UNIVERSIDADE FEDERAL DE SANTA CATARINA

Lucas Martins Petroski

Desenvolvimento de um componente App Inventor para Deep Learning

Florianópolis – SC
2020/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Lucas Martins Petroski

Desenvolvimento de um componente App Inventor para Deep Learning

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências Da Computação.

Florianópolis – SC
2020/2

Lucas Martins Petroski

Desenvolvimento de um componente App Inventor para Deep Learning

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências Da Computação.

Orientador(a):

_____

Prof. Dr. Jean Carlo Rossa Hauck
Orientador
Universidade Federal de Santa Catarina

Banca examinadora:

_____

Prof. Dr. Elder Rizzon Santos
Avaliador
Universidade Federal de Santa Catarina

_____

Prof.ª Msc. Nathalia da Cruz Alves
Avaliador
Instituto Federal de Santa Catarina

RESUMO

Mesmo com o crescente ensino de computação, ainda há dificuldades em despertar o interesse dos alunos da educação básica para o aprendizado de computação, especialmente sobre lógica de programação e algoritmos. Conforme os problemas computacionais se tornaram progressivamente mais complexos, a inteligência artificial surgiu com soluções mais acessíveis e viáveis. O uso de inteligência artificial pode auxiliar no aprendizado do pensamento computacional. Entretanto, plataformas para ensino do pensamento computacional, por exemplo o App Inventor, não oferecem opções nativas de acesso a recursos de inteligência artificial para apoiar esse processo. Este projeto propõe integrar o App Inventor com *Deep Learning*. Para isso, é desenvolvido um componente na forma de extensão, juntamente com um servidor que converte as chamadas do componente em invocações de operações que utilizam *Deep Learning*, e retornam os resultados ao componente. A ferramenta foi avaliada utilizando casos de teste com base nos requisitos levantados a fim de comprovar sua eficácia, mas sua usabilidade não pode ser avaliada devido à pandemia da COVID-19. Espera-se que a ferramenta desenvolvida possa contribuir para o aumento do engajamento dos alunos no ensino computacional, propiciando uma experiência melhor para seus usuários.

Palavras Chave: App Inventor, Deep Learning, Inteligência Artificial

Lista de Figuras

Lista de Quadros

SUMÁRIO

1. Introdução

O Pensamento Computacional é um subconjunto de competências e habilidades relacionadas à abstração e decomposição de problemas de forma a permitir sua resolução usando recursos computacionais e estratégias algorítmicas (WING, 2006). Ele introduz uma nova abordagem para o ensino da Ciência da Computação, pois parte da premissa de que a inserção dos conceitos da Ciência da Computação na educação básica desenvolve uma habilidade de abstração, a qual pode ajudar as crianças na resolução de problemas em todas as áreas da vida (DANIEL et al., 2017).

Existem diversas dificuldades para se envolver os alunos na construção de conhecimento sobre lógica de programação e algoritmos (ARAÚJO et al., 2018). Uma das plataformas utilizadas para esta finalidade é o App Inventor[1], desenvolvida pela Google e mantida pelo Instituto de Tecnologia de Massachusetts (MIT). O App Inventor é um ambiente de programação baseado em blocos que permite pessoas com pouco conhecimento em programação desenvolver aplicativos para dispositivos Android[2]. O App Inventor proporciona a experiência de desenvolver os mais diversos aplicativos de celulares por meio de um ambiente de desenvolvimento de programação visual e permite que qualquer pessoa comece a programar e construa aplicativos completos para dispositivos Android (DEMETRIO, 2017).

Conforme os problemas computacionais se tornaram progressivamente mais complexos, desenvolvedores e pesquisadores recorreram à inteligência artificial (IA) para soluções mais acessíveis e viáveis, como exemplo destas aplicações pode-se citar o reconhecimento de voz realizado pelos assistentes virtuais como a Siri[3], Cortana[4] e o Google Voice[5], o sistema de recomendação de filmes, séries e músicas, no Netflix[6], Youtube[7] e Spotify e a detecção de spam nos e-mails.

Uma das soluções atualmente empregadas na IA é o *Machine Learning* (aprendizado de máquina) e mais especificamente o *Deep Learning* (aprendizagem profunda). Segundo Michalski, Carbonell e Mitchell (1986) *Machine Learning* constitui-se no estudo e a modelagem computacional do processo de aprendizagem em suas múltiplas manifestações, sendo assim, pode-se defini-lo como a habilidade de computadores aprenderem sem serem explicitamente programados. O termo *Deep Learning* passou a se referir a uma coleção de técnicas que, juntas, demonstraram ganhos inovadores sobre os melhores algoritmos de *machine learning* existentes em vários campos. Nos últimos anos, esses métodos

[1] www.appinventor.mit.com
[2] www.android.com/intl/pt-BR_br
[3] www.apple.com/br/siri
[4] www.microsoft.com/pt-br/windows/cortana
[5] voice.google.com
[6] www.netflix.com/br
[7] www.youtube.com

revolucionaram a classificação de imagens e o reconhecimento de fala devido à sua flexibilidade e alta precisão (CHING et al., 2017).

O ensino de computação na educação básica é fundamental e estratégico, pois envolve abstrações e técnicas necessárias para a descrição e análise de informações e processos, e o empoderamento de seus conceitos fundamentais permitirá que os alunos compreendam de forma mais completa o mundo e tenham, consequentemente, maior autonomia, flexibilidade, resiliência, pró-atividade e criatividade (SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2019). Entretanto, plataformas para ensino do pensamento computacional, como o Scratch[8] e App Inventor não oferecem opções nativas de acesso a recursos de IA, como por exemplo *Deep Learning* para apoiar o ensino de computação por meio do desenvolvimento de aplicativos.

Com a finalidade de acrescentar funcionalidades de IA aos aplicativos e melhorar a experiência dos alunos no aprendizado de computação, este projeto propõe integrar o App Inventor com IA, mais precisamente *Deep Learning*. Para isso, objetiva-se desenvolver um componente na forma de extensão para a plataforma App Inventor, juntamente com um servidor que converte as chamadas do componente em invocações de operações para um framework, que utilizam *Deep Learning*, e retornam os resultados ao componente. Com a disponibilização de funcionalidades de IA no App Inventor espera-se que os alunos da educação básica possam aprender computação desenvolvendo aplicações inovadoras que utilizem *Deep Learning* sem precisarem se preocupar com a sua complexidade, e, com isso, sintam-se estimulados a aprender computação.

1.1. Objetivos

Este trabalho faz parte de um conjunto de trabalhos em *Machine Learning*, que possuem como objetivo geral apoiar o ensino de IA, no contexto da Iniciativa Computação na Escola/INE/CTC/UFSC (BAULÉ, 2020; LEHMKUHL, 2020; MARTINS, 2019; KRETZER, 2019). Complementando esse conjunto de trabalhos, o presente trabalho tem como objetivo geral integrar o App Inventor com IA, mais precisamente *Deep Learning*. A integração é realizada por meio do desenvolvimento de uma extensão, que faz requisições a um servidor para obter a resposta de uma chamada de função que utiliza uma rede neural. Além disso, também é desenvolvido um servidor que converte chamadas do componente de extensão e retorna a resposta ao componente.

Os objetivos específicos do trabalho são:

- Sintetizar a fundamentação teórica sobre *Deep Learning* e o ambiente App Inventor.

[8] scratch.mit.edu

- Analisar o estado da arte sobre componentes ou extensões de *Machine Learning* em ambientes de programação visual.
- Analisar, modelar e implementar um componente na forma de extensão para a plataforma App Inventor.
- Modelar e implementar um servidor que converte chamadas do componente e retorna a resposta ao componente.

1.2. Método de Pesquisa

Para atingir seus objetivos, este projeto foi dividido em 5 etapas:

Etapa 1 - Análise da fundamentação teórica: Para dar início ao projeto, será feita revisão da literatura sobre os conceitos relevantes para este trabalho. As atividades da primeira etapa são:

A1.1. Fundamentação teórica

A1.1.1. Ensino computação

A1.1.2. *Machine Learning*

A1.1.3. *Deep Learning*

A1.1.4. Ambiente App Inventor

Etapa 2 - Estado da Arte e trabalhos relacionados: Levantamento do estado da arte de componentes ou extensões de *Machine Learning* em ambientes de programação visual como App Inventor, Pencil Code[9], Scratch, Machine Learning For Kids (2019). Nesta etapa é realizado um estudo de mapeamento (PETERSEN, et al., 2008). Essa etapa está dividida nas seguintes atividades:

A2.1 – Definição do protocolo de estudo.

A2.2 – Execução da busca de componentes ou extensões de *Deep Learning* em ambientes de programação visual

A2.3 – Extração e análise das informações relevantes sobre componentes ou extensões e análise dessas informações, potencializando-as para serem utilizadas no componente a ser desenvolvido no trabalho.

Etapa 3 - Proposta de solução: Na terceira etapa do projeto será feita a elaboração do escopo do trabalho, análise de requisitos e a modelagem e descrição da solução proposta. As atividades desta etapa são:

A3.1. Análise de requisitos

A3.2. Modelagem e descrição da solução proposta

Etapa 4 - Desenvolvimento do componente: Na quarta etapa do projeto será feita a implementação do componente na forma de extensão do App Inventor baseado nas informações obtidas nas etapas anteriores. As atividades desta etapa são:

[9] pencilcode.net

A4.1. Implementar o componente
A4.2. Documentar a implementação
A4.3. Realizar os testes

Etapa 5 - Desenvolvimento do servidor: Na quinta etapa do projeto será feita a implementação do servidor e testes de integração com o componente baseado nas informações obtidas nas etapas anteriores. As atividades desta etapa são:

A5.1. Implementar o servidor
A5.2. Documentar a implementação
A5.3. Realizar os testes do servidor e de integração

2. Fundamentação Teórica

Este capítulo apresenta os conceitos referentes à teoria de ensino e aprendizagem, conceitos de *Machine Learning* e como estes conceitos são aplicados ao ensino da computação para crianças. Também é apresentada uma visão geral sobre o ambiente de programação App Inventor.

2.1. Ensino de Computação

Segundo a Sociedade Brasileira de Computação (SBC, 2019), nas diretrizes para ensino de Computação na educação básica, o ensino de Computação desenvolve uma série de competências nos alunos de forma única e complementar à formação dada pelas outras áreas do conhecimento. Essas competências estão sumarizadas em cinco competências (SBC, 2019):

- Interpretação e transformação do mundo: Aplicar conhecimentos de Computação para compreender o mundo e ser um agente ativo e consciente de transformação do mundo digital, capaz de entender e analisar criticamente os impactos sociais, culturais, econômicos, legais e éticos destas transformações.
- Aplicação de Computação em diversas áreas: Compreender a influência dos fundamentos da Computação nas diferentes áreas do conhecimento, reconhecendo que a Computação contribui no desenvolvimento do raciocínio lógico, do pensamento computacional, do espírito de investigação, da criatividade, e da capacidade de produzir argumentação coerente.
- Formulação, execução e análise do processo de resolução de problemas: Utilizar conceitos, técnicas e ferramentas para identificar e analisar problemas, modelá-los e resolvê-los, usando representações e linguagens para descrever processos e informação, validando estratégias e resultados.
- Desenvolvimento de projetos: Desenvolver e/ou discutir projetos de diversas naturezas envolvendo Computação.
- Computação é uma ciência: Compreender os fundamentos da Computação.

Já no CSTA (Computer Science Teachers Association) K-12, um dos currículos que servem internacionalmente de referência para a educação básica, tem como objetivo que os estudantes sejam cidadãos informados que possam se envolver criticamente em discussões públicas sobre tópicos de ciência da computação, desenvolvam-se como aprendizes, usuários e criadores de arte e conhecimento de ciência da computação, compreendam melhor o papel da computação no mundo ao seu redor e que possam aprender, atuar e se expressar em outros assuntos e interesses (CSTA, 2016).

Nos Anos Iniciais devem ser trabalhados conceitos relacionados às estruturas abstratas necessárias à resolução de problemas no eixo de Pensamento Computacional. É importante que o aluno tome consciência do processo de resolução de problemas, e compreenda a importância de ser capaz de descrever a solução em forma de algoritmo (SBC, 2019). O aluno deve entender que a computação é uma experiência criativa e uma ferramenta para expressão pessoal, que serve como um meio para representar e resolver problemas, utilizá-la para aprimorar as experiências de aprendizado em outras disciplinas (CSTA, 2017).

Os alunos devem dominar as principais operações para a construção de algoritmos (composição sequencial, seleção e repetição) e ter noções de técnicas de decomposição de problemas. Além disso, espera-se que os estudantes reconheçam a necessidade de classificar objetos em conjuntos, sendo capazes de trabalhar com elementos destes conjuntos e identificar situações concretas nas quais dados atômicos ou estruturados possam ser utilizados. Nesta etapa, é essencial que os conceitos sejam dominados através de experiências concretas, que permitirão ao estudante construir modelos mentais para as abstrações computacionais, que serão formalizadas na próxima etapa do ensino fundamental com o uso de linguagens de programação (SBC, 2019).

Nos Anos Finais, espera-se que os estudantes sejam capazes de selecionar e utilizar modelos e representações adequadas para descrever informações e processos, bem como dominem as principais técnicas para construir soluções algorítmicas. Além disso, devem conseguir descrever as soluções, de forma que máquinas possam executar partes ou todo o algoritmo proposto; construir modelos computacionais de sistemas complexos e analisar criticamente os problemas e suas soluções (SBC, 2019). Observar oportunidades em sua comunidade e sociedade para aplicar a computação de maneiras novas e que os conceitos e práticas da ciência da computação capacitam o aluno a criar mudanças autênticas em pequena e grande escala (CSTA, 2017). Nesta etapa, deve ser adquirido também um entendimento de como informações podem ser armazenadas, protegidas e transmitidas, e da estrutura e funcionamento da internet, permitindo que o aluno tenha plena compreensão do Mundo Digital, suas potencialidades e seus limites (SBC, 2019).

Não são citados diretamente os termos *machine learning* ou *deep learning* nos currículos aqui citados, mesmo estes se enquadrando em tópicos dos mesmos.

A SBC cita o entendimento dos fundamentos da IA durante o ensino médio. (SBC, 2019)

2.2. *Machine Learning*

Machine learning (aprendizado da máquina) é um método de análise de dados que automatiza a construção de modelos analíticos. É um ramo da inteligência artificial baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana (SAS, 2019).

Os métodos de *machine learning* diferem nos tipos de dados de treinamento disponíveis, na ordem e no método pelo qual os dados de treinamento são recebidos e nos dados de teste usados para avaliar o algoritmo de aprendizado. Os principais tipos de aprendizado são (MOHRI, ROSTAMIZADEH, TALWALKAR, 2018):

- Aprendizado supervisionado: a partir de um conjunto de dados rotulados previamente definido, deseja-se encontrar uma função capaz de predizer rótulos desconhecidos. Este é o cenário mais comum associado a problemas de regressão e classificação.
- Aprendizado não supervisionado: a partir de um conjunto de dados não rotulados, deseja-se encontrar similaridades entre os objetos analisados a fim de detectar similaridades e anomalias. *Clustering* e redução de dimensionalidade são exemplos de problemas de aprendizado não supervisionados.
- Aprendizagem semi-supervisionada: a partir de um conjunto de dados rotulados e não rotulados e faz previsões para rótulos desconhecidos. Normalmente utilizada quando a quantidade de dados rotulados é pequena e o custo de rotular os dados sem rótulo é muito alto. Pode ser usado tanto em problemas de classificação, onde os dados rotulados são usados no processo de rotulação dos exemplos ou em *clustering* onde os dados rotulados auxiliam na formação dos clusters.
- Aprendizado por reforço: Para coletar informações, o modelo interage ativamente com o ambiente e, em alguns casos, afeta o ambiente, e recebe uma recompensa imediata por cada ação. Seu objetivo é maximizar sua recompensa ao longo de um curso de ações e interações com o ambiente.

Esses tipos de aprendizado, são aplicáveis também a um subgrupo de técnicas de *machine learning,* chamadas de *Deep Learning* (aprendizado profundo), que geralmente utilizam redes neurais profundas (DNN) e dependem de muitos dados para o treinamento. Existem alguns pontos importantes que diferem as técnicas clássicas de *machine learning* e *deep learning*, os principais são: a quantidade de dados, o poder computacional e a flexibilidade na modelagem dos problemas (SANTANA, 2018).

A maioria das indústrias que trabalha com grandes quantidades de dados tem reconhecido o valor da tecnologia de aprendizado de máquina. O setor bancário e outros negócios na indústria financeira usam tecnologias de *machine learning* para dois propósitos principais: prevenir fraudes, identificar oportunidades de investimento, ou ajudar investidores a saber quando fazer o *trade*. A segurança pública, onde pode ajudar na detecção de fraudes e na minimização de roubos de identidade. *Machine learning* é uma tendência crescente na assistência médica graças ao advento dos dispositivos *wearables* e sensores que permitem aos profissionais de saúde acessar os dados de pacientes em tempo real, que permite aos médicos analisar dados para identificar tendências ou alertas, levando ao aperfeiçoamento de diagnósticos e tratamentos. Websites que recomendam produtos e serviços com base em suas compras anteriores estão usando *machine learning* para analisar seu histórico de compras e promover outros itens pelos quais você pode se interessar. Analisar dados para identificar padrões e tendências é essencial para a indústria de transportes, a qual depende da elaboração de rotas mais eficientes e da previsão de problemas potenciais para aumentar a rentabilidade. (SAS, 2019)

As áreas que mais avançaram no estado da arte são:

- Visão computacional. Como reconhecimento de objetos, segmentação semântica, descrição de cenário, etc. A aplicação mais direta disso tudo em conjunto são os carros autônomos.
- Reconhecimento da fala e NLP (programação neurolinguística). É notável a melhoria nos assistentes pessoais como Cortana, Siri e Google Now.
- Assistência médica. Um capítulo à parte nos avanços alcançados com DL foi na área da saúde, especificamente no diagnóstico por imagens.
- Sistemas de recomendação, filmes, livros e músicas baseadas em comportamento. A Netflix, Amazon e o Spotify utilizam bastante em suas plataformas.
- Detecção de fraudes, muito comum nas Fintech como Nubank.
- Análise de sentimento em vídeo, texto ou imagem. Muito utilizado para monitoramento de marca em redes sociais (SANTANA, 2018)

Atualmente existem diversas ferramentas para uso de *machine learning* como: TensorFlow[10], Scikit-Learn[11], Weka[12], Dask[13], Caffe[14], MXNet[15], Theano[16],

[10] https://www.tensorflow.org
[11] https://scikit-learn.org
[12] https://www.cs.waikato.ac.nz/ml/weka/
[13] https://dask.org
[14] https://caffe.berkeleyvision.org
[15] https://mxnet.apache.org
[16] http://deeplearning.net/software/theano/

DeepLearning4j[17], Keras[18], Pytorch[19], Chainer[20], entre outras. No github[21] as três ferramentas mais populares, considerando o número de estrelas, atualmente são o TensorFlow, Keras e OpenCV[22] respectivamente.

