Danilo Oliveira Sobral
Kaíque Maestrini Sacchi
Victor Fernandes Mariano Marcelino

Plataforma de complemento fisioterápico baseada em captura de imagens.

São Paulo
2020

Danilo Oliveira Sobral
Kaíque Maestrini Sacchi
Victor Fernandes Mariano Marcelino

Plataforma de complemento fisioterápico baseada em captura de imagens.

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Bacharel em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Jorge Luis Risco Becerra

São Paulo
2020

RESUMO

O foco do projeto é aumentar a adesão de pacientes, cuja debilidade motora nos membros superiores demande acompanhamento fisioterápico para manutenção e melhoria de suas capacidades de movimentação, ao processo de recuperação. Por meio da tecnologia de reconhecimento de imagem, pretende-se utilizar exercícios médicos como forma de interagir com jogos, dispensando *hardware* dedicado à captura de movimento graças ao uso da câmera do dispositivo móvel. A solução proposta é o desenvolvimento de um conjunto de ferramentas capaz de tornar mais rápida e fácil a construção de jogos voltados à complementação da fisioterapia. Tais ferramentas serão disponibilizadas na forma de *plug-ins* para *game engine*. Além de captar os movimentos e transformá-los em entrada para os jogos, trechos da sessão serão ainda registrados e disponibilizados ao profissional responsável pelo tratamento do paciente. Com isso, o produto final é a arquitetura da plataforma digital descrita, bem como arquétipos de implementação de cada nó participante - um modelo de rede neural para classificação de gestos, um jogo que faça uso deste e uma plataforma WEB que exiba trechos gravados, ao paciente e ao profissional responsável por seu tratamento, e gerencie parâmetros das sessões.

Palavras-chave: Reabilitação por jogos. Reconhecimento de gestos. Arquitetura de plataforma digital.

ABSTRACT

The aim of this project is to increase patient compliance to the recovery process, for patients whose motor weakness in upper limbs requires physical therapy to maintain and improve their movement capabilities. Through image recognition technology, it intends to use medical exercises as a way to interact with games, dismissing the use of dedicated hardware for motion capture in favor of a mobile device's camera. The proposed solution is the development of a set of tools capable of making the creation of games aimed at the complement of physiotherapy faster and easier. Said tools will be made available as game engines' plug-ins. Beyond capturing movements and translating them into game controls, portions of the session will be registered and made available to the professional responsible for the treatment. Therefore, the final product is composed of the architecture of the described digital platform, as well as implementation archetypes for each participating node - a neural network model for gesture classification, a game that makes use of it and a WEB platform that displays the recorded portions to the patient and the professional responsible for their treatment, and manages the session parameters.

Keywords: Rehabilitation through games. Gesture recognition. Digital Platform Architecture.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

.SO	*Shared Objects*
Adam	*Adaptive Moment Estimation*
API	*Application Programming Interface*
AVC	Acidente Vascular Cerebral
DLL	*Dynamic-Link Library*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*
RGB	*Red, Green, Blue*
TLS	*Transport Layer Security*
UI	*User Interface*
UML	*Unified Modeling Language*
UPM	*Unity Package Manager*
VR	*Virtual Reality*

SUMÁRIO

1. Introdução

Na fisioterapia, a adesão e o envolvimento do paciente são de extrema importância para a obtenção dos resultados esperados [1], como a recuperação motora do membro afetado ou retardo de sua deterioração. A natureza repetitiva dos exercícios realizados durante uma sessão de fisioterapia, entretanto, reduz o engajamento, o que dificulta o processo de reabilitação [2], principalmente em casos cujo tratamento tem maior duração, como recuperação após um Acidente Vascular Cerebral (AVC) ou paralisia cerebral em crianças.

A utilização de sensores de movimento e desenvolvimento de jogos surgiu como alternativa para tornar mais prazerosa a realização das sessões de tratamento. O emprego de tal *hardware* dedicado, todavia, constitui uma barreira de acesso à solução.

1.1. Objetivo

Pretende-se desenvolver uma arquitetura de plataforma digital para unir o mercado de jogos e de fisioterapia, por meio da implementação de um sistema de captação de movimentos das mãos sem o uso de *hardware* dedicado. Tal sistema deverá possibilitar a interação do usuário com jogos *mobile* através da câmera do dispositivo, de modo a dispensar o uso de sensores caros e específicos. Este software será disponibilizado na forma de *plug-in* para plataforma de desenvolvimento de jogos, para que desenvolvedores terceiros possam utilizá-lo em seus projetos. O ecossistema planejado tem como público-alvo pacientes com debilidade motora nas mãos, que poderão usá-lo para complementar as atividades fisioterápicas com conforto e diversão, de forma a auxiliar sua aderência ao tratamento.

Evitar o uso de sensores de movimento específicos confere à plataforma caráter mais acessível, para que qualquer paciente que possua um *smartphone* moderno possa usufruir de sua utilização sem a complexidade técnica e financeira de adquirir e instalar *hardware* dedicado. Disponibilizar tal sistema em formato de *plug-in* facilita a entrada de outros criadores de conteúdo neste mercado, sem precisar envolvê-los em etapas específicas ao desenvolvimento de tecnologia diretamente vinculada a tratamentos médicos. Por fim, acoplá-lo a um servidor WEB permite o armazenamento de dados, como gravação das sessões, para melhorar o serviço prestado pelo profissional responsável pelo tratamento do paciente.

1.2. Motivação

Em média, pacientes com debilidades motoras não realizam atividades funcionais em dosagem suficiente para induzir a reorganização neural necessária à recuperação [3]. A utilização de jogos no contexto da fisioterapia produz resultados positivos, e leva a uma maior adesão por parte dos pacientes no processo de reabilitação [2], [4]. Tais jogos baseiam-se no uso de sistemas já existentes no mercado, como o *Nintendo Wii* ou *Xbox Kinect*. Estes, todavia, restringem-se a ambientes controlados, por conta do custo e complexidade de uso elevados, como o doméstico, hospitalar e clínico.

Com mais de 3,4 bilhões de pessoas conectadas a redes móveis por meio de dispositivos móveis em 2018, com expectativa de atingir 5 bilhões até 2025 [5], assume-se que é possível atingir uma parcela considerável dos pacientes ao apostar na utilização de celulares para disponibilizar o acesso ao sistema, seja por tais usuários portarem-los ou terem contato direto com pessoas que dispõem de um. De tal modo, a plataforma pode fazer uso destes aparelhos sem tornar-se inviável.

Pretende-se baratear o acesso a *softwares* controlados por movimentos, por meio do uso da câmera de dispositivos móveis, como celulares e tablets, para interagir com jogos voltados ao auxílio da fisioterapia. O corte de custo possibilitará à plataforma alcançar um conjunto maior de pacientes quando comparado ao atualmente praticado. Um maior mercado serve como atrativo, também, ao crescente conjunto de desenvolvedores de jogos, de modo a aumentar a disponibilidade de opções aos pacientes.

1.3. Justificativa

Pesquisas, como “Gear VR e Leap Motion Aplicados em Reabilitação Virtual para Treinamento de Função Manual: uma Oportunidade para Reabilitação em Casa” [6] e “Sistema de Reabilitação Baseado em Técnicas de Captura de Movimento para Tratamento da Lombalgia Mecânica” [7], surgiram com o intuito de estudar o impacto do acesso a tal tecnologia, e incluem sensores mais baratos e simples. Embora mais acessíveis, captadores de movimento, como o *Leap Motion,* ainda configuram gastos de centenas de dólares e uma dificuldade a mais nos tratamentos. Este projeto busca abandonar a necessidade de *hardware* dedicado à captura de movimentos, e baseia-se no reconhecimento de imagens obtidas através da câmera de dispositivos móveis.

A concepção da plataforma busca melhorar a qualidade de vida de uma parcela da população que sofre com a falta de mobilidade dos membros superiores. Diminuir o

preço de tal sistema, bem como a dificuldade de operá-lo, é um modo de democratizar o acesso a tecnologias que promovem o bem-estar e garantem a adesão de pacientes aos tratamentos necessários. O projeto é uma continuação das investidas de outros trabalhos da área, mas tem como meta reduzir gastos por utilizar dispositivos normalmente já presentes no dia-a-dia do usuário.

O produto final é uma arquitetura de plataforma digital que, por definição, é um modelo de negócio que permite a conexão de múltiplos participantes (produtores e consumidores), que interagem entre si para criar e trocar valor [8]. A plataforma proposta une o mercado de jogos com a demanda por soluções de medicina, promovendo benefício mútuo a desenvolvedores, pacientes e profissionais da saúde.

1.4. Organização do Trabalho

Este documento foi organizado segundo as “Diretrizes para Apresentação de Dissertações e Teses” fornecidas pela Escola Politécnica da Universidade de São Paulo. O segundo capítulo, intitulado “Aspectos Conceituais”, trata da apresentação dos conceitos utilizados. O terceiro capítulo, “Tecnologias Utilizadas”, expõe e referencia as técnicas e tecnologias usadas no desenvolvimento do produto. No quarto, “Metodologia do Trabalho”, define-se a divisão do desenvolvimento do projeto em etapas, e apresenta-se os objetivos de cada fase. O quinto capítulo, denominado “Especificação de requisitos de sistema”, descreve os principais requisitos que devem ser saciados pelo sistema ao decorrer do projeto. Nele também são definidos os documentos auxiliares que serão produzidos a fim de possibilitar a execução do trabalho.

O desenvolvimento do produto é tratado a partir do sexto capítulo, “Projeto e Implementação”. Nele, todo o processo de criação e obtenção de resultados parciais e finais é descrito, e são apresentados os diagramas e documentos adicionais citados anteriormente. O sétimo, “Testes e Avaliação”, trata dos testes realizados durante e ao fim do projeto, de modo a avaliar o atendimento aos requisitos previamente levantados. Por fim, o oitavo capítulo, “Considerações Finais”, aborda as conclusões do projeto, e nele discute-se a aplicabilidade, sucessos, falhas e oportunidades para melhorias futuras.

2. Aspectos Conceituais

O aprendizado de máquina (*machine learning*) é a área de estudo da computação que trata de programas capazes de aprender autonomamente [9]. O processo acontece por meio da aplicação de modelos matemáticos e estatísticos sobre dados de entrada, de modo a estimar comportamento futuro. O campo encontra aplicação na automação de rotinas de trabalho, entendimento de fala e imagem, diagnósticos médicos e suporte à pesquisa científica [10].

Aprendizagem profunda (*deep learning*), por sua vez, é a parte de *machine learning* que soluciona problemas intuitivos ao ser humano, mas que não podem ser formalizados matematicamente. O aprendizado ocorre por meio do entendimento da hierarquia de conceitos que definem o problema, estruturados por meio de grafos, e permite à máquina realizar tarefas cujo escopo não pode ser completamente definido e, por isso, que não podiam ser automatizadas até a concepção de tal técnica [10]. Os modelos são, então, produzidos na forma de redes neurais, que simulam o comportamento dos agrupamentos de neurônios humanos, por meio de ferramentas como o TensorFlow [11], que facilitam o treinamento de máquina.

A partir do uso do *deep learning*, é possível treinar uma rede neural que realiza a identificação e classificação de imagens. O modelo é formado a partir do treinamento sobre um conjunto inicial de imagens e, após ser finalizado, pode ser usado para estimar a probabilidade de uma nova imagem apresentar um mesmo ser ou objeto contido em seu conjunto de treinamento [12]. Para realizar um treinamento adequado à tarefa, torna-se necessário um conjunto substancial de entradas para atingir precisão relevante. Essa quantidade depende diretamente da aplicação do modelo e, portanto, é obtida experimentalmente, partindo de um valor recomendado e sendo acrescida conforme necessário para obter os resultados especificados pelos requisitos.

Como alternativa, é possível re-treinar uma rede neural de reconhecimento de imagens já disponível. O conceito envolve descartar a última camada do modelo, mas manter os demais estratos, de modo a utilizá-los para agilizar uma nova sessão de aprendizagem, agora muito mais rápida e eficiente [13]. De tal modo, pode-se usar uma rede pré-treinada, como o Inception V3, da Google [14], produzido a partir da análise do ImageNet, banco de dados com milhares de imagens para mais de 80.000 classes de objetos [15], e especializá-la por meio de treinamento posterior específico com poucas centenas de imagens e alguns minutos. O resultado, todavia, pode ser uma rede mais pesada e lenta do que o necessário e, portanto, esta solução só será utilizada caso a primeira abordagem falhe.

3. Tecnologias Utilizadas

O projeto será dividido em diferentes módulos, que serão explicitados na seção ‘Metodologia do Trabalho’, e cada um necessita da aplicação de tecnologias específicas, discutidas a seguir.

3.1. Unity

Para o desenvolvimento do protótipo do jogo que servirá como base para os testes de validação do sistema, é necessário decidir qual motor de jogo será utilizado. Uma *game engine* é um programa que auxilia a criação, programação e renderização, dentre outros aspectos, de um jogo [16]. Das diversas opções de motores de jogo disponíveis no mercado, optou-se pelo Unity [17].

Tal escolha foi feita pelo fato da plataforma ser distribuída sem custos, possuir ferramentas para o desenvolvimento de aplicativos em sistema *Android*, bem como mais de vinte outras plataformas, e dispor de extensa documentação e comunidade ativa, além de ser usada para a programação de 53% dos 1000 jogos para dispositivos móveis mais lucrativos do mundo [18].

