu em desenvolver um sistema para o gerenciamento de inventário de bens móveis (patrimônio) do Instituto Federal do Paraná - campus Paranavaí. De forma semelhante ao presente trabalho, este conta também com um aplicativo móvel *Android* e uma interface *web*. As funcionalidades do sistema propostas pelos autores são muitas e incluem: gestão de espaço físico, gestão de equipamentos, movimentação de bens, conferência da localidade dos bens e relatórios para análise de conformidade e não conformidade. Neste trabalho, os QRCodes também estão presentes, atuando como forma de identificação visual dos bens, sendo afixados a eles. Com o aplicativo, é possível ler estes códigos e obter informações sobre um determinado bem, a fim de facilitar o processo de gerenciamento. Diferentemente do trabalho do autor, o sistema aqui desenvolvido implementa o conceito de *design* visual *Material Design* (melhor explicado adiante) tanto no aplicativo como na interface *web*. Desse modo, a interface de uso do sistema se torna mais unificada e mais amigável ao usuário, uma vez que este conceito preza pela simplicidade e fluidez. Além disso, o *Material Design* é utilizado nas aplicações mais populares existentes no mercado, o que faz com que a usabilidade do sistema desenvolvido se torne mais familiar ao usuário.

O trabalho de (DRESCH; EFROM; GRUMOVSKI, 2008) consistiu em criar um sistema também de gerenciamento de bens de patrimônio. Porém, diferentemente do presente trabalho e do anterior, o controle de patrimônio é feito através de etiquetas RFID. Inicialmente, as etiquetas são colocadas nos objetos presentes em um determinado ambiente. E então o sinal emitido por elas é captado automaticamente por antenas, transmitindo assim os dados relacionados a estes objetos. Segundo os autores, o RFID possui várias vantagens em relação a outros modelos de identificação, como por exemplo: a eliminação de erros de escrita e leitura de dados, coleção de dados de forma mais rápida e automática, redução de processamento de dados e maior segurança. Entretanto, um sistema envolvendo esta tecnologia possui um alto custo de implementação, uma vez que as etiquetas possuem um preço considerável e que na maioria dos casos um patrimônio possui milhares de itens, o que elevaria ainda mais o custo de implementação do sistema. Vale ressaltar que os autores apenas simularam um ambiente real de objetos de patrimônio, trabalhando apenas com protótipos.

O trabalho realizado por (NETO, 2015) consistiu em criar um sistema para dispositivos móveis que objetiva auxiliar pessoas na orientação interior de um edifício. Seu funcionamento é simples: Vários QRCodes são espalhados pelo edifício. Quando um usuário lê um destes através do *smartphone* ou *tablet*, o sistema retorna um conjunto de rotas possíveis para os lugares mais importantes do prédio, partindo de onde o usuário realizou a

leitura do código. Quando ele escolhe seu destino de interesse, um mapa é mostrado na tela juntamente com o desenho da rota a ser percorrida para chegar ao destino. Este sistema também foi desenvolvido para *Android* e se utilizou da identificação visual e computação móvel para solucionar o problema proposto.

(WANG et al., 2014) realizaram uma interessante pesquisa que envolve o uso do conceito de computação em nuvem. A pesquisa consistiu em desenvolver um sistema de remanufatura de equipamentos de lixo eletrônico, em inglês denominados *Waste Electrical and Electronic Equipment*s (WEEEs), baseando-se na computação em nuvem. Na maioria dos casos, os equipamentos eletrônicos em seu estado final de vida são descartados e jogados no lixo. Os usuários não possuem informações suficientes sobre reciclagem ou sobre como lidar com estes equipamentos. Então os autores propuseram um modelo de sistema chamado *WRCloud* que registra na nuvem informações dos equipamentos recém-fabricados. Quando eles chegam no estágio final de vida, o usuário então lê o QRCode presente no objeto através da interface de usuário do *WRCloud*. E então é retornado a ele um conjunto de serviços especializados na remanufatura destes equipamentos, conjunto este selecionado especialmente de acordo com as informações inseridas previamente no início de vida deles. Assim, o usuário escolhe a melhor opção, podendo ser para reciclagem ou para o reaproveitamento dos produtos. O trabalho citado consiste em uma pesquisa, não apresentando implementação real do sistema.

O trabalho de (COSTA; KOMATI, 2013) consistiu em desenvolver um sistema de compras rápido e seguro, baseado em QRCodes, utilizando a ideia de vitrine virtual. Uma vitrine virtual pode ser uma revista, um folheto ou encarte, em que nele há um QRCode associado ao produto que se deseja comprar. Quando o usuário lê este código através do aplicativo desenvolvido para *smartphone*, o produto é automaticamente adicionado ao carrinho de compras do usuário, mostrando informações como o nome do produto, o preço de estoque e uma imagem. Os autores quiseram priorizar a questão da segurança do sistema. Portanto para tal, as informações contidas nos QRCodes são criptografadas, garantindo assim que o código lido é exatamente o mesmo gerado pela mesma empresa que vende os produtos.

4 Desenvolvimento

Este capítulo descreve o desenvolvimento do trabalho. A seção 4.1 apresenta cada ferramenta e tecnologia relevante utilizada para a implementação do sistema. Já a seção 4.2 descreve os passos seguidos para a realização do trabalho visando alcançar os objetivos propostos.

4.1 Tecnologias e materiais utilizados

Nesta seção são listadas todas as ferramentas e tecnologias utilizadas para a implementação do sistema.

4.1.1 *SQL*

Structured Query Language (SQL) é a linguagem padrão universal para manipular bancos de dados relacionais, sendo aceita pela quase totalidade dos produtos que existem atualmente no mercado. Esta linguagem foi originalmente desenvolvida na *IBM Research* por volta dos anos 70 para ser usada em um protótipo da própria empresa, e depois foi reimplementada em seus vários outros produtos comerciais (DATE, 2004). A linguagem SQL permite realizar consultas desde as mais simples até as mais complexas em um banco de dados, garantindo um retorno de dados preciso e consistente. Além das consultas, ela permite fazer uma manipulação completa dos dados (adicionar, excluir e editar), de forma a garantir sempre a integridade dos mesmos.

4.1.2 *MySQL*

O *MySQL* é um Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto muito conhecido e amplamente utilizado para se trabalhar com a linguagem SQL. Um SGBD é uma ferramenta que tem por objetivo armazenar e gerenciar bases de dados em um sistema computacional. Ele é o intermediador entre o usuário e o banco de dados propriamente dito, permitindo que o usuário realize consultas e manipulações de dados, através da execução de comandos SQL. A maioria dos SGBDs possui uma interface gráfica para facilitar o gerenciamento do banco de dados. Neste trabalho, foi utilizada a interface própria do *MySQL* para projetar o esquema do banco de dados, o *MySQL Workbench*. Com o auxílio desta ferramenta, criar tabelas, colunas e relacionamentos tornou-se uma tarefa mais fácil e de melhor compreensão.

4.1.3 *WAMP Server*

O WAMP é um *software* que instala todas as ferramentas necessárias para criar um ambiente de servidor de *Internet* dentro de um computador. Como pode-se notar no acrônimo, cada letra é a inicial dos recursos disponibilizados neste conjunto.

- *Windows*: Indica que se deve executar estes recursos na sistema operacional *Windows*.
- *Apache*: É um *software* que transforma um computador em um servidor HTTP, possibilitando-o a receber requisições da *Internet*.
- *MySQL*: É o SGBD descrito na subseção 4.1.2.
- *PHP*: É uma linguagem de programação bastante usada para lidar com o *back-end* das páginas *web*. O Servidor que integra: *Windows, Apache, MySQL, PHP* (WAMP) já vem nativamente com um interpretador para essa linguagem, permitindo assim que as páginas executem corretamente.

Neste trabalho, o WAMP foi utilizado para testar o sistema em um computador pessoal. Assim, não foi necessário contratar um serviço de hospedagem de *websites* para executá-lo.

4.1.4 PHP

O *Hypertext Preprocessor* (PHP) é uma linguagem de desenvolvimento de código aberto amplamente utilizada e conhecida. Segundo (CONVERSE; PARK, 2003), "o PHP é uma linguagem de criação de *scripts* do lado servidor, que pode ser incorporada em HTML ou utilizada como um binário independente (embora a primeira utilização seja a mais comum)". Esta linguagem é interpretada e é usada geralmente para manter a lógica *back-end* de um sistema *web*. Além de ser bastante poderosa, é também uma linguagem simples. Um exemplo disso, é a facilidade para se integrar e interagir com um banco de dados SQL. Neste trabalho, o PHP foi utilizado para programar o funcionamento do servidor, o qual manipula as requisições recebidas do usuário e realiza as devidas atualizações no banco de dados.

4.1.5 *Slim Framework*

O *Slim* é um *microframework* PHP de código aberto criado para montar *Application Programming Interfaces* (APIs) de uma maneira mais simplificada. Ele se identifica fortemente com o modelo REST, oferecendo vários recursos para que este modelo seja implementado de forma funcional. Com ele, é possível definir facilmente rotas de recursos

(*Uniform Resource Identifiers* (URIs)) por meio dos métodos HTTP, parâmetros e condições. Além disso, com ele é possível implementar redirecionamentos, paradas e camadas de *middleware* (FRANCISCO, 2016). Neste trabalho, o *Slim* foi usado como ferramenta para implementar o modelo REST no servidor.

4.1.6 JSON

O JSON é um formato de texto que tem por finalidade serializar dados estruturados. É semelhante ao XML, porém, possui uma sintaxe mais limpa e de melhor compreensão para o ser humano. Um exemplo disso pode ser vista na Figura 4. Ela apresenta dois pequenos trechos fictícios de código. O da esquerda foi escrito em XML e o da direita em JSON, com o objetivo de se ter uma comparação visual entre a sintaxe das duas linguagens. O JSON, como o próprio nome já diz, é a notação nativa de objetos estruturados da linguagem *JavaScript*. Entretanto, ultimamente ele está sendo utilizado como formato padrão para troca de mensagens em sistemas distribuídos. Neste trabalho, o JSON foi utilizado justamente para esta finalidade, atuando como formato padrão das mensagens trocadas entre o servidor e o aplicativo/interface *web*.

Figura 4 – Comparação entre XML (à esquerda) e JSON (à direita)

xml.xml

```xml
<funcionarios>
    <funcionario>
        <nome>José da Silva</nome>
        <idade>56</idade>
    </funcionario>
    <funcionario>
        <nome>João Carlos</nome>
        <idade>45</idade>
    </funcionario>
    <funcionario>
        <nome>Paulo Souza</nome>
        <idade>32</idade>
    </funcionario>
</funcionarios>
```

json.json

```json
"funcionarios": [
    {
        "nome": "José da Silva",
        "idade": 56
    },
    {
        "nome": "João Carlos",
        "idade": 45
    },
    {
        "nome": "Paulo Souza",
        "idade": 32
    }
]
```

Fonte: Elaborado pelo autor

4.1.7 *Android Studio*

O *Android* é uma plataforma completa para dispositivos móveis que oferece vários recursos, como um sistema operacional, interface de usuário, aplicativos, APIs, entre vários outros (PEREIRA; SILVA, 2009). Por ser *open source* e por apresentar um bom desempenho e experiência de uso, rapidamente o sistema ganhou popularidade entre usuários e desenvolvedores. Atualmente, ele é um dos sistemas mais populares do mundo no que diz respeito ao mercado de *smartphones*, *tablets*, *smartwatches*, e outros dispositivos móveis.

O *Android Studio*, por sua vez, é uma *Integrated Development Environment* (IDE) criada para o desenvolvimento de aplicações para a plataforma *Android*. É uma ferramenta completa que oferece vários recursos interessantes e essenciais, como por exemplo um editor de *layouts*, emuladores, integração com o *GitHub*, *debugger*, entre outros. Esta ferramenta é baseada na IDE *IntelliJ IDEA*, criada para desenvolver aplicações em *Java* e *Kotlin*.

Neste trabalho, o *Android Studio* foi utilizado para implementar o aplicativo para *smartphones* e *tablets* que executem o sistema *Android*.

4.1.8 *Java*

Java é uma linguagem de programação muito utilizada em diversos sistemas computacionais e embarcados. Possui uma vasta quantidade de bibliotecas nativas que possibilitam aos desenvolvedores implementar quaisquer funcionalidades que possam existir em uma aplicação. Suas principais características são:

- Orientação a objetos: O paradigma de programação do *Java* é caracterizado pela abstração de objetos do mundo real. As estruturas de dados são definidas a partir de objetos com propriedades próprias e métodos, sendo possível através destes a interação mútua entre os diversos tipos de dados presentes em um sistema.
- Linguagem interpretada: Diferentemente da linguagem C, por exemplo, o *Java* não é compilado pelo sistema operacional. A partir do código fonte, o compilador gera arquivos binários *.class*. Estes arquivos, por sua vez, são interpretados pela *Java Virtual Machine* (JVM), a máquina virtual do *Java*, a qual é responsável também por executar o programa interpretado.
- Portabilidade: Pelo fato de serem executados por uma máquina virtual, programas *Java* podem rodar em quaisquer dispositivos em que a JVM possa ser instalada. E estes dispositivos não incluem somente computadores de médio a grande porte, mas também dispositivos móveis e embarcados.
- Segurança: A linguagem *Java* preza também pela segurança na execução de programas. Conta com um recurso de tratamento de exceções que permite manter a consistência do programa no caso de erros (INDRUSIAK, 1996). Além disso, a linguagem possui um padrão de segurança que garante a integridade do código.

Neste trabalho, o *Java* foi utilizado como linguagem de implementação do aplicativo para *Android*, sendo usado dentro do *Android Studio*.

4.1.9 *ZXing*

O *ZXing* (abreviação de *Zebra Crossing*) é uma biblioteca *open source* não nativa da plataforma *Android*. Com ela, é possível adicionar uma nova funcionalidade ao dispositivo móvel, que é a de ler códigos, sejam eles de barras, QRCodes, entre outros formatos, através da câmera do dispositivo. O *ZXing* oferece um alto grau de liberdade para os desenvolvedores, possibilitando a manipulação do resultado obtido da leitura e, até mesmo, a customização da tela de leitura, sendo possível adicionar novos elementos e botões na tela.

Neste trabalho, o *ZXing* foi utilizado na implementação do aplicativo *Android* para possibilitar a leitura dos QRCodes e códigos de barras dos itens de patrimônio pelo dispositivo móvel.

4.1.10 *Retrofit*

O *Retrofit* é uma outra biblioteca não nativa *open source* da plataforma *Android*. Ela oferece uma API completa em *Java* para a manipulação de requisições HTTP, prezando pela alta organização e modularização do código. O *Retrofit* permite criar requisições tanto síncronas como assíncronas, bem como adicionar parâmetros, manipular cabeçalhos HTTP e adicionar camadas interceptadoras.

Neste trabalho, o *Retrofit* foi utilizado para que o dispositivo móvel pudesse consumir, de forma assíncrona, o *web service*.

4.1.11 Linguagens para interface *web*

Para o desenvolvimento da interface *web* foram utilizadas as três linguagens principais para este fim: *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript*.

O HTML não necessariamente é uma linguagem de programação, mas sim, uma linguagem de marcação de texto. Ela é responsável por definir o conteúdo de uma página *web* através dos componentes, organizando-os de forma hierárquica. Tais componentes podem ser blocos de textos, campos de formulários, botões, listas, entre outros. Resumidamente, o HTML permite a implementação da interface de usuário da página *web*.

Enquanto o HTML define o conteúdo de uma página *web*, o CSS define como este conteúdo será apresentado, no que diz respeito ao estilo e *design*. Esta linguagem é utilizada para manipular aspectos da aparência dos componentes de uma página, a fim de mantê-la sempre com uma boa estética e aparência amigável ao usuário. Não somente aspectos de estilização, mas o CSS permite também melhorar a acessibilidade de uma página *web*. Através do recurso de responsividade, os componentes de uma página se

reorganizam dinamicamente de acordo com o tamanho da tela do dispositivo em que ela está aberta. Isto permite uma maior abrangência de dispositivos em que uma página pode ser vista de forma correta.

O *JavaScript* é a linguagem de programação padrão das páginas *web*. Enquanto o HTML e o CSS estão associados à parte visual da página (*front-end*), o *JavaScript* se encarrega de toda a lógica por trás da página, que acontece de forma invisível ao usuário (*back-end*). Com esta linguagem é possível alterar dinamicamente as propriedades dos componentes, e até mesmo criar outros novos com base em alguma ação do usuário. Assim, a página *web* deixa de ter um comportamento completamente estático, passando a ter um bom nível de dinamismo. O *JavaScript*, assim como o *Java*, é uma linguagem orientada a objetos e interpretada. Porém, existem várias diferenças entre elas, como por exemplo, o fato do *JavaScript* não ser fortemente tipado.

4.1.12 *Material Design Lite*

O *Material Design* é uma linguagem de *design* desenvolvida pela *Google* no ano de 2014. Consiste em uma série de guias e princípios básicos visuais que visam unificar a interface de aplicações, criando uma verdadeira e única identidade visual. O *Material Design* segue um conceito visual minimalista, em que só são permitidas formas geométricas simples (como círculos, quadrados e retângulos), poucas cores (porém vivas e chamativas) e animações rápidas e fluidas. Tudo isso tem o objetivo de aproximar a experiência do usuário da experiência de manusear materiais do mundo físico, mais especificamente, objetos que remetam a papel e tinta (GOOGLE, 2019). Aplicações que seguem as diretrizes propostas pelo *Material Design* possuem um *layout* mais limpo e fluido, melhorando e muito sua experiência de uso.

Por se tratar mais de um conceito do que uma ferramenta em si, o *Material Design* pode ser implementado em diversos dispositivos e plataformas, como *smartphones*, *tablets*, e até mesmo em páginas *web*. Para isso, a *Google* oferece uma extensa documentação e diversas ferramentas aos desenvolvedores.

E uma destas ferramentas é o *Material Design Lite* (MDL). Este nada mais é do que uma biblioteca de componentes padrões usados no *Material Design* para páginas *web*. O MDL possui o nome *Lite* porque é uma implementação mais simples deste conceito, não envolvendo nenhum *framework* externo. Sendo assim, o desenvolvedor precisa conhecer somente as linguagens básicas mencionadas na subseção 4.1.11 para poder utilizá-lo. Como consequência desta simplicidade, o MDL não dá suporte a grande parte das animações de objetos. Entretanto, mesmo não havendo elas, a experiência de uso não fica comprometida e o *layout* continua limpo e de fácil manuseio.

O MDL foi utilizado neste trabalho como ferramenta para adequar o *design* das

páginas da interface *web* aos princípios do *Material Design*. Também foi utilizado com o intuito de unificar a interface com a do aplicativo *Android*, uma vez que este também foi implementado baseando-se nestes princípios.

4.1.13 *jQuery*

O *jQuery* é uma biblioteca livre e de código aberto para a linguagem *JavaScript*. Possui a finalidade de tornar mais simples a escrita de códigos na linguagem, uma vez que para criar efeitos simples, por exemplo, é preciso escrever mecanicamente longas linhas de código. Com esta biblioteca, um desenvolvedor não precisa conhecer a fundo o *JavaScript*, uma vez que se trata de uma biblioteca feita com base nessa linguagem de programação (SILVA, 2013). Tal simplificação permite implementar recursos complexos de forma quase automática, como por exemplo, validação de formulários, criação de tabelas dinâmicas, e até mesmo, fazer requisições HTTP via comandos *ajax*.

Neste trabalho, o *jQuery* foi utilizado justamente para simplificar a escrita das requisições HTTP que consomem o *web service*. Neste caso, foram escritos comandos *ajax* de acordo com os recursos que o *web service* tem a oferecer aos clientes.

4.2 Metodologia

Nesta seção são detalhados os passos seguidos para a realização deste trabalho.

4.2.1 Contexto do trabalho

Primeiramente, é importante observar que no ano anterior ao início deste trabalho, foi realizado um projeto de Iniciação Científica (IC) cujo objetivo principal está interligado ao deste trabalho: desenvolver um artefato que sirva como elemento de orientação/comunicação visual, que no projeto foi denominado *Ubstone*. Parte do objetivo deste projeto foi desenvolver um aplicativo móvel para que o conceito principal pudesse ser colocado em prática, testando e validando assim as funcionalidades pretendidas. Considerando requisitos de otimização do desempenho dos dispositivos móveis, foi necessário desenvolver também um servidor de banco de dados para armazenar as informações relacionadas aos objetos presentes no espaço físico.

Como resultado do projeto de IC, o artefato foi devidamente desenvolvido e implementado. Entretanto, a parte de implementação do sistema móvel (aplicativo e servidor) não foi completamente finalizada, deixando de fornecer algumas funcionalidades importantes para um bom funcionamento do sistema como um todo.