2.3. *Deep Learning*

Métodos de *Deep Learning* são métodos de aprendizado de representação com vários níveis de representação, obtidos através da composição de módulos simples, conhecida como rede neural (NN), mas não lineares, que transformam a representação em um nível em uma representação em um nível mais alto e um pouco mais abstrato. Com composições suficientes, funções muito complexas podem ser aprendidas (LECUN, BENGIO, HINTON, 2015). O aprendizado é sobre encontrar pesos que fazem a NN exibir o comportamento desejado (SCHMIDHUBER, 2015).

O aprendizado de representação é um conjunto de métodos que permite que uma máquina seja alimentada com dados brutos e descubra automaticamente as representações necessárias para a detecção ou classificação (LECUN, BENGIO, HINTON, 2015).

Para tarefas de classificação, camadas mais altas de representação amplificam aspectos da entrada importantes para discriminação e suprimem variações irrelevantes. Uma imagem, por exemplo, vem na forma de uma matriz de valores de pixel, e os recursos aprendidos na primeira camada de representação geralmente representam a presença ou ausência de arestas em orientações e locais específicos da imagem. A segunda camada geralmente detecta motivos, identificando arranjos específicos de arestas, independentemente de pequenas variações nas posições das arestas. A terceira camada pode montar motivos em combinações maiores que correspondem a partes de objetos familiares, e as camadas subsequentes detectam objetos como combinações dessas partes. O aspecto principal do aprendizado profundo é que essas camadas de recursos não são projetadas por engenheiros humanos: elas são aprendidas com dados usando um procedimento de aprendizado de uso geral (LECUN, BENGIO, HINTON, 2015).

Existem diversos tipos de arquiteturas de NN dependendo do tipo de aprendizado e objetivo que se deseja alcançar. Por exemplo, a rede neural convolucional (CNN), é boa para o reconhecimento de imagens, processamento de vídeo e linguagem natural, outro tipo arquitetura é a *recurrent neural networks* (RNN) usadas para desenvolver outras arquiteturas de aprendizado profundo, uma

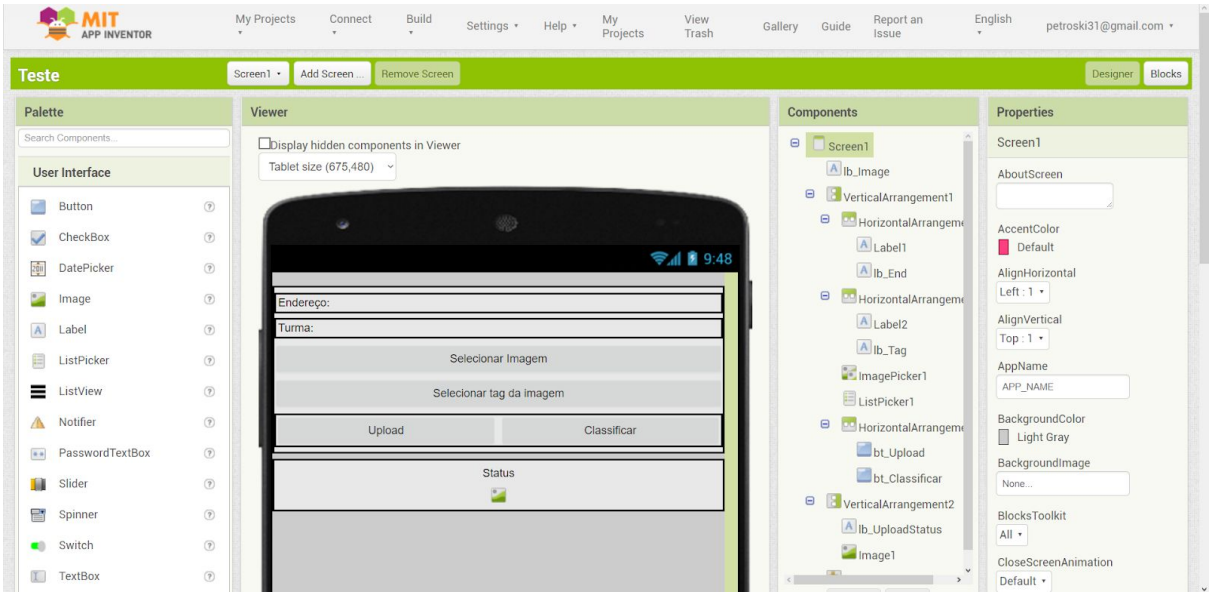
[17] https://deeplearning4j.org
[18] https://keras.io
[19] https://pytorch.org
[20] https://chainer.org
[21] https://github.com
[22] https://opencv.org

topologia popular é a long short-term memory (LSTM) que são boas para reconhecimento de discurso e caligrafia (JONES, 2017).

2.4. App Inventor

O App Inventor é um aplicativo web que usuários com experiência mínima ou até mesmo sem experiência em programação podem criar aplicativos móveis com facilidade e rapidez. Foi lançado pela primeira vez em 2010 e foi originalmente desenvolvido pelo Google; ele é mantido pelo MIT desde 2011. O App Inventor apresenta uma interface de usuário simples de arrastar e soltar para o design do layout, além de uma linguagem de programação baseada em blocos para permitir a interatividade (TANG, 2019).

Figura 1: Interface designer do App Inventor 1.



Fonte: elaborado pelo autor.

O site de desenvolvimento do App Inventor consiste em duas interfaces: a interface do designer e a interface dos blocos. A interface designer (Figura 1) contém vários componentes visíveis, como botões, imagens, rótulos, que definem o layout do aplicativo, além de componentes não visíveis, como câmeras e bancos de dados. Esses componentes estão localizados na paleta no lado esquerdo da interface. Cada componente possui propriedades que podem ser ajustadas no lado direito. Essas propriedades variam de acordo com o tipo de componente adicionado (TANG, 2019).

Os componentes visíveis são adicionados à janela de visualização do aplicativo, onde podem ser reposicionados ao gosto do usuário. Os componentes não visíveis são exibidos na parte inferior da janela de visualização para indicar que estão sendo usados. É possível que o aplicativo consista em várias telas; portanto, a segunda coluna à direita mostra todos os componentes organizados por tela. A

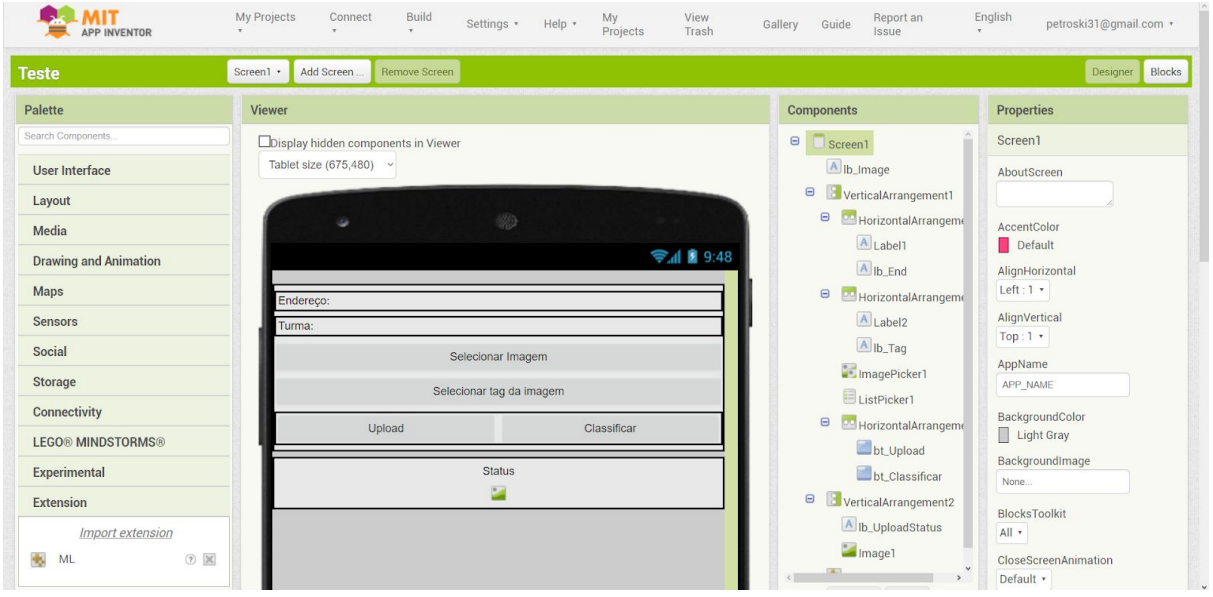
mídia também pode ser carregada (como imagens), que pode ser usada no aplicativo (TANG, 2019).

A interface dos blocos pode ser acessada clicando no botão "Blocos" no canto superior direito da interface do designer. Esses blocos definem a lógica do aplicativo e podem ser categorizados em dois grupos principais: blocos internos, por exemplo blocos de controle de fluxo e blocos lógicos, e blocos componentes (TANG, 2019).

Além disso, cada componente utilizado no designer também terá sua própria série de blocos. Geralmente, são métodos para alterar as propriedades dos componentes (por exemplo, o texto de um rótulo) ou eventos acionados pelo aplicativo (por exemplo, ao clique de um botão, executar uma série de blocos). Esses blocos formam um aplicativo que pode ser executado em um dispositivo móvel, utilizando o aplicativo App Inventor Companion no dispositivo móvel (TANG, 2019).

Existem duas maneiras principais de adicionar novos recursos ao App Inventor: adicionar um componente ou criar uma extensão. Para carregar uma extensão basta abrir a seção "Extensão" na parte inferior da paleta (Figura 2) e selecionar o arquivo de extensão apropriado em seu computador (TANG, 2019).

Figura 2: Interface designer do App Inventor 2.



Fonte: elaborado pelo autor.

Cada extensão é implementada como um componente não visível. Para carregar um em um aplicativo, é necessário primeiro adicionar um WebViewer (componente do App Inventor). O painel de propriedades da extensão mostra uma lista de WebViewers na tela e um deles deve ser escolhido para carregar o conteúdo da extensão (TANG, 2019).

3. Estado da Arte

Com o objetivo de levantar o estado da arte do ensino de programação com App Inventor e *Machine Learning* foi realizado um Mapeamento Sistemático da Literatura (MSL). Este mapeamento foi realizado de acordo com o processo proposto por Petersen et al. (2008).

3.1 Definição do protocolo do mapeamento

O objetivo deste MSL é encontrar experiências de utilização do App Inventor e *Machine Learning* no contexto do ensino da computação na educação básica e quais as suas características. Detalhando este objetivo são definidas cinco perguntas de análise:

PA1: Quais são as experiências de utilização de App Inventor com *Machine Learning*?

PA2: Quais métodos/técnicas de *Machine Learning* foram utilizadas?

PA3: Quais ferramentas são utilizadas e qual o contexto do uso dessas ferramentas? (público-alvo, ambiente, tamanho da amostra)

PA4: Quais são as tecnologias utilizadas? (frameworks, bibliotecas, ferramentas, sistemas operacionais, etc.)

PA5: Quais os resultados obtidos? (aprendizado, satisfação do usuário, o que foi medido como resultado)

Critérios de inclusão/exclusão. Somente são incluídos neste mapeamento da literatura trabalhos que tratam do ensino de programação utilizando o App Inventor ou similares e utilizando direta ou indiretamente *Machine Learning* conforme conceitos definidos na seção 2. São excluídos da pesquisa trabalhos relacionados à programação textual ou focados em outras áreas que não o apoio ao ensino de computação.

Para assegurar uma maior abrangência ao MSL, são buscados artigos científicos publicados em português e inglês. As fontes selecionadas para a realização da busca são:

- Google Scholar
- Repositório App Inventor

O Google Scholar foi utilizado por possuir uma maior abrangência em relação às bases de busca (HADDAWAY et al, 2015).

Para elaborar a string de busca da pesquisa são usados termos chave referentes ao objetivo deste mapeamento. Para minimizar o risco de exclusão de trabalhos relevantes foram também incluídos seus sinônimos e traduções para o inglês, conforme mostrado no Quadro 1.

Quadro 1: Lista de palavras-chave para busca

Palavra-chave	Sinônimos	Tradução
“App Inventor”		“App Inventor“
“Aprendizagem de máquina”	“Aprendizado profunda”, “Inteligência artificial”	“Deep learning”, “Machine learning”, “Artificial Intelligence”
Ensino	Educação, Aprendizagem	Education, Teaching, Learning, Schooling
Computação	“Ciência da computação”, Programação, “Pensamento computacional”	Computing, “Computer science”, programming, “Computational thinking”
“Educação básica”	“Ensino Básico”, “Ensino Fundamental”	“Primary school”, “Elementary school”, K12

Fonte: elaborado pelo autor.

Com base nas perguntas de pesquisa e palavras-chaves, foram feitas buscas informais para auxiliar na definição da string de busca. Após a calibração, definiu-se uma string de busca específica para aplicar nas bases de dados:

("App Inventor") AND ("Deep learning" OR "Machine learning" OR "Artificial Intelligence") AND (Education OR Teaching OR Learning)

3.2. Execução da busca

Neste mapeamento sistemático se esperava encontrar o estado da arte em relação ao ensino de *Machine Learning* com App Inventor. No entanto, uma aplicação prática não foi encontrada Machine Learning for Kids (2019), possivelmente por que não foram realizadas ainda publicações sobre a aplicação. Assim, a busca foi ampliada para o uso da ferramenta Google. Foram então analisadas as 10 primeiras páginas de resultados. A partir dessa busca a aplicação citada foi encontrada.

3.3. Extração dos dados

Para responder à questão de pesquisa, são extraídas informações relevantes às perguntas de análise conforme especificado no Quadro 2. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída, é indicada a falta desta informação como não informada (NI).

Quadro 2: Especificação das informações extraídas

Pergunta de análise	Dados a extrair	Descrição
PA1. Quais são as experiências de utilização de App Inventor com *Machine Learning*?	Nome	Autor(es)
	Referência	Referência bibliográfica
PA2. Quais métodos/técnicas de *Machine Learning* foram utilizadas?	Métodos ou técnicas	
PA3. Quais ferramentas são utilizadas e qual o contexto do uso dessas ferramentas?	Ambiente/contexto	App inventor, scratch, ...
	Público-alvo	Nível escolar ou idade
	Tamanho da amostra	Quantidade de participantes da avaliação
PA4. Quais são as tecnologias utilizadas?	Linguagens	Linguagem de programação utilizado
	Bibliotecas e Frameworks	Bibliotecas e frameworks utilizados
	Outras	Outras tecnologias utilizadas
PA5. Quais os resultados obtidos?	Método	Método de avaliação
	Resultados	Resultados obtidos

Fonte: elaborado pelo autor.

3.4. Análise dos dados

PA1: Quais são as experiências de utilização de App Inventor com *Machine Learning*?

No total foram encontrados 7 artefatos que apresentam alguma forma de experiência em utilização do App Inventor com *Machine Learning*, sendo cinco deles artigos científicos (Quadro 3) e outras duas aplicações encontradas (Quadro 4).

Quadro 3: PA01 - Artigos científicos

Citação	Referência bibliográfica
(ZHU, 2019)	ZHU, K. **An Educational Approach to Machine Learning with Mobile Applications**, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.
(TANG, 2019)	TANG, D., **Empowering Novices to Understand and Use Machine Learning With Personalized Image Classification Models, Intuitive Analysis Tools, and MIT App Inventor**, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.

(BRUMMELEN, 2019)	BRUMMELEN, J. V. **Tools to Create and Democratize Conversational Artificial Intelligence**, M.S. thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.
(ALIMISIS, LOUKATOS, 2018)	ALIMISIS, D.; LOUKATOS, D.; **STEM education post-graduate students' training in the eCraft2Learn ecosystem**, 2018
(SVANBERG, 2018)	SVANBERG, M. S. **Suggested Blocks: Using Neural Networks To Aid Novice Programmers In App Inventor**. 2018.

Fonte: elaborado pelo autor.

Quadro 4: PA01 - Outras aplicações encontradas

Citação	Referência bibliográfica
(MACHINE LEARNING FOR KIDS, 2019)	MACHINE LEARNING FOR KIDS, **Machine learning for kids worksheets**, Disponível em: <https://machinelearningforkids.co.uk/#!/worksheets> Acessado em: Set. de 2019.
(HACKERMOON, 2019)	HACKERMOON, **Your Kid Can Code a Fruit Detector with This MIT App Inventor AWS AI Services Extension**, Disponível em: <https://hackernoon.com/your-kid-can-code-a-fruit-detector-with-this-mit-app-inventor-aws-ai-services-extension-231665c8bcdc>. Acessado em: Set. de 2019.

Fonte: elaborado pelo autor.

PA2. Quais métodos/técnicas de *Machine Learning* foram utilizadas?

Quadro 5: PA02 - Métodos e técnicas de *machine learning.*

Citação	Ambiente/contexto
(ZHU, 2019)	Redes neurais convolucionais
(TANG, 2019)	Redes neurais convolucionais
(BRUMMELEN, 2019)	LSTM, RMSprop
(ALIMISIS, LOUKATOS, 2018)	NI
(SVANBERG, 2018)	DNN, FCNN, *Rectified Linear Unit*, CNN, *tree-based convolutional neural network*
(MACHINE LEARNING FOR KIDS, 2019)	NI
(HACKERMOON, 2019)	NI

Fonte: elaborado pelo autor.

Tang (2019) e Zhu (2019) não abordam muito as técnicas de *Machine Learning*, apenas citam a utilização de redes neurais convolucionais. Brummelen

(2019) utiliza LSTM (Long short-term memory) e RSMprop. Almisis e Loukatos (2018) não informam a utilização de nenhum método. E Svanberg (2018) utiliza diversos métodos como redes neurais convolucionais, redes neurais profundas, *Fully-convolutional neural network* (FCNN), *Rectified Linear Unit* e *tree-based convolutional neural network*.

PA3: Quais ferramentas são utilizadas e qual o contexto do uso dessas ferramentas?

Quadro 6: PA03 - Ferramentas e contexto de uso.