3.2. TensorFlow

O TensorFlow é uma biblioteca *Open Source* desenvolvida pela Google, focada na aplicação do aprendizado de máquina. Sua utilização facilita o desenvolvimento de redes neurais, essenciais para o módulo de reconhecimento de gestos que será utilizado neste projeto. Em particular, o TensorFlow possui uma versão *Lite*, que permite a conversão de um modelo gerado com a versão principal para um mais otimizado a dispositivos *mobile* e *IoT.* Utilizando tal abordagem, é possível produzir resultados muito satisfatórios, mesmo em *hardware* limitado, o que permitirá alcançar melhores precisões em tempo real, de acordo com os requisitos do projeto.

3.3. MobileNets e MobileNetV2

A MobileNets consiste em uma família de arquiteturas de modelos de redes neurais convolucionais para visão computacional [19], com ênfase em otimização para ambientes com grandes limitações de poder de processamento, como dispositivos *mobile* e sistemas embarcados [20]. Esse conjunto pode ser usado por meio do TensorFlow. A adoção desta solução acelera o processo de treinamento de um modelo, ao permitir fazer uso da estrutura já existente, apenas adaptando-a ao objetivo do projeto. Como resultado, tem-se uma rede eficiente, que demanda poucas operações e memória para ser utilizada e é própria para a performance de

dispositivos celulares [20]. Será utilizada sua segunda versão, denominada MobileNetV2, por ser a mais recente e eficiente disponível. Ela, ainda, permite carregar pesos obtidos através de treinamento sobre a ImageNet [21], banco de dados composto por mais de 100.000 classificações, cada uma com mais de 1000 imagens [15], para o caso de ser necessário empregar a técnica de *transfer learning*.

3.4. React

Como forma de demonstrar a aplicabilidade da arquitetura de plataforma digital proposta, será desenvolvido uma plataforma WEB para apresentar os trechos de sessão gravados ao paciente e ao profissional responsável por seu tratamento. Para tanto, optou-se por utilizar o React, biblioteca Javascript para criação de interfaces de usuário, recomendado para a criação páginas interativas [22] e preferência de membros do grupo de desenvolvimento do projeto.

3.5. HTTPS

O HTTPS (Hypertext Transfer Protocol Secure) é um protocolo de comunicação online que protege a integridade e confidencialidade dos dados trafegados entre cliente e servidor [23]. Baseado no HTTP, adiciona uma camada de segurança TLS (Transport Layer Security), que encripta a conexão e verifica a autenticidade dos nós por meio de certificados. No contexto do projeto, será utilizado para transferir, de maneira segura e confidencial, os dados provenientes do módulo de *plug-in* no jogo para o servidor da plataforma médica, assim como entre o servidor e a interface de usuário da página WEB.

3.6. Flask

Flask é um *framework* para Python, utilizado no desenvolvimento de aplicações WEB. Sua estrutura acompanha o mínimo de componentes possível, para tornar seu uso rápido e escalável. É ideal para realizar as operações de *backend* de servidores WEB, como é o caso da plataforma médica que será projetada.

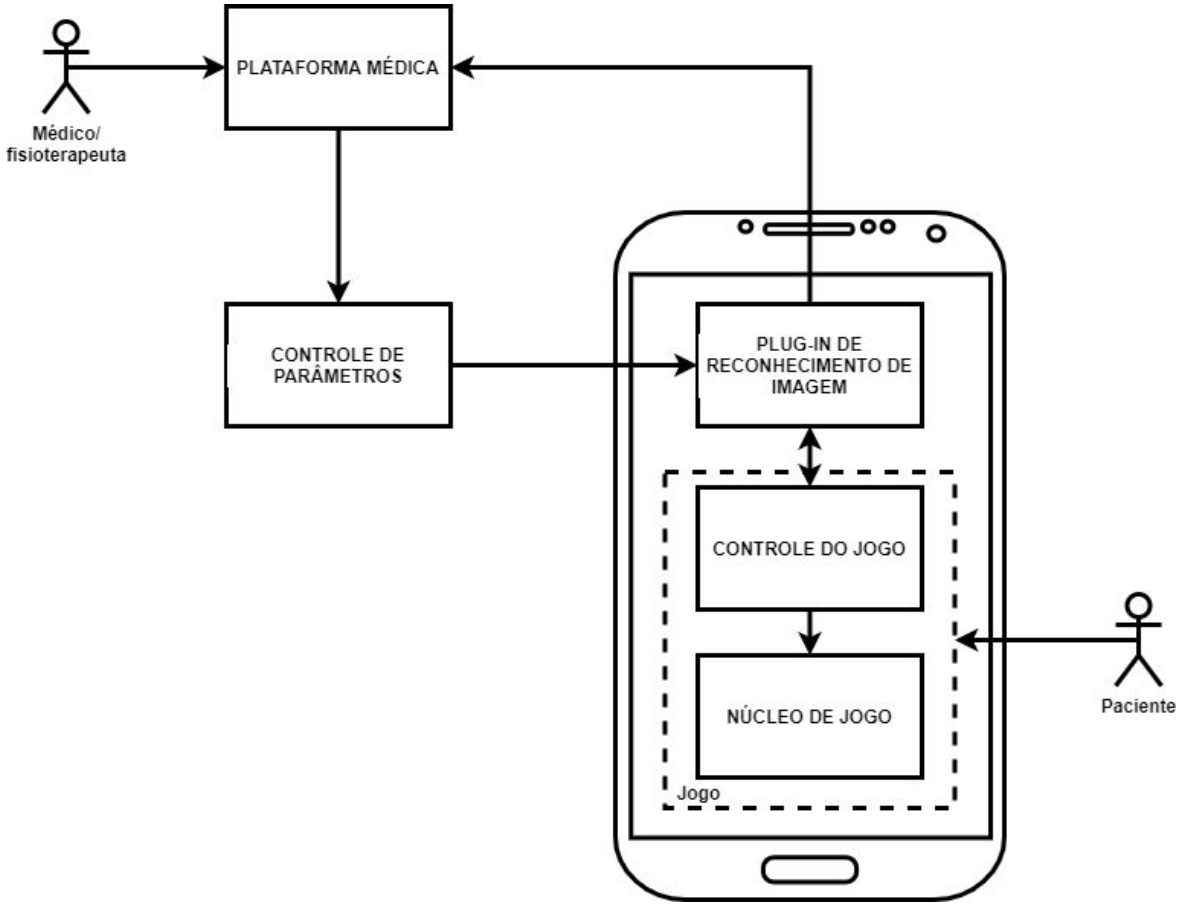
4. Metodologia do Trabalho

Nesta seção, são explicitadas as etapas e práticas de desenvolvimento que permitiram o avanço colaborativo do projeto entre os membros do grupo.

4.1. Etapas de Desenvolvimento

O desenvolvimento do projeto foi dividido em fases de concepção, planejamento e implementação. Primeiramente, durante a concepção, foi definida uma arquitetura do produto para servir de base para as demais etapas do trabalho. Tal arquitetura foi desenvolvida levando em conta os principais módulos do produto final.

Figura 1: Diagrama de blocos da arquitetura base do sistema.



Fonte: Produção própria.

Com base no esquema criado, foi proposta uma sequência simplificada de atividades, na qual definiu-se etapas de desenvolvimento dos módulos específicos e da monografia.

Tabela 1: Atividades gerais do projeto.

Etapa	Objetivos e Produtos Esperados
1	Especificação completa do projeto para a monografia; Rede Neural para reconhecimento de gestos especificados; Funcionamento do reconhecimento em *desktop*;
2	Evolução na monografia; Adição do modelo ao *Unity*; Interação entre o modelo e demais objetos da cena;
3	Monografia completa; Criação de um *plug-in* no *Unity*, que contenha o modelo treinado; Protótipo de jogo para celular que use o *plug-in*; Plataforma WEB para acesso médico;
4	Monografia revisada; Inserção do *plug-in* em um projeto *Unity open-source* ; Página de apresentação do trabalho, pôster e *press release*;

Fonte: Produção própria.

A primeira etapa do projeto é desenvolver o modelo de reconhecimento de imagens para a identificação de gestos simples, como a mão completamente aberta ou fechada, movimentos comuns à prática vista em sessões de fisioterapia para reabilitação motora dos membros superiores [24]. Para tanto, é necessário realizar o estudo de soluções de Visão Computacional disponíveis em formato *Open Source* no mercado, como o TensorFlow, com aplicação a reconhecimento de gestos ou classificação de imagens. Com a solução definida, será feito o treinamento da rede neural, de forma a adaptá-la ao contexto do projeto e, para tanto, deverá ser reunido um banco de dados de imagens dos padrões a serem reconhecidos. Tal banco será composto por imagens obtidas da Internet e também por fotos capturadas pelos membros do grupo. Uma vez desenvolvido o módulo para computador pessoal (*desktop*) com a rede neural treinada, testes serão aplicados para validar seu funcionamento e acurácia, bem como o tempo de resposta.

A segunda etapa de desenvolvimento consiste na adição do modelo ao *Unity*, para garantir que possa ser utilizado em projetos de jogos criados na *game engine*. Para tanto, o modelo deverá ser acoplado ao restante do projeto, e precisará de acesso à câmera do dispositivo. Em seguida, deve-se implementar uma interface para repasse das predições aos controles da cena. Assim, o resultado de cada classificação pode ser convertido em uma ação dentro do jogo. Todos os avanços serão, então, documentados na monografia.

A terceira etapa, por sua vez, inicia-se com a modularização do reconhecimento de imagens dentro do *Unity*, por meio da criação de um *plug-in*. Com ele, será possível integrar projetos de jogos à plataforma de reabilitação fisioterápica, sem complicações de transcrição de código. O desafio seguinte é gerar um protótipo que funcione em *hardware* de dispositivo móvel, para validar seu uso em celulares e *tablets*. Trechos de cada sessão de uso do aplicativo serão capturados e enviados a um servidor, e os vídeos poderão ser acessados pelo médico responsável pelo tratamento do paciente. Uma plataforma WEB, portanto, deve ser desenvolvida para demonstrar tal funcionalidade, que futuramente poderá ser acoplada a serviços de hospitais e clínicas especializadas. Por meio de tal plataforma, também, o profissional poderá ajustar parâmetros relacionados ao exercício do paciente em específico, para adaptá-lo melhor às suas condições e necessidades. Ao fim deste período, planeja-se ter todas as etapas do desenvolvimento já documentadas.

Na quarta etapa, por fim, o módulo de reconhecimento de imagens será adicionado a um jogo mais complexo, desenvolvido pela comunidade em modelo *open source*, para demonstrar as capacidades de compatibilidade com *software* terceiro e obter melhores resultados, de modo a tornar a experiência da fisioterapia mais lúdica e empolgante para o paciente.

4.2. Divisão e Organização do Trabalho

Para o desenvolvimento do projeto, serão aplicados os conceitos de metodologia ágil. Por tal motivo, não será elaborado um cronograma de atividades detalhado no início do projeto. A sequência de tarefas fornecida na ‘Tabela 1’ será usada para guiar as entregas progressivas. Utilizando estratégias do *framework* de organização e gestão de trabalho Scrum, cada *Sprint*, período de uma semana (durante a etapa de desenvolvimento), será marcada por reuniões realizadas por meio de chamadas de vídeo, realizadas no início - para definição de histórias a serem projetadas - e ao fim - para apresentação dos resultados entregáveis. O *backlog* do projeto, bem como a lista de tarefas atuais e finalizadas, será mantido virtualmente por meio da plataforma Trello, que utiliza o *layout* de quadro Kanban para auxiliar o gerenciamento de projetos.

O projeto desta monografia aborda diversos temas da Engenharia de Computação, como o desenvolvimento de plataforma WEB para interação com usuário, servidores para armazenamento de dados, reconhecimento de imagens por meio de modelos de redes neurais e elaboração de jogos. Por tal motivo, cada história da *Sprint* será disponibilizada e alocada aos membros do grupo de modo a aproveitar os conhecimentos técnicos específicos a cada um. O objetivo é entregar o produto de maior valor possível ao fim do período estipulado e, por isso, a distribuição eficiente de tarefas é de suma importância para o desenvolvimento.

5. Especificação de Requisitos de Sistema

Tendo em vista o objetivo a ser atingido pelo projeto, foram definidos os requisitos funcionais e não funcionais necessários para o cumprimento das metas propostas. Esta seção possui descrições de tais requisitos, divididos entre os três módulos principais do produto.

5.1. Requisitos Funcionais

Em um sistema, os requisitos funcionais representam as atividades que deve realizar, ou seja, suas funcionalidades e como são executadas. Para a plataforma de complemento fisioterápico, estas funcionalidades serão elencadas por módulo - *Plug-in*, Jogo e Plataforma Médica.

5.1.1. Módulo de *Plug-in*

Considerando a arquitetura definida na seção 4, o módulo de *plug-in* é constituído pelos componentes de reconhecimento de imagem, e possui interface com a plataforma WEB e o controlador de parâmetros. Por ser o foco da implementação, este módulo possui as funcionalidades mais importantes para o funcionamento da plataforma, sendo necessária sua integração com as demais partes do sistema. A seguir, foram elencadas suas funcionalidades:

- **Captura de imagens:** o módulo de *plug-in* deve obter acesso à câmera do dispositivo móvel e mantê-lo ininterruptamente durante o andamento de cada partida do jogo. No decorrer de tais partidas, deve capturar imagens constantemente, e enviá-las para a classificação dos gestos.

- **Classificação de imagens:** ao receber uma imagem, o *plug-in* deve ser capaz de classificá-la como uma mão aberta ou fechada. Esta classificação dependerá de parâmetros customizáveis pelo fisioterapeuta, obtidos a partir de requisições HTTPS ao servidor da plataforma WEB.