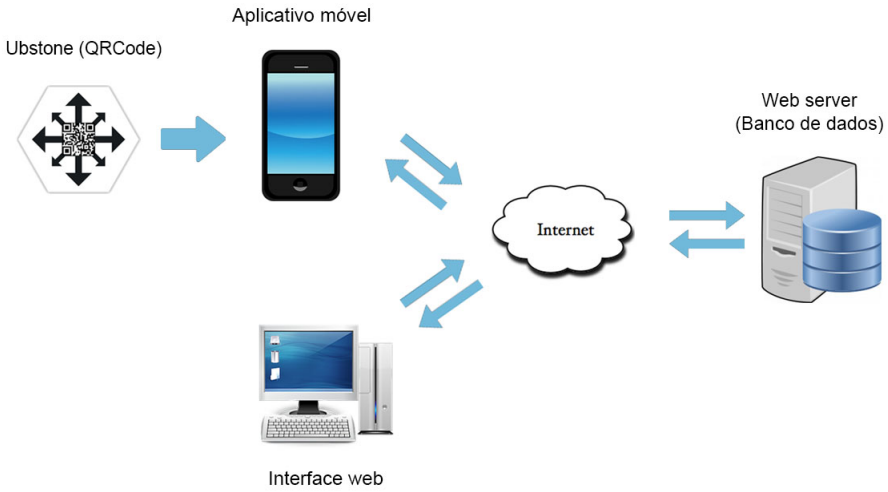
Tendo em vista este contexto, a ideia deste trabalho surgiu como uma forma de dar continuidade a implementação deste sistema. Além disso, foram estabelecidas algumas

metas adicionais para se conseguir este objetivo, como redefinir padrões de *design* e usabilidade, e também desenvolver uma interface *web* para uso administrativo. E por fim, definiu-se um nome para o sistema móvel baseado no nome do artefato desenvolvido no projeto de IC. Este nome é *Ubspaces*.

4.2.2 Visão geral do sistema

O sistema *Ubspaces* é um sistema cliente/servidor, no qual tanto o aplicativo móvel como a interface *web* assumem o papel de clientes. E o *web server* assume o papel de servidor, o qual é responsável por receber as requisições dos clientes e manipular diretamente o banco de dados. A Figura 5 mostra a estrutura básica do sistema.

Figura 5 – Estrutura do sistema *Ubspaces*



Fonte: Elaborado pelo autor

Tanto o aplicativo móvel como a interface *web* fazem uso do mesmo *web service* disponibilizado pelo servidor. Eles enviam requisições HTTP pela *Internet* destinadas ao servidor. O servidor, por sua vez, recebe estas requisições e executa os métodos solicitados. Após processá-las e fazer as devidas atualizações no banco de dados, o servidor retorna as respostas aos clientes em formato JSON. Estas respostas são recebidas pelos clientes que exibem novas informações ao usuário.

Vale ressaltar também que o dispositivo móvel, possuindo uma câmera, é capaz de ler o QRCode ou código de barras (*Ubstone* na Figura 5) de um determinado objeto. Com base na identificação extraída, o dispositivo móvel envia requisições ao servidor, que por sua vez retorna as informações referentes especificamente ao objeto lido.

Levando em consideração que este trabalho foi feito baseando-se no gerenciamento dos bens de patrimônio da UFOP, o *Ubspaces* foi desenvolvido assumindo-se que existem

três potenciais tipos de usuários para o sistema:

- **Usuário comum**: Não participa do sistema de gerenciamento dos bens de patrimônio. Nesta categoria se enquadram alunos, professores e outras pessoas que não possuem vínculo com a universidade. Um usuário comum pode somente pesquisar objetos e obter informações sobre ele. Ao usar o aplicativo, o usuário tem somente a opção de escanear o QRCode de um objeto e visualizar as informações dele a partir disso. Ao usar a interface *web*, ele pode pesquisar objetos por nome e data de compra/cadastro. Para este tipo de usuário, é satisfatório obter maior transparência a respeito dos bens de patrimônio de entidades públicas.

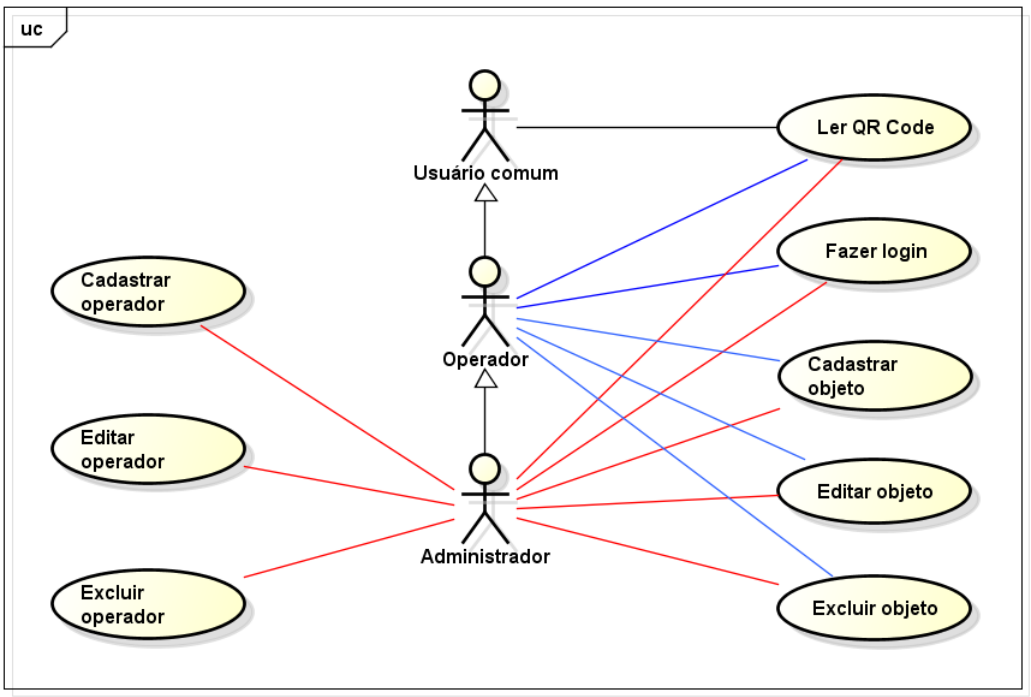
- **Operador**: Já este tipo de usuário participa ativamente do sistema de gerenciamento dos bens de patrimônio, e possui permissão para fazer alterações no mesmo. Nesta categoria estão os funcionários públicos designados especialmente para esta função. Um operador pode realizar as mesmas funções de um usuário comum, além de poder manipular os dados presentes no sistema *Ubspaces*. Ao usar o aplicativo, o operador pode pesquisar, adicionar, editar e excluir objetos. Ao usar a interface *web*, ele pode realizar estas mesmas funções, com exceção de pesquisar objetos por QRCode.

- **Administrador**: Por fim, este tipo de usuário também participa ativamente do sistema de gerenciamento dos bens de patrimônio e está um nível acima dos operadores. Nesta categoria se encontra também um funcionário público designado especificamente para esta função. O administrador é capaz de realizar todas as funções designadas aos dois outros tipos de usuários mencionados, além da função de gerência dos operadores. Ao usar o aplicativo, o administrador pode realizar as mesmas tarefas dos operadores. Já ao usar a interface *web*, o administrador pode manipular dados dos objetos, sendo possível pesquisar, cadastrar, editar e excluir operadores no sistema.

4.2.3 Casos de uso

A Figura 6 mostra o diagrama de casos de uso das principais funcionalidades do sistema *Ubspaces*. Todos os casos de uso se aplicam tanto ao aplicativo móvel quanto à interface *web*, com exceção daqueles relacionados ao gerenciamento de operadores. O diagrama em questão foi desenvolvido utilizando-se a ferramenta *Astah*.

Figura 6 – Diagrama de casos de uso do sistema



Fonte: Elaborado pelo autor

Além do diagrama de casos de uso, foram desenvolvidas também estórias textuais para cada um deles, sendo cada estória referente a cada funcionalidade presente no diagrama. Elas estão contidas nos quadros 1 ao 8.

Quadro 1 – Caso de uso: Ler QRCode

Caso de uso: Ler QRCode
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Usuário comum, operador ou administrador. Interessados: • Usuário comum: Quer realizar uma consulta sobre algum objeto do patrimônio da UFOP. • Operador: Quer realizar uma consulta sobre algum objeto do patrimônio da UFOP ou acessá-lo para alterar informações sobre ele. • Administrador: Quer realizar uma consulta sobre algum objeto do patrimônio da UFOP ou acessá-lo para alterar informações sobre ele. Pré-condições: • O dispositivo deve possuir uma câmera e precisa estar conectado à internet. • O aplicativo precisa estar aberto na tela de leitura de QR Code. Garantias de sucesso: O QR Code do objeto é reconhecido pelo dispositivo. O aplicativo automaticamente redireciona para uma tela que exibe todas as informações referentes ao objeto.
Cenário de sucesso principal: 1. O ator primário abre o aplicativo no modo de leitura de QR Code. 2. O ator primário posiciona o smartphone de modo a capturar uma imagem do QR Code através da câmera. 3. O sistema reconhece o QR Code. 4. O aplicativo exibe as informações referentes ao objeto requisitado. Extensões: 4a. O objeto requisitado não está cadastrado no banco de dados. 1. Uma mensagem de erro é exibida.
Frequência de ocorrência: Alta

Fonte: Elaborado pelo autor.

Quadro 2 – Caso de uso: Fazer *login*

Caso de uso: Fazer login
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Operador ou administrador. Interessados: • Operador: Quer inserir ou alterar informações sobre objetos do patrimônio da UFOP. • Administrador: Quer inserir ou alterar informações sobre objetos do patrimônio da UFOP, ou gerenciar os operadores do sistema, ou gerar relatórios. Pré-condições: • O dispositivo precisa estar conectado à internet. • O aplicativo/interface web precisa estar aberto na tela de login. Garantias de sucesso: O sistema autentica as informações de identificação fornecidas.
Cenário de sucesso principal: 1. O ator primário abre o aplicativo/interface web na tela de login. 2. O ator primário informa seus dados de identificação, que são um nome de usuário e uma senha. 3. O sistema reconhece e autentica as informações de identificação fornecidas. Extensões: 3a. O ator primário informa seus dados incorretamente ou eles não existem. 1. Sistema mostra uma mensagem de erro e solicita ao usuário para tentar novamente.
Frequência de ocorrência: Moderada

Fonte: Elaborado pelo autor.

Quadro 3 – Caso de uso: Cadastrar objeto

Caso de uso: Cadastrar objeto
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Operador ou administrador. Interessados: • Operador: Quer cadastrar informações de um novo objeto no sistema. • Administrador: Quer cadastrar informações de um novo objeto no sistema. Pré-condições: • O dispositivo precisa estar conectado à internet. • O operador ou administrador precisa estar logado no sistema. Garantias de sucesso: Os dados de um objeto são inseridos no sistema.
Cenário de sucesso principal: 1. O ator primário abre o aplicativo/interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Cadastrar objeto”. 3. O ator primário insere as informações do novo objeto e clica em “Confirmar”. 4. Os novos dados são salvos no sistema. Extensões: 3a. O ator primário informa um número de tombamento já existente. 1. Sistema mostra uma mensagem de erro e não salva as informações.
Frequência de ocorrência: Moderada

Fonte: Elaborado pelo autor.

Quadro 4 – Caso de uso: Editar objeto

Caso de uso: Editar objeto
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Operador ou administrador. Interessados: • Operador: Quer atualizar informações de um objeto já cadastrado no sistema. • Administrador: Quer atualizar informações de um objeto já cadastrado no sistema. Pré-condições: • O dispositivo precisa estar conectado à internet. • O operador ou administrador precisa estar logado no sistema. • O objeto a ser editado precisa estar cadastrado no sistema. Garantias de sucesso: Os dados de um objeto são atualizados e salvos no sistema.
Cenário de sucesso principal: 1. O ator primário abre o aplicativo/interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Listar objetos”. 3. O ator primário seleciona na lista o objeto que deseja editar. 4. O ator primário clica no ícone de edição. 5. O ator primário insere os novos dados do objeto e clica em “Confirmar”. 6. Os novos dados são salvos no sistema. Extensões: 5a. O ator primário informa um número de tombamento já existente. 1. Sistema mostra uma mensagem de erro e não salva as informações.
Frequência de ocorrência: Baixa

Fonte: Elaborado pelo autor.

Quadro 5 – Caso de uso: Excluir objeto

Caso de uso: Excluir objeto
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Operador ou administrador. Interessados: • Operador: Quer manter em um local separado os objetos que já foram descartados ou que não são mais utilizados. • Administrador: Quer manter em um local separado os objetos que já foram descartados ou que não são mais utilizados. Pré-condições: • O dispositivo precisa estar conectado à internet. • O operador ou administrador precisa estar logado no sistema. • O objeto a ser excluído precisa estar cadastrado no sistema. Garantias de sucesso: Um ou mais objetos são movidos para a seção de objetos excluídos.
Cenário de sucesso principal: 1. O ator primário abre o aplicativo/interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Listar objetos”. 3. O ator primário seleciona na lista um ou mais objetos que deseja excluir. 4. O ator primário clica no ícone de exclusão. 5. Os objetos são movidos para a seção de objetos excluídos.
Frequência de ocorrência: Baixa

Fonte: Elaborado pelo autor.

Quadro 6 – Caso de uso: Cadastrar operador

Caso de uso: Cadastrar operador
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Administrador. Interessados: • Administrador: Quer cadastrar informações de um novo operador no sistema. Pré-condições: • O dispositivo precisa estar conectado à internet. • O administrador precisa estar logado no sistema. Garantias de sucesso: Os dados de um operador são inseridos no sistema.
Cenário de sucesso principal: 1. O ator primário abre a interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Cadastrar operador”. 3. O ator primário insere as informações do novo operador e clica em “Confirmar”. 4. Os novos dados são salvos no sistema. Extensões: 3a. O ator primário informa um endereço de e-mail já associado a um outro operador. 1. Sistema mostra uma mensagem de erro e não salva as informações.
Frequência de ocorrência: Moderada

Fonte: Elaborado pelo autor.

Quadro 7 – Caso de uso: Editar operador

Caso de uso: Editar operador
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Administrador. Interessados: • Administrador: Quer atualizar informações de um operador já cadastrado no sistema. Pré-condições: • O dispositivo precisa estar conectado à internet. • O administrador precisa estar logado no sistema. • O operador a ser editado precisa estar cadastrado no sistema. Garantias de sucesso: Os dados de um operador são atualizados e salvos no sistema.
Cenário de sucesso principal: 1. O ator primário abre a interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Listar operadores”. 3. O ator primário seleciona na lista o operador que deseja editar. 4. O ator primário insere os novos dados do operador e clica em “Confirmar”. 5. Os novos dados são salvos no sistema. Extensões: 4a. O ator primário informa um endereço de e-mail já associado a um outro operador. 1. Sistema mostra uma mensagem de erro e não salva as informações.
Frequência de ocorrência: Baixa

Fonte: Elaborado pelo autor.

Quadro 8 – Caso de uso: Excluir operador

Caso de uso: Excluir operador
Escopo: Ubspaces. Nível: Objetivo de usuário. Ator primário: Administrador. Interessados: • Administrador: Quer descadastrar um operador do sistema. Pré-condições: • O dispositivo precisa estar conectado à internet. • O administrador precisa estar logado no sistema. • O operador a ser excluído precisa estar cadastrado no sistema. Garantias de sucesso: Um determinado operador tem seus registros excluídos do sistema.
Cenário de sucesso principal: 1. O ator primário abre a interface web na tela de login e efetua sua autenticação no sistema. 2. O ator primário seleciona a opção “Listar operadores”. 3. O ator primário seleciona na lista um ou mais operadores que deseja excluir. 4. O ator primário clica no ícone de exclusão. 5. Os operadores selecionados são excluídos do sistema.
Frequência de ocorrência: Baixa

Fonte: Elaborado pelo autor.

4.2.4 *Web server*

O servidor *web* (ou *web server*) do sistema *Ubspaces* é o componente responsável por receber as requisições dos clientes e, com base nelas, fazer as devidas manipulações no banco de dados. Neste trabalho, optou-se por executar o banco de dados na mesma máquina em que se executa o *web service*, por questões de facilidade na interação entre essas duas partes. Portanto, pode-se dizer que o *web server* e o banco de dados constituem uma só parte conjunta na estrutura do sistema.

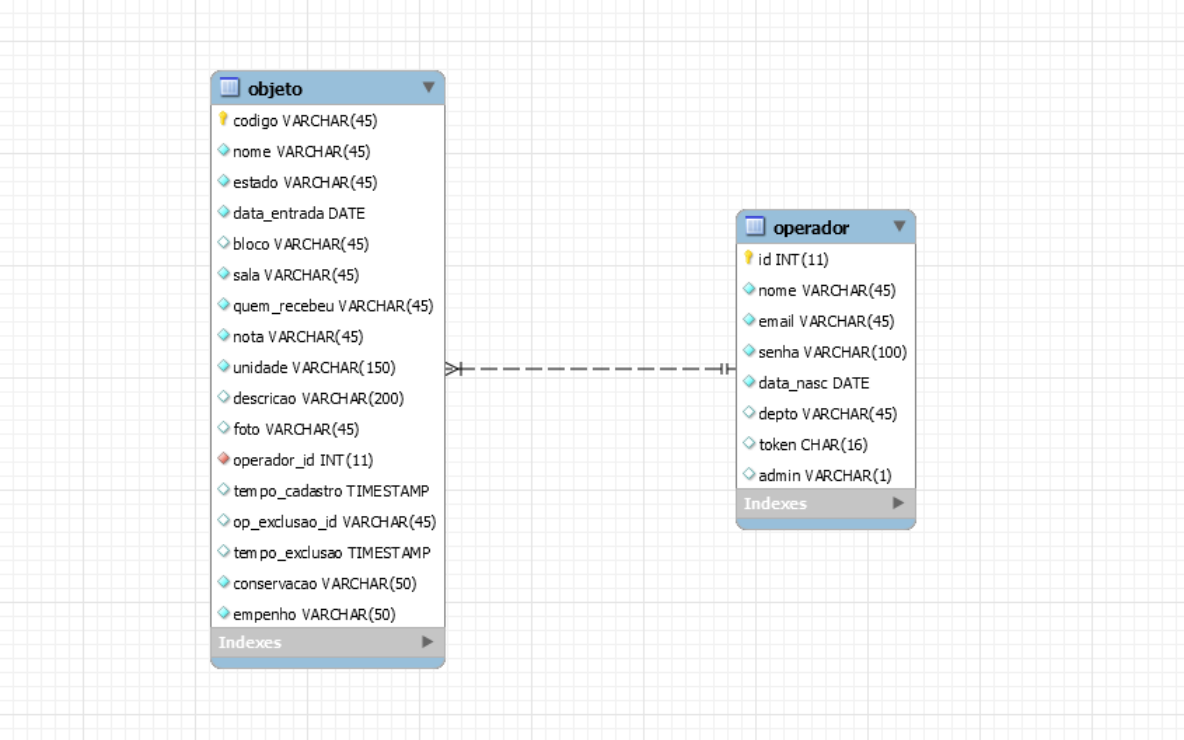
4.2.4.1 Projeto do banco de dados

De acordo com as necessidades e requisitos do sistema, elaborou-se um modelo de banco de dados relacional SQL, utilizando-se a ferramenta *MySQL Workbench*. A Figura 7 mostra o diagrama Entidade-Relacionamento (ER) do banco de dados do sistema *Ubspaces*.

O banco de dados é composto basicamente de duas tabelas. A tabela **objeto** representa cada bem de patrimônio (aqui denominado objeto) e a tabela **operador** representa cada operador e o administrador do sistema. A seguir são descritos todos os campos da tabela **objeto**.

- **codigo**: Número que identifica unicamente um objeto. É a chave primária da tabela.
- **nome**: Nome do objeto ou descrição do bem de patrimônio.
- **estado**: Situação em que o objeto se encontra na entidade pública (ex.: alocado, emprestado, etc.).
- **data_entrada**: Data em que o objeto foi adquirido ou cadastrado no sistema.
- **bloco**: Bloco físico ou edifício onde o objeto se localiza.
- **sala**: Sala onde o objeto se localiza.
- **quem_recebeu**: Nome do responsável pelo recebimento do objeto.
- **nota**: Número ou código da nota fiscal do objeto.
- **unidade**: Nome da unidade institucional a qual o objeto pertence (ex.: ICEA, ICHS, ICSA, etc.).
- **descricao**: Especificação mais detalhada do objeto.
- **foto**: Referência ao arquivo de imagem do objeto. A imagem é armazenada no sistema de arquivos do servidor, enquanto o banco de dados guarda somente o nome deste arquivo.

Figura 7 – Diagrama ER do banco de dados



Fonte: Elaborado pelo autor

- **operador_id**: Identificador do operador que cadastrou o objeto. É uma chave estrangeira para a tabela **operador**.
- **tempo_cadastro**: Data e hora em que o objeto foi cadastrado no sistema.
- **op_exclusao_id**: Identificador do operador que realizou a exclusão do objeto no sistema.
- **tempo_exclusao**: Data e hora em que o objeto foi excluído no sistema.
- **conservacao**: Estado de conservação física em que o objeto se encontra (ex.: bom, precário, sucateado, etc.).
- **empenho**: Número ou código de empenho do objeto.

A seguir são descritos todos os campos da tabela **operador**.

- **id**: Número que identifica unicamente um operador. É a chave primária da tabela.
- **nome**: Nome do operador.
- **email**: Endereço de *e-mail* do operador.
- **senha**: Senha criptografada de acesso do operador ao sistema *Ubspaces*.

- **data_nasc**: Data de nascimento do operador.
- **depto**: Departamento da instituição cujo operador faz parte.
- **token**: Sequência de caracteres que identifica a sessão ativa do operador, indicando que ele está logado no sistema. Quando não está logado, este campo assume o valor NULL.
- **admin**: Este campo indica se um operador é o administrador do sistema. Em caso positivo, este campo assume o valor 1 e 0 caso contrário.

4.2.4.2 Implementação

Ao ser realizado o projeto do banco de dados, iniciou-se a implementação do *web service* na linguagem PHP utilizando o *Slim framework*. Primeiramente, foi necessário realizar a instalação deste *framework* na máquina de testes.