Citação	Ambiente/contexto	Público-alvo	Tamanho da amostra
(ZHU, 2019)	Oficina extraclasse de uma hora, slides e exercícios práticos em cada aula. App Inventor	Entre 14 e 17 anos	Total de aulas: 6 Quantidade de alunos: NI Participantes da avaliação: Entre 6 e 14 alunos por aula
(TANG, 2019)	Oficina extraclasse de 50 minutos. Aulas com slides e atividade prática em cada aula. App Inventor e Servidor Web	Entre 15 e 17 anos	Total de aulas: 2 Quantidade de alunos: 26 e 28 Participantes da avaliação: 8 e 15 alunos
(BRUMMELEN, 2019)	Oficina extraclasse de uma hora. App Inventor	Entre 9 e 12 anos	Total de aulas: 6 Quantidade de alunos: Entre 7 e 15 Participantes da avaliação: Entre 5 e 7
(ALIMISIS, LOUKATOS, 2018)	Oficina extraclasse de 6 horas. Utilizando o ecossistema eCraft2Learn. Scratch, Scratch4Arduino, toolSnap, Ardublock, Pocket code e App Inventor	Entre 13 e 17 anos	Total de aulas: 2 Quantidade de alunos: 11 e 17 Participantes da avaliação: NI
(SVANBERG, 2018)	App Inventor	NI	NI
(MACHINE LEARNING FOR KIDS, 2019)	App Inventor e Scratch	NI	NI
(HACKERMOON, 2019)	App Inventor	NI	NI

Fonte: elaborado pelo autor.

Zhu (2019), Tang (2019) e Brummelen (2019) utilizaram de abordagens similares, todos com oficinas extraclasse de aproximadamente uma hora e utilizando de pré e pós questionários para a avaliação. Almisis e Loukatos (2018) utilizaram de uma oficina mais longa com seis horas de duração, sua avaliação foi

feita em cima de um relatório da atividade feito pelos alunos fora da oficina. Todos os autores acima citados acima realizaram aula e exercícios práticos em suas oficinas.

PA4: Quais são as tecnologias utilizadas?

Quadro 7: PA04 - Tecnologias.

Citação	Linguagens	Bibliotecas/Frameworks	Outras Tecnologias
(ZHU, 2019)	Java, JavaScript	TensorFlow, Tesseract	MobileNet, Teachable Machine
(TANG, 2019)	CSS, HTML, Java, Javascript	TensorFlow	MobileNet, SqueezeNet
(BRUMMELEN, 2019)	Javascript, Python	AlexaSDK, CloudDB, JOVO, Keras, Redis, TensorFlow	Amazon Web Services, Alexa
(ALIMISIS, LOUKATOS, 2018)	NI	NI	ToolSnap!, Snap4Arduino
(SVANBERG, 2018)	Python, XML	Cuda, CuDNN, Keras, Numpy, Scikit-learn, TensorFlow	App Inventor summarizer program, Os (object serialization packages), Pickle
(MACHINE LEARNING FOR KIDS, 2019)	Python	NI	NI
(HACKERMOON, 2019)	NI	NI	Amazon Polly, Amazon Translate e Amazon Rekognition

Fonte: elaborado pelo autor.

Analisando as linguagens utilizadas, podemos ver uma maior utilização nas linguagens Python e Javascript e nas bibliotecas e frameworks o Tensorflow. Pode-se destacar uma grande variedade de outras tecnologias utilizadas entre todos os artigos avaliados.

PA5: Quais os resultados obtidos?

Zhu (2019), Tang (2019) e Brummelen (2019) afirmam, com base nos dados coletados pelos autores, que os alunos dos seus seminários conseguiram aprender os conceitos de *Machine Learning* mesmo sem conhecimento prévio em inteligência artificial. Alimissis e Loukatos (2018) não aplicaram diretamente os conceitos de *Machine Learning*, apenas citam que é possível utilizando o método proposto.

Svanberg (2018) não possibilita a utilização de mecanismos de *Machine Learning* para o usuário, seu modelo utiliza destes para sugerir blocos de programação.

3.4. Outras Ferramentas de Apoio ao Ensino de *Machine Learning*

Dentre as iniciativas encontradas, mas que não utilizam App Inventor, destacam-se aqui algumas dentre as mais relevantes, que atendiam os outros critérios de inclusão.

HACKERMOON (2019) utiliza o Thunkable Classic[23] e extensões para utilizar nos aplicativos os serviços de IA da Amazon, como o Amazon Lex, serviço de *chatbot*, o Amazon Polly, serviço de conversão de texto em fala, e o Amazon Rekognotion, serviço de tradução de idiomas.

Outra ferramenta encontrada é o MACHINE LEARNING FOR KIDS (2019) que permite treinar sua rede neural e importar uma extensão com sua rede treinada ao Scratch, App Inventor ou em Python. Seu site possui vários projetos voltados ao ensino com diferentes propósitos e com materiais diferentes para professores e alunos.

Existem diversas ferramentas que podem ser usadas de apoio ao ensino de *machine learning*, no Quadro 6 foram algumas das ferramentas encontradas durante a pesquisa além do App Inventor, sendo assim não se encaixando no critério de aceite do mapeamento sistemático mas que valem a pena serem citadas.

Quadro 8: Outras ferramentas de apoio ao ensino de *machine learning.*

AI for Kids	https://www.ai4children.org
Scratch	https://scratch.mit.edu
Orange Biolab	https://orange.biolab.si/
Machine Learning for Kids	https://machinelearningforkids.co.uk
Cognimates	http://cognimates.me/home/
Pencil Code	https://pencilcode.net
Apps For Good	https://www.appsforgood.org/courses/machine-learning
Amazon Web Services (Alexa, NLP tools, DeepLens, Amazon Polly, Amazon Translate e Amazon Rekognition)	https://aws.amazon.com/pt/
Experiments with Google	https://experiments.withgoogle.com/collection/ai

[23] http://app.thunkable.com

App Inventor Spin-outs (Thunkable, AppyBuilder, Kodular, DroidMaker, Hybro Studio)	

Fonte: elaborado pelo autor.

3.5. Discussão

Após o mapeamento sistemático pode-se verificar que existem diversas ferramentas distintas que podem ser utilizadas para o ensino de *Machine Learning*. Entretanto, dentre essas, poucos artigos foram encontrados especificamente utilizando App Inventor.

Dentre os artigos encontrados, a maioria se utilizou de questionários para fazer a validação, e em todos eles a amostra que respondeu os questionários era pequena e que no melhor dos casos correspondia aproximadamente 50% dos participantes, o que pode ameaçar sua validade.

Dos quatro artigos que utilizam App Inventor, dois são similares ao proposto neste trabalho, e um deles agrega mais a possibilidade de treinar sua rede neural. Entre todos os artigos que utilizam o App Inventor, todos desenvolveram uma extensão para conseguir utilizar *Machine Learning* em seus projetos, o que demonstra a necessidade da implementação de uma extensão mais genérica que permita a utilização de *Machine Learning* de maneira mais fácil.

3.5.1. Ameaças à validade do mapeamento

Como em qualquer mapeamento sistemático, existem potenciais ameaças à validade dos resultados. Foram identificadas ameaças potenciais e aplicadas estratégias de mitigação para minimizar os impactos.

A fim de mitigar o risco da omissão de estudos relevantes, a string de busca foi construída para ser abrangente considerando não apenas os principais conceitos, mas também sinônimos, mas muitos resultados irrelevantes foram encontrados com os termos inicialmente pesquisados como “Scratch”, por isso foi limitado ao App Inventor. Tentou-se minimizar o risco de não encontrar trabalhos relevantes deixando mais abrangentes os mecanismos de busca, no caso o Google, Google Scholar e o repositório do App Inventor.

Mapeamentos sistemáticos podem sofrer do viés comum de que os resultados positivos têm maior probabilidade de serem publicados do que os negativos. No entanto, os resultados dos artigos encontrados possuem pouca influência sobre esse mapeamento sistemático, uma vez que foi buscado experiências de aplicações do uso do App Inventor no ensino de *machine learning*.

4. Análise e Projeto

Neste capítulo é apresentada a proposta de solução para melhorar o ensino de *machine learning* com App Inventor.

4.1. Análise de requisitos

Nesta seção são apresentados os requisitos de software, que por definição da ISO/IEC 12207 consiste em uma declaração que traduz ou expressa uma necessidade e suas restrições e condições associadas. Estes requisitos são separados em requisitos funcionais, que são declarações que identificam quais resultados um produto ou processo deve produzir (ISO/IEC/IEEE 24765f, 2016) e não funcionais, que descrevem não o que o software fará, mas como o software fará (ISO/IEC/IEEE 24765f, 2017).

Os requisitos são levantados por meio da análise dos resultados do estado da arte e de entrevistas não estruturadas com um especialista de domínio. Na sequência são apresentados os requisitos funcionais e não-funcionais identificados.

4.1.1. Requisitos Funcionais

A extensão possui requisitos funcionais listados abaixo:

Quadro 9: Requisitos funcionais da extensão

ID	Requisito	Descrição
RF01.01	Enviar requisição de classificação	Enviar ao servidor a requisição de classificação de uma entrada para um modelo enviando uma imagem ou texto.
RF01.02	Receber requisição de classificação	Receber resposta do servidor da requisição de classificação enviada.
RF01.03	Enviar requisição de lista de modelos.	Enviar ao servidor a requisição de lista de modelos treinados do servidor e responder ao cliente o que foi solicitado.
RF01.04	Receber requisição de lista de modelos.	Receber resposta do servidor da requisição de lista de modelos enviada.
RF01.05	Enviar requisição de lista de *labels* de um modelo.	Enviar ao servidor a requisição da lista de *labels* de um determinado modelo e responder ao cliente o que foi solicitado.
RF01.06	Receber requisição de lista de *labels* de um modelo.	Receber resposta do servidor da requisição de lista de *labels* enviada.
RF01.07	Configurar endereço	Configurar Ip e Porta de envio das requisições

Fonte: elaborado pelo autor.

O servidor possui os requisitos funcionais listados abaixo:

Quadro 10: Requisitos funcionais do servidor

ID	Requisito	Descrição
RF02.01	Receber requisição de classificação	Receber a requisição de classificação de uma entrada para um modelo e responder ao cliente com a *label* de classificação, probabilidade ou uma mensagem de erro.
RF02.02	Receber lista de modelos.	Receber a requisição de lista de modelos treinados do servidor e responder ao cliente o que foi solicitado.
RF02.03	Receber lista de *labels* de um modelo.	Receber a requisição da lista de labels de um determinado modelo e responder ao cliente o que foi solicitado.
RF02.04	Classificar uma entrada com base em um modelo	Classificar uma entrada de acordo com um modelo pré-treinado existente.
RF02.05	Exibir modelos disponíveis	Exibir os modelos pré-treinados disponíveis com uma breve descrição e *labels* existentes (web)
RF02.06	Login	Efetuar login
RF02.07	Inserir novos modelos	Manter modelos pré-treinados

Fonte: elaborado pelo autor.

4.1.2. Requisitos Não-Funcionais

Os requisitos não-funcionais com IDs iniciados com RNF01 pertencem a extensão e os IDs iniciados com RNF02 pertencem ao servidor.

Quadro 11: Requisitos não-funcionais

ID	Requisito	Descrição
RNF1.1	Linguagem de programação Java	A ferramenta deve ser desenvolvida na linguagem de programação Java compatível com a versão 7, pois o projeto existente que está continuado é Java versão 7.
RNF1.2	Usabilidade	Eficácia: 90% dos usuários da avaliação devem conseguir completar a tarefa de classificar uma entrada para um modelo.
RNF1.3	Extensibilidade	O sistema deve permitir que novos modelos pré-treinados sejam adicionados.
RNF1.4	Tamanho de arquivo da requisição	O sistema deve permitir o envio de arquivos em requisições de até 16 MB,

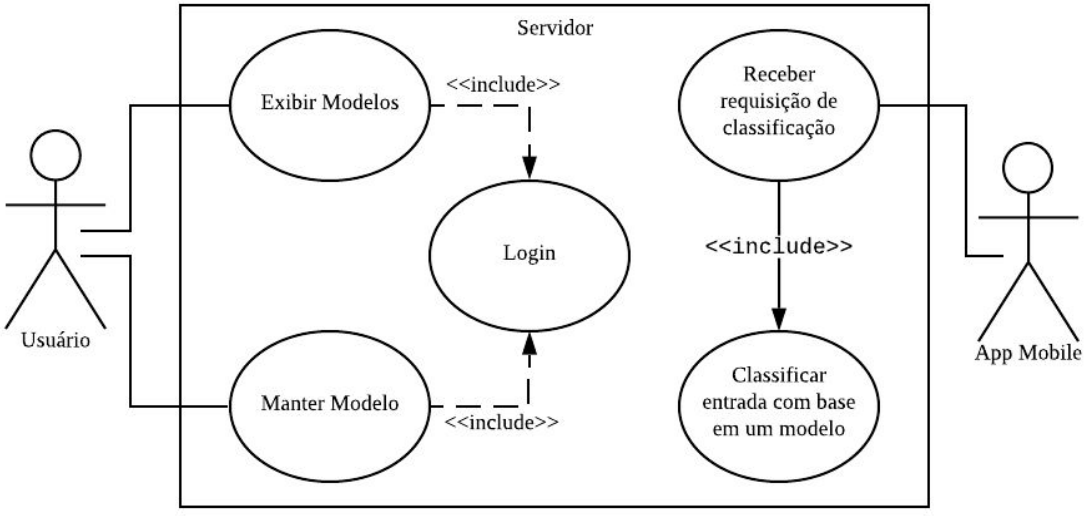
RNF 1.5	Compatibilidade com App Inventor	A extensão deve ser compatível com a versão 2.41 do App Inventor
RNF 2.1	Servidor Web	O servidor deve ser acessado via navegador Web com conexão à internet. Navegadores compatíveis: Google Chrome versão 77.
RNF 2.2	Linguagem de programação Javascript, HTML, CSS	A ferramenta deve ser desenvolvida nas linguagens de programação Javascript, HTML5 e CSS3 pois são linguagens muito conhecidas facilitando a manutenção do sistema.
RNF 2.3	Os métodos de classificação assíncronos	Os métodos de classificação devem ser assíncronos.
RNF 2.4	Tempo de resposta	O tempo de resposta do servidor deve ser de até 5 segundos.

Fonte: elaborado pelo autor.

4.2. Casos de uso

Nesta seção são apresentados os casos de uso identificados e seus principais fluxos de execução, considerando os requisitos previamente levantados. Casos de uso são narrativas em texto, descrevendo a unidade funcional, e são amplamente utilizados para representar requisitos funcionais nos sistemas (VAZQUEZ; SIMÕES, 2016).

Figura 3: Diagrama de casos de uso



Fonte: elaborado pelo autor.

Existem elementos computacionais e usuários que interagem com o sistema e são parte fundamental da aplicação como um todo, esses elementos são os atores do sistema.

[A01] Usuário – Acessa e mantém os modelos do servidor.

[A02] App Mobile – Utiliza os recursos do servidor.

Caso de Uso 1 – UC02.01 - Login

Atores: ● Usuário
Fluxo base: 1. Usuário acessa o site do servidor. 2. Usuário digita suas credenciais. 3. O sistema autentica o usuário e é redirecionado à página principal;
Fluxo de Exceção: FE01 – Sistema não encontra credenciais ou credenciais inválidas. Sistema informa com a mensagem de erro: “Usuário e senha não identificados pelo sistema”.

Caso de Uso 2 – UC02.02 - Exibir modelos

Atores: ● Usuário
Pré-condição: ● UC02.01
Fluxo base : 1. Sistema exibe lista dos modelos disponíveis, exibindo nome, descrição, labels e as opções de excluir e editar cada modelo.
Fluxo de Exceção: FE01 – Nenhum modelo previamente cadastrado. Sistema informa uma mensagem: “Nenhum modelo cadastrado.”.

Caso de Uso 3 – UC02.03 - Manter modelos

Atores: ● Usuário
Pré-condição: ● UC02.01
Fluxo base - Cadastrar : 1. Usuário clica no botão cadastrar. 2. Sistema redireciona usuário para página de cadastro. 3. Usuário informa nome, descrição, labels e importa modelo.

4. Usuário clica em salvar. 5. Sistema redireciona usuário para página principal. Fluxo base - Excluir: 1. Usuário clica no botão excluir. 2. Usuário confirma a exclusão. 3. Sistema atualiza a tela principal. Fluxo base - Editar: 1. Usuário clica no botão editar. 2. Sistema redireciona usuário para página de edição. 3. Usuário altera as informações. 4. Usuário clica em salvar / cancelar. 5. Sistema redireciona usuário para página principal.
Fluxo de Exceção: FE01 – Nome já existente, sistema apresenta a mensagem “Já existe um modelo com este nome”. FE02 – Erro ao importar modelo, sistema apresenta a mensagem “Erro ao importar modelo”.

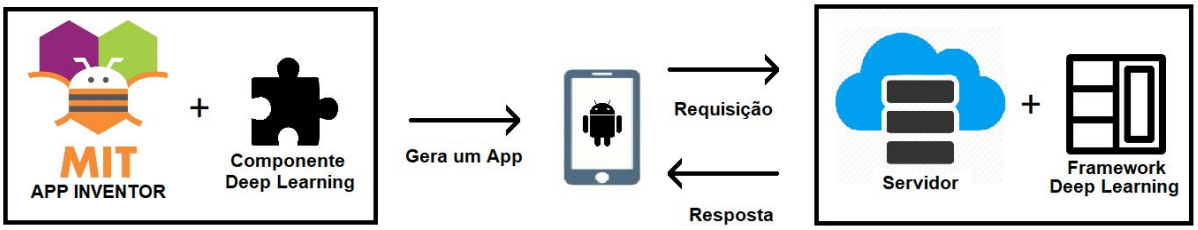
Caso de Uso 4 – UC02.04 - Receber requisição de classificação

Atores: ● App Mobile
Fluxo base: 1. App Mobile envia requisição de classificação ao servidor. 2. Sistema executa classificação de acordo com o modelo solicitado. 3. Sistema envia ao App Mobile a requisição com a resposta da classificação.
Fluxo de Exceção: FE01 – Tipo de arquivo incorreto: Sistema envia ao App Mobile, uma resposta de erro. FE02 – Modelo inexistente: Sistema envia ao App Mobile, uma resposta de erro. FE03 – Erro na classificação: Sistema envia ao App Mobile, uma resposta de erro.

4.3. Visão geral da arquitetura

Para possibilitar o atendimento dos requisitos funcionais e não-funcionais, é definido um modelo conceitual da arquitetura dividido em dois módulos principais: a extensão do App Inventor e o Servidor Web, conforme ilustrado na Figura 4.