- **Captura de vídeos para análise médica:** o módulo de *plug-in* deve, durante uma sessão de fisioterapia, gravar vídeos dos movimentos das mãos do paciente e enviá-los aos servidores da plataforma WEB, onde ficarão armazenados. Cada vídeo pode ser capturado a qualquer momento da sessão, e possui duração de 5 segundos. Para isso, o módulo deve ser capaz de enviar os arquivos de vídeo por meio de requisições HTTPS.

- **Comunicação com o jogo:** as informações geradas pelo classificador de gestos devem ser disponibilizadas de forma estruturada, por meio de uma

interface, aos demais componentes do jogo ao qual o *plug-in* foi adicionado. Dessa forma, o módulo deve ser construído tendo em mente a integração com a *game engine* utilizada, visando facilitar a transmissão dos dados entre os módulos.

- **Instalação em projetos distintos:** Para poder ser utilizado em projetos de desenvolvedores terceiros, o módulo deve ser empacotado de modo a propiciar sua inserção em outro projeto *Unity*, novo ou já existente, respeitando restrições de versionamento da *game engine*.

5.1.2. Módulo da Plataforma Médica

O módulo da plataforma médica deve possuir funcionalidades que permitam a verificação do funcionamento da arquitetura da plataforma digital. Tais funcionalidades foram listadas a seguir:

- **Autenticação de usuários:** objetivando manter a privacidade do paciente, a plataforma deve possuir mecanismos de autenticação do usuário, de modo que apenas ele mesmo e o responsável pelo seu tratamento tenham acesso às informações coletadas durante uma sessão de fisioterapia. Esse mecanismo será construído com base em autenticação de fator único, por meio de *login* com email e senha cadastrados.

- **Recebimento e disponibilização dos vídeos armazenados:** a plataforma médica deve ser capaz de receber vídeos provenientes de jogos que utilizam o *plug-in* produzido, armazená-los e disponibilizá-los por meio de sua interface WEB aos usuários com permissão correspondente.

- **Interface para controle de parâmetros:** a partir da plataforma, o responsável pelo tratamento deve ser capaz de alterar os parâmetros customizáveis do controle de jogo, de modo a customizá-lo para melhor experiência do paciente. Esta customização será feita através da escolha de configurações que controlam a dificuldade do jogo.

5.1.3. Módulo do Jogo

O projeto de jogos voltado à área da saúde enquadra-se na classificação de *Serious Games* (Jogos Sérios) [25], [26]. Desafios, como abrangência e especificidade ao domínio da aplicação, afetam o ciclo de desenvolvimento de *softwares* deste tipo [27]. Alguns parâmetros a atentar-se no contexto da criação de Jogos Sérios com propósito de melhoria da saúde são recompensa, desafio, *feedback*, interatividade, mecânicas e objetivos claros e socialização [2], [27]. A aplicação de tais quesitos no

projeto das funcionalidades do jogo pode trazer maior motivação ao jogador, levando ao aumento da adesão do paciente ao tratamento.

Dessa forma, o projeto das *features* de um jogo é um processo detalhado e a atenção a certos aspectos o torna mais efetivo. Entretanto, no momento inicial do desenvolvimento da plataforma de complemento fisioterápico, o módulo de jogo criado terá por objetivo demonstrar as capacidades da arquitetura, e não adequar-se à proposta de produto final disponibilizado a clientes. Já é possível, todavia, identificar requisitos que são comuns, ou pelo menos semelhantes, ao trabalho que poderá ser realizado por desenvolvedores terceiros.

- **Integração com reconhecimento de gestos:** um dos componentes internos do módulo de jogo é o 'controlador de jogo'. Este componente é responsável por criar comandos para o jogo a partir da classificação do gesto. Para tanto, é necessária a implementação da integração entre estes dois componentes, por meio de funções disparadas pelo evento de classificação de imagem.

- **Recompensa:** o jogo deve ser capaz de recompensar o jogador por sua performance. Abordagens para atingir tal objetivo são intrinsecamente relacionadas ao jogo desenvolvido, e podem ser compostas por uma ou mais das seguintes estratégias: progresso, dado por níveis diferentes; pontuação obtida, com registro dos melhores resultados; conquistas, alcançadas ao cumprir requisitos pré-determinados, de curto ou longo prazo.

- **Apresentação da captura de imagens:** é importante dar ao usuário o *feedback* visual da captura de imagens realizada pela câmera, de modo a possibilitar ajustes no posicionamento e inclinação da mão durante a sessão. Para isso, objetiva-se expor, em uma região da tela, a imagem recebida da câmera do dispositivo.

5.2. Requisitos Não-Funcionais

Os requisitos não-funcionais de um sistema representam seus aspectos comportamentais, ou seja, a forma como ele executa as ações, bem como limitações que restringem suas funções. Tais requisitos são listados abaixo, tendo em vista o sistema como um todo.

- **Confidencialidade:** é a garantia de segurança dada por um sistema de que os dados que nele trafegam só poderão ser acessados por usuários autorizados [28]. Isso é de particular importância, pois os dados coletados na plataforma, como os vídeos das sessões, são protegidos pelo sigilo médico-paciente. Dessa forma, o acesso a essas informações deve ser feito

apenas mediante autenticação do usuário, o que afeta a implementação do servidor e o método de transferência de dados entre ele e o módulo de *plug-in*. Por tal motivo, a transmissão dos vídeos e estatísticas deverá ser implementada utilizando o protocolo HTTPS, e o acesso às funcionalidades de gravação e armazenamento de vídeo só será permitido mediante autenticação no jogo e na plataforma WEB.

- **Privacidade:** a privacidade se refere à possibilidade do usuário controlar os dados que são enviados, no que se refere ao seu destino, uso e quem pode acessá-los [29]. O sistema deve, então, garantir que apenas o profissional de saúde e seu paciente tenham acesso aos dados gerados no uso da plataforma, além de informar ambas as partes sobre a utilização desses dados. Ao profissional de saúde, deve ser apresentado um termo de uso, pelo qual aceita utilizar as informações geradas apenas para fins médicos e, para o paciente, aceitação semelhante também será requerida. O vínculo entre os dois atores deverá ser estabelecido através de um convite na plataforma WEB, proveniente do médico e direcionado ao paciente, e só será firmado mediante aprovação direta do segundo. Para existir um controle dos dados gerados, gravações só serão realizadas mediante autenticação no jogo com a mesma conta usada na plataforma WEB.

- **Disponibilidade:** o usuário final deve ser capaz de utilizar o serviço do módulo (por meio do jogo) a qualquer momento, mesmo na ausência de conexão com a Internet, com exceção de jogos em que o desenvolvedor terceiro requeira tal conexão. Para tanto, em tal caso, opta-se por permitir o uso da rede neural, abrindo mão da comunicação com o servidor e, por conseguinte, da gravação e envio de vídeos da sessão. Quanto à plataforma WEB, o protótipo produzido pelo grupo será hospedado localmente, para possibilitar a demonstração. No caso de implantação futura em um servidor cliente, a disponibilidade dependerá da infraestrutura fornecida, sendo idealmente constante.

- **Acurácia:** a capacidade de realizar predições corretas relaciona-se diretamente com a acurácia do reconhecimento de comandos do paciente para o jogo. Erros de reconhecimento do gesto realizado resultam em insatisfação e frustração. Precisões muito altas, todavia, são custosas em quesito de desempenho na produção (processo de obtenção de lotes de treinamento e o processo de treino em si) e uso do modelo. Como métrica para este projeto, pretende-se obter um valor de 90% de precisão para a rede neural usada sobre um conjunto de validação.

- **Portabilidade:** em um contexto de heterogeneidade nos modelos de aparelhos celulares [30], a capacidade de um sistema se adaptar ao ambiente no qual está inserido é essencial. Com isso em mente, optou-se por desenvolver a plataforma para o sistema operacional *Android*, que representa mais de 70% do mercado de celulares [31]. Utilizando este sistema operacional e as ferramentas da *game engine* Unity, é possível criar um sistema capaz de adequar-se ao *hardware* em que é executado. Em uma possível evolução do projeto, também será possível incluir celulares com sistema operacional iOS. Produzir aplicativos para o sistema da Apple, todavia, é um processo com significativa barreiras de entrada, tanto financeiramente (é necessário adquirir um computador da marca para compilar o produto final) quanto tecnicamente, pela necessidade de atender requisitos rígidos da plataforma proprietária e, por isso, tal tarefa não se enquadra no escopo do projeto atual. A plataforma WEB, por sua vez, será desenvolvida para acesso por meio de navegadores para *desktop* e *notebooks*.

- **Desempenho:** A análise de uma imagem pela rede neural pode ser custosa demais, dependendo das especificações do dispositivo. Para reduzir tais impactos, pretende-se diminuir ao máximo a complexidade da rede e seu custo de processamento, treinando um modelo voltado ao uso em celulares, por meio do *TensorFlow Lite*. Para a câmera, a frequência de captura pretendida é de 60 *frames* por segundo, valor correspondente à taxa de atualização da tela de boa parte dos celulares atuais. Como esse parâmetro, todavia, é limitado também pelo *hardware* presente no dispositivo, será usada a configuração nativa mais próxima. Para fazer uso das imagens coletadas, portanto, objetiva-se que a rede seja capaz de realizar as mesmas 60 verificações por segundo, conferindo maior fluidez aos controles do jogo. Quanto ao espaço, é comum que jogos atuais ocupem centenas de MegaBytes. Por isso, considera-se aceitável que o *plug-in* não ultrapasse 10MB.

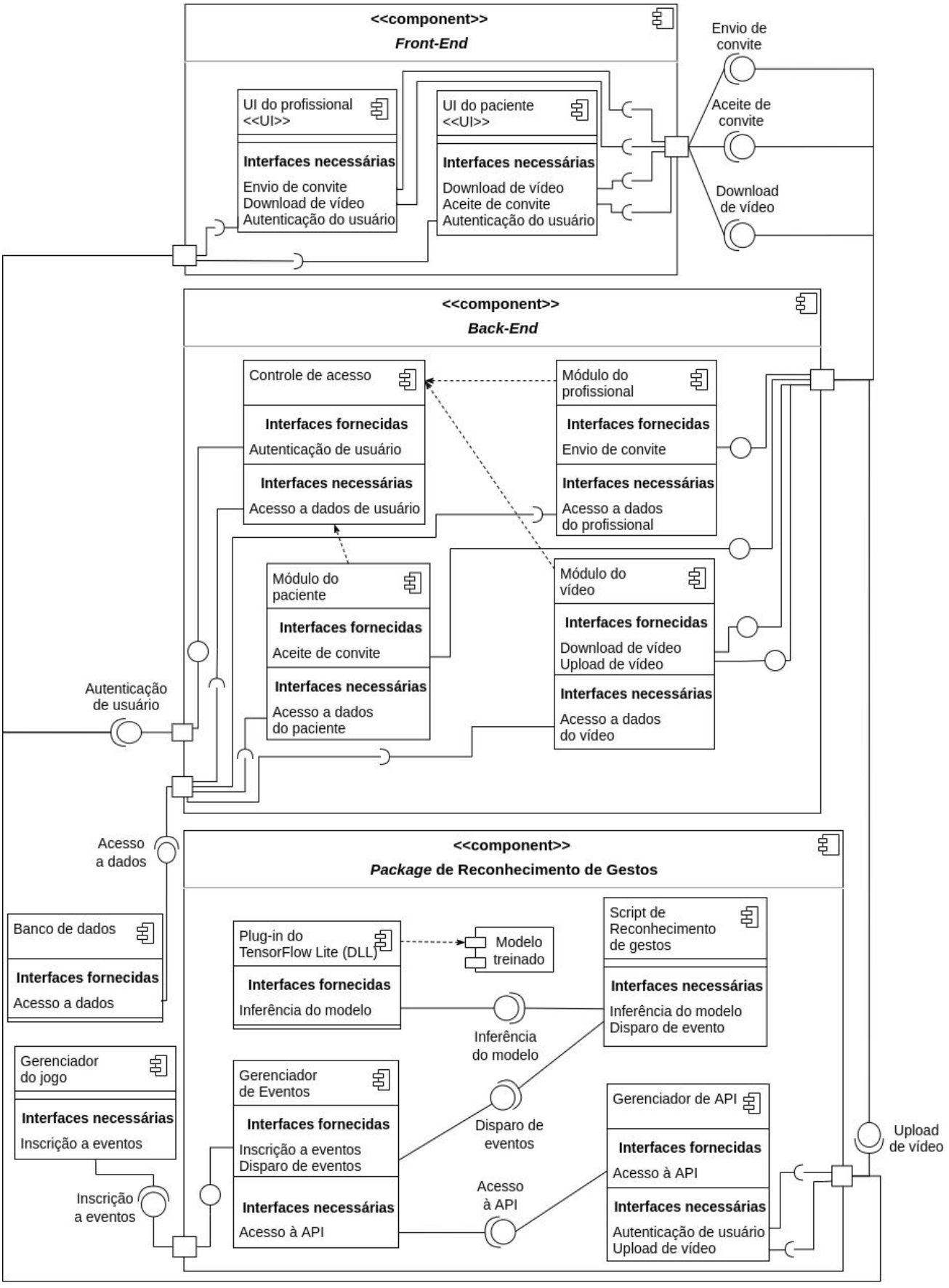
5.3. Arquitetura do sistema

Uma vez definidos os requisitos da plataforma, é necessário detalhar a estrutura dos diversos módulos da plataforma, ressaltando sua componentização e intercomunicação. Para tanto, optou-se pela representação em Diagrama de Componentes, artefato presente na descrição do UML [32]. A ênfase deste diagrama não é a funcionalidade do sistema, mas sim uma descrição estrutural de seus componentes e as interfaces de comunicação, representadas em notação *Lollipop* no diagrama. Nele, são representados os três grandes componentes com funcionalidades desenvolvidas neste projeto - o módulo de *plug-in* do jogo, o

front-end e o *back-end* da plataforma - e suas interações com os elementos de infraestrutura (banco de dados) e com os sistemas externas que o utilizarão (o controlador do jogo). Tal diagrama é apresentado na Figura 2 a seguir.