Com a instalação deste *framework*, foi possível definir uma série de recursos no arquivo **index.php**, gerado pelo próprio *Slim*. Estes recursos nada mais são do que métodos acessados via *Uniform Resource Locator* (URL) pelos clientes que realizam as devidas manipulações no banco de dados de acordo com as requisições recebidas. A seguir são listados os principais recursos desenvolvidos e que são disponibilizados pelo *web service*:

- **addObject**: Recebe como parâmetro um novo objeto via JSON e o adiciona ao banco.
- **getObject**: Recebe como parâmetro o código de um determinado objeto e retorna todos os campos do banco de dados referentes a ele.
- **delete**: Recebe como parâmetro o código de um determinado objeto e atualiza seu estado para "Excluído".
- **edit**: Recebe novos dados de um objeto já existente e atualiza suas informações no banco de dados.
- **listall**: Retorna um *array* de objetos contendo somente algumas informações para serem exibidos na tela de listagem de objetos.
- **filter**: Recebe um conjunto de parâmetros de filtragem definidos pelo usuário e retorna um *array* contendo somente objetos condizentes com estes parâmetros.
- **uploadImg**: Recebe uma imagem serializada em formato JSON e a salva na pasta de imagens do servidor.
- **addOperator**: Recebe como parâmetro dados referentes a um novo operador e os adiciona ao banco.

- **editOperator**: Recebe como parâmetro novos dados de um operador já existente e atualiza suas informações no banco.
- **deleteOperator**: Recebe como parâmetro o identificador de um operador e o exclui do banco.
- **getOperator**: Recebe como parâmetro o identificador de um operador e retorna todos os campos referentes a ele.
- **validateLogin**: Recebe como parâmetro o *e-mail* e senha de um operador que quer fazer *login* no sistema. Se estes dados existirem no banco, é gerado e armazenado um *token* de acesso para este operador, e são retornadas as informações dele.

A máquina escolhida para assumir o papel do *web server* durante todo o desenvolvimento deste trabalho foi um *laptop ASUS A45A* que possui as seguintes configurações:

- Processador Intel Core i5 3210M 2.50GHz
- Memória RAM de 6GB
- 500GB de HD
- Sistema operacional Windows 8.1

4.2.5 Aplicativo móvel

Após o projeto e implementação inicial do *web server*, iniciou-se o projeto do aplicativo móvel. Primeiramente, foi necessário fazer uma análise das funcionalidades requeridas no sistema para então idealizar modelos de *layout* visual para interação com o usuário. Depois da análise, passou-se para o processo de prototipagem do aplicativo.

4.2.5.1 Prototipagem

A prototipagem é o processo no qual os modelos de *layout* são transferidos para o papel, de modo que se obtenha um melhor planejamento da interface e uma melhor visualização de como ficarão as telas.

Neste trabalho foram desenvolvidos protótipos de baixa e alta fidelidade. Protótipos de baixa fidelidade consistem em esboços ou rascunhos que são compreensíveis o suficiente para representar as ideias planejadas. Nesse tipo de protótipo não se faz necessário uma representação tão fiel à realidade. Já os protótipos de alta fidelidade consistem em artefatos visuais que se aproximam o máximo possível de modelos visuais reais de um *software*. Este tipo de protótipo retrata em minúcias cada elemento visual presente em um *layout* de *software*.

Os protótipos de baixa fidelidade foram feitos em manuscrito. As Figuras 8 e 9 mostram os protótipos de baixa fidelidade desenvolvidos para o aplicativo móvel.

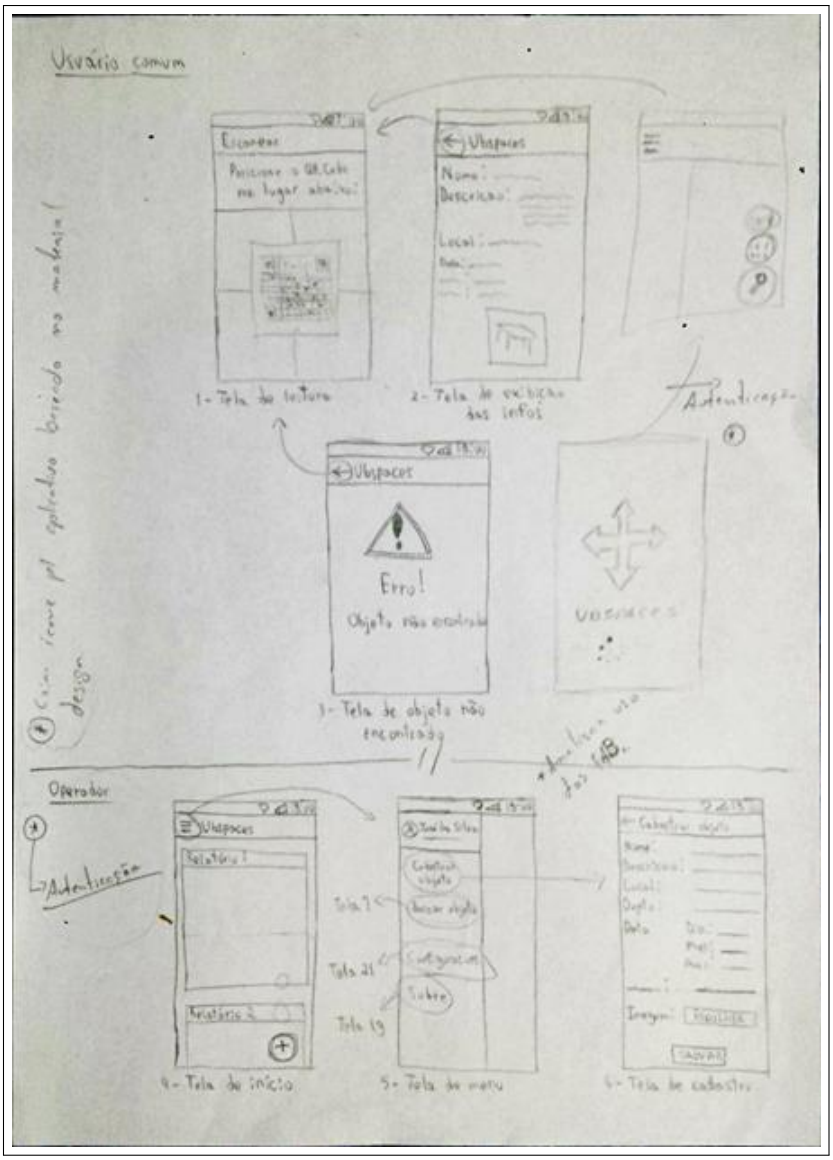
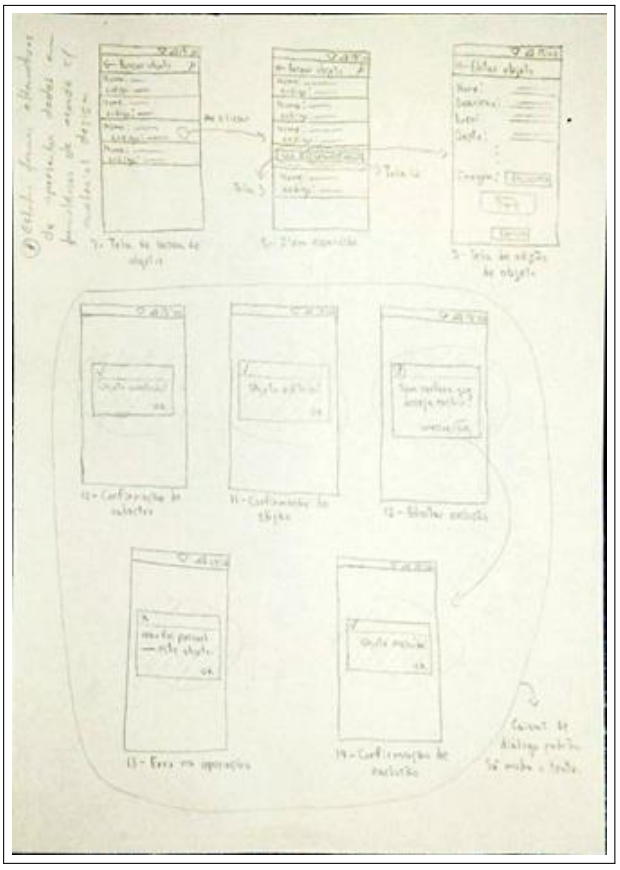
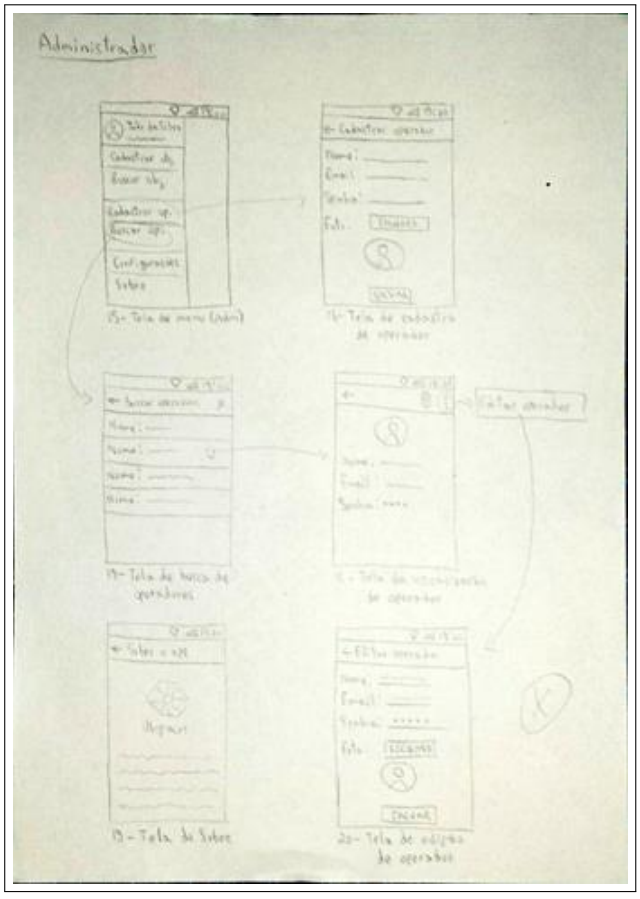


Figura 8 – Protótipos de baixa fidelidade do aplicativo móvel



(a)

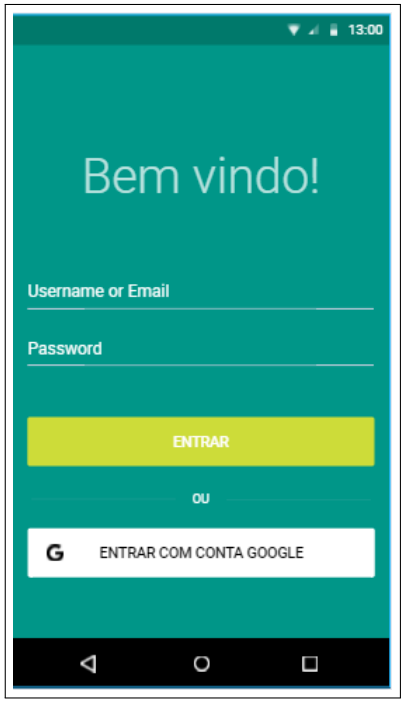


(b)

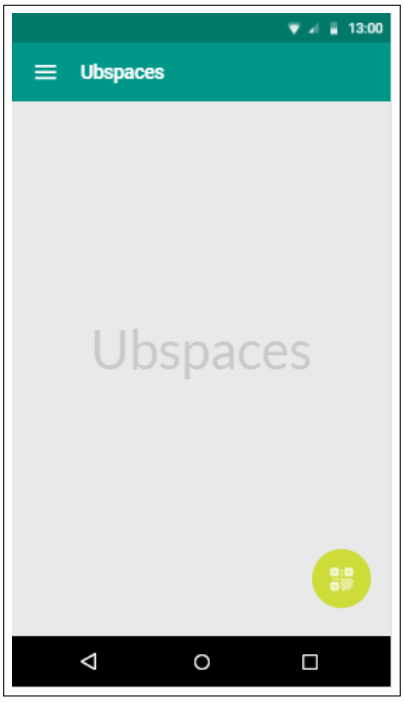
Figura 9 – Protótipos de baixa fidelidade do aplicativo móvel (continuação)

Fonte: Elaborado pelo autor

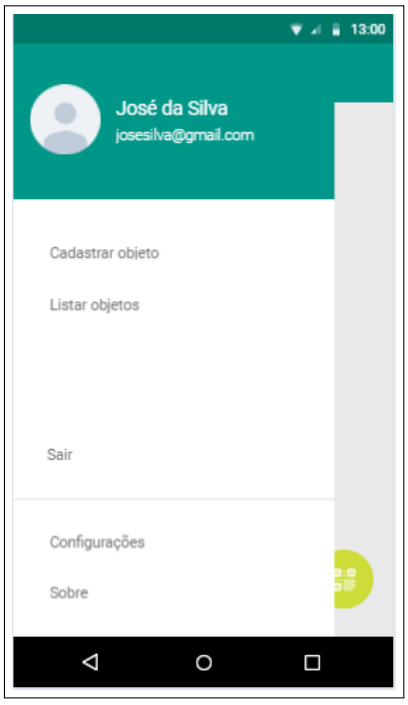
Os protótipos de alta fidelidade foram desenvolvidos utilizando-se a ferramenta *Marvel*, especializada no desenvolvimento de *design* de *softwares*. Ela está disponível na *Internet* de forma gratuita e *online*. As Figuras 10, 11 e 12 mostram os protótipos de alta fidelidade desenvolvidos neste trabalho.



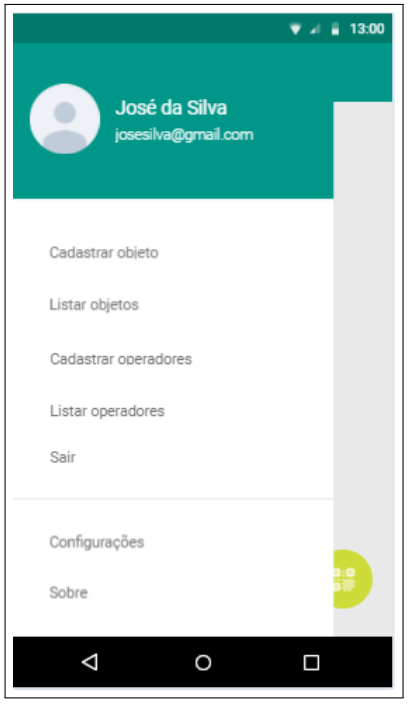
(a) Tela de *login*



(b) Tela inicial limpa



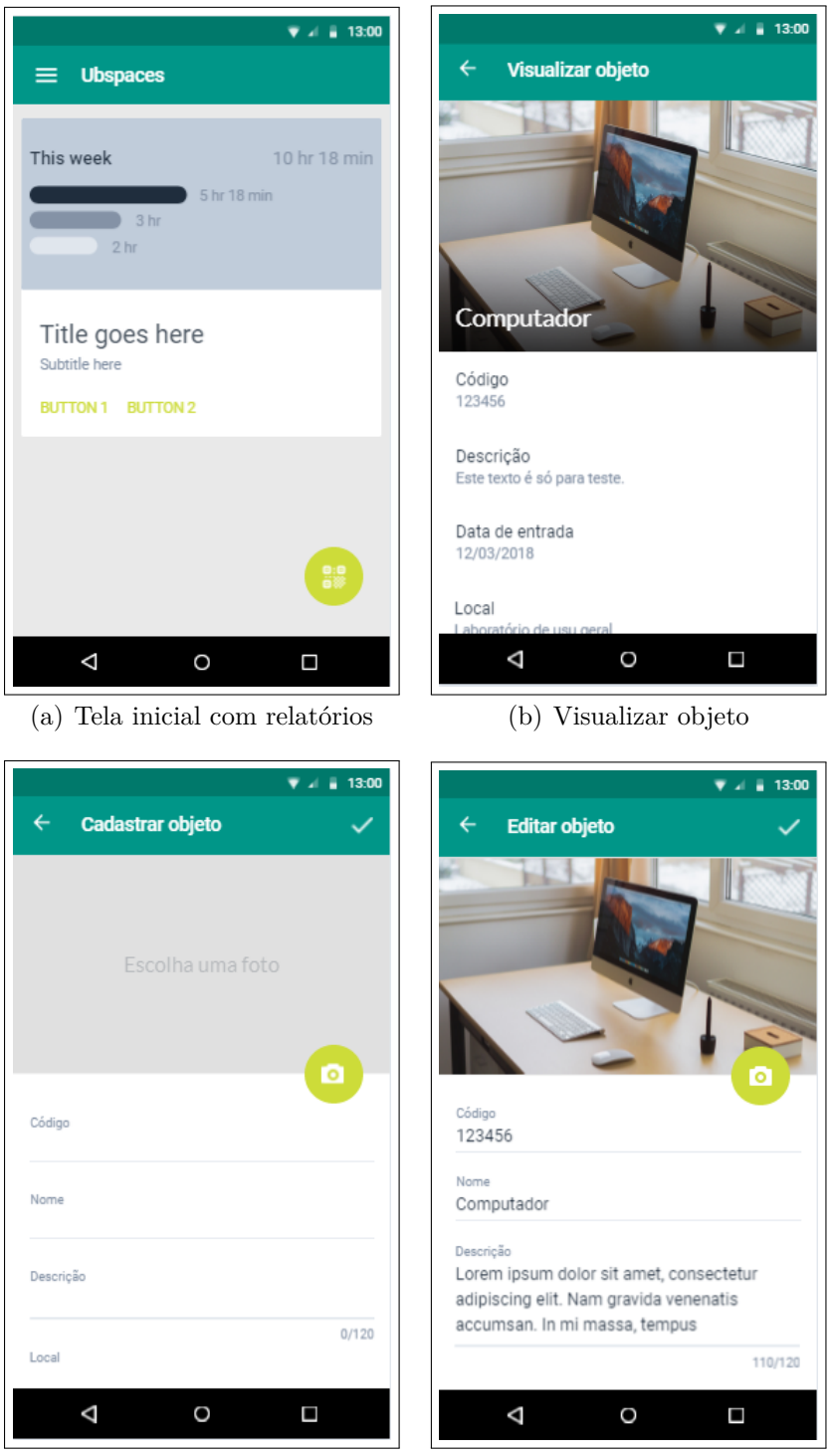
(c) Menu do operador



(d) Menu do administrador

Figura 10 – Protótipos de alta fidelidade do aplicativo móvel (parte 1)

Fonte: Elaborado pelo autor



(a) Tela inicial com relatórios

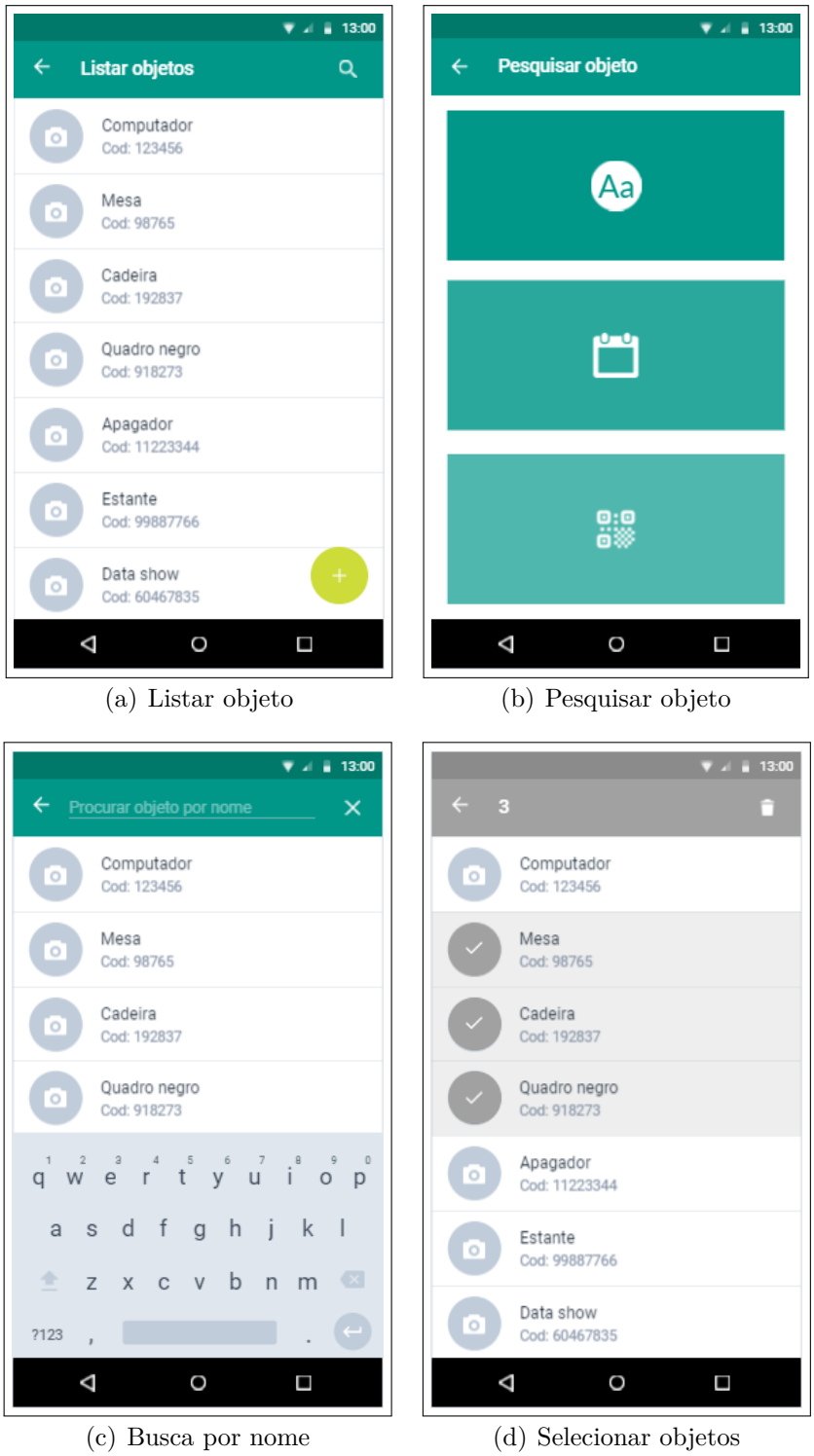
(b) Visualizar objeto

(c) Cadastrar objeto

(d) Editar objeto

Figura 11 – Protótipos de alta fidelidade do aplicativo móvel (parte 2)

Fonte: Elaborado pelo autor



(a) Listar objeto (b) Pesquisar objeto

(c) Busca por nome (d) Selecionar objetos

Figura 12 – Protótipos de alta fidelidade do aplicativo móvel (parte 3)

Fonte: Elaborado pelo autor

4.2.5.2 Implementação

Após a realização da prototipagem do aplicativo, iniciou-se o processo de implementação do mesmo. Como ele já estava sendo desenvolvido no projeto de IC, algumas

funcionalidades desenvolvidas no projeto foram adaptadas para o novo sistema. Como exemplo tem-se a leitura de códigos através da câmera, requisições para comunicação com o servidor, tela para visualização de informações de objetos, entre outras. A partir daí procurou-se implementar cada tela desenvolvida na prototipagem de alta fidelidade, utilizando-se a princípio os recursos nativos do *Android Studio*. Foram utilizadas algumas bibliotecas de terceiros, como o *Zebra Crossing* e o *Retrofit*, já abordados na seção 4.1.