Figura 4: Ilustração da solução proposta

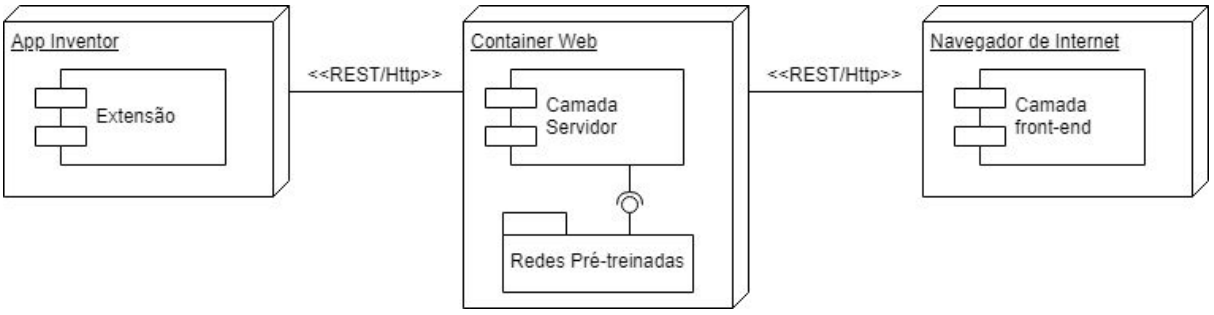


Fonte: elaborado pelo autor.

Assim, quando o componente do aplicativo fizer uma requisição ao servidor, o servidor processa a requisição e retorna a resposta ao aplicativo, no caso de uma classificação de imagem, por exemplo, o servidor executa a classificação para um modelo específico e responde ao aplicativo qual é a label e sua probabilidade de acerto.

Visando uma aplicação escalável a longo prazo, principalmente no lado do servidor, foi definido o modelo de arquitetura do sistema. Os dois módulos do sistema são divididos em três nodos (Figura 5), um container da Extensão do App Inventor, um container web para o servidor com um pacote de redes pré-treinadas, e o navegador de internet do usuário onde pode ser acessadas informações sobre a API do servidor, como por exemplo as redes pré-treinadas.

Figura 5: Diagrama de componentes da solução proposta



Fonte: elaborado pelo autor.

Servidor

O servidor é uma API restful que é responsável por atender às requisições da extensão e executar as predições dos modelos definidos. O resultado da requisição contém o mapeamento da predição do modelo para as labels ou erro no caso de erro. Ele também responde a extensão para listar os modelos definidos e suas respectivas *labels*. Além disso, o servidor também possui uma aplicação Web para que seus usuários possam manter os modelos previamente treinados.

Extensão do App Inventor

A extensão executa os blocos quando chamados, realizando as requisições ao servidor e aguardando a resposta do mesmo, para isso utiliza os seguintes blocos:

Quadro 12: Tipos de blocos do App Inventor

Tipo do Bloco	Descrição
Blocos de evento (Marrom)	Executa ao receber a resposta de uma requisição associada.
Blocos de Execução (Roxo)	Executa a requisição associada.
Blocos de Input (Verde Escuro)	Usado para definir atributos da extensão.

Fonte: elaborado pelo autor.

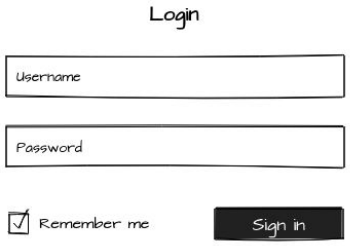
4.4. Protótipos de tela

Nesta seção são apresentados os protótipos de tela do módulo Servidor Web, separados pelos casos de uso previamente definidos.

Caso de Uso 1 – UC02.01 - Login

Nesta tela (Figura 6) o usuário fará *login* no servidor, o que permitirá acesso às funcionalidades de listagem e manutenção dos modelos.

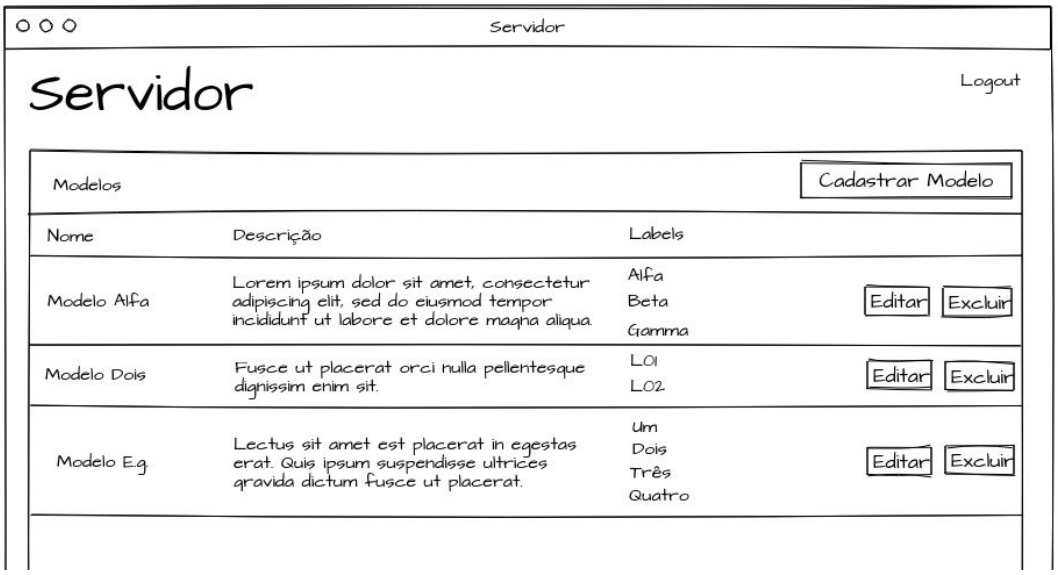
Figura 6: Protótipo de tela - Login



Fonte: elaborado pelo autor.

Caso de Uso 2 – UC02.02 - Exibir e Manter modelos

Figura 7: Protótipo de tela - Lista de modelos

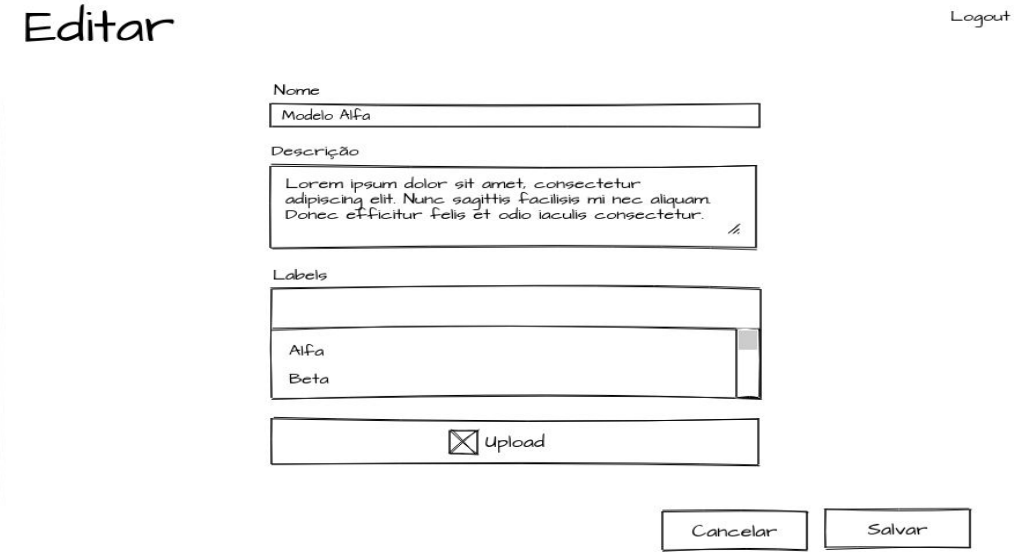


Fonte: elaborado pelo autor.

Nesta tela o usuário possui acesso a várias funcionalidades como cadastro de novos modelos, edição ou exclusão de modelos, a listagem dos modelos disponíveis, mostrando nome, descrição e labels de cada modelo, e a opção de *logout* do sistema.

Caso de Uso 3 – UC02.03 - Editar/Cadastrar modelos

Figura 8: Protótipo de tela - Editar modelo



Fonte: elaborado pelo autor.

Nesta tela o usuário pode editar as informações do modelo selecionado e/ou refazer o *upload* do modelo, clicando em salvar após alterar as informações

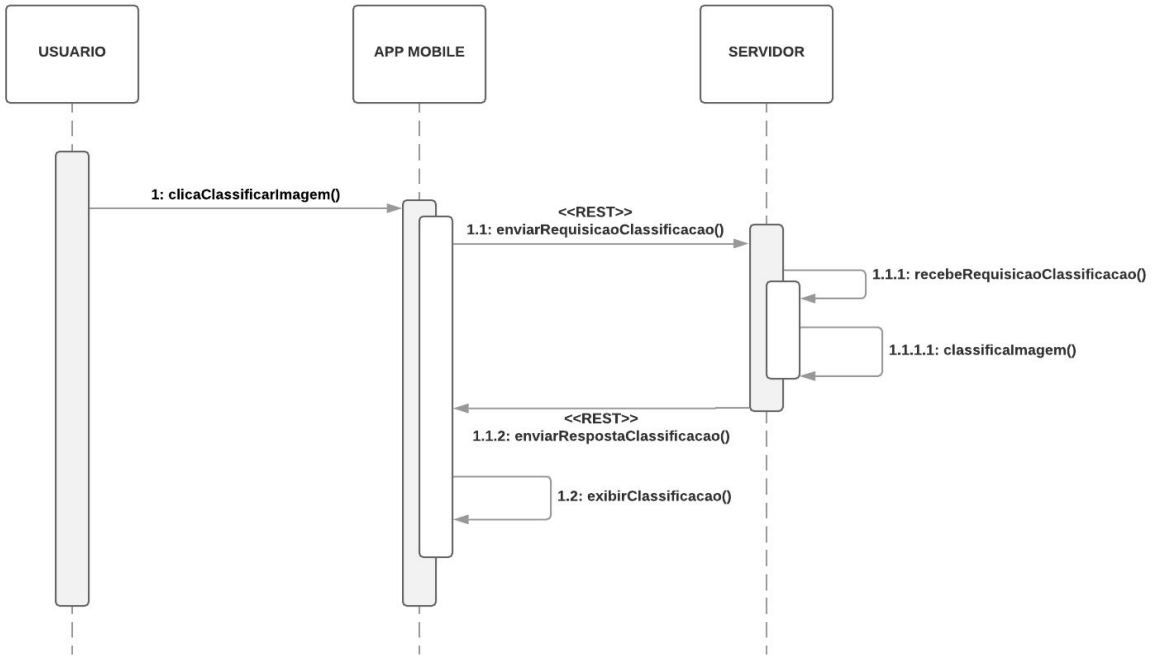
desejadas. O usuário também tem as opções de cancelar e *logout* do sistema. A tela de cadastro possui três diferenças para a tela de edição:

1. Título Editar vira Cadastrar;
2. Botão de Salvar vira Cadastrar;
3. e todos os campos vem em branco ao abrir a tela;

4.5. Diagrama de Sequência

Por fim, para ilustrar a comunicação do componente junto ao servidor da aplicação apresentada na Figura 4, foi criado um diagrama de sequência. A figura abaixo (Figura 9), apresenta o diagrama de sequência das trocas de mensagens para o fluxo base do caso de uso UC02.04, receber requisição de classificação, que faz a comunicação entre servidor e a extensão utilizando REST (via protocolo HTTP).

Figura 9: Diagrama de sequência do fluxo base do UC02.04



Fonte: elaborado pelo autor.

5. Desenvolvimento

Este capítulo tem como objetivo apresentar as principais decisões técnicas e detalhar as principais dificuldades encontradas durante o desenvolvimento da ferramenta e como foram solucionadas e apresentando as funcionalidades implementadas.

5.1. Tecnologias Utilizadas

Diversas tecnologias e ferramentas são utilizadas para a implementação da extensão App Inventor e do Servidor. Esta seção expõe as principais tecnologias aplicadas no desenvolvimento que foram escolhidas a partir de pesquisa do autor deste trabalho.

Quadro 13: Detalhamento das tecnologias

Tecnologia	Descrição	Referência
Python	Linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.	*https://www.python.org/*
Fast.ai	Biblioteca Python de código aberto que simplifica o treinamento de redes neurais. É construído sobre o PyTorch.	*https://www.fast.ai/*
Flask	Microframework multiplataforma que provê um modelo simples para o desenvolvimento web.	*https://flask.palletsprojects.com/en/1.1.x/*
HTML	Linguagem de marcação utilizada na construção de páginas na Web.	*https://www.w3.org/html/*
CSS	É um mecanismo para adicionar estilos a um documento Web.	*https://www.w3.org/Style/CSS/*
Javascript	Linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da Web	*https://www.javascript.com/*
Bootstrap	Framework web para desenvolvimento de componentes de interface e front-end para sites e aplicações web.	*https://getbootstrap.com.br/*
Java	Linguagem de programação orientada a objetos, utilizada no desenvolvimento da extensão.	*https://www.java.com/pt_BR/*

Fonte: elaborado pelo autor.

5.2. Preparação do Ambiente de Desenvolvimento

O primeiro passo da implementação foi a preparação do ambiente de desenvolvimento. Foi iniciado com a preparação do ambiente de desenvolvimento da extensão, para tal, seguiu-se os passos de instalação e de pré-requisitos disponivels no site do App Inventor (How to build App Inventor from the MIT sources, https://docs.google.com/document/u/1/d/1Xc9yt02x3BRoq5m1PJHBr81OOv69rEBy8LVG_84j9jc/pub), o código fonte foi obtido do repositório Gitlab da UFSC, que é uma customização do software App Inventor realizada pela equipe da Computação na Escola/GQS/INCoD/INE/UFSC e foi escolhido o Eclipse (https://www.eclipse.org/) como IDE. Nessa etapa, um dos principais problemas encontrados foi a instalação do Java 7, que teve que ser instalado manualmente.

Para o servidor, iniciou-se com a instalação do Fast.ai, Flask, e das outras bibliotecas citadas no Quadro 13. Foram seguidos os passos de instalação do site do FastAi (https://docs.fast.ai/) e outros guias encontrados na internet a fim de contornar pequenos problemas encontrados. Como IDE foi escolhido o PyCharm (https://www.jetbrains.com/pt-br/pycharm/). Nessa etapa, um dos principais problemas encontrados foi a configuração do ambiente virtual de desenvolvimento (venv), que era necessária para a depuração do servidor.

5.3. Desenvolvimento

Conforme definido na seção 4.3, são desenvolvidos dois componentes distintos: (i) uma Extensão App Inventor que serve para possibilitar o desenvolvimento de aplicativos que utilizem *Machine Learning* e (ii) um Servidor, que permite a utilização de modelos pré-treinados de *Machine Learning*, o gerenciamento das turmas de alunos que irão utilizar o servidor, receber as imagens (re)treinar o modelo e classificar novas imagens recebidas.

5.3.1 Desenvolvimento da Extensão

A extensão é basicamente uma adaptação de um componente Web do App Inventor, contendo especializações do métodos para o fim desejado, como o envio de imagens e a obtenção do resultado da classificação. Durante o desenvolvimento da extensão, não foi encontrado uma forma de depurar a extensão em tempo de execução, o que atrapalhou o seu desenvolvimento, e assim tendo que ser testada durante a execução da extensão no App Inventor Companion (https://drive.google.com/file/d/1Eq7kpPDaKA-7VChk6RQMgkO13JHXRI9T/view?usp=sharing).

Tendo o componente Web do App Inventor como base, primeiramente foram adicionados os parâmetros de configuração da extensão e seus respectivos

métodos acessores, como por exemplo o endereço de servidor exibido no Quadro 14.

Quadro 14:: Atributo enderecoServidor e seus métodos acessores.

```
private String enderecoServidor = "";

@DesignerProperty(editorType = PropertyTypeConstants.PROPERTY_TYPE_STRING,
defaultValue = "")
@SimpleProperty
public void EnderecoServidor(String enderecoServidor) {
        if (enderecoServidor != null) {
                this.enderecoServidor = enderecoServidor.trim();
        }
}

@SimpleProperty
public String EnderecoServidor() {
        return this.enderecoServidor != null ? this.enderecoServidor : "";
}
```

Fonte: elaborado pelo autor.

A classe aninhada *CapturedProperties* também foi alterada para aceitar os novos atributos, o caminho da imagem, o endereço do servidor e a classe da imagem.

Para a execução das requisições, foi alterado o método *performRequest,* onde foi adicionado o código do Quadro 15, que adicionam ao *header* da requisição a extensão e a classe da imagem obtidos e mais abaixo um *If-else* para invocar o método de disparo do evento passado como parâmetro do performRequest.

Quadro 15: Alterações no método *performRequest*.

```
// dentro do performRequest
int index = this.imagePath.lastIndexOf('.');
String extension = index == -1 ? "" : this.imagePath.substring(index + 1);
if (extension != "") {
        connection.addRequestProperty("extension", extension);
        connection.addRequestProperty("tagImagem", tagImagem);
}

// ... continuação do código ...

if (method == UPLOAD) {
        // Dispatch the event.
        activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                        Upload(ML.getStatus(), ML.getError());
                }
        });
} else if (method == PREDICT) {
        // Dispatch the event.
        activity.runOnUiThread(new Runnable() {
                @Override
```

```
                public void run() {
                        Classify(ML.getPredict(), ML.getError());
                }
        });
} else if (method == CLASSIFICATIONS) {
        // Dispatch the event.
        activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                        Classifications(ML.getClasses(), ML.getError());
                }
        });
}
```

Fonte: elaborado pelo autor.

Para as requisições, com base nos métodos Get e PostFile do componente Web, foram escritos os métodos para executar a requisição (performRequest) e aguardar o retorno da mesma ou em caso de erro, disparar o evento de erro. Como exemplo dessa alteração, no Quadro 16, o método UploadImage e o método de disparo do evento *upload, que* contém a resposta da requisição invocada no performRequest.

Quadro 16: Método uploadImage e disparo do evento upload.

```
@SimpleFunction
public void UploadImage() {
        clearResponseStrings();
        this.urlString = this.enderecoServidor + "/upload/" + this.tagTurma;

        // Capture property values before running asynchronously.
        final CapturedProperties webProps = captPropertyValues(UPLOAD);
        if (webProps == null) {
                // already called form.dispatchErrorOccurredEvent
                return;
        }

        AsynchUtil.runAsynchronously(new Runnable() {
                @Override
                public void run() {
                        try {
                        performRequest(webProps, null, webProps.imagem, "POST", UPLOAD);
                } catch (PermissionException e) {
                        form.dispatchPermissionDeniedEvent(ML.this, UPLOAD, e);
                } catch (FileUtil.FileException e) {
                        form.dispatchErrorOccurredEvent(ML.this, UPLOAD,
e.getErrorMessageNumber());
                } catch (RequestTimeoutException e) {
                        form.dispatchErrorOccurredEvent(ML.this, UPLOAD,
ErrorMessages.ERROR_WEB_REQUEST_TIMED_OUT, webProps.urlString);
                } catch (Exception e) {
                        form.dispatchErrorOccurredEvent(ML.this, UPLOAD,
ErrorMessages.ERROR_WEB_UNABLE_TO_POST_OR_PUT_FILE, webProps.imagem,
webProps.urlString);
                 }
```

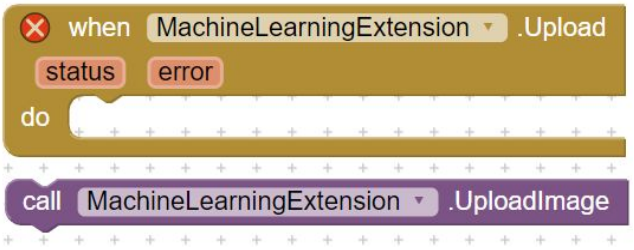
```
                    }
            });
}

// Disparo do evento upload
@SimpleEvent
public void Upload(String status, String error) {
            EventDispatcher.dispatchEvent(this, "Upload", status, error);
}
```

Fonte: elaborado pelo autor.