Além do Diagrama de Componentes, durante a etapa de execução do projeto serão elaborados o Diagrama de Classes para a plataforma WEB e o Diagrama de Casos de Uso. Ambos serão usados para guiar a prototipação dos módulos propostos.

Figura 2: Diagrama de Componentes da plataforma



Fonte: Produção própria

6. Projeto e Implementação

6.1. Treinamento da Rede Neural

A primeira etapa de implementação do projeto baseia-se no treinamento de uma rede neural, capaz de identificar os gestos pretendidos por meio do processo de classificação de imagens em tempo real.

6.1.1. Obtenção de Imagens para o Treinamento

Para o primeiro protótipo, foram usadas cerca de 200 imagens para cada classe. Pode-se medir, ao fim, a eficácia e tamanho do modelo gerado, e adicionar novas imagens ao lote de treinamento caso a acurácia não seja satisfatória. Duas abordagens foram usadas para popular o lote de treinamento: a captura de fotos realizada pelos próprios membros do grupo e o *download* de imagens disponíveis na Internet. O primeiro grupo de imagens tem por objetivo dar foco às posições de mão esperadas durante o uso do produto final, levando em consideração a distância da câmera, fundos neutros e caóticos e possíveis rotações de pulso. O segundo, por sua vez, teve como meta trazer maior variedade física, para reduzir o enviesamento do lote causado pelas características fisionômicas da pequena amostra composta pelos integrantes do grupo.

6.1.2. Formatação das Imagens

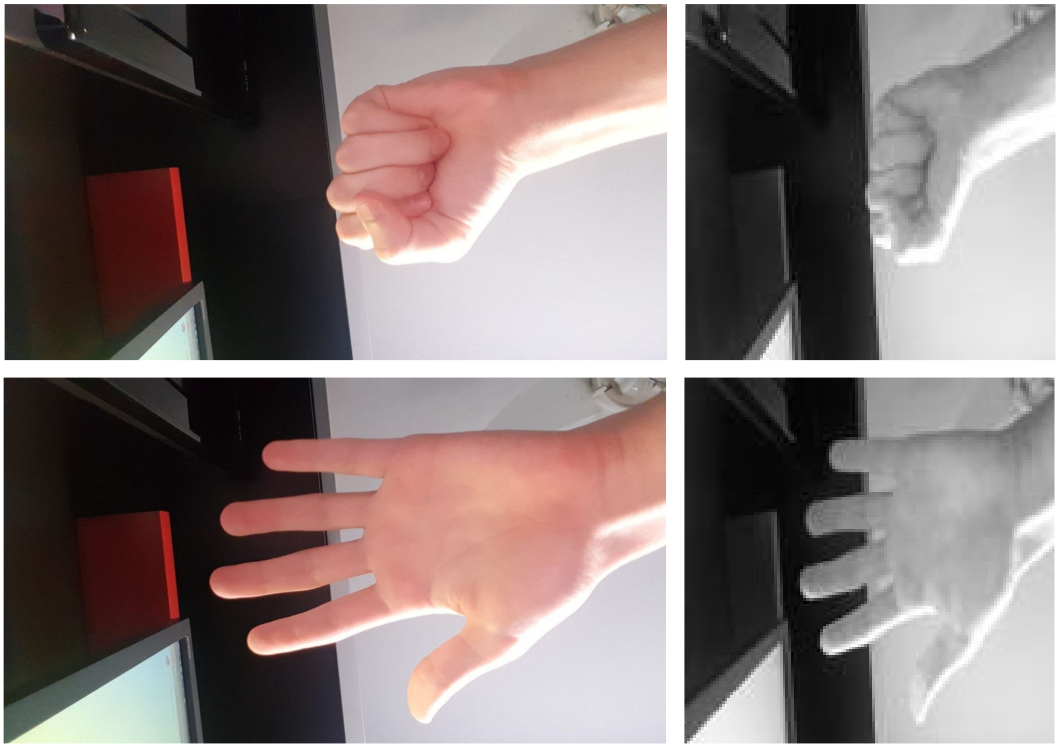
Uma das mais importantes características de uma rede neural é ser capaz não só de aprender sobre seu conjunto de dados de treinamento, mas também de aplicar tais conhecimentos para entradas desconhecidas, habilidade denominada ‘generalização’ [10]. Para isso, é necessário evitar casos de ‘overfitting’, identificados pela diferença entre o êxito de predições sobre entradas de treino e de imagens novas [10]. São causados pela falta de diversidade no dataset de treinamento, ou seja, as imagens disponíveis não são capazes de representar toda a pluralidade presente nos testes de ‘mundo real’. Como exemplo, caso todas as imagens do lote de treinamento possuam fundo branco, o modelo pode apresentar resultados pouco satisfatórios caso seja usado para identificar imagens com fundos coloridos.

Além disso, variáveis irrelevantes à tarefa que constantemente aparecem em um subconjunto de imagens, percebidas e confundidas como padrões importantes durante o treinamento, bem como ruídos - imagens presentes no lote de treinamento que não descrevem o padrão proposto e, por isso, que não deveriam estar listadas - resultam em um modelo que descreve relações aleatórias, ao invés das características pretendidas [33]. Por tal motivo, é necessário remover o máximo de informações irrelevantes das imagens de treinamento, como componentes de cor,

detalhes pequenos, tamanhos de imagens e orientações. Também é imperativo checar todas as imagens, para garantir a corretude dos exemplos. Para isso, deve-se garantir que toda imagem pertencente à classe intitulada ‘mão fechada’ realmente apresente a foto de uma mão fechada, e assim por diante.

Deste modo, todas as imagens obtidas foram formatadas. Optou-se por remover as componentes de cor ‘RGB’, mantendo apenas a escala de cinza para preservar o contraste entre as mãos e o fundo. Todas as imagens foram, também, redimensionadas para um tamanho padrão de 128 por 128 pixels. O valor foi escolhido de modo a manter um nível suficiente de nitidez e reduzir o tamanho das entradas, diminuindo o tamanho do modelo gerado e tornando a análise das entradas mais rápida. Durante o treinamento, algumas das imagens serão rotacionadas, ampliadas, distorcidas e/ou terão o valor de seus pixels invertidos, para conferir maior variedade ao dataset e reduzir tendências. As imagens produzidas a partir das originais são criadas e removidas a cada época de treinamento, de forma aleatória, para reduzir ao máximo qualquer enviesamento e conferir ao lote maior diversidade. Na ‘Figura 3’, tem-se exemplos das transformações aplicadas às imagens. À esquerda, imagens capturadas pelo grupo, a primeira de uma mão fechada e, a segunda, de uma mão aberta. Ambas apresentam cor e alta resolução. À direita de cada uma, sua equivalente formatada, apenas com escala de cinzas e baixa resolução.

Figura 3: Exemplos de formatação das imagens de treinamento.



Fonte: Produção própria.

O código produzido e utilizado para fazer a formatação encontra-se no Apêndice A deste documento e no repositório usado pela equipe [34].

6.1.3. Elaboração do Treinamento

Deseja-se que o treinamento seja suficientemente longo para que o modelo aprenda a executar sua tarefa, ou seja, pretende-se garantir que as informações captadas e modeladas sejam suficientes para perceber as características que distinguem as posições de mão pretendidas. Para evitar o *'underfitting'*, definido como a incapacidade de reconhecer os padrões pretendidos [33] e identificado pela baixa acurácia das predições sobre entradas de treinamento [10], o aprendizado foi realizado com um máximo de 10.000 épocas, aliado ao uso da técnica de *'Early-Stopping*', que encerra o processo antecipadamente caso não haja melhoria nos resultados [35] por, no caso, 300 épocas seguidas. Tais valores foram obtidos experimentalmente, de modo a obter resultado com acurácia satisfatória em um curto período de treinamento (alguns minutos).

Durante o treinamento, o modelo avalia todo o *dataset* de treino a cada época, repartido em lotes, denominados *batches*. O tamanho dos lotes é fixado ao início do processo de aprendizado e, para o problema em questão, o ideal é utilizar grupos de 32 imagens [36] [37].

Para aprimorar a escolha dos pesos em cada neurônio do modelo durante o treino, é necessário optar por um algoritmo de otimização. Para problemas de classificação de imagens, como é o caso, indica-se utilizar o *Adam* (Adaptive Moment Estimation), que apresenta os melhores resultados quando comparado às demais alternativas [38]. Para calcular a taxa de erro (*loss)* durante o avanço das épocas, optou-se pelo método de *Categorical Cross-Entropy*, que é o mais indicado para categorização com múltiplas opções. Embora o modelo do produto a ser desenvolvido aborde duas possíveis posições e, portanto, duas opções de saída, trabalhos futuros podem envolver variações adicionais de posição da mão como melhoria ao sistema, ou seja, diversas possibilidades de saída para adequar-se melhor a cada paciente.

O algoritmo desenvolvido para realizar o treinamento encontra-se no Apêndice B do relatório, bem como no repositório usado pelo grupo [34]. Ao fim do processo de treinamento, testes mais específicos devem ser realizados para determinar a taxa de acertos, e seus resultados indicarão a necessidade de alterar tal valor.

6.1.4. Treinamento da Rede Neural

Utilizando o código desenvolvido na etapa anterior, o treinamento do modelo de rede neural foi realizado localmente, utilizando o computador de um dos membros do grupo. O suporte dado pelas bibliotecas de computação paralela da NVIDIA

promove maior desempenho às aplicações de aprendizagem de máquina, e possibilitou a execução do programa em aproximadamente vinte minutos com o auxílio da placa de vídeo. As etapas e resultados do treinamento foram monitorados com o auxílio da plataforma *Weights and Biases* [39], que registra valores parciais de acurácia do modelo e uso de recursos, dentre outros. A seguir, na 'Tabela 2', o compilado de métricas obtidos ao fim do treinamento.

Tabela 2: Métricas do treinamento.

Métrica	Valor
Épocas de treinamento	2404
Acurácia sobre conjunto de treinamento	98,60%
Acurácia sobre conjunto de validação	92,31%
Perda sobre conjunto de treinamento	0.06655
Perda sobre conjunto de validação	0.19520

Fonte: Produção própria.

A acurácia, aqui, é o fator de maior interesse para análise dos resultados do treinamento. O valor obtido para o conjunto de validação é um indicativo da capacidade de generalização do modelo criado. A seguir, na 'Figura 4', o gráfico que mostra o avanço da acurácia sobre o conjunto de validação com o passar das épocas de treino.

Figura 4: Gráfico da acurácia sobre o conjunto de validação.

Fonte: Dashboard do Weights and Biases para o projeto.

Como pode-se observar, o valor obtido, de 92,31%, está acima dos 90% previstos pelos requisitos não funcionais do projeto e, por esse motivo, o modelo produzido será considerado suficiente para a aplicação.

6.2. Desenvolvimento do *Package* para *Unity*

Para a inserção e modularização do código responsável pelo reconhecimento de gestos em projetos *Unity*, foi utilizada uma abordagem incremental, ou seja, o processo foi subdividido em funcionalidades menores, cada uma requerendo pesquisas e testes das possíveis soluções. As tarefas estão destacadas a seguir.

6.2.1. Criação de *package* simples

No contexto da *game engine Unity*, existe uma série de maneiras de se modularizar um trecho de jogo, de modo a permitir sua disponibilização à comunidade de desenvolvedores. As duas principais abordagens são o *Unity Plug-in* e o *Unity Package*.

De forma genérica, um *plug-in* é um trecho de *software* que, quando instalado em um programa, permite a extensão e melhoria de suas funcionalidades [40]. Entretanto, dentro do contexto do motor de jogos em questão, *plug-ins* adquirem um significado particular: representam bibliotecas compiladas que disponibilizam uma interface de funções que podem ser utilizadas pelo jogo criado [41]. Tais bibliotecas possuem duas categorias: *plug-ins* gerenciados e *plug-ins* nativos. Ambos devem ser compilados em ambiente externo ao *Unity*, resultando em arquivos de formatos específicos para cada plataforma - *Dynamic-Link Libraries* (DLLs) no sistema Windows ou arquivos '.so' em arquiteturas Unix e Android - para, em seguida, serem adicionados ao projeto do jogo no momento de ligação (*Linking*).

Por ser um trecho de código externo à *game engine*, os *plug-ins* não possuem limitação de linguagem, podendo ser codificados em C#, como os demais programas gerados no *Unity* (estes são os *plug-ins* nativos), ou em outras linguagens (*python, C++,* entre outras) [41]. Este segundo tipo representa o *plug-in* gerenciado, que deve ser compilado como *assemblies* para *.NET*, ou seja, um programa capaz de ser executado pelo ambiente de *runtime* .NET.