Para cada nova funcionalidade implementada foram realizados testes de aceitação juntamente com o *web server* para validar o funcionamento do aplicativo no contexto do sistema. Esses testes consistiram no cadastro e manipulação de dados fictícios, e também alguns reais, a respeito dos bens de patrimônio da UFOP.

Ao longo do desenvolvimento do aplicativo, conforme surgiram as necessidades, as funcionalidades e *layouts* definidos na prototipagem de alta fidelidade sofreram leves alterações, como pode-se observar no Capítulo 5. O dispositivo móvel principal usado durante todo o processo de desenvolvimento foi um *smartphone Samsung Galaxy S4 I9505*.

4.2.6 Interface *web*

Após a implementação do aplicativo móvel, iniciou-se o processo de implementação da interface *web* do sistema *Ubspaces*.

Como no aplicativo móvel foi utilizado o conceito de *Material Design* para a definição de padrões de *design* e usabilidade, então procurou-se um *template front-end* de páginas *web* que seguisse este mesmo conceito. Decidiu-se então por utilizar um *framework* oficial do *Google* (empresa que desenvolveu o conceito): o *Material Design Lite* (MDL). Este é melhor abordado na seção de tecnologias e ferramentas utilizadas (seção 4.1).

A partir daí procurou-se implementar as mesmas telas presentes no aplicativo móvel, seguindo inclusive, os mesmos padrões de *layout*, a fim de desenvolver uma interface de uso unificada para ambas as partes. Para isso, foi necessário primeiramente estudar mais a fundo o MDL e sua coleção de componentes *web*. Além disso, foi necessário também consultar outras fontes de informação, como o *Stack Overflow*, tutoriais no *YouTube*, e também um curso *online* de *JavaScript*.

Foi também utilizado o *jQuery* para facilitar a implementação de algumas funcionalidades mais complexas. No caso deste trabalho, ele foi utilizado para o desenvolvimento das requisições HTTP por meio de métodos *ajax*.

De modo semelhante ao processo de desenvolvimento do aplicativo móvel, à medida que as funcionalidades da interface *web* foram implementadas, testes de aceitação e usabilidade foram realizados a fim de aprimorar o funcionamento do sistema. Durante todo o período de desenvolvimento da interface *web*, a máquina principal utilizada para executá-la e testá-la foi a mesma utilizada como *web server* neste trabalho. Porém, a

interface *web* foi também testada em outras máquinas e dispositivos móveis, a fim de averiguar a responsividade das páginas *web*.

5 Resultados

Neste capítulo são apresentados os resultados obtidos da implementação do sistema *Ubspaces*. Na primeira seção é apresentado o aplicativo móvel e suas funcionalidades. Na segunda seção é apresentada a interface *web* e suas funcionalidades.

Nas seções seguintes do texto são utilizados alguns termos estrangeiros específicos do contexto de interface de aplicações móveis. Tais termos e seus significados constam a seguir:

- Menu *drawer*: É um menu de opções que, a princípio, não fica visível ao usuário. Ele se torna visível quando um botão (geralmente localizado no canto superior esquerdo) é acessado, ou então, quando o usuário o arrasta do canto esquerdo da tela para o canto direito.
- *Floating Action Button* (FAB): É um elemento presente no conceito de *Material Design*. O FAB é um botão circular que se sobrepõe aos outros elementos da tela e denota alguma funcionalidade de alta importância ou a mais usada.
- *Snackbar*: É um elemento visual que mostra breves mensagens na parte inferior da tela, informando ao usuário a resposta do sistema com base em alguma ação realizada.
- *Splash Screen*: É uma tela mostrada logo no início da execução de uma aplicação, enquanto o sistema carrega suas configurações iniciais. Nesta tela é mostrada a identidade visual da aplicação.
- *Swipe*: É uma forma de interação do usuário com um dispositivo móvel que possui uma tela sensível ao toque. Esta forma consiste em deslizar o dedo na tela ao longo de uma direção de forma a completar alguma ação do sistema.

5.1 Aplicativo móvel

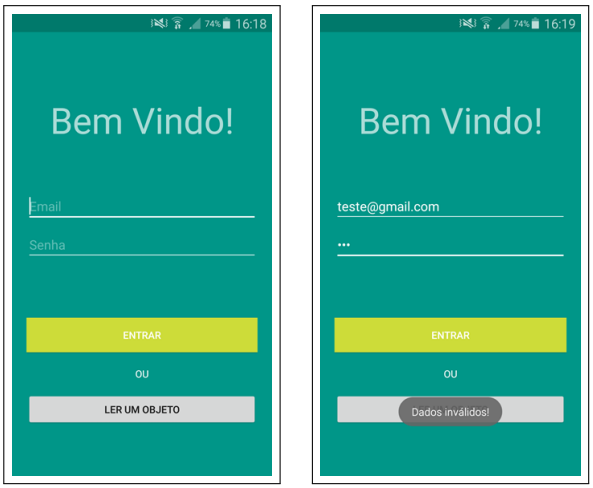
Nesta seção é apresentado o funcionamento do aplicativo móvel desenvolvido, abordando cada funcionalidade juntamente com as respectivas capturas de tela.

5.1.1 Tela de *login*

Ao se executar o aplicativo, uma *Splash Screen* é mostrada. Esta pode ser vista na Figura 13. Ela permanece na tela por uma pequena porção de tempo, aproximadamente 3 segundos.

Figura 13 – Aplicativo - *Splash Screen*

Fonte: Elaborado pelo autor



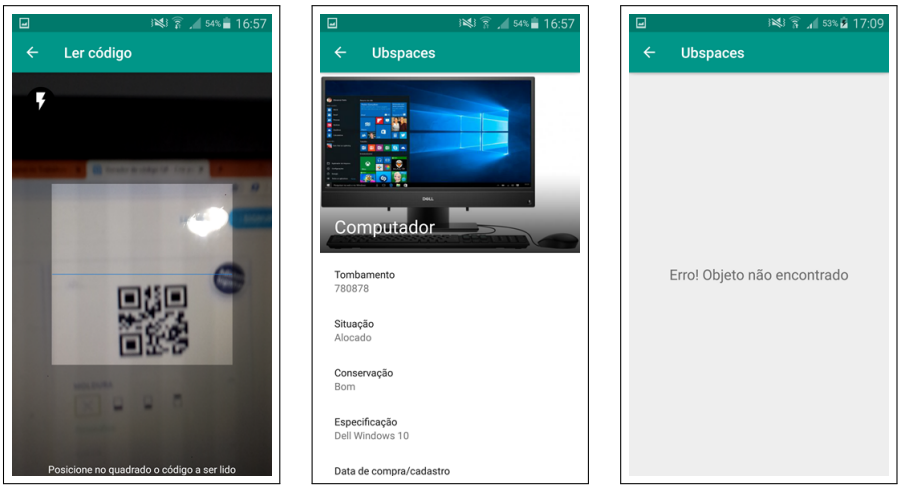
(a) Antes de informar os dados de acesso

(b) Dados de acesso incorretos

Figura 14 – Aplicativo - Tela de *login*

Fonte: Elaborado pelo autor

Logo após é mostrada a tela de *login* (Figura 14). Este é o ponto de separação entre o usuário comum e os operadores, pois somente os operadores e administrador possuem cadastro no sistema *Ubspaces*. Os usuários comuns podem somente consultar informações sobre um determinado objeto. Para isso, basta acessar a opção “Ler um objeto”.



(a) Tela de leitura do código (b) Visualização de objeto para usuário comum (c) Tela indicativa de que o objeto não está cadastrado

Figura 15 – Aplicativo - Leitura de código para usuário comum

Fonte: Elaborado pelo autor

Quando esta opção é acessada é exibida a tela de leitura de código (Figura 15(a)). Assim que o código é lido, é mostrada a tela de visualização de objetos para usuário comum, se o objeto está cadastrado no sistema (Figura 15(b)). Caso contrário, é mostrada uma mensagem indicando que o objeto não foi encontrado (Figura 15(c)).

Voltando a atenção aos operadores, eles precisam fornecer seus dados de acesso ao sistema, que no caso são o *e-mail* e a senha. Ao clicar na opção "Entrar", uma requisição é enviada ao servidor com os dados informados. Se o operador não está cadastrado no sistema, então é exibida uma mensagem indicando que os dados estão incorretos (Figura 14(b)). Caso contrário, é exibida a tela inicial do sistema (Figura 16), que é melhor detalhada na subseção 5.1.2.

5.1.2 Tela inicial

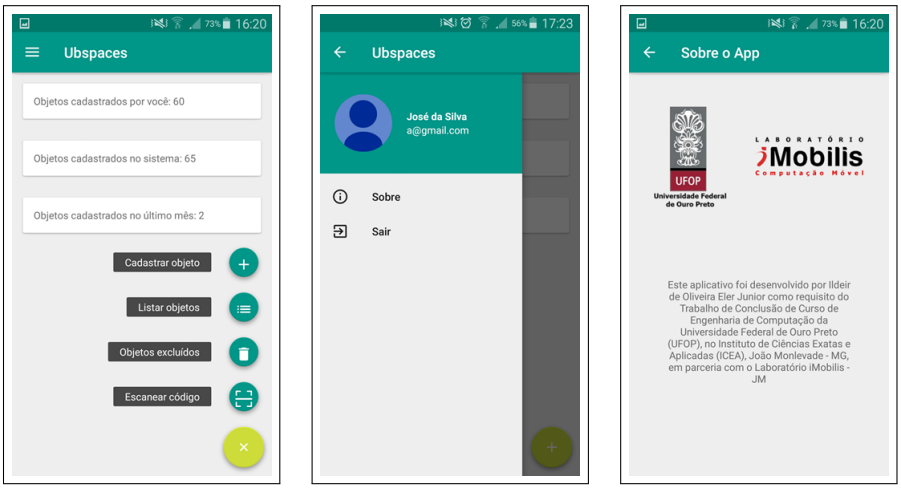
Figura 16 – Aplicativo - Tela inicial

Fonte: Elaborado pelo autor

A tela inicial do aplicativo mostrada depois do *login* contém alguns metadados buscados no servidor, que assumem o papel de pequenos relatórios de uso do sistema. Ela mostra a quantidade total de objetos cadastrados no sistema, a quantidade de objetos cadastrados pelo usuário logado e a quantidade de objetos cadastrados no mês corrente.

Ao se clicar no botão localizado no canto superior esquerdo da tela, um menu *drawer* (Figura 17(b)) aparece contendo o nome e *e-mail* do usuário logado e algumas opções: exibir a tela "Sobre" (Figura 17(c)) e fazer *logout* no sistema, ou seja, encerrar a sessão do usuário.

Ao se clicar no FAB localizado no canto inferior direito da tela, são exibidas as principais funcionalidades do aplicativo: "Cadastrar objeto", "Listar objetos", "Objetos excluídos" e "Escanear código" (Figura 17(a)).



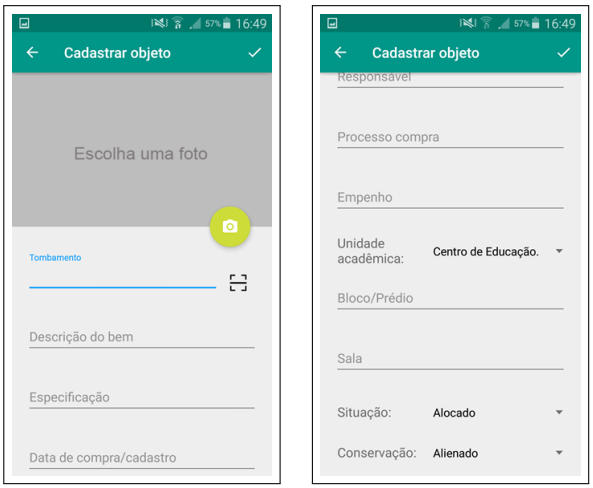
(a) Menu de funcionalidades principais (b) Menu *drawer* (c) Tela "Sobre" do aplicativo

Figura 17 – Aplicativo - Funcionalidades da tela inicial

Fonte: Elaborado pelo autor

5.1.3 Cadastro de objetos

Ao acessar a opção "Cadastrar objeto", é mostrada uma tela (Figura 18) contendo todos os campos relacionados a um bem de patrimônio da UFOP, para então serem preenchidos pelo usuário.



(a) Início da tela (b) Final da tela

Figura 18 – Aplicativo - Tela de cadastro de objeto

Fonte: Elaborado pelo autor

Todos os campos, com exceção de **Especificação** e **Bloco/Prédio**, são obrigatórios. Portanto, se o usuário deixar de preencher algum dos que são obrigatórios, é mostrada uma marcação neste campo, conforme pode ser visto na Figura 19.

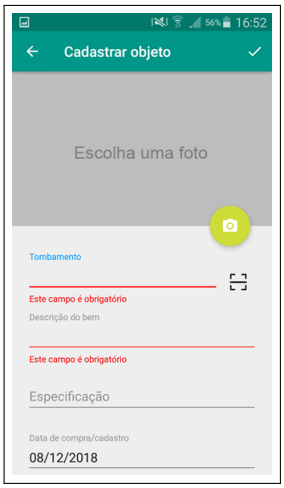
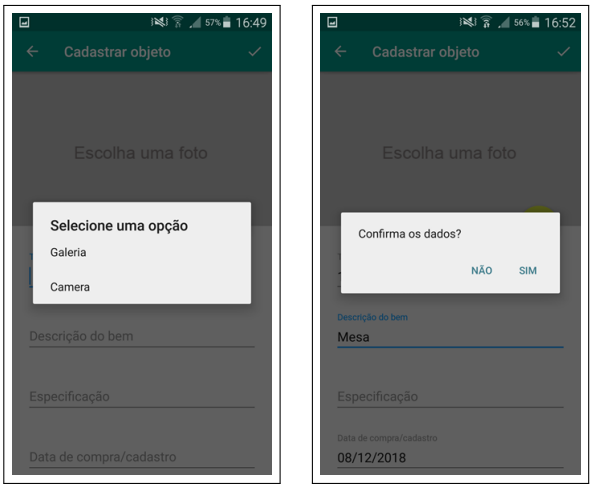


Figura 19 – Aplicativo - Validação de campos obrigatórios do cadastro

Fonte: Elaborado pelo autor

É possível armazenar também uma fotografia do objeto, clicando no botão de câmera. Então é mostrada uma janela com duas opções: "Galeria" para selecionar uma imagem já armazenada, e "Camera" para fazer uma nova fotografia e já selecioná-la (Figura 20(a)).

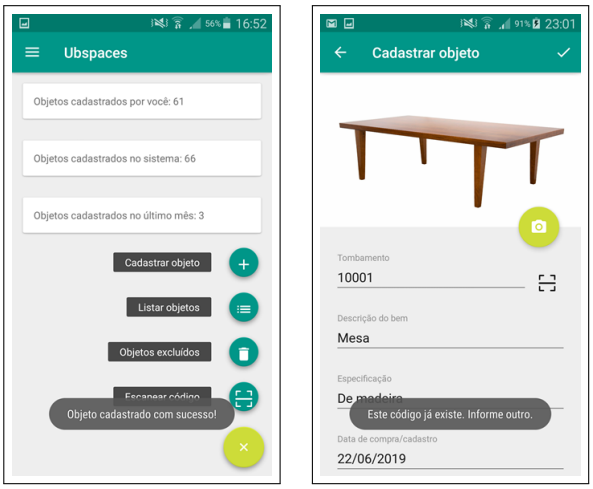


(a) Janela com opções de imagem (b) Janela de confirmação

Figura 20 – Aplicativo - Janelas da tela de cadastro

Fonte: Elaborado pelo autor

Quando o usuário já inseriu todas as informações corretamente e clica no botão "*Check*" localizado no canto superior direito da tela, então é mostrada uma janela de confirmação dos dados (Figura 20(b)). Ao confirmar, se os dados foram salvos corretamente no servidor, então é mostrada uma mensagem de confirmação (Figura 21(a)) e o aplicativo volta para a tela inicial. Caso contrário, aparece uma mensagem indicando o problema e o aplicativo permanece na mesma tela (Figura 21(b)).



(a) Mensagem de sucesso (b) Mensagem de erro

Figura 21 – Aplicativo - Mensagens de resposta do cadastro de objeto

Fonte: Elaborado pelo autor

5.1.4 Listagem de objetos

Ao acessar a opção “Listar objetos”, é mostrada uma lista contendo todos os objetos cadastrados com suas respectivas informações: miniatura de imagem, descrição do bem, tombamento e data de compra/cadastro (Figura 22). Os objetos são obtidos dinamicamente do servidor a medida que a lista é rolada para baixo.

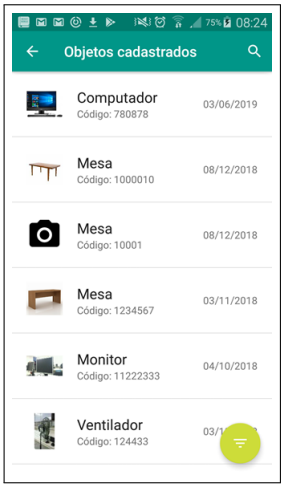


Figura 22 – Aplicativo - Tela de listagem de objetos

Fonte: Elaborado pelo autor

Ao se clicar em um item dessa lista, é exibida a tela de visualização dos dados do objeto acessado, contendo todas as informações cadastradas referentes a ele.

No canto superior direito da tela há um ícone de lupa. Ao acessá-lo, é exibido um campo de texto numérico para que o usuário possa buscar um objeto diretamente pelo seu

tombamento (Figura 23). O usuário então informa o código desejado e clica no botão de pesquisa (nesse caso, o ícone de lupa do teclado numérico). Caso o objeto que contém o tombamento informado esteja cadastrado no sistema, é mostrada a tela de visualização do objeto contendo todas as suas informações. Caso contrário, é exibida a tela indicando que o objeto não foi encontrado (Figura 15(c)).

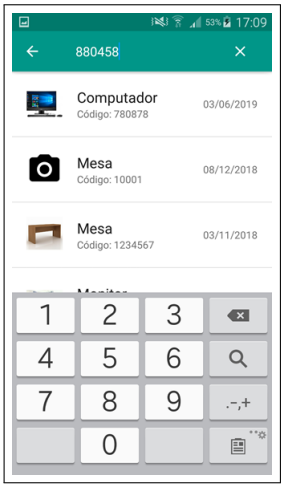
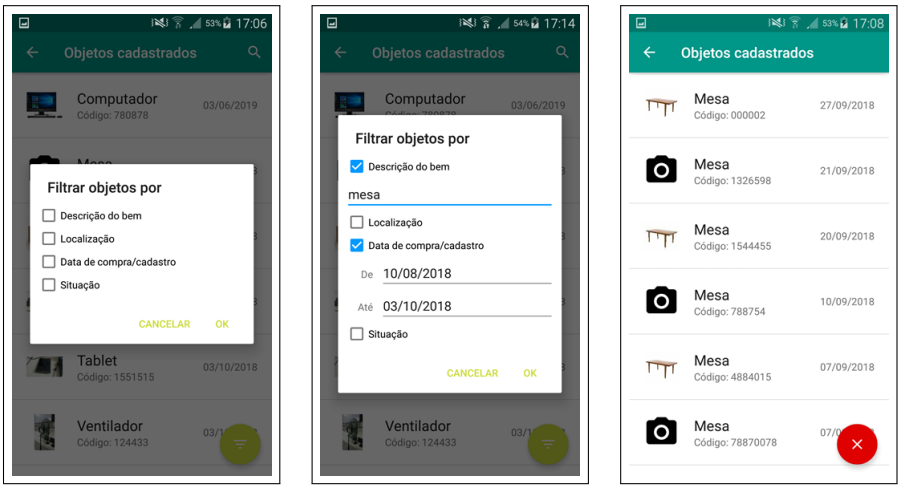


Figura 23 – Aplicativo - Pesquisar objeto por tombamento

Fonte: Elaborado pelo autor

Voltando a tela de listagem, no canto inferior direito há um FAB com um ícone de filtro. Ao acessá-lo, é exibida uma janela de filtragem de objetos (Figura 24(a)). Esta funcionalidade permite que o usuário filtre objetos baseado em quatro parâmetros: descrição do bem, localização, data de compra/cadastro e conservação. Na janela de filtragem estão presentes estas quatro opções, podendo ser ativadas independentemente umas das outras conforme a vontade do usuário.