A notação *SimpleFuction* no método UploadImage, resulta em um bloco de procedimento, usado para invocar o método associado, e a notação SimpleEvent no método Upload, resulta em bloco de evento, usado para realizar um determinada ação quando a extensão invocar o método associado, nesse caso, o resultado da requisição do *upload*. Os blocos de procedimento UploadImage e evento Upload, dentro do App Inventor ficam como demonstra a Figura 10 e os resultados recebidos das requisições ao servidor são todos JSON, descritos no capítulo 5.3.2.1.

Figura 10: Bloco de procedimento UploadImage e bloco de evento upload



Fonte: elaborado pelo autor.

5.3.2 Desenvolvimento do Servidor

Para facilitar a implementação do servidor, os modelos e turmas, foi decidido salvá-los em arquivo nas pastas “models” e “turmas” respectivamente, seguindo a hierarquia de arquivos apresentada no Quadro 17.

Dentro da pasta de cada modelo, além do arquivo de serialização de objetos Python da biblioteca *pickle (https://docs.python.org/3/library/pickle.html*), usado para salvar o modelo já treinado. Nesta mesma pasta existe um arquivo chamado “classes.txt”, que contém um dicionário de dados usado na tradução das classes de cada modelo. Na aplicação ainda não existe uma interface gráfica para a adição de novos modelos, mas pode ser feita incluindo esses arquivos em uma nova pasta com o nome do modelo.

A pasta ‘*static’* contém os arquivos CSS, Javascript e imagens do formato *gif*, todos dentro de suas respectivas pastas, que são utilizados nas páginas HTML do servidor, que ficam dentro da pasta *templates*, seguindo o padrão de hierarquia de arquivos do Flask.

Quadro 17: Hierarquia de arquivos

```
../servidor
   /models
      /<nome_do_modelo>
         classes.txt
         <nome_do_modelo>.pkl
   /static
      /css
      /gif
      /js
   /templates
   /turmas
      /<codigo_da_turma>
         /upload/train
            /<classe>
         descricao.txt
         <nome_do_modelo>.pkl
   application.py
   model_service.py
   server.py
   turma_service.py
```

Fonte: elaborado pelo autor.

A pasta ‘turma’, contém uma cópia do arquivo pkl do modelo, o arquivo ‘descricao.txt’ contendo a descrição da turma criada durante o cadastro da turma e as imagens enviadas pela extensão, dentro da subpasta da classe a que foi submetida.

Para finalizar a parte de arquivos do servidor, sua implementação foi dividida em quatro arquivos:

- application.py que contém as configurações da aplicação Flask e sua execução inicia o servidor Web,
- server.py que contém as rotas do servidor,
- model_service.py que contém os métodos de modelos e,
- turma_service.py que contém os métodos de turmas.

Dentro do server.py, possui as principais chamadas de métodos do servidor, como o upload do arquivo (Quadro 18), onde é obtido o formato do arquivo e o código da turma do *header* da requisição, verifica-se se o código e o formato da extensão são válidos, caso não haja erro, é chamado o método *upload* do turma_server.py, onde é salvo o *array* de *bytes* no formato enviado dentro da pasta da turma. A classificação é feita de forma similar, executando o método *predict* no lugar do *upload*, onde é carregado o modelo da turma, chamada a classificação deste modelo para o *array* de *bytes* da imagem e retorna a classificação recebida do modelo.

Quadro 18: Método de upload

```
def upload_file(token : str) -> json:
```

```
if request.method == 'POST':
    extension =  request.headers.get('extension')
    tag = request.headers.get('tagImagem')
    error = verify(token, extension)
    if error == None :
        ts.upload(token, tag, request.get_data(), extension)
        return jsonify({"status":"ok"})
    else :
        return jsonify(error)
return jsonify({"error": "Not POST method"})
```

Fonte: elaborado pelo autor.

Outro método importante é o de retreinar o modelo, onde após carregar o modelo, é criado um novo *dataset* (Quadro 19) a partir da pasta de imagens da turma, utilizando o nome das pastas para categorizar as imagens e 20% das imagens são utilizadas para validação do treino. Em seguida, é definido esse dataset para o modelo e é efetuado o treinamento usando o comando *fit(n)* do Fast.ai realizando n ciclos de treino, sendo n, o valor do epoch enviado pelo usuário.

Quadro 19: Criação do novo dataset

```
data : ImageDataBunch = (ImageList.from_folder(path)
       .split_by_rand_pct().label_from_folder()
       .transform(tfms)
       .databunch(num_workers=0, bs=4))
```

Fonte: elaborado pelo autor.

Durante a implementação, a utilização de diversas tecnologias que nunca havia utilizado, como Fast.ai, Flask, Bootstrap, tornaram mais lento o progresso da implementação, isso foi amenizado devido a grande quantidade de tutoriais disponíveis na internet das tecnologias utilizadas.

Como a ferramenta utiliza o conceito de *Transfer Learning* (TORREY & SHAVLIK, 2010), ou seja, modelos pré-treinados são utilizados para serem re-treinados com novas imagens fornecidas pelo usuário, é necessário re-treinar o modelo após a inclusão de novas imagens. Isso gerou outro problema em relação ao tempo necessário para treinar o modelo, pois a proposta inicial era utilizar as imagens do dataset do modelo e acrescentar mais imagens para o re-treino, mas isso gerou um grande tempo execução, já que realizar um ciclo do *fit* com apenas as imagens do modelo demorava em torno de 25 minutos no computador do autor (Intel® Core™ i7-3770 CPU @ 3.40GHz × 4, 16GB de memória ram), tornando inviável para ser usado em sala de aula. Sendo assim, a proposta foi substituída pelo apresentado acima, onde o treinamento acrescenta mais camadas no modelo utilizando as imagens enviadas. Usando 60 imagens o tempo de execução diminuiu para próximo de 1 minuto por ciclo. O problema dessa abordagem, é que devido à baixa quantidade de imagens do cenário proposto de uso, provavelmente irá resultar

em uma taxa de acurácia menor do que o modelo inicial, dependendo da qualidade das imagens e da categorização fornecidas pelos usuários.

5.3.2.1 Serviços implementados no servdor

Nesta seção são apresentados os serviços implementados no servidor para manutenção de turmas, envio de imagens e para classificação de imagens.

O Quadro 20 apresenta os endpoints relacionados à extensão, com uma breve descrição, parâmetros, possíveis JSONs de saída e um exemplo da requisição em Curl.

Quadro 20: Tabela de *endpoints* para a extensão

Endpoint	**Descrição**	**Parâmetros**	**JSONs de saída**
Exemplo da requisição em Curl			
POST /upload/<token>	Realiza o *upload* de uma imagem com uma classe (tagImagem) para uma turma	URI: token (código da turma) header: extension (formato do arquivo) header: tagImagem (Classe da imagem) data-binary: <caminho do arquivo> (Imagem em binário)	{"status": "ok"} {"error":<descrição erro>}
curl --location --request POST 'http://192.168.0.10:5000/upload/a434fd' \ --header 'extension: jpg' \ --header 'Content-Type: text/plain' \ --data-binary '<path>/plastic14.jpg'			
POST /predict/<token>	Obtém a classificação da imagem pelo modelo da turma.	URI: token (código da turma) header: extension (formato do arquivo) data-binary: <caminho do arquivo> (Imagem em binário)	{"result": <classe>} {"error":<descrição erro>}
curl --location --request POST 'http://0.0.0.0:5000/predict/087aae \ --header 'extension: jpg' \ --header 'Content-Type: text/plain' \ --data-binary '<path>/ferro.jpg'			
GET /classificacoes/<token>	Obtém as classes do modelo de turma	URI: token (código da turma)	{"classes": <String das classes>} {"error": "invalid token."}
curl --location --request GET 'http://0.0.0.0:5000/classificacoes/a434fd'			

Fonte: elaborado pelo autor.

O Quadro 21 apresenta todos os outros endpoints disponíveis, utilizados no funcionamento do sistema web.

Quadro 21: Tabela de *endpoints* para a servidor

Endpoint	Descrição
/	redireciona o usuário para /turmas ou página de login
/login	usado para realizar o *login* do usuário
/logout	usado para realizar o *logout* do usuário
/cadastro_turma	usado para cadastrar uma turma
/editar_turma/<token>	usado para salvar a edição de uma turma
/turma/<token>/<tag>/<name>	retorna a imagem <name> de classe <tag> e turma <token>
/delete/<token>	usado para deletar uma imagem de uma turma
/turma/<token>	página da turma
/move_picture_to_tag	usado para mover imagens de uma turma entre classes
/turmas	página da listagem de turmas
/cadastrar	página de cadastro de turma
/editar	página da edição de turma
/excluir	usado para deletar uma turma
/clean	usado para deletar todas as imagens de uma turma
/train/<token>	usado para treinar o modelo de uma turma

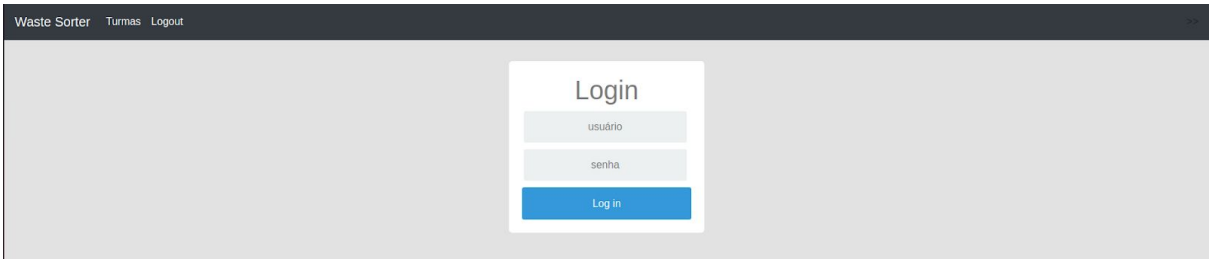
Fonte: elaborado pelo autor.

5.4. Operação do servidor

Nesta seção é descrito o funcionamento da aplicação, apresentando as funcionalidades implementadas no Servidor conforme seus casos de uso.

Ao iniciar a aplicação, a tela de login é apresentada (Figura 11), o usuário pode então informar o seu login e senha para poder se autenticar no sistema (UC02.01). Ao tentar acessar qualquer outra rota sem estar autenticado, o usuário é redirecionado a tela de login.

Figura 11: Tela de login



Fonte: elaborado pelo autor.

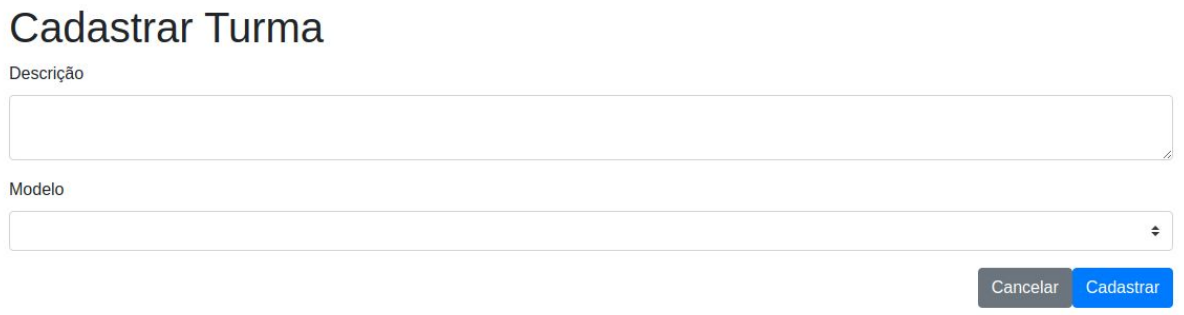
Após autenticação, o usuário é redirecionado a tela de turmas (Figura 12), a qual possui uma lista de turmas cadastradas (UC02.02A), um botão para cadastrar novas turmas e para cada turma apresenta um botão para visualizar, editar e excluir (UC02.03A).

Figura 12: Tela de listagem de turmas

Fonte: elaborado pelo autor.

Ao acessar o botão cadastrar, o usuário é redirecionado para tela de cadastro de turma (Figura 13). Esta tela permite o cadastro de uma nova turma, ela apresenta um campo texto descrição e um campo para selecionar o modelo, onde seus valores são obtidos automaticamente a partir dos modelos pré-cadastrados disponíveis na pasta ‘model’. Ao efetuar o cadastro, o código da turma é gerado e o usuário é redirecionado a tela de turmas.

FIgura 13: Tela de cadastro de turma



Fonte: elaborado pelo autor.

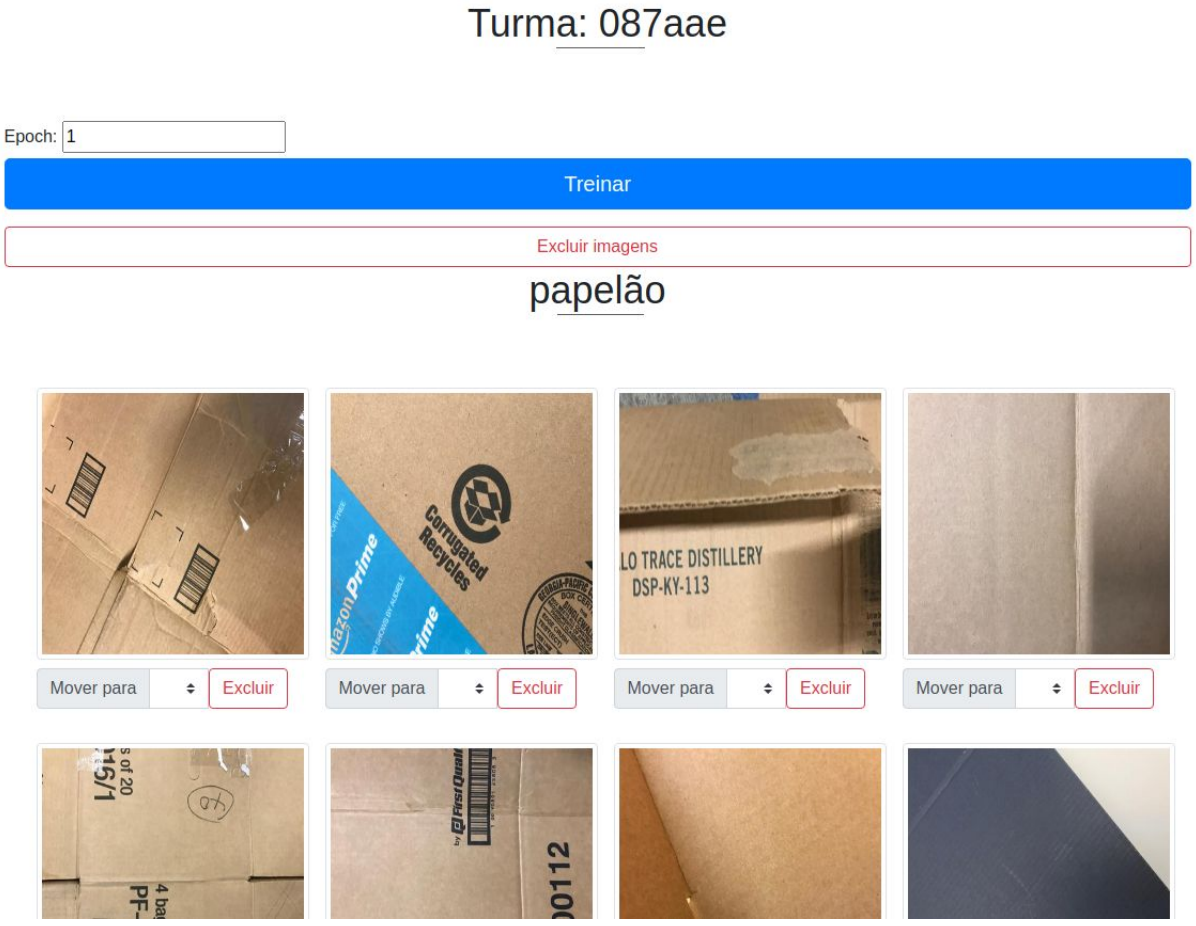
A tela de edição de turma (Figura 14), é similar a de cadastro, apenas trocando o título, onde é apresentado “Editar” no lugar de “Cadastrar” e o código da turma e o botão “Cadastrar” virá “Salvar”.

Figura 14: Tela de edição de turma

Fonte: elaborado pelo autor.

A tela de visualização de turma (Figura 15), apresenta ao usuário as configurações de treino (Epoch), a opção de treinar, a opção de excluir todas as imagens, e as imagens separadas por classe. Cada imagem possui as opções de exclusão e de mover para outra classe, caso uma imagem esteja em uma classe errada. As imagens apresentadas na tela de visualização, são as imagens obtidas das requisições de *upload* para cada turma. Ao clicar no botão “Treinar”, o sistema mostra ao usuário “aguarde o treinando” enquanto a função de treino ocorre, utilizando as imagens categorizadas nas classes apresentadas nesta tela para o re-treinar o modelo de turma.