O *Unity Package*, por sua vez, é uma estrutura geral disponibilizada pelo motor de jogos que permite a exportação/importação de arquivos de quaisquer tipos entre projetos, podendo disponibilizar diversos módulos que implementam funcionalidades baseadas no *core* da *game engine* [42]. Com ele, é possível compartilhar *assets* de jogos (como modelos e texturas), bem como *scripts* de comportamento dos *assets*. Os *packages* criados, então, podem ser disponibilizados via ferramenta de versionamento ou via *Unity Asset Store* [43], uma plataforma de obtenção de

pacotes de outros desenvolvedores mantida pela própria *Unity*. Além disso, o ambiente de desenvolvimento *Unity* fornece uma ferramenta chamada *Unity Package Manager* (UPM), que simplifica o processo de importação de pacotes.

Para o contexto deste projeto, optou-se por uma abordagem mista, em que o código relativo às bibliotecas do TensorFlow estão incluídos em um *plug-in*, enquanto as funcionalidades relativas ao jogo (que fazem uso das bibliotecas), estão estruturadas em um *package*. Dessa forma, é possível adicionar o *plug-in* dentro de um pacote e disponibilizá-lo via ferramenta de versionamento, como o *Git.*

6.2.2. Obtenção do *plug-in* com acesso à câmera e adição do modelo

Uma vez decidida a estrutura de disponibilização do *software* construído, estudou-se a forma de adicionar os *plug-ins* referentes ao TensorFlow Lite em um projeto. Estes *plug-ins* são essenciais para o reconhecimento de gestos, pois eles possuem as funções utilizadas no processo de inferência com base no modelo fornecido.

Para isso, utilizou-se como base o repositório *TensorFlow Lite for Unity Samples*, disponível em [44]. Este projeto contém exemplos de como utilizar o TensorFlow Lite dentro do motor de jogos e, portanto, já traz as bibliotecas da ferramenta compiladas para as diversas plataformas (Android, Windows, MacOS, iOS e Linux). Tais bibliotecas estão disponíveis na forma de *package* no projeto base.

A partir da estrutura fornecida, foi criado um programa que implementa o acesso à câmera do dispositivo e reconhece o gesto a partir do modelo treinado na seção anterior. Mais detalhes desta implementação encontram-se no Apêndice C deste documento.

6.2.3. Sistema de eventos

Com o projeto *Unity* integrado ao interpretador do TensorFlow Lite, o próximo passo é traduzir o resultado da inferência para um controle válido para o jogo. A solução escolhida foi a implementação de um sistema de eventos. Um sistema de eventos é uma variante do padrão de design *Publisher-Subscriber*, no qual uma entidade (*Publisher*) dispara um evento que é recebido por um *broker* e este *broker* notifica outras entidades (*Subscribers*) que esperavam pelo seu disparo [45]. Um evento é um objeto ou mensagem usado para notificar outros módulos de uma mudança de estado [46].

Este é um *design pattern* extremamente vantajoso para o contexto da plataforma, pois, no padrão, o *Publisher* não possui informações sobre os *Subscribers* que esperam por determinado evento. As funcionalidades do plug-in devem operar de modo independente ao jogo em que será inserido. Logo, a construção de um

sistema de eventos que dispara mensagens quando ocorre uma detecção de abertura/fechamento de mão, acompanhado do seu controlador, é uma solução válida para traduzir o gesto em controle do jogo. Em particular, é possível definir os eventos de forma assíncrona, o que significa que o *Publisher* pode continuar seu processamento, sem que seja necessário esperar pelo fim da reação do *Subscriber*. Isso é um fator importante, pois garante que o interpretador detectará os gestos de forma contínua, enquanto os demais componentes reagem à mensagem recebida.

O motor de jogos já possui uma infraestrutura de eventos. Porém, ela é voltada a ações pré-estabelecidas, como detectar o pressionamento de um botão. Assim, deve ser desenvolvido um controlador de eventos customizado, que se baseia nas classes fornecidas pelo *Unity*, mas executa ações diferentes.

Considerando as funcionalidades determinadas para o *package*, percebe-se a necessidade da criação de quatro eventos, acionados em momentos diferentes. Sua integração com o controlador deve permitir não apenas a detecção do evento, mas também o envio de alguns parâmetros, para o correto processamento da ação a ser realizada.

- **Detecção de gesto:** este evento é acionado após o resultado de cada inferência do interpretador. Assim, ele será gerado pelo próprio *script* de reconhecimento de gestos para ser enviado a uma entidade do jogo. Como parâmetro, deve enviar se o gesto detectado é de “mão aberta” ou “mão fechada”;
- **Requisição de autenticação:** este evento é gerado por alguma ação no jogo (como apertar um botão) e surte efeito no módulo de comunicação com a API do pacote desenvolvido. Como parâmetros, deve enviar o email e senha do usuário, bem como a indicação de pretensão de manter a sessão ativa, e resulta no envio da requisição de login ao servidor.
- **Requisição para envio de vídeo:** análogo à requisição de autenticação, este evento é gerado por alguma ação do jogo e surte efeito no módulo de comunicação com a API. Seu acionamento resulta no envio da requisição para armazenar o vídeo da sessão no servidor.
- **Requisição de parâmetros:** análogo às demais requisições ao servidor, este evento deve ser gerado por alguma ação do jogo após a autenticação do usuário e deve obter os parâmetros de customização do jogo, como dificuldade.

Por possuir uma pequena quantidade de eventos gerados, não é necessário implementar um módulo de *broker* com funcionalidades de criar filas de requisição ou executar balanceamento de carga. O processo de desenvolvimento do sistema de eventos é detalhado no Apêndice E deste documento.

6.2.4. Captura de vídeo e comunicação com API

A última etapa do desenvolvimento do *plug-in* (ou *package*) é a criação de uma estrutura de comunicação com o restante da arquitetura, em particular, com a plataforma WEB. Com isso, também mostra-se necessária a implementação de uma forma de captura de vídeos durante uma sessão de exercícios, para que eles possam ser enviados ao profissional de saúde para análise e acompanhamento do tratamento.

Existem alguns pacotes disponibilizados na *Asset Store* que são capazes de gravar vídeos, porém, apresentam limitações que os tornam incompatíveis com o projeto. Por exemplo, alguns são incompatíveis com a plataforma *Android* (como o '*WatchMeAlways Video Recorder'*), outros são pagos (como o '*Android Native Screen Recorder'*) ou necessitam que o usuário pare de jogar para ter acesso à câmera (como o '*Android Native Camera'*). Como o reconhecimento de gestos e a gravação de vídeos devem compartilhar o recurso da câmera, a adoção de dois *packages* que o acessam simultaneamente torna-se inviável. Para evitar este problema, optou-se pelo desenvolvimento de uma solução própria.

A solução proposta baseia-se na captura de *frames* da textura gerada pela imagem do dispositivo de câmera no jogo. Tal captura já era feita, pois os quadros são utilizados como entrada do interpretador para o reconhecimento de gestos. Então, para adequá-la a esta funcionalidade, certos *frames* devem ser salvos, de acordo com a parametrização do início e duração do vídeo.

Com os quadros armazenados, é necessário convertê-los em vídeo. Isso pode ser feito utilizando a biblioteca *FFmpeg.* Esta é uma ferramenta capaz de, dentre outras funcionalidades, converter formatos de vídeo, criar arquivos de áudio e, mais relevante para o projeto, transformar um conjunto de imagens em um vídeo [47]. *Packages* e *plug-ins* que implementam o *FFmpeg* estão disponibilizados em ferramentas de compartilhamento como o *GitHub* ou em sites de empresas de desenvolvimento. Mesmo assim, também possuem algumas limitações como a incompatibilidade com a plataforma *Android* ou o custo de licença do produto. Dessa forma, optou-se por uma solução alternativa em que os *frames* armazenados são enviados ao *back-end*, onde, posteriormente, são convertidos em vídeo por meio da implementação de bibliotecas semelhantes ao *FFmpeg* em *Python* no servidor.

O envio das imagens e das demais requisições necessárias para o projeto, como a autenticação, também deve ser implementado. Esta funcionalidade, porém, pode ser obtida utilizando soluções pertencentes ao *core* do motor de jogos, a saber, a classe '*UnityWebRequest*' [48]. Esta é uma classe do *Unity* responsável pela comunicação HTTP/HTTPS com servidores WEB, e possui a funcionalidade de utilização dos

padrões REST de requisição, ideais para a troca de informações com a API a ser desenvolvida. Os procedimentos para a implementação da captura de vídeo e comunicação com o servidor estão detalhados no Apêndice E deste documento.

6.2.5. Finalização do *package* para reconhecimento de gestos

Com todas as funcionalidades anteriores implementadas e testadas isoladamente, é possível criar o *package* que servirá como produto final desenvolvido neste projeto. Para isso, todos os *scripts* que foram gerados ao longo desta etapa foram unificados em um único projeto, disponibilizado em repositório do *GitHub* (https://github.com/danilosobral/unity-gesture-recognition). Neste projeto, a versão mais atual do *package* está disponível para *download* na área de *Releases*, além de estar acompanhado de um arquivo '*README'*, contendo as instruções de instalação, uso e compilação do jogo.

O *package* fornecido é adaptado apenas para jogos no ambiente *Windows* ou aparelhos *Android* com arquitetura ARM64, por conta de limitações das bibliotecas do TensorFlow Lite, uma vez que os arquivos compilados foram gerados apenas para estas plataformas.

6.3. Desenvolvimento da Plataforma WEB

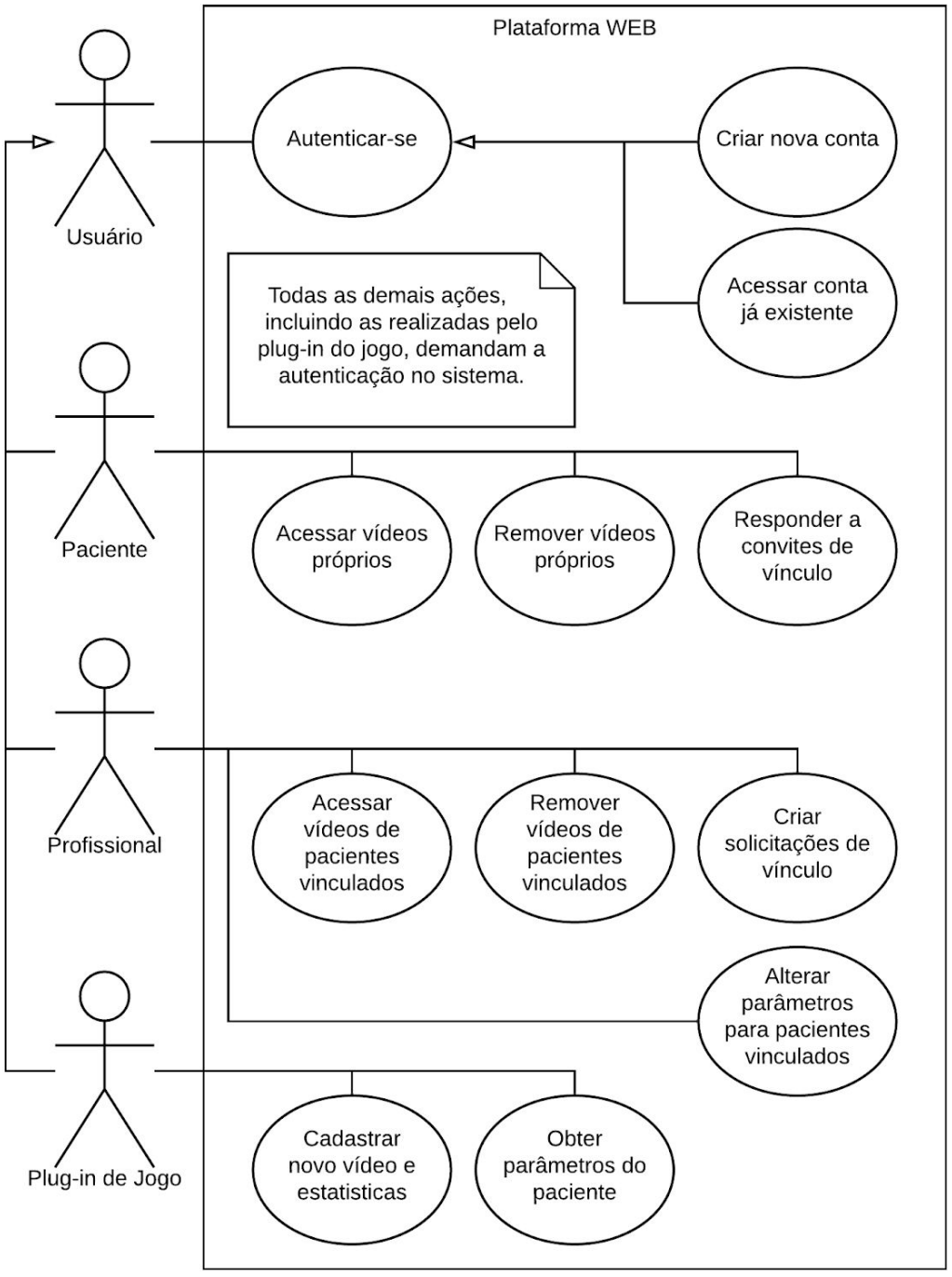
O módulo da plataforma WEB tem por objetivo demonstrar a capacidade de interação de pacientes e profissionais com os dados coletados durante as sessões de jogos. Para o desenvolvimento, serão necessárias as etapas de definição do sistema, prototipação da interface e o desenvolvimento do *front-end* e *back-end*.