Assim que o usuário informa os parâmetros da filtragem (Figura 24(b)) e clica em "OK", é exibida uma nova lista de objetos, porém contendo somente aqueles que condizem com os parâmetros informados pelo usuário no filtro (Figura 24(c)). Para sair da lista de objetos filtrados, basta clicar no FAB vermelho com um ícone "X", localizado no canto inferior direito da tela.



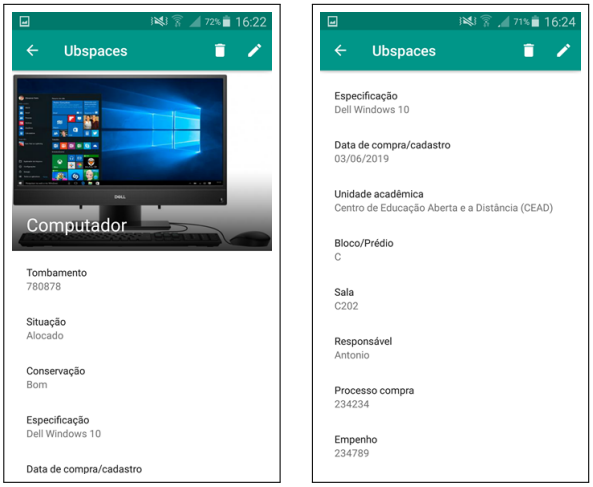
(a) Antes da seleção de parâmetros (b) Durante a seleção de parâmetros (c) Lista de objetos filtrados

Figura 24 – Aplicativo - Filtragem de objetos

Fonte: Elaborado pelo autor

5.1.5 Visualização dos dados dos objetos

A tela de visualização, mostrada na Figura 25, exibe as informações cadastradas de um determinado objeto, incluindo sua imagem. No canto superior direito da tela há dois ícones: um de lixeira e outro de lápis. Ao clicar no ícone de lixeira, o objeto em questão é movido para a seção de objetos excluídos. Ao clicar no ícone de lápis, é exibida a tela de edição deste objeto.



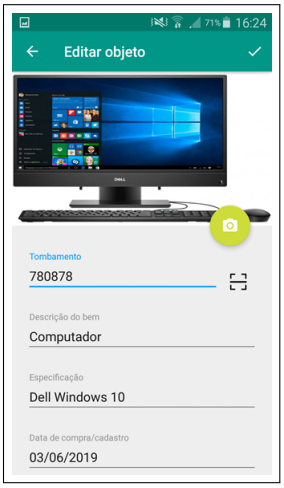
(a) Início da tela (b) Final da tela

Figura 25 – Aplicativo - Visualização dos dados de um objeto

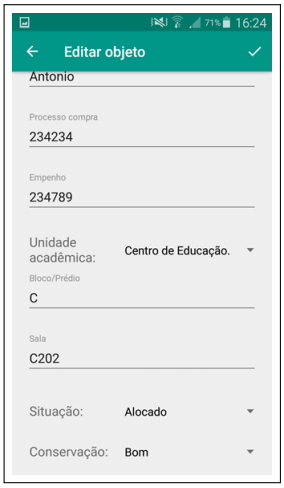
Fonte: Elaborado pelo autor

5.1.6 Edição de objetos

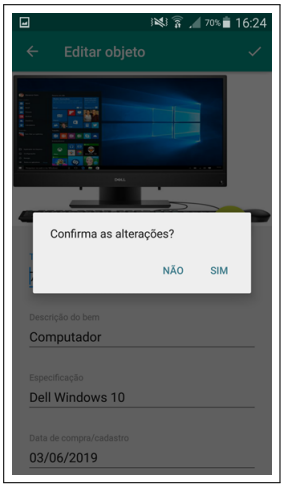
A tela de edição de objetos é semelhante à tela de cadastro, com a diferença de que nessa os campos de texto já vêm preenchidos com os dados referentes ao objeto que se deseja editar (Figura 26). Os campos embora preenchidos, ainda continuam editáveis para o usuário alterar os dados do objeto conforme sua vontade. O funcionamento da tela é igual ao da tela de cadastro, porém as mensagens de interface com o usuário e o modo como os dados são processados no aplicativo e no *web server* diferem um pouco, por se tratar de uma edição.



(a) Início da tela

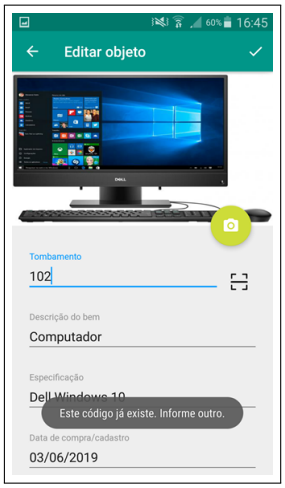


(b) Final da tela



(c) Janela de confirmação

(d) Mensagem de sucesso



(e) Mensagem de erro

Figura 26 – Aplicativo - Tela de edição de um objeto

Fonte: Elaborado pelo autor

5.1.7 Exclusão de objetos

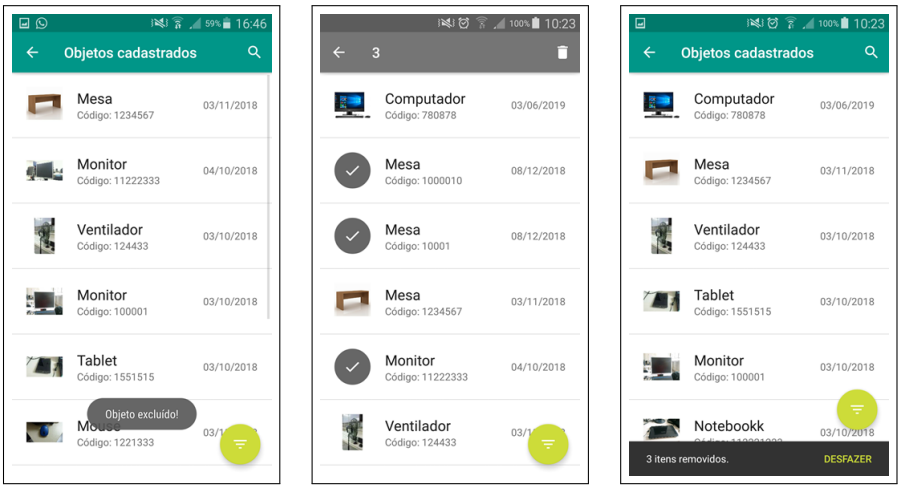
A exclusão de um objeto pode ser feita de três maneiras: clicando no ícone de lixeira na tela de visualização, selecionando múltiplos itens na lista de objetos ou arrastando para

a esquerda (*swipe*) um item da lista.

No primeiro modo, estando na tela de visualização, o usuário clica no ícone de lixeira. O objeto é movido para a seção de excluídos e o aplicativo volta para a tela de listagem após exibir uma mensagem de confirmação (Figura 27(a)).

No segundo modo, estando na tela de listagem de objetos, o usuário pode selecionar um ou mais itens da lista. Ele pode fazer isso clicando e segurando um item para entrar no modo de seleção, ou então clicando sobre as miniaturas dos objetos. Assim que o usuário seleciona quantos objetos deseja excluir, um ícone de lixeira aparece no canto superior direito da tela (Figura 27(b)). Ao clicar nesse ícone, os objetos são movidos para a seção dos excluídos e são removidos da lista. Além disso, é mostrado um *Snackbar* na parte inferior da tela mostrando uma mensagem de confirmação e um botão "Desfazer" (Figura 27(c)). Ao clicar neste botão, a ação de excluir é desfeita e os objetos voltam para onde estavam na lista.

No terceiro modo, estando na tela de listagem de objetos, o usuário pode arrastar um item da lista para a esquerda (Figura 27(d)). Ao fazer isso, o objeto em questão é removido da lista e vai para a seção de objetos excluídos.



(a) Mensagem de confirmação

(b) Selecionando múltiplos objetos

(c) *Snackbar* de exclusão

(d) Exclusão via *swipe*

Figura 27 – Aplicativo - Exclusão de objetos

Fonte: Elaborado pelo autor

5.1.8 Objetos excluídos

Em um contexto de gerenciamento de bens patrimoniais de um local público, é interessante manter um histórico dos objetos relacionados a este local, até mesmo aqueles que já foram descartados, ou que por algum motivo já não têm mais utilidade. Por esta razão, o *Ubspaces* dispõe de uma seção especial que lista todos os objetos excluídos do sistema.

Para acessá-la, estando na tela inicial do aplicativo, basta expandir o FAB e clicar em "Objetos excluídos". Então é mostrada uma lista semelhante à dos objetos não excluídos da Figura 22. A diferença é que nesta os elementos de texto são escritos em vermelho com o intuito de deixar mais visível essa diferenciação (Figura 28).

Ao clicar em um item da lista, é mostrada uma tela de visualização do objeto,

semelhante à da Figura 25. Porém, nesta não existem os botões de exclusão e edição, uma vez que o objeto em questão já não faz mais parte de maneira ativa da coleção de bens patrimoniais.

Na tela de listagem dos objetos excluídos existem também as funcionalidades de pesquisa por tombamento e de filtragem. Estas são as mesmas funcionalidades presentes na tela da Figura 22 e funcionam da mesma maneira, mas sempre pesquisando somente entre os objetos excluídos. E na janela do filtro, já não há mais o parâmetro "Estado", uma vez que o estado de um objeto deste tipo já é definido como excluído.

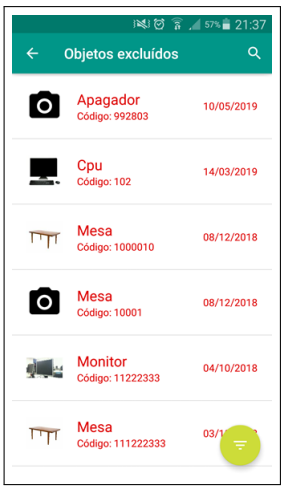


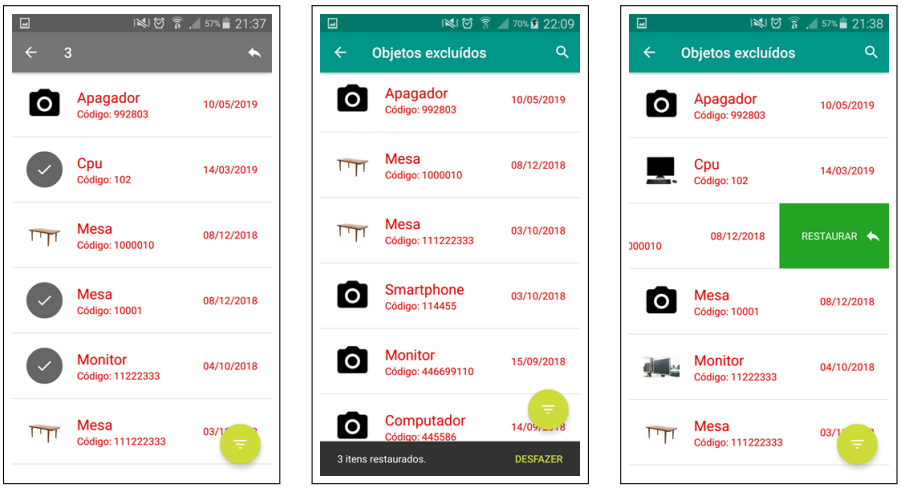
Figura 28 – Aplicativo - Objetos excluídos

Fonte: Elaborado pelo autor

Caso haja a necessidade, o usuário pode restaurar objetos desta seção, levando-os de volta para a seção de objetos não excluídos. No aplicativo móvel, existem duas formas de fazer isso: selecionando múltiplos itens da lista ou arrastando um item para a esquerda.

O primeiro modo ocorre da mesma forma que no processo de exclusão de objetos por seleção. A diferença é que no modo de seleção, ao invés de ser mostrado um ícone de lixeira no canto superior direito da tela, é mostrado um ícone de uma seta direcionada para a esquerda (Figura 29(a)). Ao se clicar neste ícone, os objetos selecionados são restaurados, sendo removidos da lista e movidos para a listagem de objetos não excluídos. Neste momento é mostrado também um *snackbar* com uma mensagem confirmando que os objetos foram restaurados e um botão "Desfazer" (Figura 29(b)). Ao se clicar neste botão, a ação de restaurar os objetos selecionados é desfeita e eles voltam para a lista de objetos excluídos.

O segundo modo também ocorre da mesma maneira que no processo de exclusão via *swipe*. Ao arrastar um item da lista para a esquerda, ele é removido da lista e movido para a seção de objetos não excluídos. A diferença consiste no aspecto visual, pois é mostrada uma cor verde com o ícone de restauração e o escrito "Restaurar" (Figura 29(c)).



(a) Seleção de objetos (restauração) (b) Snackbar (restauração) (c) Swipe (restauração)

Figura 29 – Aplicativo - Exclusão de objetos

Fonte: Elaborado pelo autor

5.1.9 Escanear código

Assim como o usuário comum, os operadores e o administrador também podem acessar a funcionalidade de ler/escanear o código de um determinado objeto. Este código pode ser tanto um QRCode como um código de barras.

Estando na tela inicial, ao expandir o FAB e clicar na opção "Escanear código" é mostrada a tela de leitura de código da Figura 15(a). Essa tela se utiliza da câmera do dispositivo móvel para capturar e interpretar a informação codificada. Além disso, caso este possua uma lanterna, um ícone de raio é mostrado no canto superior esquerdo da tela. Ao clicar nele, a lanterna é ativada/desativada para facilitar a leitura em ambientes escuros.

Para que um código seja devidamente reconhecido, basta posicioná-lo dentro da moldura quadrada centralizando-o de acordo com a linha azul e esperar até que se obtenha uma resposta. Caso o objeto com o código lido esteja cadastrado no sistema, a tela de visualização da Figura 25 é mostrada contendo as informações do objeto. Caso contrário, é exibida a tela da Figura 15(c), que contém a mensagem de que o objeto não foi encontrado.

5.2 Interface *web*

Nesta seção é apresentado o funcionamento da interface *web*, abordando-se cada funcionalidade juntamente com as respectivas capturas de tela.

5.2.1 *Landing page*

As *landing pages*, páginas de aterrissagem em português, são páginas *web* que possuem o objetivo de atrair um internauta ou cliente para o consumo de um determinado conteúdo. Podem ser consideradas como cartões de visita de algum *site* ou serviço *web*. Geralmente são as primeiras páginas a serem exibidas quando um internauta deseja acessar algum conteúdo de seu interesse.

A *landing page* do sistema *Ubspaces* contém um link para a realização de autenticação dos operadores e administrador, uma seção de pesquisa de objetos para usuários comuns e também algumas informações sobre o projeto desenvolvido (Figura 30).

(a) Seção inicial

(b) Seção “Pesquisar”

(c) Seção “Sobre”

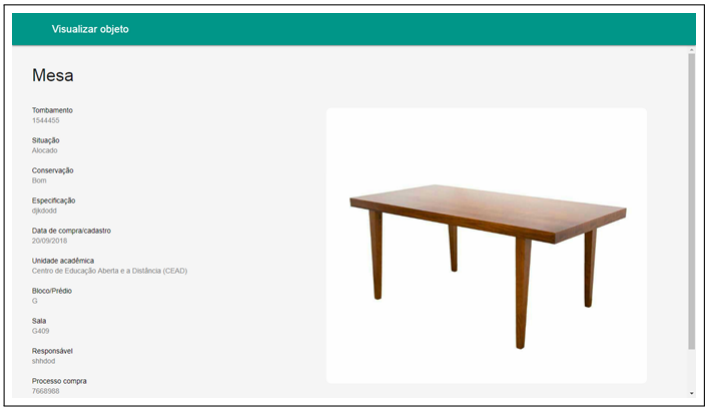
Figura 30 – Interface *web* - *Landind page* do sistema *Ubspaces*

Fonte: Elaborado pelo autor

Para os usuários comuns, há a funcionalidade de pesquisa de objetos baseada nos parâmetros **Descrição do bem** e **Data de compra/cadastro**, sendo este último um período de tempo (Figura 31(a)). Ao inserir os dados de busca e em seguida clicar em "Pesquisar", é mostrada uma lista com todos os objetos retornados do servidor que condizem com os parâmetros informados (Figura 31(b)). Ao clicar em um item da lista, então é exibida uma tela de visualização com as informações do objeto selecionado (Figura 31(c)).

(a) Campos de pesquisa preenchidos

(b) Resultados da pesquisa



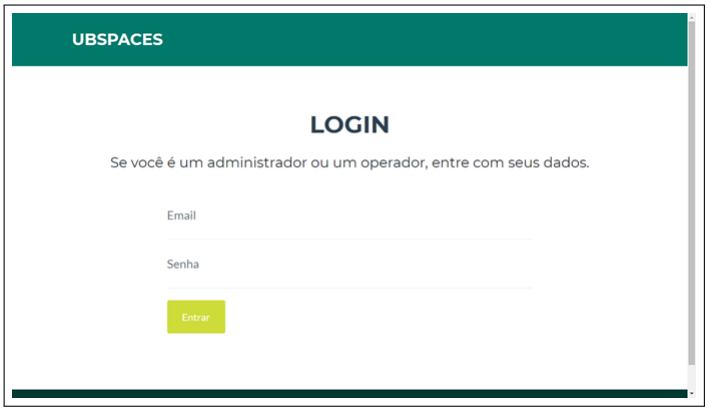
(c) Visualização de objeto para usuário comum

Figura 31 – Interface *web* - Pesquisa de objetos para usuário comum

Fonte: Elaborado pelo autor

5.2.2 Página de *login*

Ao acessar a opção "Login" localizada no canto superior direito da *landing page*, é exibida uma outra página contendo os campos de acesso para operadores e administrador (Figura 32(a)). O usuário informa seu *e-mail* e senha e clica em "Entrar" para realizar o *login* no sistema *Ubspaces*. Se o usuário não estiver cadastrado no banco de dados, uma mensagem de erro é exibida na tela (Figura 32(b)). Caso contrário, é exibida a página inicial da interface *web*.



(a) Campos de acesso

(b) Dados de acesso incorretos

Figura 32 – Interface *web* - Página de *login*

Fonte: Elaborado pelo autor

Em razão da interface *web* ser de caráter administrativo, ela possui funcionalidades de gerenciamento de operadores, algo que não está presente no aplicativo móvel. Portanto, se faz necessário realizar a distinção entre o gerenciamento de objetos de patrimônio e o gerenciamento de operadores. Na subseção 5.2.3 são apresentadas as funcionalidades relacionadas aos bens de patrimônio, que em essência são as mesmas presentes no aplicativo móvel. Na subseção 5.2.4 são apresentadas as funcionalidades referentes aos operadores, que são habilitadas quando o usuário logado é o administrador do sistema.

5.2.3 Objetos

Nesta subseção são apresentadas as funcionalidades presentes na interface *web* referentes aos objetos de patrimônio.

5.2.3.1 Tela inicial

A tela inicial da interface *web* exibe os mesmos metadados buscados do servidor presentes na tela da Figura 16.

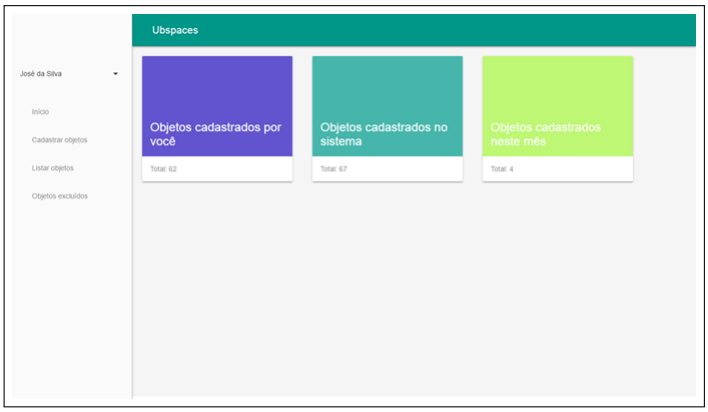
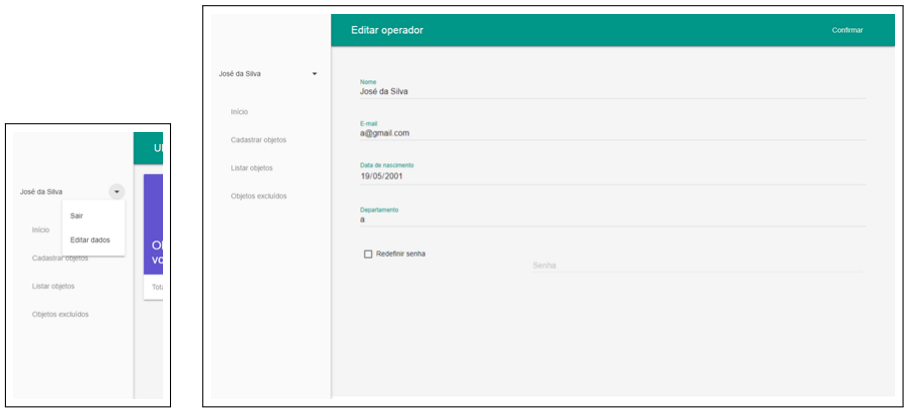


Figura 33 – Interface *web* - Tela inicial

Fonte: Elaborado pelo autor

5.2.3.2 Menu lateral

A interface *web* possui um menu lateral fixo, ou seja, ele é exibido durante todo o tempo de navegação pela interface (Figura 34(a)). Na parte superior desse menu é mostrado o nome do usuário logado e um botão que dá acesso a duas opções: “Sair” e “Editar dados”. Ao clicar na primeira, o usuário é “deslogado” do sistema, voltando para a tela de *login*. Ao clicar na segunda, é exibida uma página de edição dos dados do usuário (Figura 34(b)).