FIgura 15: Tela de visualização de turma

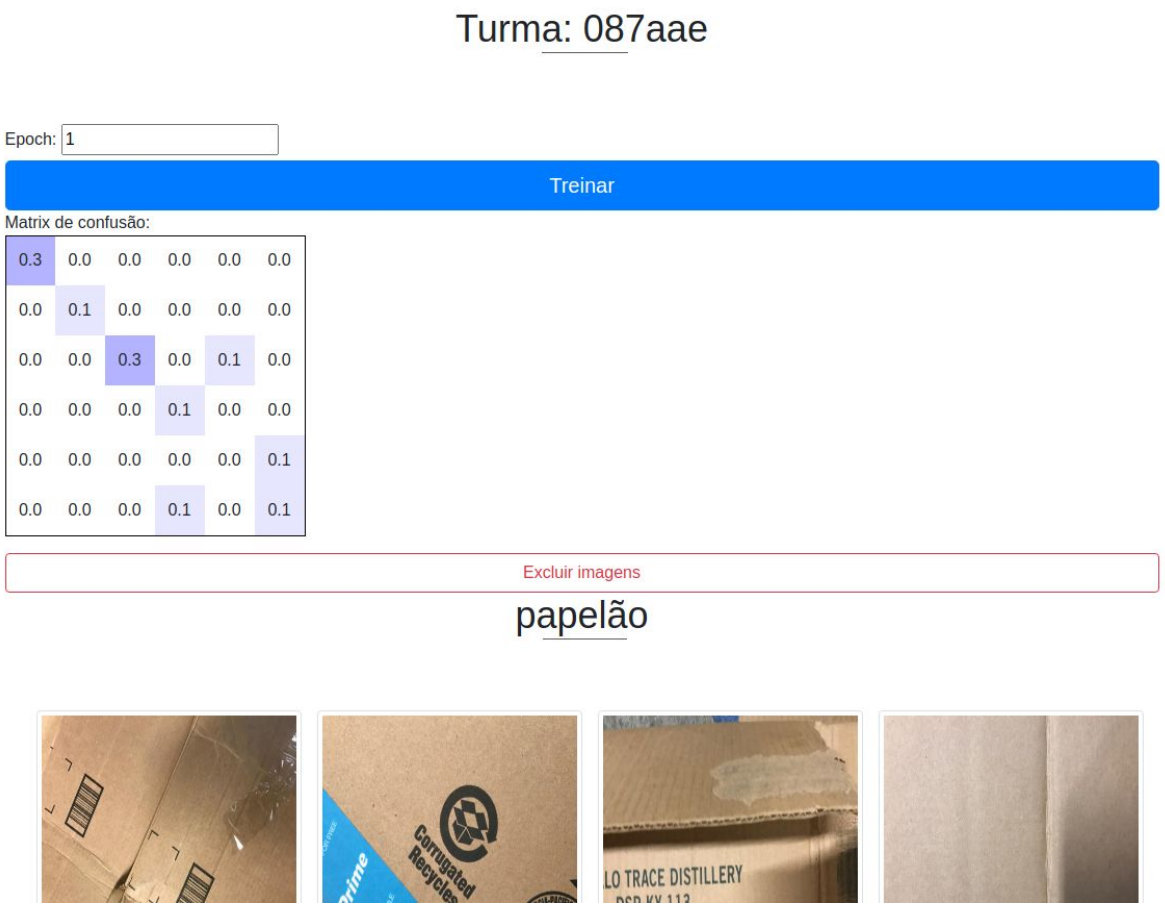


Fonte: elaborado pelo autor.

Após o (re)treinamento do modelo, a tela de visualização de turma mostra a matriz de confusão gerada pelo treino realizado, como mostra a Figura 16. Como uma das funcionalidades a serem desenvolvidas, a representação visual matriz de confusão ainda está incompleta, não mostrando as classes nos eixos X e Y e não apresentando a legenda das cores.

Em qualquer página da aplicação, o menu superior é apresentado possibilitando ao usuário retornar a página de listagem das turmas ou deslogar.

Figura 16: Tela de visualização de turma após treino



Fonte: elaborado pelo autor.

5.5. Discussão

Durante a implementação, alguns requisitos funcionais já não estavam coerentes devido ao rumo que o projeto tomou. Em reuniões com os *stakeholders*, foi constatado que era mais relevante para o usuário cadastrar as turmas de alunos que iriam utilizar o sistema do que manter o cadastro dos modelos de ML que estava inicialmente previsto. Então, a partir disso, algumas mudanças nos requisitos iniciais foram realizadas conforme o Quadro 22. Um ponto importante de ser levantado, para não precisar alterar todos os requisitos, é que ao citar ‘modelo’

refere-se ao modelo da turma, sendo a única exceção o RF02.07 (Inserir novos modelos Manter modelos pré-treinados).

Assim, os requisitos RF01.03, RF01.04 e RF02.02 foram removidos por não fazerem mais sentido quando analisado o objetivo do usuário, que deseja enviar as requisições para o modelo de uma turma específica de alunos, logo ele deve ter o código da turma para efetuar as requisições, não sendo necessário realizar uma requisição para tal. O RF01.01 foi simplificado e o RF02.07 não foi desenvolvido, por causa do curto período de desenvolvimento. E para contemplar as outras alterações, foram criados outros requisitos.

Considerando essas alterações nos requisitos, todos os requisitos foram alcançados, com exceção do RF02.07 e RNF1.2 que não foi avaliado (complementado no capítulo 6).

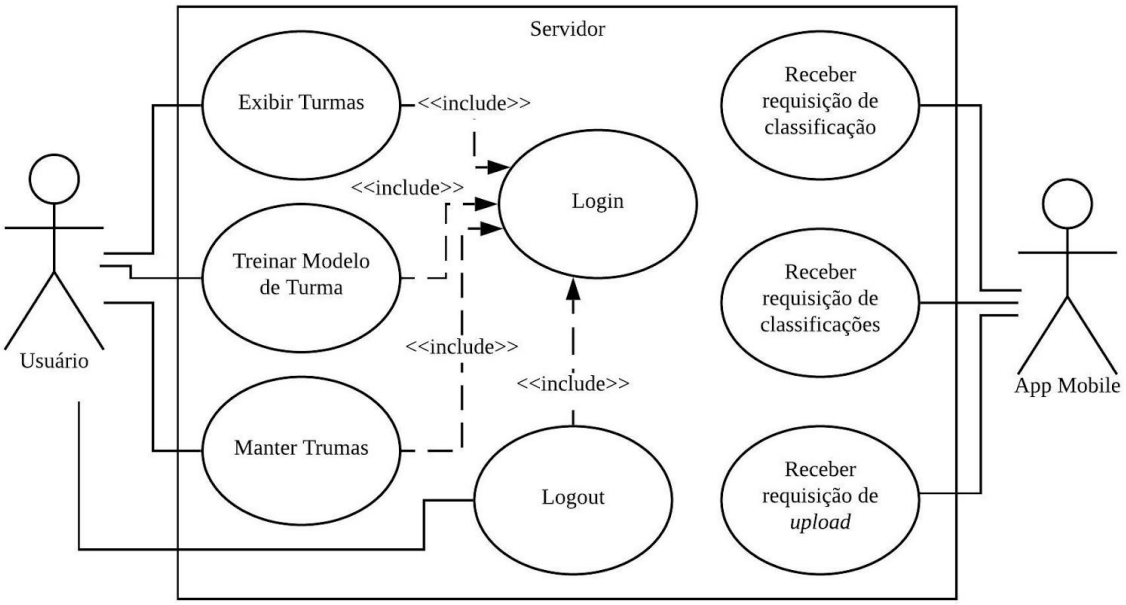
Quadro 22: Alterações requisitos

ID	Requisito	Descrição	Mudança
RF 01. 01	Enviar requisição de classificação	Enviar ao servidor a requisição de classificação de uma entrada para um modelo enviando uma imagem ou texto.	Alterado para apenas imagens.
RF 01. 03	Enviar requisição de lista de modelos.	Enviar ao servidor a requisição de lista de modelos treinados do servidor e responder ao cliente o que foi solicitado.	Requisito removido.
RF 01. 04	Receber requisição de lista de modelos.	Receber resposta do servidor da requisição de lista de modelos enviada.	Requisito removido.
RF 02. 02	Receber lista de modelos.	Receber a requisição de lista de modelos treinados do servidor e responder ao cliente o que foi solicitado.	Requisito removido.
RF 01. 08	Enviar requisição de *upload*	Enviar ao servidor a requisição de *upload* de imagem para uma turma.	Novo requisito
RF 02. 08	Inserir novas turmas	Manter turmas.	Novo requisito
RF 02. 09	Treinar turmas	Possibilitar o re-treinamento do modelo da turma.	Novo requisito
RF 02. 10	Apresentar resultado do treinamento	Apresentar feedback ao usuário após término do treinamento	Novo requisito

Fonte: elaborado pelo autor.

Com os novos requisitos funcionais fez-se necessário refazer o diagrama de casos de uso, como mostrado na Figura 17.

Figura 17: Novo diagrama de casos de uso



Fonte: elaborado pelo autor.

Juntamente com a descrição dos casos de uso atualizados.

Caso de Uso 1 – UC02.01 - Login

Atores: ● Usuário
Fluxo base: 1. Usuário acessa o site do servidor. 2. Usuário digita suas credenciais. 3. O sistema autentica o usuário e é redirecionado à página principal;
Fluxo de Exceção: FE01 – Sistema não encontra credenciais ou credenciais inválidas. Sistema informa com a mensagem de erro: “Usuário e senha não identificados pelo sistema”.

Caso de Uso 2 – UC02.02A - Exibir Turmas

Atores: ● Usuário

Pré-condição: ● UC02.01
Fluxo base : 1. Sistema exibe lista de turmas disponíveis, exibindo nome, descrição, labels e as opções de excluir e editar cada modelo.
Fluxo de Exceção: FE01 – Nenhum modelo previamente cadastrado. Sistema informa uma mensagem: “Nenhuma turma cadastrada.”.

Caso de Uso 3 – UC02.03A - Manter turmas

Atores: ● Usuário
Pré-condição: ● UC02.01
Fluxo base - Cadastrar : 1. Usuário clica no botão cadastrar. 2. Sistema redireciona usuário para página de cadastro. 3. Usuário informa nome. descrição e seleciona um modelo disponível. 4. Usuário clica em salvar. 5. Sistema redireciona usuário para página principal. Fluxo base - Excluir: 1. Usuário clica no botão excluir. 2. Usuário confirma a exclusão. 3. Sistema atualiza a tela principal. Fluxo base - Editar: 1. Usuário clica no botão editar. 2. Sistema redireciona usuário para página de edição. 3. Usuário altera as informações. 4. Usuário clica em salvar / cancelar. 5. Sistema redireciona usuário para página principal.

Caso de Uso 5 – UC02.05 - Receber requisição de classificação

Atores: ● App Mobile
Fluxo base: 1. App Mobile envia requisição de classificação ao servidor.

2. Sistema executa classificação de acordo com o modelo solicitado. 3. Sistema envia ao App Mobile a requisição com a resposta da classificação.
Fluxo de Exceção: FE01 – Tipo de arquivo incorreto: Sistema envia ao App Mobile, uma resposta de erro. FE02 – Modelo inexistente: Sistema envia ao App Mobile, uma resposta de erro. FE03 – Erro na classificação: Sistema envia ao App Mobile, uma resposta de erro.

Caso de Uso 5 – UC02.05 - Treinar o Modelo

Atores: ● Usuário
Pré-condição: ● UC02.01
Fluxo base: 1. Usuário clica no botão treinar. 2. Sistema re-treina o modelo da turma, usando as imagens da turma. Enquanto o processo ocorre é apresentado: Aguarde o treinamento. 3. Sistema salva o modelo re-treinado. 4. Sistema apresenta ao usuário o *feedback* do treinamento.

Caso de Uso 6 – UC02.06 - Receber requisição de upload

Atores: ● App Mobile
Fluxo base: 1. App Mobile envia requisição de upload ao servidor. 2. Sistema executa o *upload* para a turma solicitada para a classe solicitada. 3. Sistema envia ao App Mobile a requisição com a resposta da requisição.
Fluxo de Exceção: FE01 – Sem classe ou classe inválida: Sistema faz o upload da imagem para a primeira classe do modelo. FE02 – Tipo de arquivo incorreto: Sistema envia ao App Mobile, uma resposta de erro. FE03 – Turma inexistente: Sistema envia ao App Mobile, uma resposta de erro.

Caso de Uso 7 – UC02.07 - Receber requisição de classificações

Atores: ● App Mobile
Fluxo base: 1. App Mobile envia requisição de classificações ao servidor. 2. Sistema envia ao App Mobile a requisição com a resposta contendo as classes do modelo de turma.
Fluxo de Exceção: FE01 – Turma inexistente: Sistema envia ao App Mobile, uma resposta de erro.

Caso de Uso 8 – UC02.08 - Logout

Atores: ● Usuário
Pré-condição: ● UC02.01
Fluxo base: 1. Usuário clica em logout no menu superior. 2. Sistema encerra a sessão do usuário e o redireciona a tela de login.
Fluxo de Exceção: FE01 – Usuário sem sessão: Sistema redireciona usuário para página de login.

Atualmente a construção do *dataset* para o treino é feito de maneira específica (Quadro 19), sendo assim modelos que na construção do seu *dataset* não se encaixarem nessa maneira, provavelmente vão dar erro ao serem retreinados. Uma solução seria salvar o modo como esse *dataset* é construído para cada modelo, por exemplo, salvando o nome do método que constrói o dataset e os valores de parâmetros. Assim, teríamos uma maneira genérica para construção de diferentes tipos de datasets com suas devidas configurações.

Outro ponto relevante é que a arquitetura do servidor é flexível, sendo assim é possível implementar outros tipos de aprendizado, ou realizar regressão no lugar de classificação, mas isso traz complexidade à generalização do sistema, tanto *backend* quanto *frontend*. A integração com outras APIs, como o Tensorflow, são pontos a serem analisados individualmente.

6. Avaliação

Neste capítulo é apresentado a avaliação da solução desenvolvida com o objetivo de verificar a adequação funcional e operabilidade da ferramenta e suas funcionalidades. Inicialmente tinha-se o objetivo de realizar uma avaliação prática em sala de aula durante algum projeto da Computação na Escola, onde os alunos seguiriam um tutorial para construir uma aplicação mobile utilizando recursos de ML, enviando diversas fotos ou imagens de diferentes tipos de lixo reciclável, separando essas imagens nas classes , realizando o treino. No entanto, devido à pandemia da COVID-19, não foi possível efetuar a avaliação com os usuários, sendo assim, a avaliação foi reduzida para testes de funcionamento de sistema com base nos casos de uso e fluxos de exceção. Pelo mesmo motivo, o RNF1.2, que fala sobre usabilidade, também não pode ser avaliado.

6.1. O modelo de ML utilizado

Para a realização dos testes do sistema desenvolvido foi necessário selecionar um modelo de ML já pré-treinado que permitisse classificar imagens de tipos diferentes de materiais recicláveis. Para esse fim foi selecionado o modelo trashnet (https://github.com/garythung/trashnet).

O modelo foi desenvolvido por Gary Thung e Mindy Yang como projeto final para a aula de ML CS 299 da Universidade de Stanford em 2016-2017. É uma CNN desenvolvida para fazer a classificação de um objeto reciclável, foi utilizado um *dataset* contendo 2527 imagens divididos em 70/13/17 (treinamento/validação/teste) e foram capazes de atingir cerca de 75% de precisão no teste.

6.2. Definição do cenário para os testes

Para a realização dos testes definiu-se como um cenário hipotético a realização de uma oficina de ensino de programação, utilizando como problema de fundo a classificação de imagens de lixo e materiais recicláveis para a conscientização da sociedade para a importância da reciclagem.

A oficina ocorre durante as aulas de ciência para uma turma do 9º ano e conta com a presença de 15 alunos. A oficina será dividida em 2 dias, totalizando em torno de 4 horas de duração Além do professor, um monitor se faz presente, para auxiliar os alunos, principalmente durante as atividades práticas.

No primeiro dia, o professor apresenta inicialmente os conceitos básicos de *machine learning* e depois ensina como desenvolver um app que captura imagens e, utilizando a extensão, envia essas imagens para o servidor. Depois o professor divide os alunos em grupos de 3 alunos e pede para cada grupo tirar novas fotos de materiais e enviar para o servidor, cada grupo deve tirar pelo menos 3 fotos de cada

categoria de material reciclável (papelão, vidro, metal, papel, plástico e lixo) para a próxima aula.

No dia seguinte, os alunos terminam de enviar as fotos que faltam e em seguida, o professor avalia as imagens na tela do sistema juntamente com os alunos, se necessário, remove as imagens incorretas ou move imagens categorizadas erradas, retreina o modelo e analisa junto com os alunos a matriz de confusão.

6.3. Realização dos testes

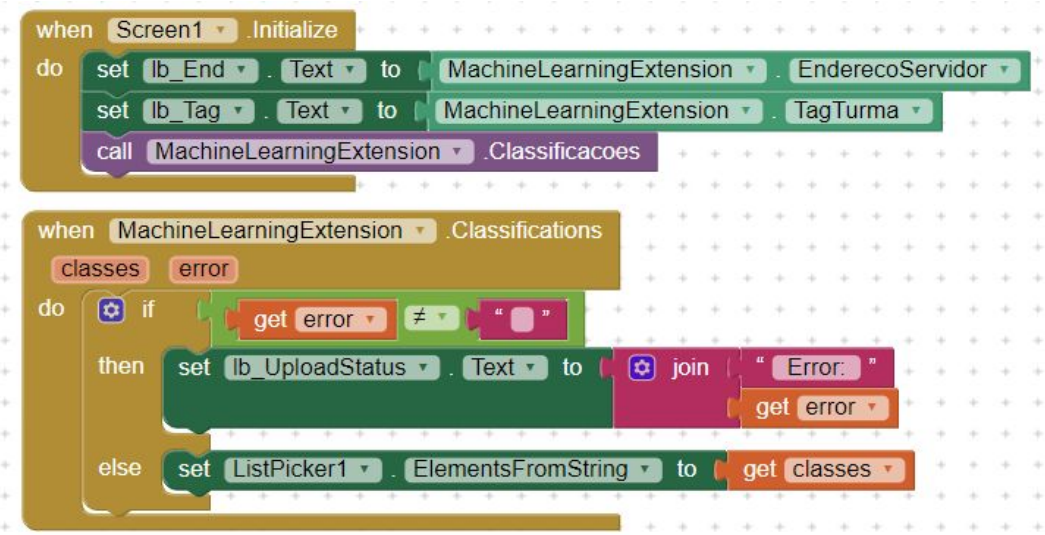
Para realizar os testes para avaliação e durante o desenvolvimento, foi preciso fazer um aplicação no App Inventor (disponível em: https://drive.google.com/file/d/1Eq7kpPDaKA-7VChk6RQMgkO13JHXRI9T/view?usp=sharing). Após incluir os componentes de tela necessários, foram definidos os dois parâmetros de configuração da extensão (Figura 18), o endereço do servidor (EndereçoServidor) juntamente com a porta, sendo obrigatório possuir “http://” ou “https://”, por exemplo: *http://192.168.0.10:5000,* e o código da turma (TagTurma), que define qual modelo será utilizado para o envio de imagens e classificação.

FIgura 18: Parâmetros de configuração da extensão

Fonte: elaborado pelo autor.

E foram utilizados os seguintes blocos de código do App Inventor:

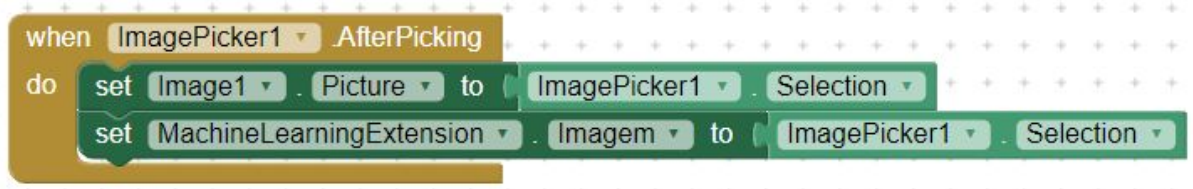
FIgura 19: Bloco de código do App Inventor - Lista de classes



Fonte: elaborado pelo autor.