6.3.1. Definição do Sistema

Para iniciar o desenvolvimento do sistema, primeiro é necessário definir exatamente do que ele será capaz, e quem poderá com ele interagir. Começa-se, então, com o Diagrama de Casos de Uso. Para ele, definimos quatro atores primários: o primeiro representa todos os usuários, humanos ou demais sistemas, e leva o nome genérico 'Usuário'. Dele, herdam os três outros tipos de usuário, o 'Paciente', que representa uma pessoa que utiliza o sistema para seu tratamento, 'Profissional', que atua na área de reabilitação de seus pacientes, e 'Plug-in do Jogo', um sistema externo que faz uso do módulo de *plug-in* para utilizar a funcionalidade de reconhecimento de gestos.

No diagrama, cada ator tem acesso às ações às quais está ligado, bem como às recebidas por herança (relacionadas à autenticação). Como todas as ações requerem autenticação no sistema, tal necessidade fica explicitada por meio da nota adicionada ao diagrama.

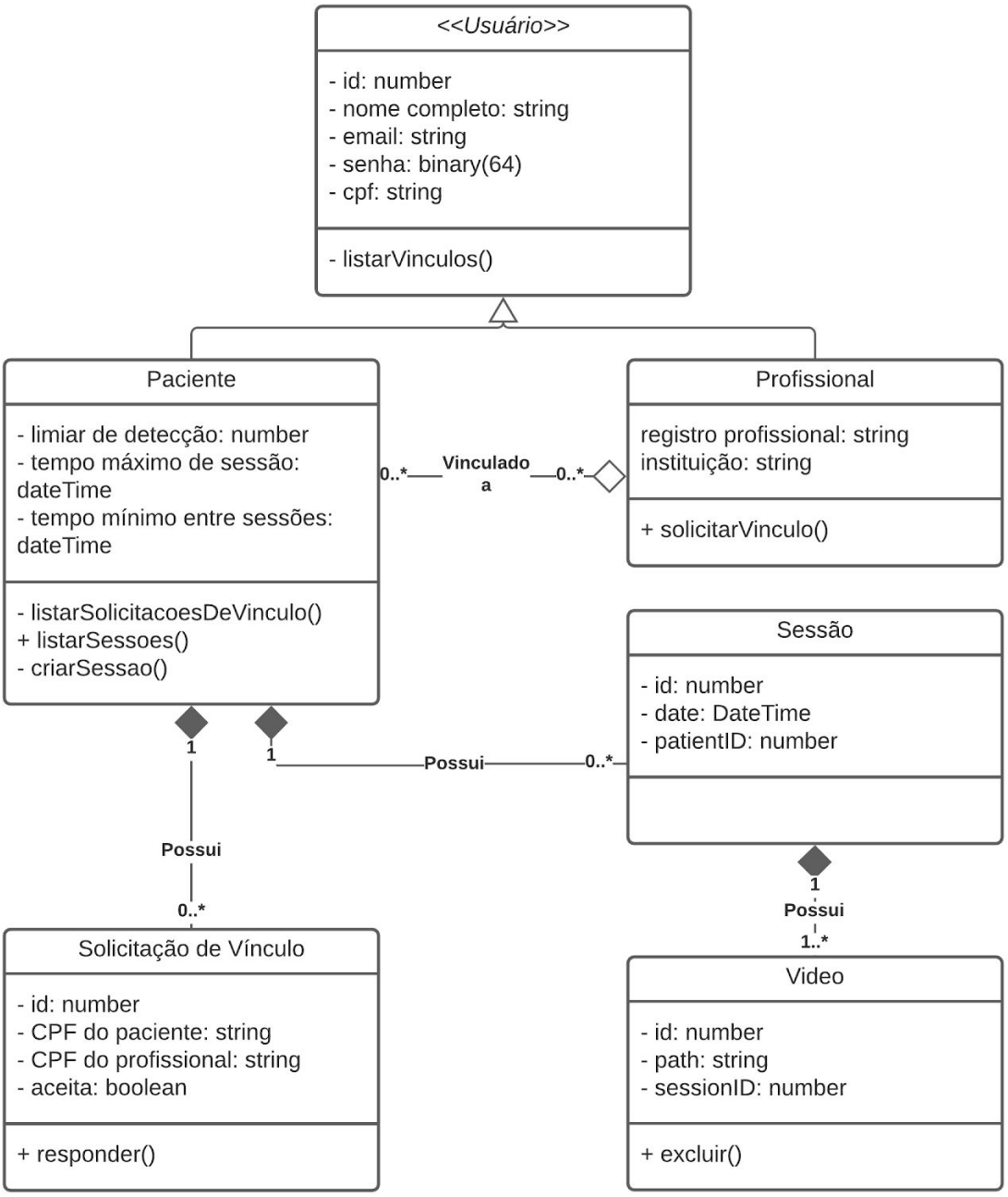
Figura 5: Diagrama de Casos de Uso da plataforma WEB.



Fonte: Produção própria.

Em seguida, pode ser montado um diagrama de classes, especialmente útil para a estruturação do banco de dados:

Figura 6: Diagrama de Classes da plataforma WEB.



Fonte: Produção própria.

6.3.2. Prototipação do *Front-End*

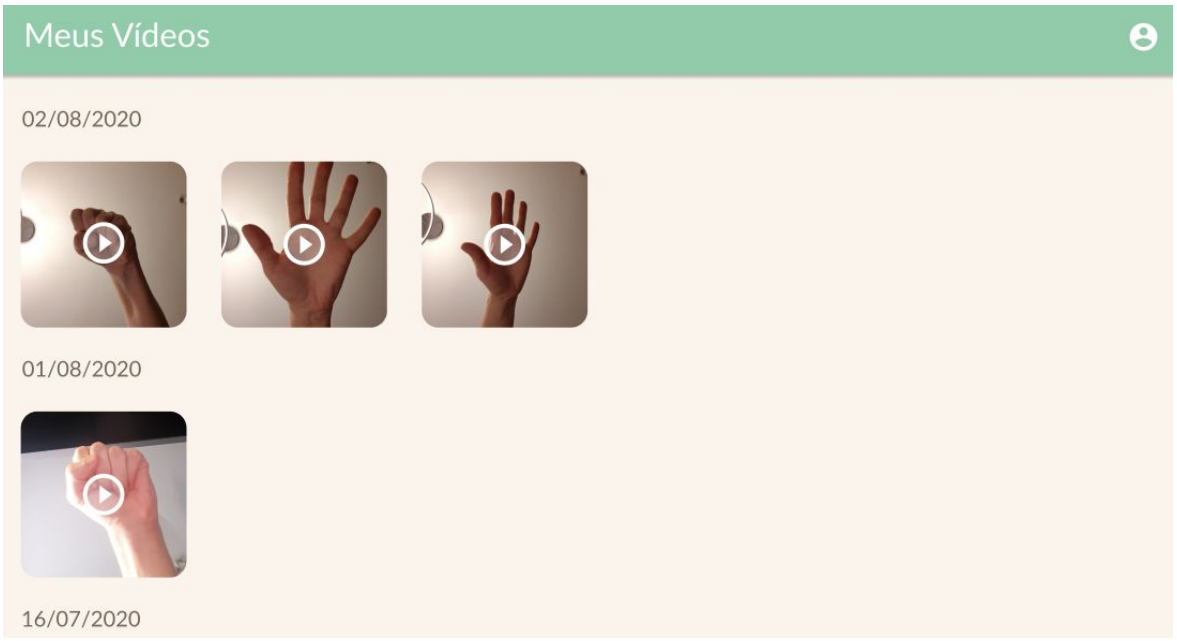
Para realizar a prototipação, foi utilizado o serviço do editor gráfico *online* Figma [49]. O objetivo desta etapa é desenvolver uma interface de usuário, de modo a expor as funcionalidades propostas. Ao acessar a plataforma, deseja-se que o usuário seja apresentado a uma tela de autenticação, onde possa criar ou acessar sua conta. Dependendo do tipo de usuário, terá acesso a telas específicas. Pacientes só possuem acesso à página que lista seus vídeos de sessões, e nela

também deverão receber convites para vincularem-se a profissionais. Estes, por sua vez, conseguem alternar entre a exibição dos vídeos de cada um de seus pacientes vinculados, bem como acessar uma tela com configuração de parâmetros para a sessão do paciente em questão. A prototipação completa pode ser acessada a partir do seguinte *link:* <https://www.figma.com/file/SqjUXMeBKzWYdooxgJNX80/TCC?node-id=0%3A1>. A seguir, alguns recortes relevantes.

Figura 7: Recorte da tela inicial de autenticação.

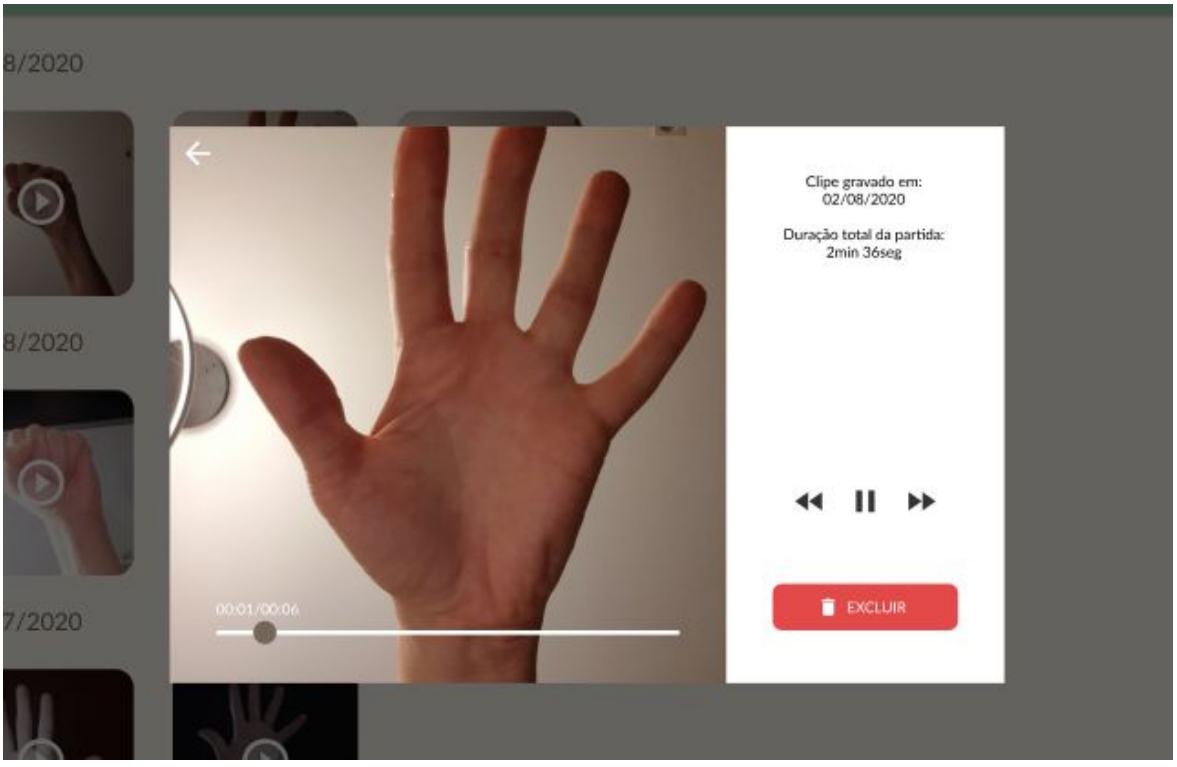
Fonte: Produção própria em projeto Figma.

Figura 8: Recorte da tela de listagem de vídeos.



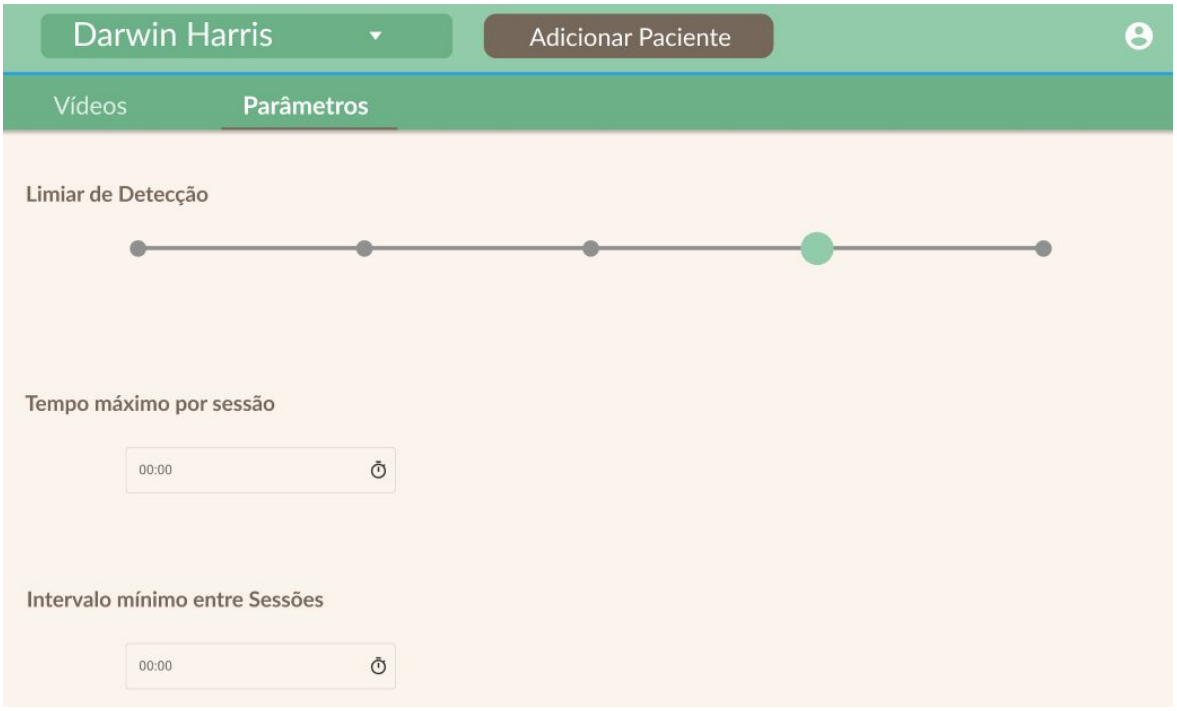
Fonte: Produção própria em projeto Figma.