(a) Menu lateral (b) Edição de dados do usuário logado

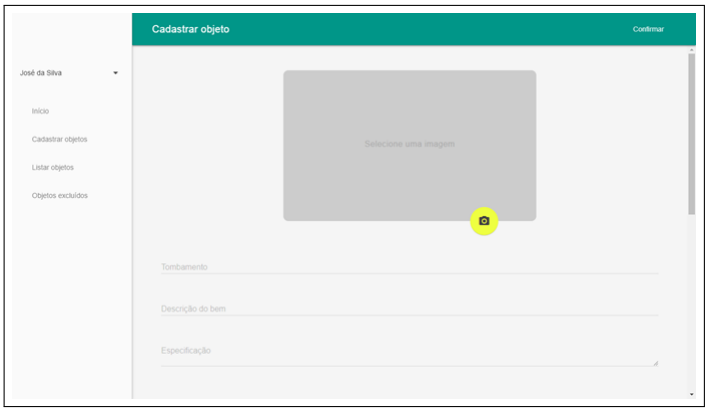
Figura 34 – Interface *web* - Menu lateral e página de edição de dados do operador logado

Fonte: Elaborado pelo autor

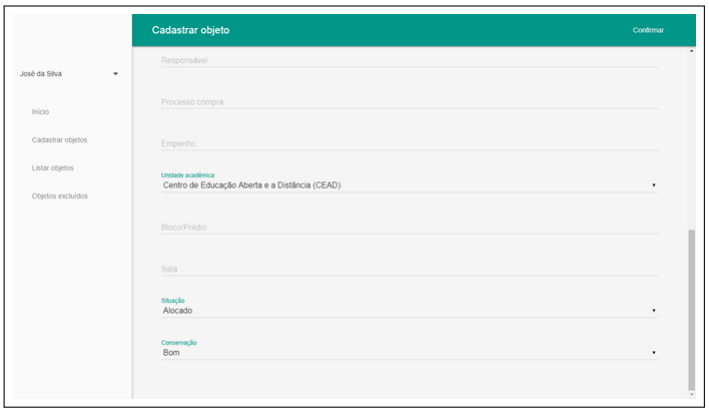
Ainda no menu lateral, abaixo do nome do usuário logado ainda há quatro opções: “Início”, “Cadastrar objetos”, “Listar objetos” e “Objetos excluídos”. Ao clicar em “Início” é exibida a tela inicial da interface, apresentada anteriormente na Figura 33. As outras opções serão melhor detalhadas nos tópicos seguintes.

5.2.3.3 Cadastro

Ao clicar em “Cadastrar objetos” no menu lateral, é mostrada a página de cadastro de objetos (Figura 35). Esta página possui todos os campos de informações relacionadas aos bens de patrimônio, inclusive uma opção de adicionar uma imagem. Ao clicar no ícone de câmera, é aberta uma janela do sistema operacional para a seleção de uma imagem.



(a) Início da página



(b) Final da página

Figura 35 – Interface *web* - Página de cadastro

Fonte: Elaborado pelo autor

Da mesma forma que ocorre no aplicativo, todos os campos, com exceção de **Especificação** e **Bloco/Prédio**, são obrigatórios. Portanto, caso o usuário confirme o cadastro sem preencher ao menos um deles, uma marcação é exibida nos que não foram preenchidos (Figura 36). Ou então se algum foi preenchido de forma errada, também é mostrada a marcação com uma mensagem.

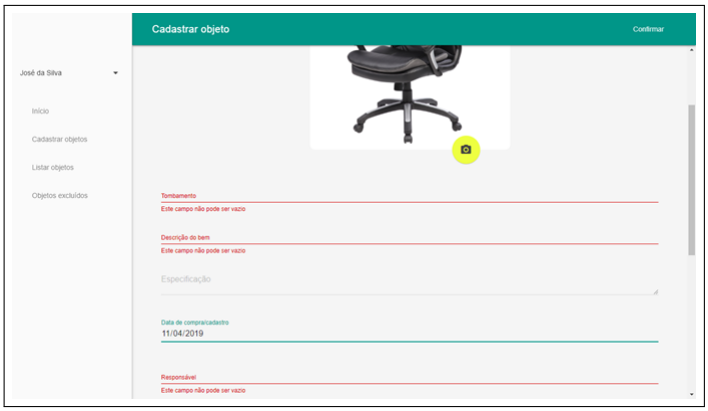
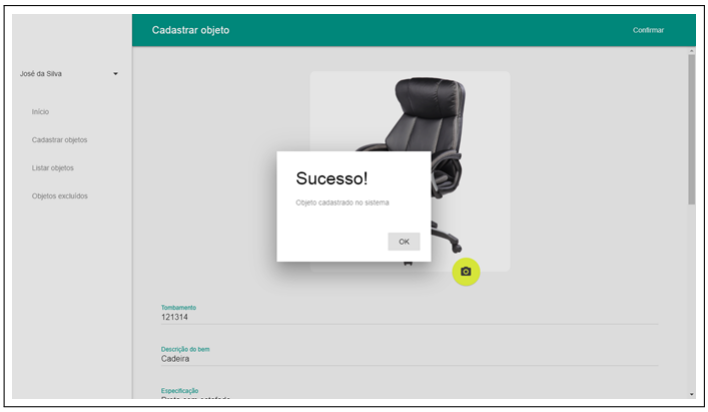


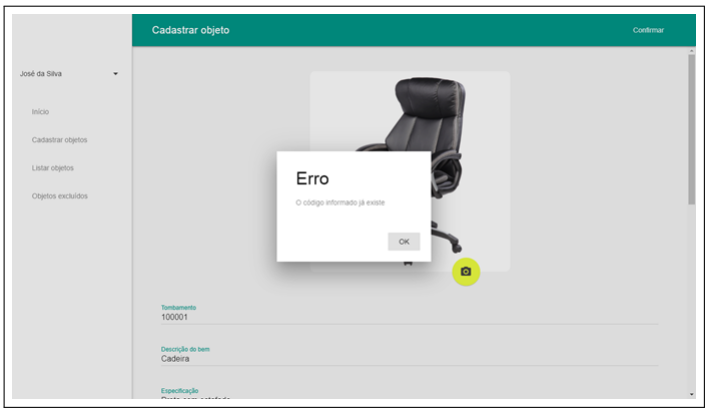
Figura 36 – Interface *web* - Validação de campos incorretos

Fonte: Elaborado pelo autor

Para confirmar o cadastro, basta clicar na opção “Confirmar” localizada no canto superior direito da página. Caso a operação ocorra corretamente, será exibida uma janela com uma mensagem de sucesso (Figura 37(a)). Caso contrário, será exibida uma janela com uma mensagem de erro (Figura 37(b)).



(a) Sucesso



(b) Erro

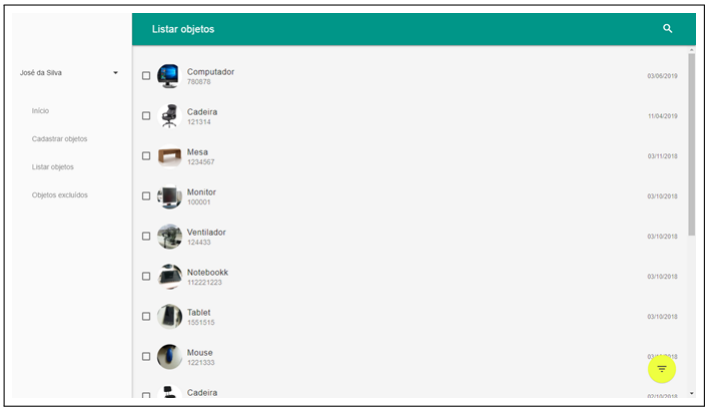
Figura 37 – Interface *web* - Janelas de confirmação de cadastro

Fonte: Elaborado pelo autor

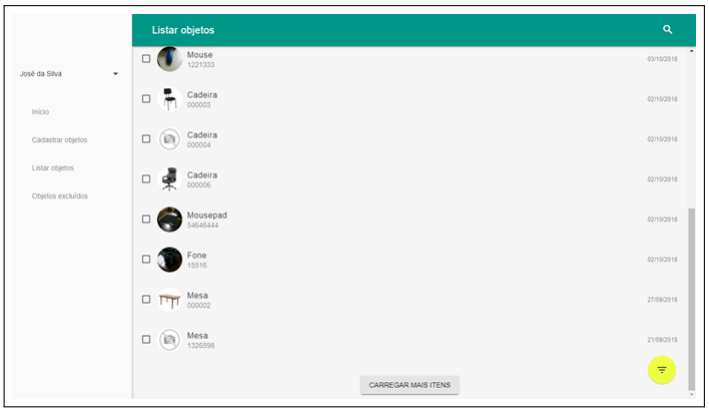
5.2.3.4 Listagem

Ao clicar em "Listar objetos" no menu lateral, é mostrada a página de listagem dos objetos cadastrados e ativos (Figura 38). A lista mostra algumas informações sobre cada objeto: miniatura da imagem, descrição do bem, tombamento e data de compra/cadastro.

Os objetos são carregados do servidor por partes, com o objetivo de não sobrecarregar o sistema quando há um grande volume de objetos cadastrados. Por esta razão, para ver mais objetos cadastrados é necessário chegar ao final da lista e clicar em "Carregar mais itens" (Figura 38(b)). Ao fazer isso, mais uma porção de objetos é requisitada do servidor, concatenando-se à lista corrente com os novos objetos.



(a) Início da lista



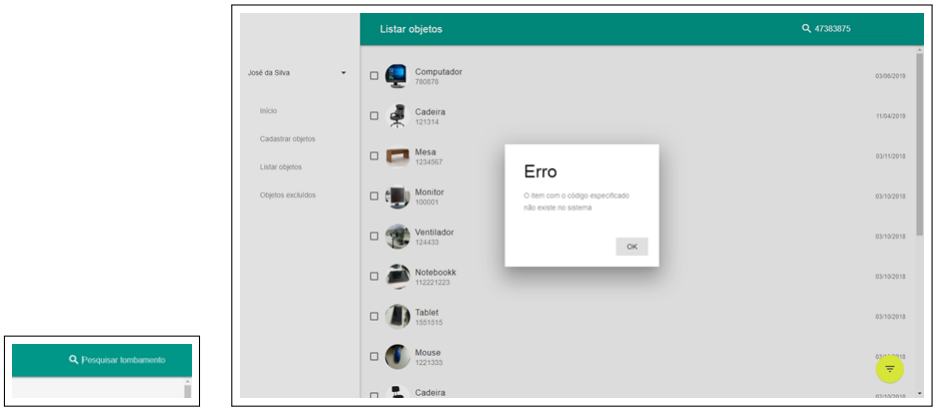
(b) Final da lista

Figura 38 – Interface *web* - Página de listagem de objetos

Fonte: Elaborado pelo autor

Ao clicar no ícone de lupa localizado no canto superior direito da página, é aberto um campo numérico para que o usuário possa pesquisar um objeto diretamente pelo número do tombamento (Figura 39). Assim que ele informa o código desejado e pressiona a tecla "Enter", uma requisição é enviada ao servidor para retornar o objeto desejado. Caso ele não esteja cadastrado no banco de dados, uma janela é mostrada com uma mensagem indicando que o objeto não existe (Figura 39(b)). Caso contrário, é exibida a página de

visualização de objetos mostrando as informações do objeto pesquisado.



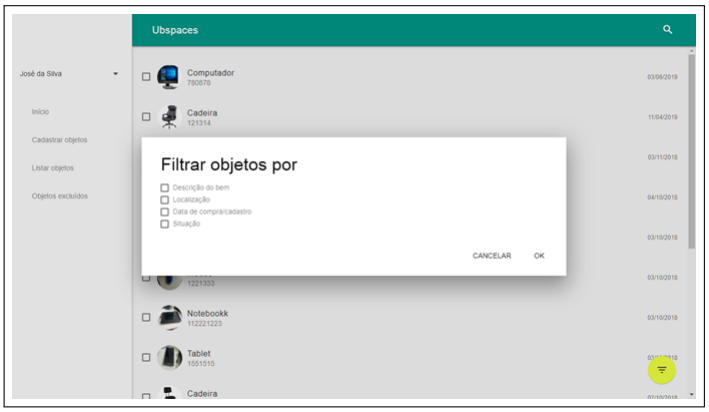
(a) Campo de pesquisa

(b) Mensagem de erro

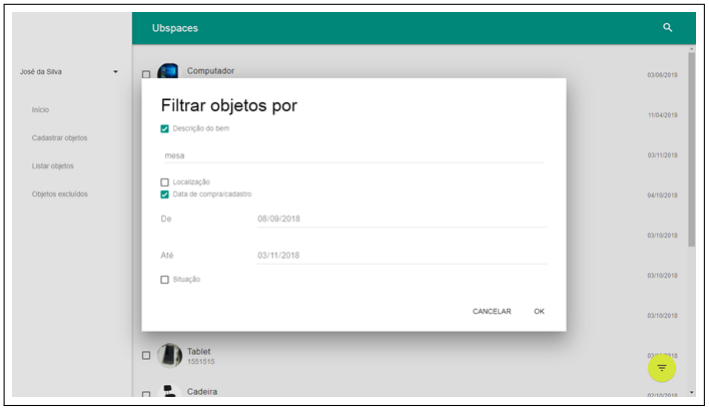
Figura 39 – Interface *web* - Pesquisa por tombamento

Fonte: Elaborado pelo autor

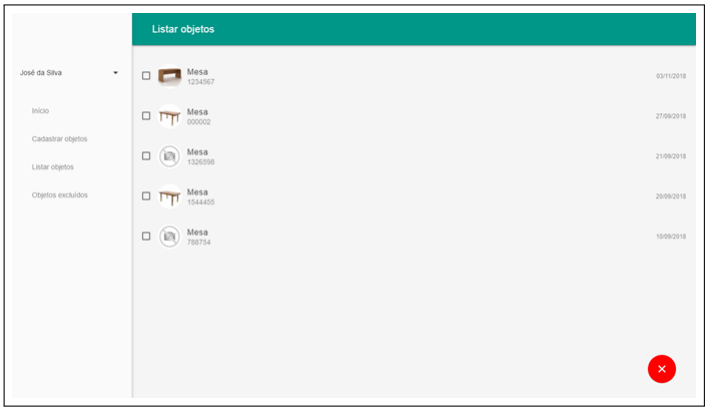
Ainda na página de listagem de objetos, ao clicar no FAB localizado no canto inferior direito, é mostrada a janela de filtragem dos objetos (Figura 40(a)). O filtro é exatamente o mesmo encontrado no aplicativo móvel, possuindo os mesmos parâmetros de busca que funcionam de forma independente. Ao inserir os dados de busca (Figura 40(b)) e clicar em "OK", o servidor retorna uma lista contendo somente os objetos condizentes com os parâmetros informados (Figura 40(c)). Para sair do modo de filtragem, basta clicar no FAB vermelho com um "X".



(a) Janela de filtragem



(b) Parâmetros de filtragem informados



(c) Resultados da filtragem

Figura 40 – Interface *web* - Filtragem de objetos

Fonte: Elaborado pelo autor

Na lista de objetos em questão, ao clicar em um item correspondente a um objeto, é mostrada a página de visualização de objeto.

5.2.3.5 Visualização

A página de visualização de objetos (Figura 41) mostra todas as informações relacionadas a um determinado bem de patrimônio, incluindo sua respectiva imagem em um tamanho maior. No canto superior direito da página há dois ícones: um de lixeira e outro de lápis, de forma semelhante ao aplicativo. Ao clicar no primeiro, o objeto é movido para a seção de objetos excluídos (será detalhado mais adiante). Ao clicar no segundo, é mostrada a página de edição do respectivo objeto.

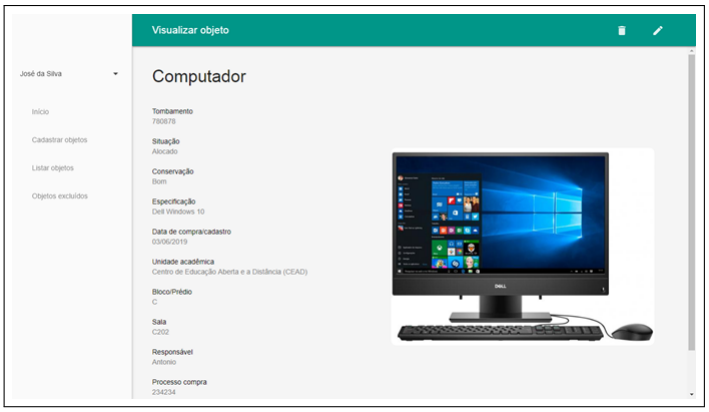
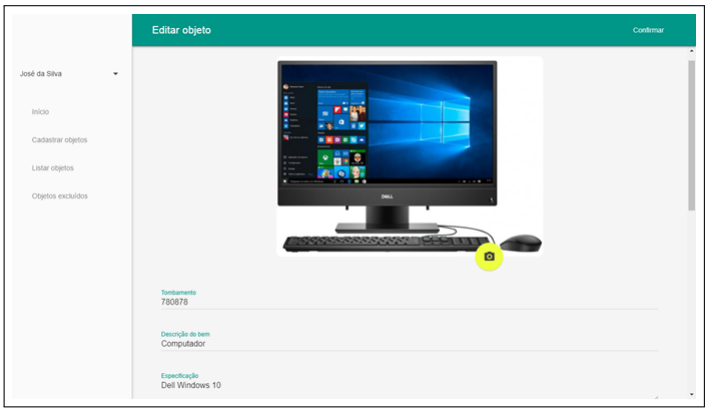


Figura 41 – Interface *web* - Visualização de um objeto

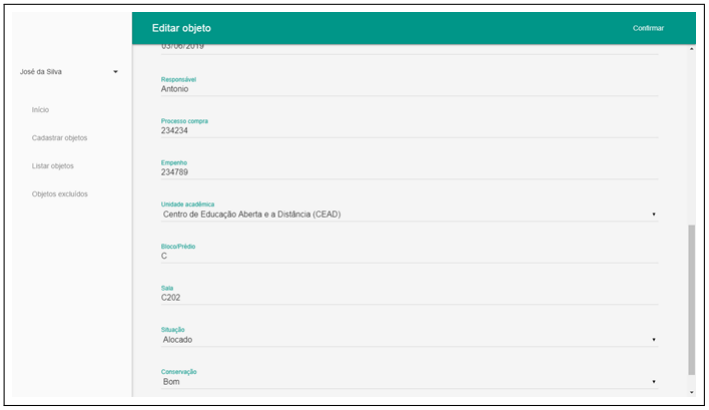
Fonte: Elaborado pelo autor

5.2.3.6 Edição

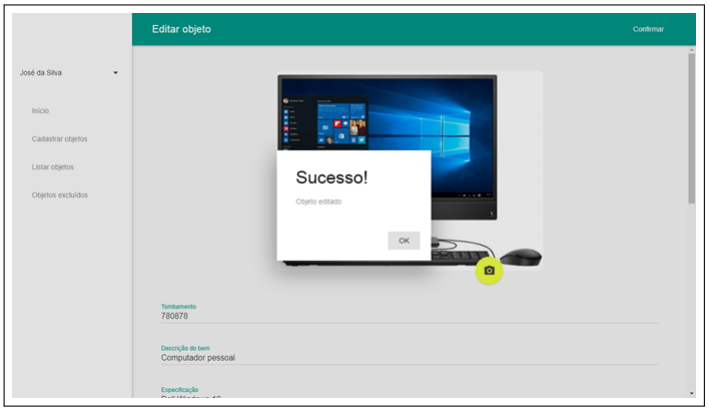
A página de edição de um objeto (Figura 42) é a mesma que a de cadastro. Porém nessa, os campos de entrada já vêm preenchidos com as informações do objeto a ser editado, continuando ainda assim editáveis para que o usuário possa alterar as informações. Esta página possui o mesmo funcionamento da página de cadastro, salvo algumas diferenças internas por se tratar de uma edição.



(a) Início da página



(b) Final da página



(c) Mensagem de sucesso

Figura 42 – Interface *web* - Página de edição

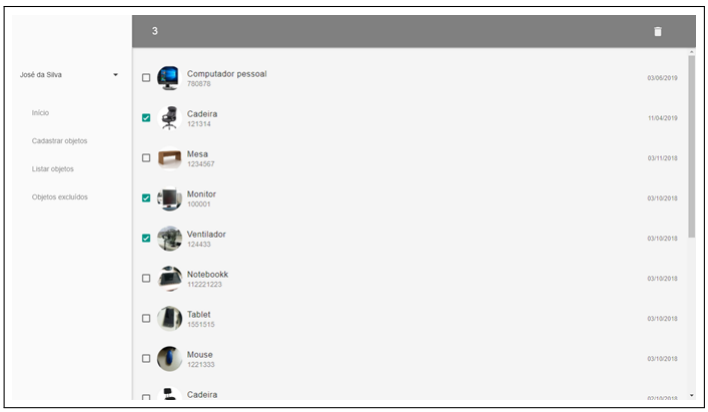
Fonte: Elaborado pelo autor

5.2.3.7 Exclusão

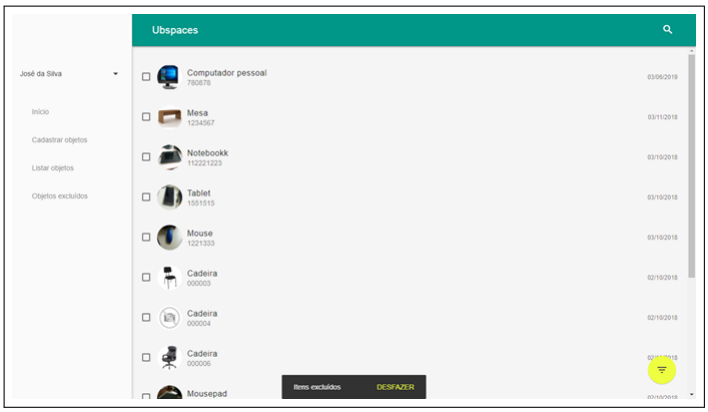
A exclusão de um ou mais objetos na interface *web* pode ocorrer de duas maneiras: selecionando-se múltiplos itens da lista de objetos ou clicando no ícone de lixeira na tela de visualização.