Os blocos acima (Figura 19), o primeiro bloco é usado para invocar o método para obter as classificações do modelo ao inicializar a tela inicial e passar os valores de configuração para a extensão, no segundo bloco, é obtido o retorno do para criar a lista de classes do modelo ou exibir erro caso ocorra.

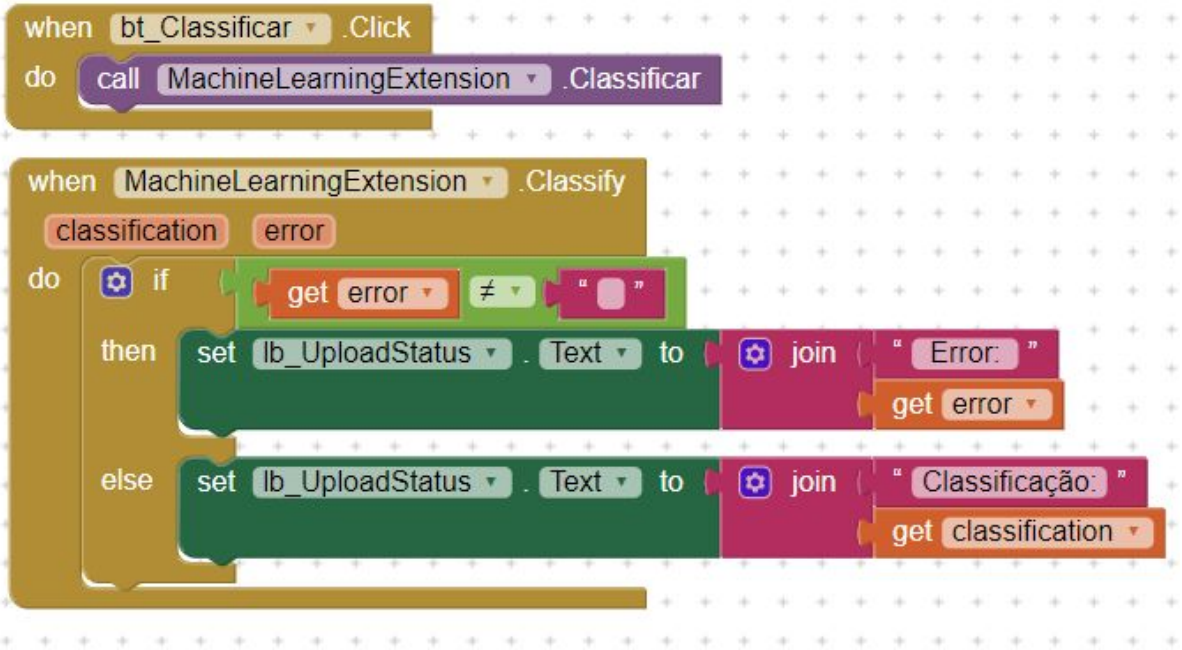
FIgura 20: Bloco de código do App Inventor - Seleção da imagem



Fonte: elaborado pelo autor.

O bloco acima (Figura 20), após seleção da imagem, apresenta a mesma para o usuário, e define seu valor para a extensão, para o *upload* ou classificação da imagem.

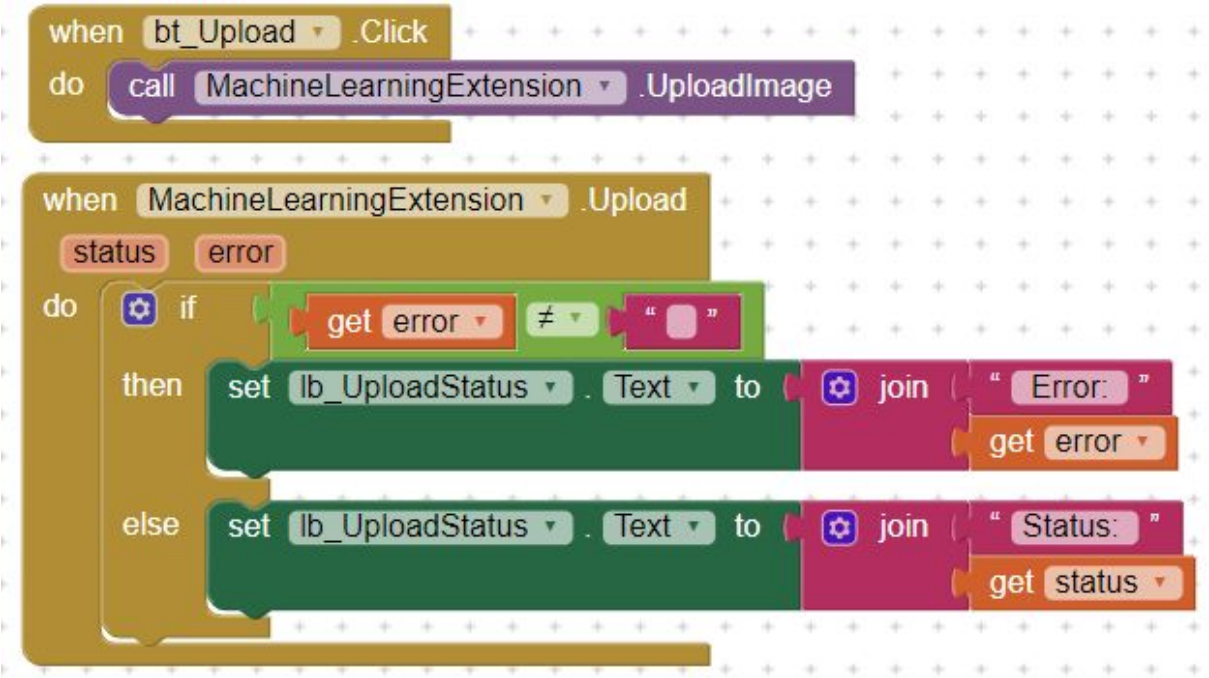
FIgura 21: Bloco de código do App Inventor - Classificação



Fonte: elaborado pelo autor.

O bloco acima (Figura 21), é usado para realizar a ação de classificar e obter o seu retorno.

FIgura 22: Bloco de código do App Inventor - *Upload*



Fonte: elaborado pelo autor.

E o bloco acima (Figura 22), é usado para realizar a ação de fazer o *upload* e obter seu retorno. A Figura 23 apresenta algumas imagens da aplicação mobile desenvolvida durante alguns dos casos de teste.

Figura 23: Imagens da aplicação mobile durante os testes

Fonte: elaborado pelo autor.

Para planejar a realização dos testes foram definidos 30 casos de testes, de forma a contemplar os requisitos e casos de uso estabelecidos para o sistema. conforme apresenta o Quadro 23, onde CT1 são testes da extensão e CT2 são

testes da aplicação, uma descrição resumida do caso de teste e uma descrição de problemas encontrados.

Os casos de teste foram então executados no sistema e os resultados foram classificados em três cores, verde representando que não houve problemas nesse caso de teste, amarelo representado um problema pequeno mas que não impossibilita a utilização do sistema e vermelho um problema grave que pode impossibilitar a utilização do sistema.

Quadro 23: Quadro de casos de teste

Caso de teste	Descrição	Problema encontrado
CT1.01.1	Tentar classificar imagem utilizando a extensão (UC01.01).	
CT1.01.2	Variação do CT1.01.1, extensão inválida	Apresentam a mensagem de erro do servidor (invalid extension)
CT1.01.3	Variação do CT1.01.1, código turma inválido	Apresentam a mensagem de erro do servidor (invalid token)
CT1.01.4	Variação do CT1.01.1, sem imagem	Apresenta erro 1104 (Unable to post or put the file \%s\ with the specified URL: %s)
CT1.01.5	Variação do CT1.01.1, sem classificação	Faz o upload para a primeira classe do modelo
CT1.02.1	Tentar fazer upload de imagem utilizando a extensão (UC01.02).	
CT1.02.2	Variação do CT1.02.1, extensão inválida	Apresentam a mensagem de erro do servidor (invalid extension)
CT1.02.3	Variação do CT1.02.1, código turma inválido	Apresentam a mensagem de erro do servidor (invalid token)
CT1.02.4	Variação do CT1.02.1, sem imagem	Apresenta erro 1104 (Unable to post or put the file \%s\ with the specified URL: %s)
CT2.01.1	Login	
CT2.01.2	Variação do CT2.01.1, sem usuário ou usuário incorreto	Não é apresentada nenhuma mensagem ao usuário.
CT2.01.3	Variação do CT2.01.1, sem senha ou senha incorreto	Não é apresentada nenhuma mensagem ao usuário.
CT2.02.1	Exibir modelos	
CT2.02.2	Variação do CT2.02.1, sem nenhuma turma cadastrada (UC02.02 - FE01)	
CT2.03.1	Ações de cadastro, visualização, edição e	

	exclusão da tela de listagem de turmas	
CT2.04.1	Cadastro turma	
CT2.04.2	Variação do CT2.04.1, sem selecionar o modelo	
CT2.04.3	Variação do CT2.04.1, sem modelos disponíveis.	
CT2.05.1	Editar turma	
CT2.05.2	Variação do CT2.05.1, sem selecionar o modelo	
CT2.05.3	Variação do CT2.05.1, sem modelos disponíveis.	
CT2.06.1	Visualização de turma	
CT2.06.2	Variação do CT2.06.1, ações de excluir todas imagens	
CT2.06.3	Variação do CT2.06.1, ações de mover e excluir imagens	
CT2.07.1	Treinamento	
CT2.07.2	Variação do CT2.07.1, epoch zero ou negativo	É usado 1 no epoch
CT2.07.3	Variação do CT2.07.1, sem imagens	Erro ao efetuar o *fit.*
CT2.07.4	Variação do CT2.07.1, com poucas imagens	Com 2 imagens em cada classe, ocorre erro ao efetuar o *fit,* com 3 imagens em cada classe executa
CT2.07.5	Variação do CT2.07.1, sem imagens de uma ou mais classe	Erro ocorre a gerar a matriz de confusão
CT2.07.6	Variação do CT2.07.1, quantidade diferente de imagens para as classes	

Fonte: elaborado pelo autor.

Todos os CT1 que apresentaram problema foram amarelos, mas foram ambos quando a imagem não foi selecionada, o que pode ser solucionado na aplicação mobile do App inventor. Já nos CT2 houve muitos erros na parte de treinamento, os CT2.07.3 e CT2.07.4 poderiam ser resolvidos fazendo a validação da quantidade de imagens de cada classe antes de iniciar a execução, o que também resolveria o problema no CT2.07.5. Esses poucos erros encontrados serão corrigidos na entrega final da ferramenta a ser disponibilizada em um servidor da Computação na Escola.

Alguns fluxos de exceção não foram alcançados devido a forma que foi desenvolvida a aplicação, como o código de turma já existente, que agora é gerado automaticamente.

7. Conclusão

Neste trabalho foi desenvolvida uma ferramenta com a finalidade de auxiliar e estimular o processo de aprendizagem de computação na educação básica, envolvendo para isso, o desenvolvimento de aplicações inovadoras utilizando *Machine Learning* de maneira fácil.

Inicialmente foi realizada a fundamentação teórica dos principais conceitos relacionados ao tema deste trabalho, como o Ensino de Computação, *Machine Learning*, *Deep Learning* e o App Inventor. Com o objetivo de identificar outras aplicações similares existentes atualmente, foi realizado um mapeamento sistemático da literatura. Onde se comprovou a necessidade de desenvolver uma extensão para utilizar os recursos de *Deep Learning* no App Inventor.

A ferramenta foi modelada e desenvolvida utilizando tecnologias atuais. Os módulos desenvolvidos contemplaram quase que integralmente os requisitos e casos de uso definidos. Os resultados obtidos a partir das avaliações por meio dos casos de teste, indicam que a ferramenta é funcional, mas sua usabilidade e aplicabilidade não puderam ser avaliadas com o público-alvo devido à pandemia da COVID-19, mesmo assim, espera-se que a ferramenta desenvolvida possa ajudar no ensino de computação e IA. Os erros identificados pelos casos de teste, estão sendo corrigidos e serão finalizados até a entrega final da ferramenta.

Assim foi alcançado o objetivo de desenvolver uma extensão do App Inventor para utilização de *Deep Learning* e um servidor Web para interagir com a extensão. Assim espera-se contribuir com o engajamento dos alunos e a melhoria do ensino de computação na educação básica.

Como trabalhos futuros, além do requisito funcional que não foi implementado, existem diversas possíveis melhorias, mas as principais são a melhoria nas configurações de treinamento (taxa de aprendizado e momentum), a possibilidade de criar um modelo novo, por exemplo usando um *dataset* do MNIST, disponivel pelo proprio fast.ai (https://docs.fast.ai/data.external), apresentar ao usuário mais informações sobre o resultado do treinamento e métricas de acurácia, precisão e revocação e acrescentar um texto para ajudar a compreensão da matriz de confusão, assim melhorando a flexibilidade da ferramenta e o *feedback* ao usuário para melhorar a parte de treinamento.

REFERÊNCIAS

ALIMISIS, D.; LOUKATOS, D. STEM education post-graduate students' training in the eCraft2Learn ecosystem, 2018

ARAÚJO, L.; DA SILVEIRA, H. U. C.; MATTOS, M. Ensino do pensamento computacional em escola pública por meio de uma plataforma lúdica. Anais dos Workshops do Congresso Brasileiro de Informática na Educação, v. 7, n. 1, p. 589, 2018.

BAULÉ, D. DE S., Desenvolvimento de um Modelo de Geração Automática de Wireframes no App Inventor a partir de Sketches usando Deep Learning. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade Federal de Santa Catarina. 2020.

BRUMMELEN, J. V. Tools to Create and Democratize Conversational Artificial Intelligence, M.S. thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.

CHING, T. et al. Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface, v. 15, n. 141, p. 20170387, 2018.

COMPUTER SCIENCE TEACHERS ASSOCIATION (CSTA), K–12 Computer Science Framework. 2016. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K–12-Computer-Science-Framework.pdf>. Acesso em: Set. 2019.

COMPUTER SCIENCE TEACHERS ASSOCIATION (CSTA), K–12 Computer Science Standards, 2017. Disponível em: <https://www.csteachers.org/Page/standards>. Acesso em: Nov. 2020.

DANIEL, G. T.; WANGENHEIM, G. von W.; MEDEIROS, G. A. e S.; ALVES, N. da C. Ensinando a Computação por meio de Programação com App Inventor. Proc. of Computer on the Beach, Florianópolis, Brazil, 2017.

HADDAWAY, N. R.; COLLINS, A. M.; COUGHLIN, D.; KIRK, S. The role of Google Scholar in evidence reviews and its applicability to grey literature searching. PloS one, v. 10, n. 9, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. 24765-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering--Vocabulary, 2017.

JONES, T. Deep learning architectures, The rise of artificial intelligence. 2017. Disponível em: <https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/>. Acesso em: Nov. 2020.

KRETZER, F. M. Desenvolvimento de uma Unidade Instrucional para Formação de Professores da Educação Básica para o Ensino de Computação, Trabalho de Conclusão de Curso (Graduação em Ciência da Computação), Universidade Federal de Santa Catarina, 2019

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. nature, v. 521, n. 7553, p. 436-444, 2015.

LEHMKUHL, G. Desenvolvimento de um modelo de avaliação de criatividade no ensino da computação na educação básica, Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação), Universidade Federal de Santa Catarina, 2020.

MACHINE LEARNING FOR KIDS. Disponível em: <https://machinelearningforkids.co.uk>, Acesso em: Set. 2019.

MARTINS, O. P. H. R. Desenvolvimento de um Modelo para Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando Deep Learning, Trabalho de Conclusão de Curso (Graduação em Ciência da Computação), Universidade Federal de Santa Catarina, 2019.

MITCHELL, T. M.; CARBONELL, J. G.; MICHALSKI, R. S. Machine learning: a guide to current research. Springer Science & Business Media, 1986.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. Foundations of Machine Learning. 2. ed. MIT Press, 2018.

PATTERSON, J.; GIBSON, A. Deep learning: A practitioner's approach. "O'Reilly Media, Inc.", 2017.

SAS. Machine Learning. Disponível em <https://www.sas.com/pt_br/insights/analytics/machine-learning.html> Acesso em: Set. 2019.

SANTANA, M. Deep Learning: do Conceito às Aplicações. Jul 19, 2018. Disponível em <https://medium.com/data-hackers/deep-learning-do-conceito-às-aplicações-e8e91a7c7eaf> . Acesso em: Set. 2019.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. Neural networks, v. 61, p. 85-117, 2015.

SVANBERG, M. S. Suggested Blocks: Using Neural Networks To Aid Novice Programmers In App Inventor. 2018.

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC), Diretrizes para ensino de Computação na Educação Básica. 2019. Disponível em:

<https://www.sbc.org.br/documentos-da-sbc/send/203-educacao-basica/1220-bncc-em-itinerario-informativo-computacao-2> Acesso em: Nov. 2020

TANG, D., Empowering Novices to Understand and Use Machine Learning With Personalized Image Classification Models, Intuitive Analysis Tools, and MIT App Inventor, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.

TORREY, L.; SHAVLIK, J. Transfer learning. In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global, 2010. p. 242-264.

VAZQUEZ, C.; SIMÕES, G. Engenharia de Requisitos: Software Orientado ao Negócio. Brasport. 2016.

ZHU, K. An Educational Approach to Machine Learning with Mobile Applications, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2019.

WING, J. M. Computational thinking. Communications of the ACM, v. 49, n. 3, p. 33–35, mar 2006.