Figura 9: Recorte com o *Player* de vídeo para reprodução de sessões



Fonte: Produção própria em projeto Figma.

Figura 10: Visão da tela para profissional, na aba de ajuste de parâmetros da sessão.



Fonte: Produção própria em projeto Figma.

6.3.3. Desenvolvimento do Front-end e Back-end

Com a especificação completa e o protótipo produzido, é possível iniciar o desenvolvimento do front-end e back-end da plataforma WEB. O *front-end* será desenvolvido utilizando o *React*, *framework Javascript*. O *back-end*, por sua vez, será escrito em *Python*, com o *framework Flask*. Para realizar as requisições, será utilizado o Axios. É possível encontrar o código-fonte do *front-end* no repositório em <https://github.com/kaiquesacchi/physiotherapy-web-frontend>, enquanto o *back-end* está em <https://github.com/vMarcelino/physiotherapy-web-backend>.

6.4. Criação do Jogo

Para demonstrar a utilização do *plug-in* no desenvolvimento de jogos em Unity, deseja-se criar um jogo que, de preferência, utilize componentes já criados pela comunidade da plataforma, uma vez que o objetivo é que ele seja facilmente acoplado a projetos de programadores terceiros. Além disso, dado que o modelo de rede neural produzido é capaz de diferenciar apenas dois tipos de entrada (mão aberta ou fechada), a complexidade de interação com o usuário é fator determinante para a escolha do jogo a ser produzido.

6.4.1. Obtenção dos Recursos *Open-Source*

Dadas tais restrições, optou-se por utilizar os *assets* fornecidos juntamente a um tutorial criado pela própria equipe da Unity [50]. O projeto em questão é intitulado ‘Space Shooter’, e compõe um jogo de plataforma no qual uma nave espacial deve desviar e destruir inimigos e asteróides, evitando colisões e somando pontos. A versão original foi projetada para interação a partir do teclado e, por isso, adaptações serão feitas, para simplificar a interface com o usuário e reduzir os controles para apenas dois.

O pacote fornecido é composto por modelos 3D, *meshes*, texturas e *scripts*. Em adição, uma série de vídeos com explicações passo-a-passo para a montagem do jogo também foi disponibilizada. Com base neste conteúdo, uma versão simplificada do jogo será desenvolvida, de modo a operar com os controles provenientes da inferência de classificação de imagens.

6.4.2. Adaptação do Jogo

Pensando na futura adição do módulo de reconhecimento de imagem, o jogo original foi modificado para comportar apenas a interação do usuário por meio dos gestos captados pela câmera. Sendo assim, os movimentos de abrir e fechar a mão controlarão a subida e descida da nave protagonista, no plano vertical. A ‘Figura 11’ abaixo auxilia a ilustração do fato.

Figura 11: Jogo adaptado sem reconhecimento de imagem.

Fonte: Produção própria em projeto Unity.

Na imagem, à esquerda, é possível identificar a nave pilotada pelo jogador. Neste momento, ela é controlada pelas teclas direcionais do teclado, as setas para cima e para baixo, que guiam sua subida e descida. À direita, observam-se asteróides, que se movem na direção horizontal, em rota de colisão com o jogador. O objetivo, então, é desviar dos objetos para evitar a colisão. O jogo produzido se enquadra na categoria de *Endless-runners*, popularizada por títulos como '*Temple Run'* e *'Jetpack Joyride'*, que atingiram as primeiras colocações nas lojas de aplicativos *mobile* anos atrás.

6.4.3. Inserção do *Plug-in* de Reconhecimento de Imagens

Com um jogo já funcional, é possível realizar a etapa de adição do *plug-in* de reconhecimento de imagens produzido. A inserção foi feita seguindo a documentação desenvolvida no processo de finalização do *package*, disponibilizada no formato de README no repositório do *GitHub* do pacote (https://github.com/danilosobral/unity-gesture-recognition). Nela, são detalhados os procedimentos para:

- Obtenção de bibliotecas dependentes;
- Importação do *package* ao projeto;
- Integração à interface do sistema de eventos para controle do jogo;
- Integração à interface do sistema de eventos para comunicação com API;
- Compilação para o sistema *Android*.

Conforme explicado na documentação, foram feitas pequenas alterações no projeto do jogo original, sendo a principal a troca do sistema tradicional de controle pelo acionamento da interface de eventos gerados pela detecção de gestos. O procedimento de adição do *plug-in* pode ser observado em vídeo disponibilizado na página do projeto, em <https://sites.google.com/usp.br/complemento-fisioterapico/>.

7. Testes e Avaliação

Durante o desenvolvimento do produto, cada etapa acompanhou testes para validação da implementação. O módulo de *plug-in* de reconhecimento de gestos foi adicionado a outros projetos Unity, e sua operação foi testada tanto em *desktop*, no computador dos integrantes do grupo, tanto usando o Windows 10 quanto o Linux (Ubuntu 20.04). Os protótipos também foram compilados para Android, e testados nos dispositivos Samsung Galaxy S8, Galaxy S9 e Galaxy Note 9. A inspeção de funcionamento do reconhecimento de imagens foi feita de maneira qualitativa, avaliando o comportamento do produto, e os resultados foram satisfatórios. A análise quantitativa fica por parte do resultado obtido sobre o conjunto de validação durante o treinamento do modelo, que atingiu acurácia de 92%, acima do valor mínimo previsto pelos requisitos não-funcionais, demonstrado na seção anterior.

O jogo desenvolvido recebeu o mesmo tratamento, sendo avaliado antes e depois da adição do *plug-in* em *desktop* e apenas após o processo em *dispositivos móveis* (por conta da limitação de métodos de entrada causada pela ausência de teclado). A plataforma WEB, por sua vez, teve constante teste de funcionalidade para adequação entre *front-end* e *back-end*. Operações como autenticação e criação de vínculos profissional-paciente foram postos à prova, e cada caso de uso relacionado foi testado. Ao fim do processo, a arquitetura foi montada e deu-se início aos testes de integração.

7.1. Testes de Funcionalidade e Integração

Os testes de integração englobam as múltiplas funcionalidades desenvolvidas no projeto, ressaltando a corretude da comunicação entre os diferentes módulos. Dessa forma, são propostos cinco cenários de teste que serão percorridos do ponto de vista do profissional de saúde e do paciente, simulando as interações dos usuários com a plataforma. Em alguns casos de teste, foi necessária a utilização de guias anônimas autenticadas em contas diferentes para simular o comportamento de dois usuários simultâneos no sistema. A execução dos testes seguiu a ordem normal de utilização da plataforma, sequência esperada para a utilização real do produto.

7.1.1. Autenticação

A autenticação de usuários é o primeiro passo contato com a plataforma WEB. Este cenário de teste engloba o cadastro de dois novos usuários, um fisioterapeuta e um paciente.

Autenticação de paciente:

Situação inicial:
Usuário sem cadastro no sistema, na página de criação de conta.

Tarefas:
1. Selecionar a opção para criação de contas de ‘Paciente’;
2. Informar nome, email e senha (mínimo de 8 caracteres) e selecionar ‘Continuar’;
3. Informar CPF e selecionar ‘Criar Conta’;
4. Aceitar ‘Termos de Uso’.

Resultado esperado:
Paciente cadastrado e autenticado no sistema, redirecionado à página ‘Meus Vídeos’. Nela, nenhum vídeo disponível.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

Autenticação de médico:

Situação inicial:
Usuário sem cadastro no sistema, na página de criação de conta.

Tarefas:
1. Selecionar a opção para criação de contas de ‘Profissional’;
2. Informar nome, email e senha (mínimo de 8 caracteres) e selecionar ‘Continuar’;
3. Informar CPF, registro profissional e instituição na qual trabalha e selecionar ‘Criar Conta’;
4. Aceitar ‘Termos de Uso’.

Resultado esperado:
Profissional cadastrado e autenticado no sistema, redirecionado à página ‘Meus Pacientes’.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

7.1.2. Convites

Uma vez cadastrados, o próximo cenário de testes envolve a criação do vínculo entre o profissional responsável pelo tratamento e seu paciente. Essa tarefa se dá

por meio do envio de convites por parte do profissional, o que podem ser aceitos ou recusados pelo paciente.

Recusar convite:

Situação inicial:
Profissional e paciente autenticados no sistema em guias anônimas do navegador, sem vínculo existente.

Tarefas:
1. Profissional seleciona ‘Adicionar Paciente’;
2. Profissional digita o CPF do paciente e seleciona ‘Enviar Convite’;
3. Paciente recarrega a página para listar novo convite;
4. Paciente seleciona ‘Recusar’.

Resultado esperado:
Profissional e paciente sem vínculos após recusar o convite. Profissional não é capaz de selecionar o paciente em questão no seletor do topo da página.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

Aceitar convite:

Situação inicial:
Profissional e paciente autenticados no sistema, sem vínculo.

Tarefas:
1. Profissional seleciona ‘Adicionar Paciente’;
2. Profissional digita o CPF do paciente e seleciona ‘Enviar Convite’;
3. Paciente recarrega a página para listar novo convite;
4. Paciente seleciona ‘Aceitar’.

Resultado esperado:
Profissional e paciente vínculos após aceitar o convite. Profissional é capaz de selecionar o paciente em questão no seletor do topo da página.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

7.1.3. Gravação de vídeos

Com o paciente e profissional vinculados na plataforma, deve ser testado o acesso dos usuários aos vídeos gravados durante uma sessão. Para isso, também será testada a integração com o jogo, passando pelo processo de autenticação e envio das gravações pelo aplicativo.

Gravação de vídeo:

Situação inicial:
Paciente com jogo instalado no celular.

Tarefas:
1. Abrir o aplicativo do jogo;
2. Selecionar botão ‘Login’;
3. Adicionar email e senha aos campos correspondentes, e selecionar ‘Entrar’;
4. Selecionar ‘Jogar’;
5. Jogar por alguns segundos e, então, colidir com um asteróide.

Resultado esperado:
Paciente com acesso aos vídeos gravados na plataforma WEB.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado

Acesso de profissional autorizado ao vídeo:

Situação inicial:
Profissional autenticado e vinculado a paciente que possui vídeos salvos na plataforma.

Tarefas:
1. Selecionar paciente em questão, por meio do seletor no topo da página.
2. Selecionar algum dos vídeos listados e o assistir.

Resultado esperado:
Profissional com acesso aos vídeos gravados na plataforma WEB. Cada vídeo possui 5 segundos de duração, e pode ser assistido pela plataforma.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

7.1.4. Remoção de vídeo

O teste a ser realizado após a gravação dos vídeos é a possibilidade de sua remoção, o que pode ser feito pelo profissional ou pelo paciente.

Remoção de vídeo pelo paciente:

Situação inicial:
Paciente com vídeo gravado autenticado no sistema e vinculado ao fisioterapeuta.

Tarefas:
1. Selecionar algum dos vídeos disponíveis;
2. Selecionar ‘Excluir’.

Resultado esperado:
Player de vídeo é fechado. Vídeo é apagado do sistema para o paciente e fisioterapeuta, e deixa de ser listado.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

Remoção de vídeo pelo profissional:

Situação inicial:
Profissional autenticado no sistema e vinculado a paciente com vídeo gravado. Paciente já selecionado por meio do seletor no topo da página.

Tarefas:
1. Selecionar algum dos vídeos disponíveis;
2. Selecionar ‘Excluir’.

Resultado esperado:
Player de vídeo é fechado. Vídeo é apagado do sistema para o paciente e fisioterapeuta, e deixa de ser listado.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

7.1.5. Modificação de parâmetros

Com as demais funcionalidades testadas, é necessário verificar a possibilidade do profissional de saúde alterar os parâmetros do jogo dependendo das condições do paciente

Alteração de parâmetros do jogo para um paciente:

Situação inicial:
Profissional autenticado na plataforma. Paciente selecionado por meio do seletor no topo da página.

Tarefas:
1. Selecionar aba de ‘Parâmetros’;
2. Modificar posição dos *sliders*, ambos para o valor máximo (100).

Resultado esperado:
Parâmetros alterados e salvos. Notificação de sucesso recebida e mostrada na parte inferior da tela. Os novos valores são mantidos mesmo ao recarregar ou fechar e reabrir a página.

Resultado obtido:
Resultados obtidos estão de acordo com o esperado.

Perceber alterações no jogo:

Situação inicial:
Profissional acaba de fazer a mudança de parâmetros descrita no passo anterior. Paciente possui o jogo instalado no celular e já informou seu email e senha na área de ‘Login’.

Tarefas:
1. Iniciar o jogo, selecionando ‘Jogar’;
2. Notar que a nave e os asteróides se movem mais rápido do que anteriormente.

Resultado esperado:
A nave e os asteroides devem se mover mais rápido do que nas partidas anteriores, de acordo com o valor selecionado pelo profissional na etapa de teste anterior.

Resultado obtido:
Devido à implementação da mudança de configurações no jogo teste, é necessário esperar uma rodada para perceber as alterações do parâmetros. Após esta rodada, o resultado obtido fica de acordo com o esperado. Como o problema não impede o uso do sistema, não será tratado.