No primeiro modo, estando na página de listagem de objetos, ao clicar em um *checkbox* localizado à esquerda da miniatura de imagem, o item em questão é selecionado e o modo de seleção é ativado (Figura 43(a)). Enquanto este modo está ativado, a barra do topo da página fica cinza e surge um ícone de lixeira localizado no canto superior direito. Ao clicar neste ícone, os itens com o *checkbox* marcado são removidos da lista e movidos para a seção de objetos excluídos. No mesmo instante é mostrado um *snackbar* com uma mensagem de sucesso e um botão "Desfazer" (Figura 43(b)). Ao clicar neste botão, a operação é desfeita e os itens excluídos voltam para a lista nos lugares em que estavam.

No segundo modo, estando na página de visualização de um objeto, ao clicar no ícone de lixeira localizado no canto superior direito, uma janela de confirmação é exibida para o usuário (Figura 44(a)). Ao clicar em "Sim", o objeto é excluído e uma mensagem de sucesso é mostrada (Figura 44(b)).



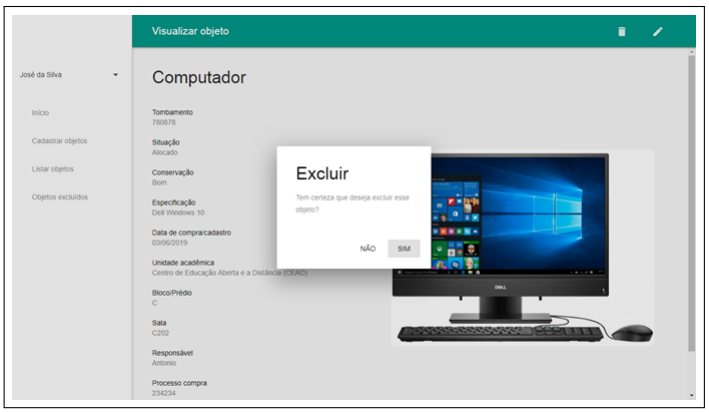
(a) Modo de seleção



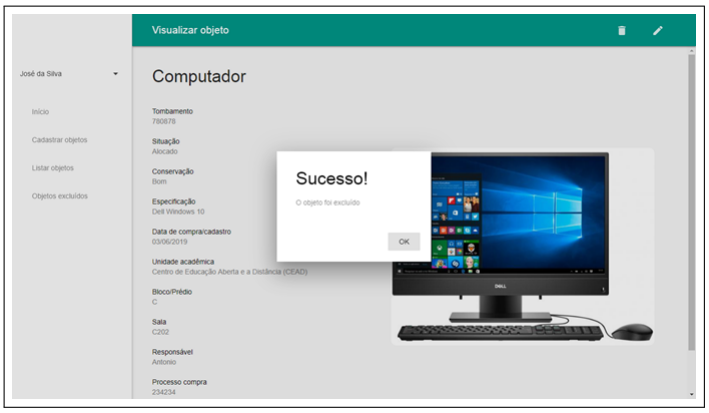
(b) *Snackbar*

Figura 43 – Interface *web* - Exclusão de objetos pelo modo de seleção

Fonte: Elaborado pelo autor



(a) Mensagem de sucesso



(b) Mensagem de sucesso

Figura 44 – Interface *web* - Exclusão de objetos pela tela de visualização

Fonte: Elaborado pelo autor

5.2.3.8 Objetos excluídos

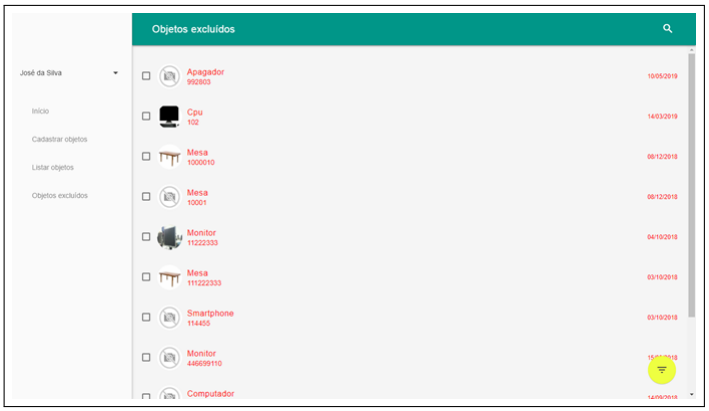
Assim como no aplicativo, a interface *web* também conta com a seção de objetos excluídos para mostrar os que já foram descartados ou que já não estão mais sendo utilizados. Para acessá-la, basta clicar na opção "Objetos excluídos" no menu lateral.

Ao clicar nesta opção, é mostrada a página de listagem dos objetos excluídos (Figura 45(a)). Essa lista contém as mesmas informações presentes na página de listagem dos objetos ativos. Porém nessa, os elementos de textos são escritos de vermelho para indicar que se tratam dos objetos excluídos, de forma semelhante ao aplicativo.

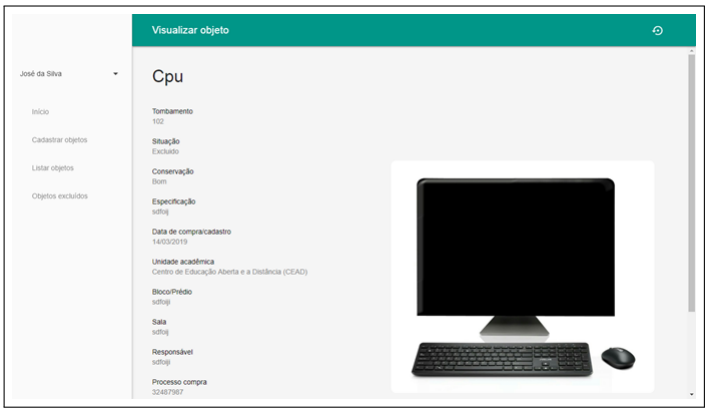
Ao clicar no ícone de lupa localizado no canto superior direito da página, também é aberto um campo de texto para que o usuário possa pesquisar algum objeto diretamente pelo seu número de tombamento. Mas estando na seção dos excluídos, essa pesquisa se dá somente entre os objetos nessa situação.

Ao clicar no FAB localizado no canto inferior direito da página, é mostrada uma janela de filtragem semelhante àquela presente na listagem de objetos ativos. Os parâmetros de busca são os mesmos, com exceção do parâmetro **Situação**. Este não se encontra presente pois os objetos nesta seção já possuem a situação definida como "Excluído".

Ao clicar em um item da lista, é mostrada a página de visualização do objeto selecionado, contendo todas as informações relacionadas a ele, incluindo sua respectiva imagem em um tamanho maior (Figura 45(b)). No canto superior direito da página há um ícone de uma espécie de seta circular. Ao clicar neste ícone é executada a operação de restaurar este objeto (será melhor detalhada adiante).



(a) Listagem



(b) Visualização de objeto

Figura 45 – Interface *web* - Seção de objetos excluídos

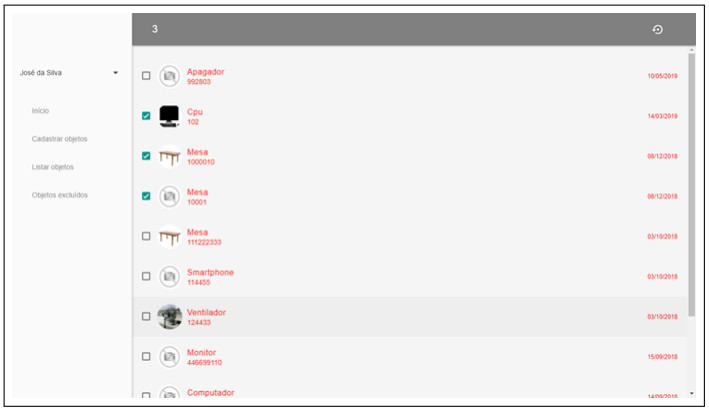
Fonte: Elaborado pelo autor

Na seção de objetos excluídos da interface *web* também é possível restaurar objetos à sua situação normal, caso haja necessidade. Para fazer isso, existem duas formas: selecionando-se múltiplos itens da lista através dos *checkboxes*, ou clicando no ícone de restauração na tela de visualização da Figura 45(b).

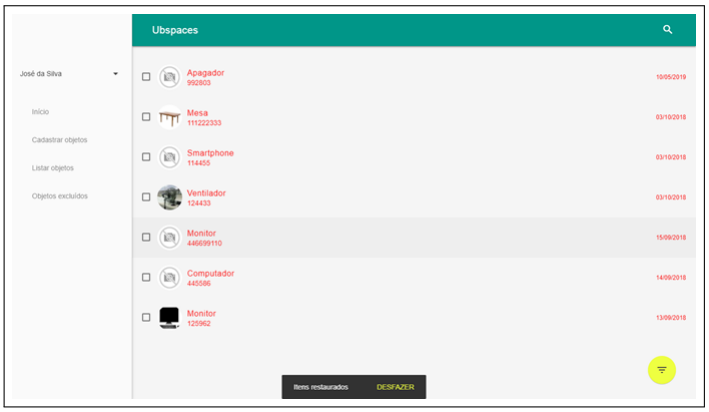
Na primeira forma, ao clicar no *checkbox* de algum item da lista, este item é selecionado e o modo de seleção é ativado, de forma semelhante à listagem dos objetos ativos (Figura 46(a)). O usuário pode selecionar um ou vários itens da lista conforme sua vontade. E então, ao clicar no ícone de restauração localizado no canto superior direito da página, os objetos selecionados são removidos da lista e movidos para a listagem padrão, dos objetos ativos. Aqui também aparece um *snackbar* contendo uma mensagem de sucesso e um botão "Desfazer" (Figura 46(b)). Ao clicar neste botão, a operação de restauração é desfeita e os objetos voltam para a lista de objetos excluídos em seus respectivos lugares de origem.

Na segunda forma, estando na página de visualização de um objeto excluído, ao clicar no ícone de restauração localizado no canto superior direito, é mostrada uma

janela de confirmação, solicitando ao usuário se ele deseja realmente restaurar o objeto (Figura 47(a)). Ao clicar em "Sim", é exibida uma janela com uma mensagem de sucesso e o objeto é então restaurado para sua situação padrão (Figura 47(b)).



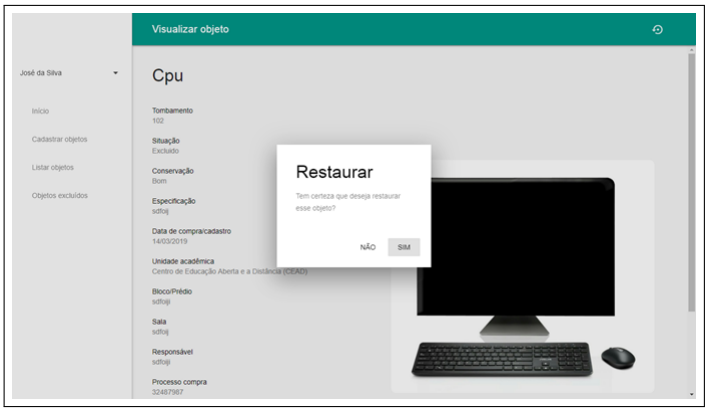
(a) Selecionando objetos



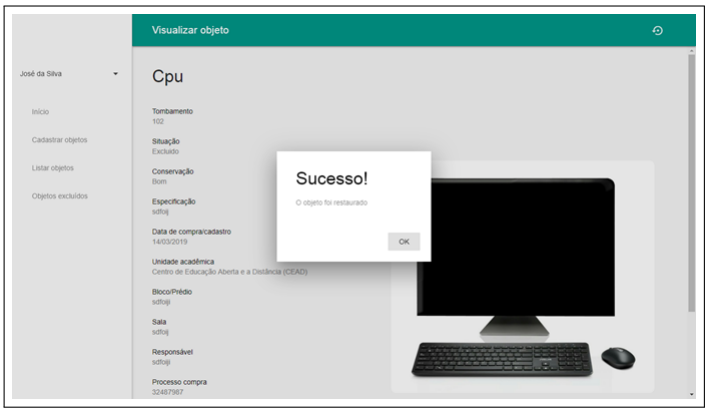
(b) *Snackbar*

Figura 46 – Interface *web* - Restauração de objetos por seleção

Fonte: Elaborado pelo autor



(a) Janela de confirmação



(b) *Mensagem de sucesso*

Figura 47 – Interface *web* - Restauração de objetos pela tela de visualização

Fonte: Elaborado pelo autor

5.2.4 Operadores

Nesta subseção são apresentadas as funcionalidades presentes na interface *web* referentes ao gerenciamento de operadores do sistema *Ubspaces*.

Como visto na subseção 4.2.2, somente o administrador do sistema possui permissão para gerenciar os operadores. Sendo assim, quando ele está logado no sistema, aparecem mais duas opções no menu lateral, que são “Cadastrar operador” e “Listar operadores”. Isto pode ser visto na Figura 48.

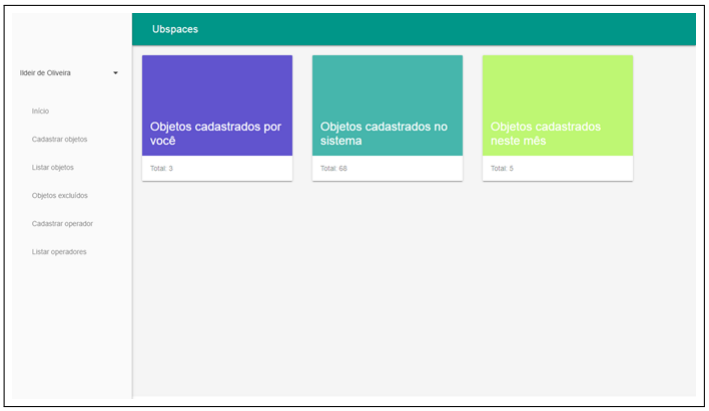
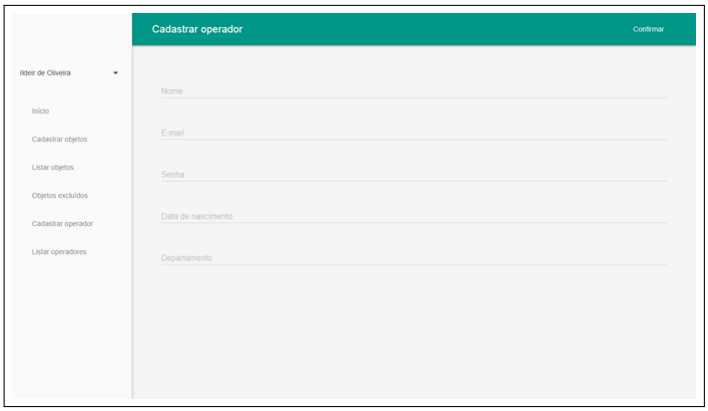


Figura 48 – Interface *web* - Tela inicial para o administrador

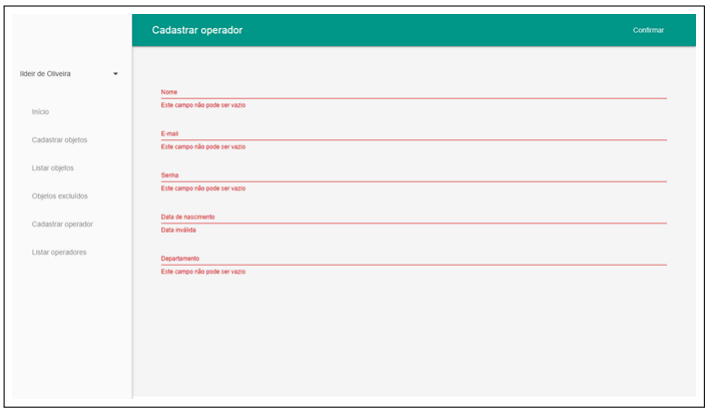
Fonte: Elaborado pelo autor

5.2.4.1 Cadastro

Ao clicar na opção "Cadastrar operador" localizada no menu lateral, é mostrada uma página contendo campos de formulário para o cadastro de um operador (Figura 49(a)). Neste caso, todos os campos são obrigatórios. Portanto, caso o administrador deixe de preencher um ou mais campos, ou preencha de forma incorreta, esses são marcados em vermelho com mensagens de validação (Figura 49(b)).



(a) Campos a serem preenchidos

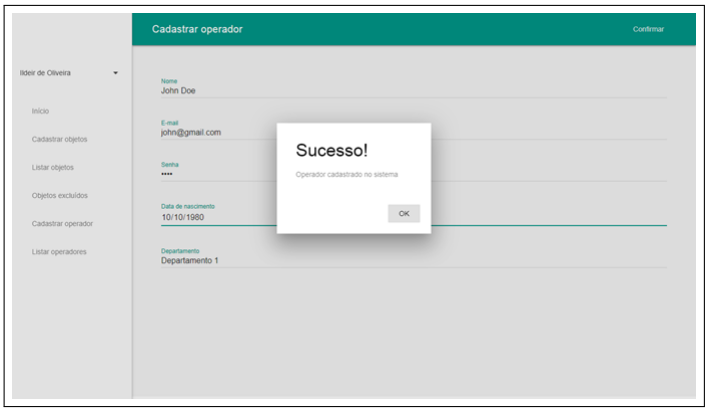


(b) Validação de campos preenchidos de forma indevida

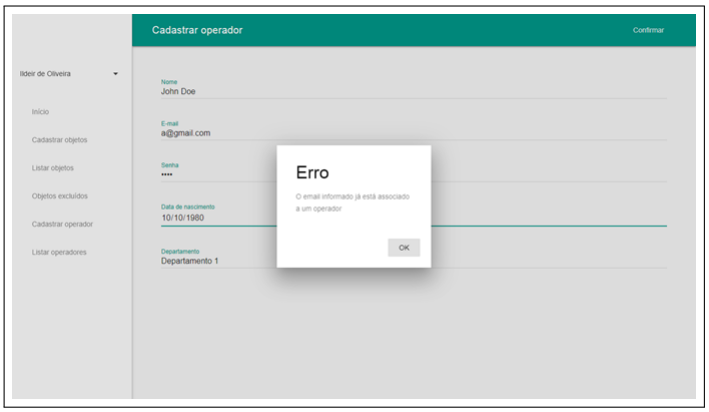
Figura 49 – Interface *web* - Página de cadastro de operador

Fonte: Elaborado pelo autor

Ao clicar na opção “Confirmar”, localizada no canto superior direito da página, é mostrada uma janela com uma mensagem de sucesso, caso o operador seja cadastrado corretamente no banco de dados (Figura 50(a)). Caso contrário, a janela exibe uma mensagem de erro, informando qual problema ocorreu (Figura 50(b)).



(a) Mensagem de sucesso



(b) Mensagem de erro

Figura 50 – Interface *web* - Retorno do servidor para o cadastro de operadores

Fonte: Elaborado pelo autor

5.2.4.2 Listagem

Ao clicar na opção "Listar operadores" localizada no menu lateral, é mostrada uma página que contém uma lista de todos os operadores cadastrados no sistema (Figura 51). Em cada item estão presentes algumas informações sobre o respectivo operador, que são o nome, *e-mail*, departamento a que pertence e sua data de nascimento.