Apêndice A

Código fonte disponivel em: <https://codigos.ufsc.br/100000000394729/machine-learning-app-inventor-server>

Apêndice B

Desenvolvimento de um componente App Inventor para Deep Learning

Lucas Martins Petroski

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis, Brasil

petroski31@gmail.com

Abstract. *Even with the growing educational computing, there is still difficulty in awakening the interest of basic education students to computer learning. The use of artificial intelligence can assist in the learning of computational thinking, however, platforms for teaching computational thinking do not offer native options for accessing artificial intelligence resources. This project proposed to integrate App Inventor with Deep Learning, developing a component in the form of an extension, together with a server that converts the component's calls into invocations of operations that use Deep Learning, and returns the results to the component. The tool was evaluated using test cases based on the requirements and it is hoped that the tool developed can contribute to increasing student engagement in computational education.*

Resumo. *Mesmo com o crescente ensino de computação, ainda há dificuldades em despertar o interesse dos alunos da educação básica para o aprendizado de computação. O uso de inteligência artificial pode auxiliar no aprendizado do pensamento computacional, entretanto, as plataformas para ensino do pensamento computacional não oferecem opções nativas de acesso a recursos de inteligência artificial. Esse projeto propôs integrar o App Inventor com Deep Learning, desenvolvendo um componente na forma de extensão, juntamente com um servidor que converte as chamadas do componente em invocações de operações que utilizam Deep Learning, e retornam os resultados ao componente. A ferramenta foi avaliada utilizando casos de teste com base nos requisitos e espera-se que a ferramenta desenvolvida possa contribuir para o aumento do engajamento dos alunos no ensino computacional.*

1. Introdução

O Pensamento Computacional é um subconjunto de competências e habilidades relacionadas à abstração e decomposição de problemas de forma a permitir sua resolução usando recursos computacionais e estratégias algorítmicas (WING, 2006). A inserção dos conceitos da Ciência da Computação na educação básica desenvolve uma habilidade de abstração, a qual pode ajudar as crianças na resolução de problemas em todas as áreas da vida (DANIEL et al., 2017).

Existem diversas dificuldades para se envolver os alunos na construção de conhecimento sobre lógica de programação e algoritmos (ARAÚJO et al., 2018). Uma das plataformas utilizadas para esta finalidade é o App Inventor, desenvolvida pela Google e mantida pelo Instituto de Tecnologia de Massachusetts (MIT). O App Inventor é um ambiente de programação baseado em blocos que permite pessoas com pouco conhecimento em programação desenvolver aplicativos para dispositivos Android. Entretanto, plataformas para ensino do pensamento computacional, como o Scratch e App Inventor não oferecem opções nativas de acesso a recursos de IA, como por exemplo Deep Learning para apoiar o ensino de computação por meio do desenvolvimento de aplicativos.

Uma das soluções atualmente empregadas na IA é o *Machine Learning* (ML - aprendizado de máquina) e mais especificamente o *Deep Learning* (aprendizagem profunda). Segundo Michalski, Carbonell e Mitchell (1986) ML constitui-se no estudo e a modelagem computacional do processo de aprendizagem em suas múltiplas manifestações, sendo assim, pode-se defini-lo como a habilidade de computadores aprenderem sem serem explicitamente programados.

Com a finalidade de acrescentar funcionalidades de IA aos aplicativos e melhorar a experiência dos alunos no aprendizado de computação, este projeto propõe integrar o App Inventor com IA, mais precisamente *Deep Learning*. Para isso, foi desenvolvido um componente na forma de extensão para a plataforma App Inventor, juntamente com um servidor que converte as chamadas do componente em invocações de operações para um *framework*, que utilizam *Deep Learning*, e retornam os resultados ao componente. Com a disponibilização de funcionalidades de IA no App Inventor espera-se que os alunos possam aprender computação desenvolvendo aplicações inovadoras sem precisarem se preocupar com a sua complexidade.

2. Metodologia

A metodologia de pesquisa utilizada neste trabalho é dividida em cinco etapas principais. A primeira etapa foi a fundamentação teórica, que consiste em estudar, analisar e sintetizar os conceitos principais e a teoria referente aos temas abordados. A segunda etapa é realizado um mapeamento sistemático de literatura seguindo o processo proposto por Petersen et al. (2008) para levantar o estado da arte de componentes ou extensões de ML em ambientes de programação visual. Na terceira etapa é feita a elaboração do escopo do trabalho, análise de requisitos e a modelagem e descrição da solução proposta. Na quarta etapa é feita a implementação do componente na forma de extensão do App Inventor baseado nas informações obtidas nas etapas anteriores. Por fim, na quinta etapa é feita a implementação do servidor e testes de integração com o componente baseado nas informações obtidas nas etapas anteriores.

3. Estado da Arte

O objetivo é encontrar experiências de utilização do App Inventor e ML no contexto do ensino da computação na educação básica e quais as suas características. Detalhando este objetivo são definidas cinco perguntas de análise: Quais são as experiências de utilização de App Inventor com ML? Quais métodos/técnicas de ML foram utilizadas?

Quais ferramentas são utilizadas e qual o contexto do uso dessas ferramentas? Quais são as tecnologias utilizadas? Quais os resultados obtidos?

As buscas foram realizadas dentro das bases do repositório do App Inventor e Google Scholar, considerando somente trabalhos que tratam do ensino de programação utilizando o App Inventor ou similares e utilizando direta ou indiretamente ML e artigos científicos publicados em português e inglês.

Analisando os resultados das buscas foram encontrados somente 7 artefatos que satisfazem os critérios de inclusão e exclusão. No entanto, uma aplicação prática não foi encontrada Machine Learning for Kids (2019), possivelmente por que não foram realizadas ainda publicações sobre a aplicação. Assim, a busca foi ampliada para o uso da ferramenta Google. Foram então analisadas as 10 primeiras páginas de resultados. A partir dessa busca a aplicação citada foi encontrada.

Dentre os artigos encontrados, quatro artigos que utilizam App Inventor, dois são similares ao proposto neste trabalho, e um deles agrega mais a possibilidade de treinar sua rede neural. Entre todos os artigos que utilizam o App Inventor, todos desenvolveram uma extensão para conseguir utilizar ML em seus projetos, o que demonstra a necessidade da implementação de uma extensão mais genérica que permita a utilização de ML de maneira mais fácil.

4. Análise e Projeto

Os requisitos foram levantados por meio da análise dos resultados do estado da arte e de entrevistas não estruturadas com um especialista de domínio. Durante o processo de desenvolvimento, alguns dos requisitos foram alterados para ficarem mais coerentes com o rumo do projeto.

Abaixo, no Quadro 1, são apresentados os requisitos funcionais da extensão (RF01) e do servidor (RF02).

Quadro 1: Requisitos funcionais.

ID	Requisito	Descrição
RF 01. 01	Enviar requisição de classificação	Enviar ao servidor a requisição de classificação de uma entrada para um modelo enviando uma imagem.
RF 01. 02	Receber requisição de classificação	Receber resposta do servidor da requisição de classificação enviada.
RF 01. 06	Receber requisição de lista de *labels* de um modelo.	Receber resposta do servidor da requisição de lista de *labels* enviada.
RF 01. 07	Configurar endereço	Configurar Ip e Porta de envio das requisições
RF 01.	Enviar requisição de *upload*	Enviar ao servidor a requisição de *upload* de imagem para uma turma.

08		
RF 02. 01	Receber requisição de classificação	Receber a requisição de classificação de uma entrada para um modelo e responder ao cliente com a *label* de classificação, probabilidade ou uma mensagem de erro.
RF 02. 02	Receber lista de modelos.	Receber a requisição de lista de modelos treinados do servidor e responder ao cliente o que foi solicitado.
RF 02. 03	Receber lista de *labels* de um modelo.	Receber a requisição da lista de *labels* de um determinado modelo e responder ao cliente o que foi solicitado.
RF 02. 04	Classificar uma entrada com base em um modelo	Classificar uma entrada de acordo com um modelo pré-treinado existente.
RF 02. 05	Exibir modelos disponíveis	Exibir os modelos pré-treinados disponíveis com uma breve descrição e *labels* existentes (web)
RF 02. 06	Login	Efetuar login
RF 02. 07	Inserir novos modelos	Manter modelos pré-treinados
RF 02. 08	Inserir novas turmas	Manter turmas.
RF 02. 09	Treinar turmas	Possibilitar o re-treinamento do modelo da turma.
RF 02. 10	Apresentar resultado do treinamento	Apresentar feedback ao usuário após término do treinamento

E no Quadro 2, os requisitos não-funcionais da extensão (RNF01) e do servidor (RNF02).

Quadro 2. Requisitos não-funcionais.

ID	Requisito	Descrição
RNF 1.1	Linguagem de programação Java	A ferramenta deve ser desenvolvida na linguagem de programação Java compatível com a versão 7, pois o projeto existente que está continuado é Java versão 7.
RNF	Usabilidade	Eficácia: 90% dos usuários da avaliação devem

1.2		conseguir completar a tarefa de classificar uma entrada para um modelo.
RNF 1.3	Extensibilidade	O sistema deve permitir que novos modelos pré-treinados sejam adicionados.
RNF 1.4	Tamanho de arquivo da requisição	O sistema deve permitir o envio de arquivos em requisições de até 16 MB,
RNF 1.5	Compatibilidade com App Inventor	A extensão deve ser compatível com a versão 2.41 do App Inventor
RNF 2.1	Servidor Web	O servidor deve ser acessado via navegador Web com conexão à internet. Navegadores compatíveis: Google Chrome versão 77.
RNF 2.2	Linguagem de programação Javascript, HTML, CSS	A ferramenta deve ser desenvolvida nas linguagens de programação Javascript, HTML5 e CSS3 pois são linguagens muito conhecidas facilitando a manutenção do sistema.
RNF 2.3	Os métodos de classificação assíncronos	Os métodos de classificação devem ser assíncronos.
RNF 2.4	Tempo de resposta	O tempo de resposta do servidor deve ser de até 5 segundos.

Considerando os requisitos previamente levantados, os casos de uso identificados e seus principais fluxos de execução resultando no diagrama de casos de uso apresentado na Figura 1.

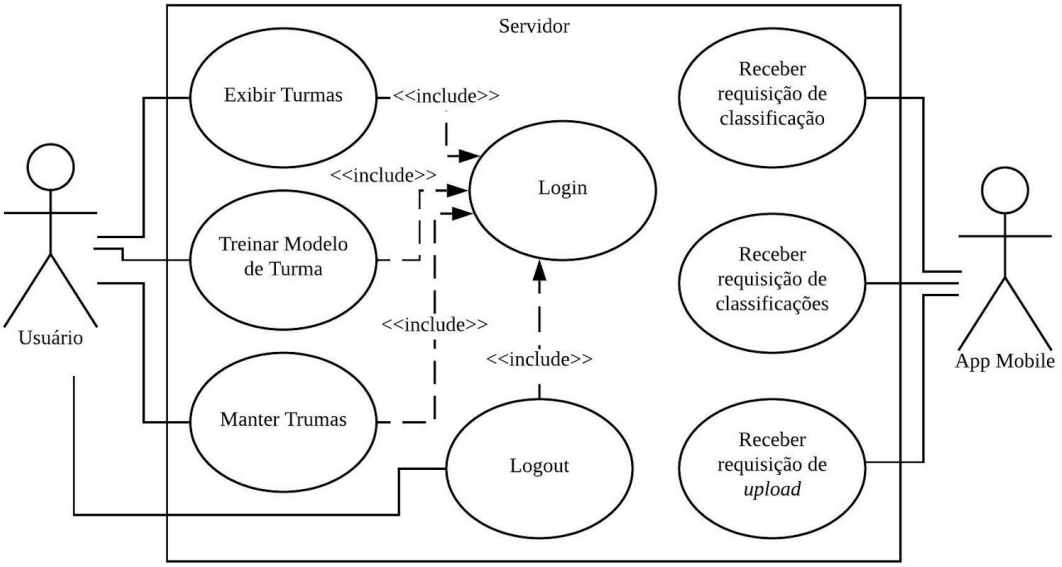


Figura 1. Diagrama de casos de uso.

Para possibilitar o atendimento dos requisitos, foi definido um modelo conceitual da arquitetura dividido em dois módulos principais: a extensão do App Inventor e o Servidor Web. Assim, quando o componente do aplicativo fizer uma requisição ao servidor, o servidor processa a requisição e retorna a resposta ao aplicativo, no caso de uma classificação de imagem, por exemplo, o servidor executa a classificação para um modelo específico e responde ao aplicativo qual é a classificação.

O servidor é uma API restful responsável por atender às requisições da extensão e executar as predições dos modelos definidos. O resultado da requisição contém o mapeamento da predição do modelo para as *labels* ou erro no caso de erro. Ele também responde a extensão para listar os modelos definidos e suas respectivas *labels*. Além disso, o servidor também possui uma aplicação Web para que seus usuários possam manter os modelos previamente treinados. E a extensão executa os blocos quando chamados, realizando as requisições ao servidor e aguardando a resposta do mesmo.

5. Desenvolvimento

Diversas tecnologias e ferramentas são utilizadas para a implementação da extensão App Inventor e do Servidor. As principais tecnologias aplicadas no desenvolvimento foram Python, Fast.ai e Flask para o servidor e Java para a extensão.

5.1. Extensão

A extensão é basicamente uma adaptação de um componente Web do App Inventor, contendo especializações do métodos para o fim desejado, como o envio de imagens e a obtenção do resultado da classificação. Primeiramente foram adicionados os parâmetros de configuração da extensão e seus respectivos métodos acessores. A classe aninhada ‘CapturedProperties’ foi alterada para aceitar os novos atributos, o caminho da imagem, o endereço do servidor e a classe da imagem.

Para a execução das requisições, foi alterado o método ‘performRequest’ que adicionam ao *header* da requisição a extensão e a extensão da imagem e um If-else para invocar o método de disparo do evento passado como parâmetro no ‘performRequest’.

Para as requisições, com base nos métodos ‘Get’ e ‘PostFile’ do componente Web, foram escritos os métodos para executar a requisição e aguardar o retorno da mesma ou em caso de erro, disparar o evento de erro.

5.2. Servidor

Para facilitar a implementação do servidor, foi decidido salvar os modelos e turmas em arquivo nas pastas “models” e “turmas” respectivamente, seguindo uma hierarquia de arquivos. Dentro da pasta de cada modelo, além do arquivo de serialização de objetos Python da biblioteca pickle, usado para salvar o modelo já treinado. Nesta mesma pasta existe um arquivo chamado “classes.txt”, que contém um dicionário de dados usado na tradução das classes de cada modelo. E a pasta ‘turma’, contém uma cópia do arquivo pkl do modelo, o arquivo ‘descricao.txt’ contendo a descrição da turma criada durante o cadastro da turma e as imagens enviadas pela extensão, dentro da subpasta da classe a que foi submetida.

Para finalizar a parte de arquivos do servidor, sua implementação foi dividida em quatro arquivos:

- application.py: que contém as configurações da aplicação Flask e sua execução inicia o servidor Web,
- server.py: que contém as rotas do servidor,
- model_service.py: que contém os métodos de modelos e,
- turma_service.py: que contém os métodos de turmas.

O server.py possui as principais chamadas de métodos do servidor, como o upload do arquivo, onde é obtido o formato do arquivo e o código da turma do *header* da requisição, verifica-se se o código e o formato da extensão são válidos, caso não haja erro, é chamado o método upload do turma_service.py, onde é salvo o array de bytes no formato enviado dentro da pasta da turma. A classificação é feita de forma similar, executando o método predict no lugar do upload, onde é carregado o modelo da turma, chamada a classificação deste modelo para o array de bytes da imagem e retorna a classificação recebida do modelo.

Outro método importante é o de retreinar o modelo, onde após carregar o modelo, é criado um novo dataset a partir da pasta de imagens da turma, utilizando o nome das pastas para categorizar as imagens e 20% das imagens são utilizadas para validação do treino. Em seguida é efetuado o treinamento realizando n ciclos de treino, sendo n, o valor do epoch enviado pelo usuário.

Como a ferramenta utiliza o conceito de Transfer Learning (TORREY & SHAVLIK, 2010), é necessário re-treinar o modelo após a inclusão de novas imagens. Isso gerou um problema em relação ao tempo necessário para treinar o modelo, pois a proposta inicial era utilizar as imagens do dataset do modelo e acrescentar mais imagens para o re-treino, mas isso gerou um tempo de execução inviável para ser usado em sala de aula. Sendo assim, a proposta foi substituída, onde o treinamento acrescenta mais camadas no modelo utilizando as imagens enviadas. O problema dessa abordagem, é que devido à baixa quantidade de imagens do cenário proposto de uso, provavelmente irá resultar em uma taxa de acurácia menor do que o modelo inicial, dependendo da qualidade das imagens e da categorização fornecidas pelos usuários.

6. Avaliação

Inicialmente tinha-se o objetivo de realizar uma avaliação prática em sala de aula durante algum projeto da Computação na Escola, onde os alunos seguiriam um tutorial para construir uma aplicação mobile utilizando recursos de ML, enviando diversas fotos ou imagens de diferentes tipos de lixo reciclável, separando essas imagens nas classes , realizando o treino. No entanto, devido à pandemia da COVID-19, não foi possível efetuar a avaliação com os usuários, sendo assim, a avaliação foi reduzida para testes de funcionamento de sistema com base nos casos de uso e fluxos de exceção. Pelo mesmo motivo, o requisito não-funcional que fala sobre usabilidade, também não pode ser avaliado.

Para planejar a realização dos testes foram definidos 30 casos de testes, sendo 9 para a extensão e 21 para o servidor, de forma a contemplar os requisitos e casos de uso estabelecidos para o sistema. Os casos de teste que apresentaram algum tipo de erro

foram corrigidos, os principais eram devido a falta de verificação na quantidade de imagens disponíveis para o re-treino do modelo, sendo assim, foi adicionado uma condição para execução do treino, onde deve-se ter no mínimo um total de 11 imagens e de 3 em cada classificação.

7. Conclusão

Este artigo apresenta uma ferramenta desenvolvida com a finalidade de auxiliar e estimular o processo de aprendizagem de computação na educação básica, envolvendo para isso, o desenvolvimento de aplicações inovadoras utilizando ML de maneira fácil.

A ferramenta foi modelada e desenvolvida utilizando tecnologias atuais. Os módulos desenvolvidos contemplaram quase que integralmente os requisitos e casos de uso definidos. Os resultados obtidos a partir das avaliações por meio dos casos de teste, indicam que a ferramenta é funcional, mas sua usabilidade e aplicabilidade não puderam ser avaliadas com o público-alvo, mesmo assim, espera-se que a ferramenta desenvolvida possa ajudar no ensino de computação e IA. Assim espera-se contribuir com o engajamento dos alunos e a melhoria do ensino de computação na educação básica.

Referências

Araújo, L., da Silveira, H. U. C., & Mattos, M. (2018, October). Ensino do pensamento computacional em escola pública por meio de uma plataforma lúdica. In Anais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 7, No. 1, p. 589).

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., ... & Xie, W. (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, *15*(141), 20170387.

Daniel, G. T. et al (2017). Ensinando a Computação por meio de Programação com App Inventor. In: Proc. of Computer on the Beach, Florianópolis/Brazil.

Mitchell, T. M., Carbonell, J. G., & Michalski, R. S. (Eds.). (1986). Machine learning: a guide to current research (Vol. 12). Springer Science & Business Media.

Patterson, J., Gibson, A. (2017). Deep learning: A practitioner's approach. " O'Reilly Media, Inc.".

Wing, J. M. (2006) “Computational thinking.” In: Communications of the ACM, v. 49, n. 3, pages. 33–35.