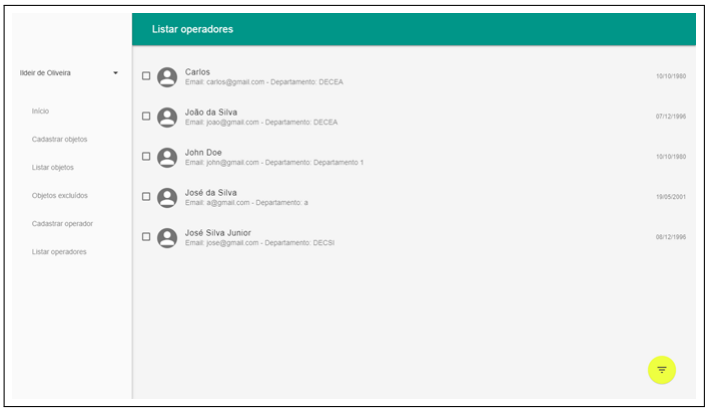
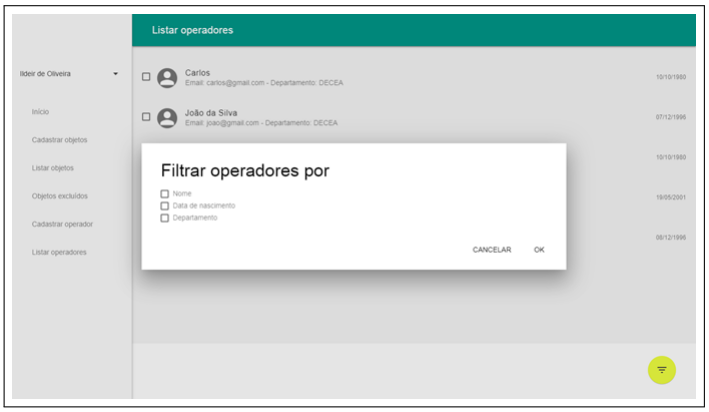


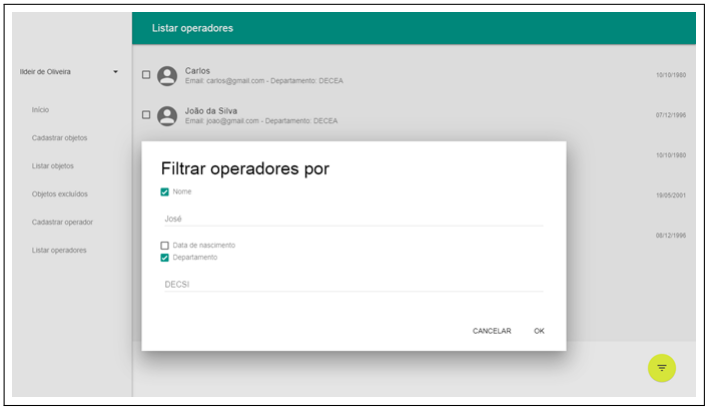
Figura 51 – Interface *web* - Página de listagem de operadores

Fonte: Elaborado pelo autor

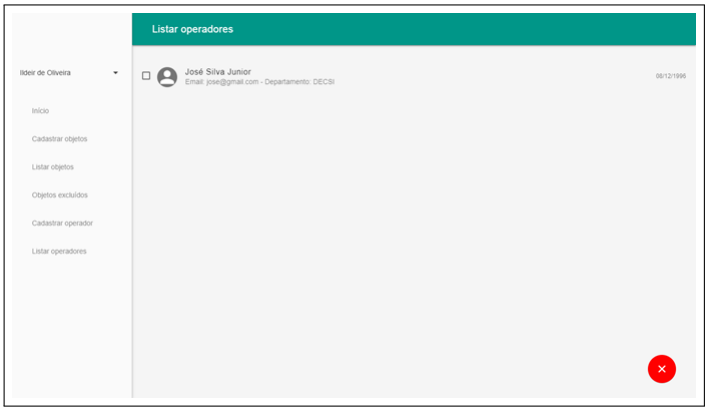
Ao clicar no FAB localizado no canto inferior direito da página, é exibida a janela de filtragem de operadores (Figura 52(a)). O filtro em questão funciona de forma semelhante ao filtro de objetos. Todavia nesse, os parâmetros de busca são diferentes. São eles: **Nome**, **Data de nascimento** e **Departamento**. Conforme a vontade do administrador, os parâmetros podem ser ativados ou desativados de forma independente (Figura 52(b)). Depois destes serem informados, ao clicar em "OK", é exibida uma nova lista contendo somente os operadores cujas informações são condizentes com os parâmetros de filtragem informados (Figura 52(c)). Para sair da pesquisa, basta clicar no FAB vermelho com um ícone de "X".



(a) Parâmetros de filtragem a serem preenchidos



(b) Parâmetros de filtragem preenchidos



(c) Resultado da filtragem

Figura 52 – Interface *web* - Janela de filtro de operadores

Fonte: Elaborado pelo autor

Ainda na tela de listagem dos operadores, ao clicar em um item da lista, é mostrada a página de edição do operador clicado. Está é mais detalhada no próximo tópico.

5.2.4.3 Edição

A página de edição de um operador contém campos de formulário referentes às suas informações (Figura 53). Por se tratar de uma edição, estes campos já vêm preenchidos com as informações atuais, para que então o administrador possa atualizá-las conforme a necessidade. Esta página é muito semelhante à de cadastro de operador. Entretanto, há uma pequena diferença na parte de alteração da senha.

Como a senha de um operador é salva no banco de dados de forma criptografada, não é possível recuperá-la em seu formato original. Portanto, só é possível manter a senha atual ou criar uma nova. Para isso, há um *checkbox* na página com o rótulo "Redefinir senha". Caso o administrador deseje alterar a senha do operador a ser editado, ele deve então marcar este *checkbox* e o campo de texto ao lado ficará habilitado para que ele informe uma nova senha. Vale ressaltar que ao fazer isto, o campo deve ser obrigatoriamente preenchido. Caso contrário, o *checkbox* deve ficar desmarcado. Então o campo de texto ao lado fica desativado e a senha permanece a mesma.

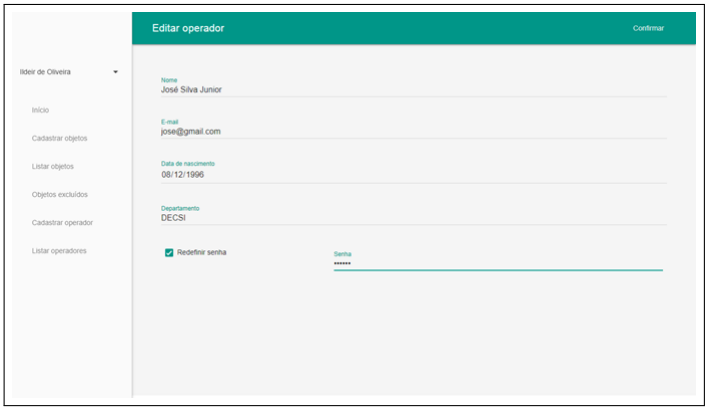
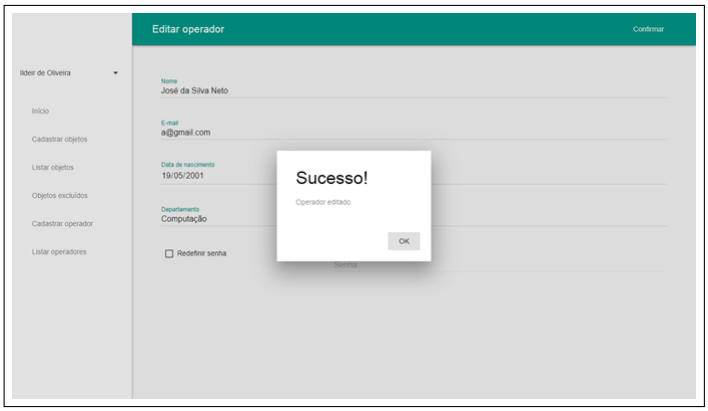


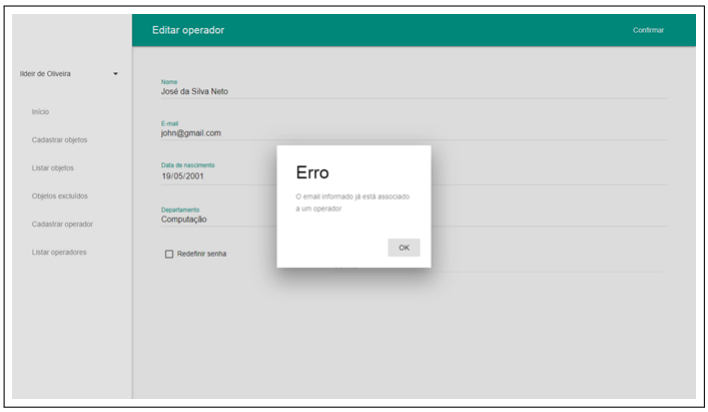
Figura 53 – Interface *web* - Página de edição de operador

Fonte: Elaborado pelo autor

Ao clicar na opção "Confirmar", localizada no canto superior direito da página, então é exibida uma janela com uma mensagem de sucesso, caso os dados tenham sido corretamente alterados no servidor (Figura 54(a)). Caso contrário, esta janela exibe uma mensagem informando qual foi o problema ocorrido (Figura 54(b)).



(a) Mensagem de sucesso



(b) Mensagem de erro

Figura 54 – Interface *web* - Mensagens de retorno do servidor para a edição de um operador

Fonte: Elaborado pelo autor

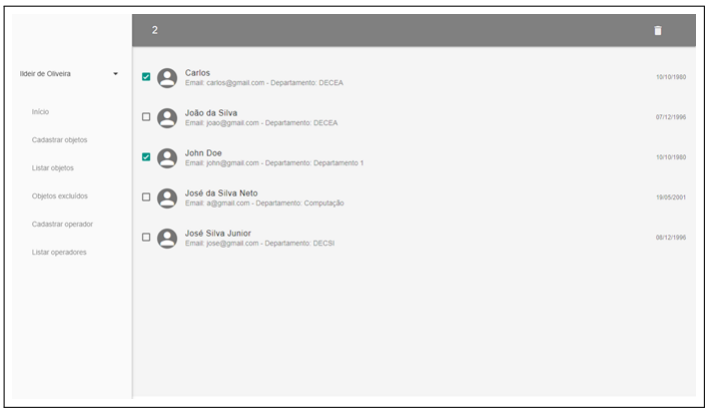
Vale observar também que esta página de edição é exatamente a mesma da Figura 34(b). A diferença é que na outra página o próprio operador pode alterar seus dados, enquanto que nesta é o administrador quem os altera.

5.2.4.4 Exclusão

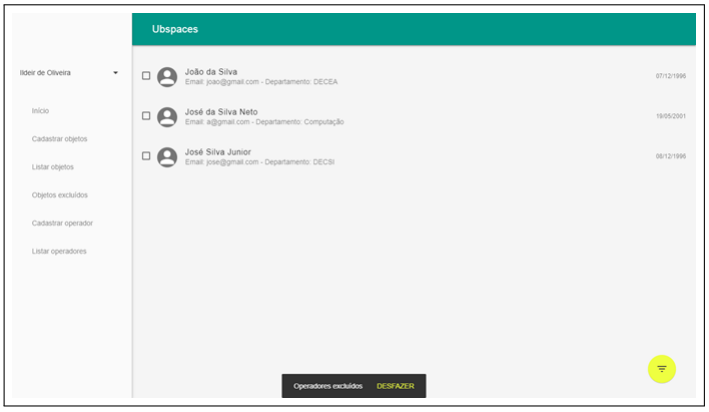
Diferentemente dos objetos, o sistema *Ubspaces* não possui uma seção que armazena usuários excluídos, pois isso não convém ao problema de gerenciamento, já que se trata de permissões de acesso ao sistema. Portanto, ao excluir um determinado operador, ele é apagado efetivamente do banco de dados.

O processo de exclusão de um operador se dá de forma semelhante à exclusão de objetos pela seleção de múltiplos itens da lista. Ao clicar em um *checkbox* de algum item da lista, então o modo de seleção é ativado e um ícone de lixeira aparece no canto superior direito da página. A partir daí o administrador pode excluir um ou vários operadores de uma só vez (Figura 55(a)). Ao concluir a seleção e clicar no ícone de lixeira, os operadores selecionados são removidos da lista. Então é mostrado um *snackbar* contendo

uma mensagem de sucesso da operação e um botão “Desfazer” (Figura 55(b)). Ao clicá-lo, a operação de exclusão é desfeita e os operadores voltam para a lista em seus respectivos lugares de origem. Caso o administrador não clique neste botão, então os operadores selecionados são excluídos permanentemente do banco de dados.



(a) Modo de seleção



(b) *Snackbar*

Figura 55 – Interface *web* - Exclusão de operadores

Fonte: Elaborado pelo autor

6 Conclusão

Este trabalho apresentou *Ubspaces*, um sistema computacional móvel para o gerenciamento de informações relacionadas a objetos do espaço físico, que neste trabalho, foi aplicado aos bens de patrimônio da UFOP. Este sistema se utiliza dos conceitos de Computação Móvel e Computação em Nuvem para a definição de sua estrutura básica de funcionamento. Com ele, é possível extrair informações presentes em QRCodes ou códigos de barras de objetos, e com base neles, realizar processamento e manipulação dos dados de forma móvel.

O sistema é composto basicamente de um *web server* REST, de um aplicativo móvel para sistema *Android* e de uma interface *web* para navegadores. O *web server* hospeda o banco de dados do sistema e faz a ligação entre as requisições dos usuários (clientes) e os dados armazenados no banco. O aplicativo móvel é o *software* utilizado pelos operadores e administrador para o gerenciamento de informações dos bens de patrimônio. E a interface *web* é o *software* utilizado pelo administrador para o gerenciamento das informações de objetos, e também de operadores.

No sistema *Ubspaces*, se tornou mais fácil acessar informações referentes a um bem de patrimônio e poder manipulá-las no mesmo instante. O mesmo processo, em teoria, seria mais complicado de ser feito da forma convencional, uma vez que as informações estão registradas em meios físicos e centralizados em um mesmo local. Apesar de ainda não ter ido a campo para ser testado com as cargas reais de dados, o sistema desenvolvido demonstrou resultados satisfatórios suficientes para cumprir os objetivos propostos neste trabalho.

A principal limitação encontrada no desenvolvimento deste trabalho foi executar e testar o sistema somente em redes locais. Isso se deveu à impossibilidade de contratação de um serviço de hospedagem e também a dificuldade de encontrar máquinas que possuem endereço IP real na *Internet*. Essa limitação certamente impactou o desempenho do sistema, pois em redes locais não ocorrem os problemas comuns da *Internet*, como por exemplo, congestionamentos. Sendo assim, ainda não se sabe o comportamento do sistema quando este for colocado em funcionamento para toda a rede, possuindo um grande volume de clientes e requisições.

Algumas melhorias podem ser feitas para aprimorar ainda mais o desempenho e usabilidade do sistema *Ubspaces*. Ademais, elas podem ser realizadas em trabalhos futuros. São elas listadas abaixo:

- Desenvolver uma versão do aplicativo para *iOS*;

- Permitir a geração de relatórios mais abrangentes (por exemplo, objetos em uma sala que estão fora do seu local designado);
- Permitir o cadastro de unidades de localização, que neste trabalho são a unidade acadêmica, bloco/prédio e sala;
- Manter um histórico de todos os locais por onde um objeto já passou e
- Distribuir a carga computacional do *web server* para mais máquinas (sistema distribuído).

Por fim, vale mencionar que o conceito principal deste trabalho possui muito potencial de exploração. Um sistema de computação móvel para gerenciamento de informações relacionadas a objetos do espaço físico pode ser implementado para atuar em diversas áreas. Abaixo, são indicadas algumas delas juntamente com suas possíveis aplicações para serem trabalhadas em pesquisas futuras:

- Turismo: Utilizar o QRCode como fonte de acesso a informação de locais turísticos e obras de arte;
- Obras públicas: Utilizar o QRCode como fonte de acesso a informações *in locu* de obras e projetos públicos;
- Controle de presença: Utilizar o QRCode como fonte de acesso a registro de presença de pessoas em reuniões marcadas em espaços definidos e
- Automóveis: utilizar o QRCode como fonte de entrada para informações (documentos, registros de manutenção, etc.) de veículos.

Referências

ABREU, K. C. K. História e usos da internet. *Biblioteca on-line de Ciências da Comunicação. Universidade da Beira Interior. Covilhã*, 2009. Citado na página 20.

BAX, M. P.; LEAL, G. J. Serviços web e a evolução dos serviços em ti. *DataGramaZero: Revista de Ciência da Informação. Rio de Janeiro*, v. 2, n. 2, 2001. Citado na página 25.

BROOKSHEAR, J. G. *Ciência da Computação-: Uma Visão Abrangente*. [S.l.]: Bookman Editora, 2013. Citado na página 20.

CONVERSE, T.; PARK, J. *PHP: a bíblia*. [S.l.]: Gulf Professional Publishing, 2003. Citado na página 30.

COSTA, E. R.; KOMATI, K. S. Um sistema móvel de compras rápido e seguro via qr code e vitrine virtual. *X Encontro Anual de Computação (ENACOMP), Catalão, Goiás*, 2013. Citado na página 28.

COULOURIS, G. et al. *Sistemas Distribuídos-: Conceitos e Projeto*. [S.l.]: Bookman Editora, 2013. Citado 2 vezes nas páginas 24 e 25.

DATE, C. J. *Introdução a sistemas de bancos de dados*. [S.l.]: Elsevier Brasil, 2004. Citado na página 29.

DIAS, A. C.; ZAVAN, A. R.; OLIVEIRA, F. W. C. de. Sistema de gerenciamento de inventário de bens móveis para o instituto federal do paraná-campus paranavaı. 2018. Citado na página 27.

DRESCH, A.; EFROM, D. R.; GRUMOVSKI, D. Controle de patrimônio por sistema wireless (rfid) system control of property through rfid. *Revista E-Tech: Tecnologias para Competitividade Industrial-ISSN-1983-1838*, v. 1, n. 1, p. 47–57, 2008. Citado na página 27.

FIGUEIREDO, C. M.; NAKAMURA, E. Computação móvel: Novas oportunidades e novos desafios. *T&C Amazônia*, v. 1, n. 2, p. 21, 2003. Citado 2 vezes nas páginas 17 e 20.

FRANCISCO, M. R. e R. Web services rest conceitos, anÁlise e implementaÇÃo. *Educação, Tecnologia e Cultura - E.T.C.*, n. 14, 2016. ISSN 2525-3859. Disponível em: <https://publicacoes.ifba.edu.br/index.php/etc/article/view/25>. Acesso em: 10 jun. 2019. Citado na página 31.

GOOGLE. *Introduction - Material Design*. [S.l.], 2019. Disponível em: <https://material.io/design/introduction/>. Acesso em: 11 jun. 2019. Citado na página 34.

INDRUSIAK, L. S. Linguagem java. *Grupo JavaRS JUG Rio Grande do Sul*, 1996. Citado na página 32.

LOPES, C. J. F.; RAMALHO, J. C. Web services: Metodologias de desenvolvimento. 2004. Citado na página 25.

MASALHA, F.; HIRZALLAH, N. et al. A students attendance system using qr code. *International Journal of Advanced Computer Science and Applications*, v. 5, n. 3, p. 75–79, 2014. Citado na página 23.

MATEUS, G. R.; LOUREIRO, A. A. F. Introdução à computação móvel. DCC/IM, COPPE/UFRJ, 1998. Citado na página 21.

MILIES, C. P. A matemática dos códigos de barras. Mini-curso apresentado na Bienal da Sociedade Brasileira de Matemática. UFG, 2006. Citado na página 22.

NETO, I. Relatório de projeto-aplicação android-sistema de orientação interior com recurso a códigos qr. 2015. Citado na página 27.

NIST. *Cloud Computing*. [S.l.], 2018. Disponível em: <https://csrc.nist.gov/projects/cloud-computing>. Acesso em: 14 jun. 2019. Citado na página 24.

NUNES, S.; DAVID, G. Uma arquitectura web para serviços web. *XATA-2005*, 2005. Citado na página 26.

PEREIRA, L. C. O.; SILVA, M. L. da. *Android para desenvolvedores*. [S.l.]: Brasport, 2009. Citado na página 31.

PERLES, J. B. Comunicação: conceitos, fundamentos e história. *Biblioteca on-line de Ciências da Comunicação*, 2007. Citado na página 20.

ROSSETTI, A. G.; MORALES, A. B. T. O papel da tecnologia da informação na gestão do conhecimento. *Ciência da Informação*, SciELO Brasil, v. 36, n. 1, p. 124–135, 2007. Citado na página 17.

RUSCHEL, H.; ZANOTTO, M. S.; MOTA, W. d. Computação em nuvem. *Curitiba, abr*, p. 1–3, 2010. Citado na página 24.

SILVA, M. S. *jQuery-A Biblioteca do Programador JavaScript-3ª Edição: Aprenda a criar efeitos de alto impacto em seu site com a biblioteca JavaScript mais utilizada pelos desenvolvedores web*. [S.l.]: Novatec Editora, 2013. Citado na página 35.

SILVA, S. E. et al. Arquitetura da internet dos espaços: Modelagem de sua aplicação em um ambiente de manutenção industrial. *SIMPEP*, 2018. Citado na página 23.

SOON, T. J. Qr code. *Synthesis Journal*, v. 2008, p. 59–78, 2008. Citado na página 22.

TAURION, C. *Cloud computing-computação em nuvem*. [S.l.]: Brasport, 2009. Citado 2 vezes nas páginas 17 e 24.

VIECELLI, M. E. A importância do controle patrimonial para as entidades públicas: um estudo de caso no centro de educação superior do norte do rio grande do sul (cesnors). *Revista de Administração*, v. 11, n. 20, p. 9–28, 2013. Citado na página 17.

WANG, L. et al. A clould based approach for weee remanufacturing. *CIRP Annals - Manufacturing Technology*, v. 63, p. 409–412, 2014. Citado na página 28.

TERMO DE RESPONSABILIDADE

Eu, **Ildeir de Oliveira Eler Junior** declaro que o texto do trabalho de conclusão de curso intitulado "***Sistema de Computação Móvel para Gerenciamento de Informações Relacionadas a Objetos do Espaço Físico***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 11 de julho de 2019

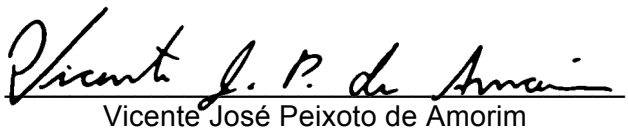
Ildeir de Oliveira Eler Junior

Ildeir de Oliveira Eler Junior

ANEXO IX – DECLARAÇÃO DE CONFORMIDADE

Certifico que o(a) aluno(a) **Ildeir de Oliveira Eler Junior**, autor do trabalho de conclusão de curso intitulado "**Sistema de Computação Móvel para Gerenciamento de Informações Relacionadas a Objetos do Espaço Físico**" efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

João Monlevade, 22 de Julho de 2019.



Vicente José Peixoto de Amorim