8. Considerações Finais

8.1. Conclusões

Após a execução do projeto, pôde-se verificar a viabilidade técnica da plataforma proposta. Criou-se uma ferramenta modularizada capaz de prover a um jogo de celular a interface via controle gestual e, ademais, integrá-lo a uma plataforma WEB que pode ser usada pelo médico como ferramenta de monitoração do tratamento.

Analisando os requisitos propostos, foram implementadas as diversas funcionalidades básicas que comprovam o funcionamento da arquitetura, desde o reconhecimento de gestos dentro do jogo até a visualização de trechos da sessão de exercícios por parte do fisioterapeuta. O produto criado configura, assim, um protótipo que pode ser demonstrado a clínicas e hospitais, visando captação de investimento para sua contínua melhoria e adequação às necessidades do paciente e do médico.

Além disso, os requisitos não funcionais elencados neste documento foram sempre utilizados como métricas para guiar a tomada de decisões sobre os detalhes da implementação. Conforme previsto, atingiu-se acurácia superior aos 90% previstos para a rede neural, garantindo responsividade no jogo e, devido às tecnologias usadas, observa-se também um *footprint* relativamente pequeno do *plug-in*, quando comparado ao tamanho usual de um jogo, de aproximadamente 1,3MB. A escolha da *game engine* foi essencial para o cumprimento do requisito de portabilidade, já que o motor de jogos possibilitou a criação de uma ferramenta que funciona tanto em *desktops* quanto aparelhos *Android*, enquanto a preocupação com segurança e privacidade motivou o uso de uma comunicação segura entre o servidor e os módulos de interface com o usuário.

Com isso, o produto resultante deste projeto confirmou a possibilidade do funcionamento da arquitetura e abre caminho para análise junto a um corpo médico e de desenvolvimento de jogos, que verifique a aplicabilidade deste protótipo em um contexto real.

8.2. Contribuições

O projeto desenvolvido comprova a viabilidade técnica da opção de realizar o tratamento por jogos em um ambiente *mobile*, mais acessível a grande parte dos pacientes, tornando a adoção desta alternativa mais democratizada. Tal viabilidade foi possibilitada não só pelo uso de ferramentas voltadas para a inclusão de interpretadores de modelos de rede neural no ambiente de uma *game engine* - como

a biblioteca do TensorFlow Lite para o *Unity*, desenvolvido por Koki Ibukuro [44] -, mas também pela inserção dessa ferramenta em uma infraestrutura modularizada, que permite integração com uma variedade de jogos já desenvolvidos dependente de pequenas modificações. Desta forma, este projeto é capaz de facilitar a criação de *games* no contexto da fisioterapia, diminuindo a barreira técnica.

Além disso, a integração da ferramenta a uma plataforma WEB mostra a possibilidade de um monitoramento remoto do tratamento por parte do fisioterapeuta, seja observando a evolução por meio dos vídeos ou ajustando os exercícios por meio da customização de parâmetros. Assim, o projeto é relevante como um passo inicial para a criação de funcionalidades voltadas para uma melhoria da Telemedicina no contexto da reabilitação motora.

8.3. Perspectivas de continuidade

Existem diversas melhorias que podem ser feitas à plataforma visando sua maior adequação ao mercado, conforme descrito a seguir:

- **Refinamento da capacidade de reconhecimento de gestos:** Na fisioterapia, existem múltiplos movimentos que são usados nos exercícios, não só o abrir e fechar das mãos. Além disso, uma análise realista do ambiente de reabilitação motora mostra uma grande quantidade de pacientes incapazes de realizar os movimentos de forma completa. Dessa forma, a obtenção de imagens mais diversas e a criação de múltiplas categorias de gestos abre caminho para um pacote de reconhecimento de gestos mais adequado ao tratamento.

- **Adaptação às necessidades dos tratamentos:** Durante o desenvolvimento do projeto, objetivou-se demonstrar as capacidades da plataforma proposta. Existem, portanto, métricas e configurações que podem mostrar-se úteis para o cenário e que não foram adicionadas. Para a parametrização do jogo, por exemplo, foram adicionadas apenas as opções para modificação da dificuldade e velocidade da nave controlada, mas outras configurações poderão ser adicionadas caso sejam solicitadas, como um limite de tempo por sessão.

- **Expansão do modelo de negócios**: Para a aplicação deste projeto em um contexto real, é necessária uma adequação do modelo de negócio, no qual a plataforma assuma um papel mais ativo no intermédio entre médicos e desenvolvedores de jogo. A intenção é criar, junto a especialistas, uma lista de normas e diretrizes para jogos voltadas ao tratamento fisioterápico e, com base nessa lista, realizar uma curadoria dos jogos sujeitos a usar a

plataforma, garantindo assim que os participantes desta infraestrutura atendam os requisitos mínimos necessários para um tratamento eficaz.

REFERÊNCIAS

[1] KALMMANN, Marcelo; CAMPORESI, Carlo; HAN, Jay. **VR-Assisted Physical Rehabilitation: Adapting to the Needs of Therapists and Patients.** In: INTERNATIONAL DAGSTUHL SEMINAR, 20013, 2015, Schloss Dagstuhl, Alemanha. Disponível em: <http://graphics.ucmerced.edu/papers/15-dagbook-vpt-s.pdf> Acesso em: 15/03/2020

[2] LOHSE, Keith PhD; SHIRZAD, Navid MASc; VERSTER, Alida; HODGES, Nicola PhD; VAN DER LOOS, H. F. Machiel PhD. **Video Games and Rehabilitation: Using Design Principles to Enhance Engagement in Physical Therapy**. Journal of Neurologic Physical Therapy: Dez. 2013 - Volume 37 - Issue 4 - p 166-175. Disponível em: <https://journals.lww.com/jnpt/Fulltext/2013/12000/Video_Games_and_Rehabilitation__Using_Design.4.aspx>. Acesso em: 25/01/2020.

[3] LANG, Catherine; MACDONALD, Jillian; REISMAN, Darcy; et al. **Observation of amounts of movement practice provided during stroke rehabilitation.** Arch Phys Med Rehabil. 2009;90(10):1692–1698. doi:10.1016/j.apmr.2009.04.005. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3008558/>. Acesso em: 14/03/2020.

[4] BÔAS, A. V.; FERNANDES, W. L. M.; SILVA, A. M.; SILVA, A. T. **Efeito da Terapia Virtual na Reabilitação Motora do Membro Superior de Crianças Hemiparéticas**. Revista Neurociências, v. 21, n. 4, p. 556-562, 31 mar. 2001. Disponível em: <https://periodicos.unifesp.br/index.php/neurociencias/article/view/8148>. Acesso em: 16/02/2020.

[5] GSM Association, **The Mobile Economy 2019.** Disponível em: <https://www.gsmaintelligence.com/research/?file=b9a6e6202ee1d5f787cfebb95d3639c5&download>. Acesso em: 16/02/2020.

[6] CORRÊA, Ana Grasielle D.; KINTSCHNER, Natália R.; CAMPOS, Victor Z.; et al. **Gear VR and leap motion sensor applied in virtual rehabilitation for manual function training: an opportunity for home rehabilitation.** In Proceedings of the 5th Workshop on ICTs for improving Patients Rehabilitation Research Techniques (REHAB ’19). Association for Computing Machinery, New York, NY, USA, 148–151. Disponível em <https://doi.org/10.1145/3364138.3364169>. Acesso em: 21/04/2020.

[7] DAMASCENO, Eduardo Filgueiras. **Sistema de Reabilitação Baseado em Técnicas de Captura de Movimento para Tratamento da Lombalgia Mecânica.**

Tese de doutorado na Universidade Federal de Uberlândia, 2013. Disponível em: <http://clyde.dr.ufu.br/bitstream/123456789/14321/1/EDUARDO%20FILGUEIRAS.pdf>. Acesso em: 20/04/2020.

[8] CHOUDARY, S.P. (n.d.); **The Platform Stack: For Everyone Building a Platform… and For Everyone Else: A unifying framework for digital business models.** Disponível em <http://platformed.info/platform-stack/>. Data de acesso: 17/11/2020.

[9] HOSCH, William L. **Machine learning**. Encyclopædia Britannica, October 08, 2019. Disponível em: <https://www.britannica.com/technology/machine-learning>. Data de acesso: 21/04/2020.

[10] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. MIT Press (2016). Disponível em: <http://www.deeplearningbook.org/contents/intro.html>. Acesso em: 21/04/2020.

[11] **TensorFlow**. Disponível em: <https://www.tensorflow.org/>. Acesso em: 21/04/2020.

[12] **Classificação de imagens**. Disponível em: <https://www.tensorflow.org/lite/models/image_classification/overview>. Acesso em: 21/04/2020.

[13] ZHAO, K.; MATSUKAWA, T.; SUZUKI, E. **Retraining: A Simple Way to Improve the Ensemble Accuracy of Deep Neural Networks for Image Classification**. 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, 2018, pp. 860-867. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8545535>. Acesso em 21/04/2020.

[14] Google. **Inception V3**. Disponível em <https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=pt-br>. Acesso em 21/04/2020.

[15] **ImageNet**. Disponível em <http://www.image-net.org/about-overview>. Acesso em 21/04/2020.

[16] BAKER, Maverick. **How Do Game Engines Work?**. Interesting Engineering, 02 nov. 2016. Disponível em: <https://interestingengineering.com/how-game-engines-work> Acesso em: 15/03/2020

[17] **Unity**. Disponível em <https://unity.com/pt>. Acesso em 18/11/2020.

[18] Unity**. Quem somos.** Disponível em <https://unity.com/pt/our-company>. Data de acesso: 18/11/2020

[19] HOWARD, Andrew G. **MobileNets: Open-Source Models for Efficient On-Device Vision**. Google AI Blog. 14 jun. 2017. Disponível em <https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html> Acesso em: 15/03/2020

[20] SANDLER, Mark; HOWARD, Andrew; ZHU, Menglong; et al. **MobileNetV2: Inverted Residuals and Linear Bottlenecks.** Google Inc. Disponível em: <https://arxiv.org/abs/1801.04381>. Data de acesso: 02/10/2020.

[21] **Documentação do Keras para MobileNet.** Disponível em <https://keras.io/api/applications/mobilenet/>. Data de acesso: 02/10/2020.

[22] **React.** Disponível em <https://pt-br.reactjs.org/>. Data de acesso: 18/11/2020.

[23] Google Developers. **Secure your site with HTTPS**. Disponível em <https://developers.google.com/search/docs/advanced/security/https?hl=en&visit_id=637413457389853292-3556506591&rd=1>. Data de acesso: 18/11/2020.

[24] TAKAHASHI, Craig D.; DER-YEGHIAIAN, Lucy; LE, Vu; MOTIWALA, Rehan R.; CRAMER, Steven C. **Robot-based hand motor therapy after stroke.** Brain, [*s. l.*], v. 131, ed. 2, p. 425-437, Fev 2008. Disponível em: <https://academic.oup.com/brain/article/131/2/425/406042>. Acesso em: 20/04/2020.

[25] MARNE, B.; WISDOM, J.; HUYNH-KIM-BANG, B.; LABAT, J.M. The Six Facets of Serious Game Design: A Methodology Enhanced by Our Design Pattern Library. *In*: RAVENSCROFT, A.; LINDSTAEDT, S.; KLOOS, C.D.; HERNÁNDEZ-LEO, D. **21st Century Learning for 21st Century Skills**. Lecture Notes in Computer Science: Springer, 2012. v. 7563. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-642-33263-0_17>. Acesso em: 20/04/2020.

[26] DJAOUTI, Damien; ALVAREZ, Julian; JESSEL, Jean-Pierre. Classifying Serious Games: the G/P/S model. *In*: **HANDBOOK of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches**. [*S. l.*: *s. n.*], 2011. Disponível em: <http://www.ludoscience.com/files/ressources/classifying_serious_games.pdf>. Acesso em: 20/04/2020.

[27] ROCHA, Rafaela Vilela da; BITTENCOURT, Ig Ibert; ISOTANI, Seiji. Análise, Projeto, Desenvolvimento e Avaliação de Jogos Sérios e Afins: uma revisão de desafios e oportunidades. **Brazilian Symposium on Computers in Education**

(Simpósio Brasileiro de Informática na Educação - SBIE), [S.l.], p. 692, out. 2015. ISSN 2316-6533. Disponível em: <https://br-ie.org/pub/index.php/sbie/article/view/5342/3705>. Acesso em: 20/04/2020.

[28] MDN. Confidencialidade, Integridade e Disponibilidade. *In*: MDN. **MDN web docs**. [*S. l.*], 18 mar. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Security/B%C3%A1sico_de_Seguran%C3%A7a_da_Informa%C3%A7%C3%A3o/Confidencialidade,_Integridade,_e_Disponibilidade>. Acesso em: 20/04/2020.

[29] OPENTEXT. Information Security and Privacy: Overview. *In*: **Information Governance**. Disponível em: <https://www.opentext.com/products-and-solutions/business-needs/information-governance/ensure-compliance/information-security-and-privacy>. Acesso em: 20/04/2020.

[30] EADICICCO, Lisa. **Apple just got knocked out of the top 3 smartphone makers in the world — here's how it stacks up against rivals like Samsung, Huawei, and LG**. Business Insider, [*s. l.*], 8 ago. 2019. Disponível em: <https://www.businessinsider.com/biggest-smartphone-makers-in-the-world-apple-slips-2019-8>. Acesso em: 20/04/2020.

[31] STATCOUNTER. **Mobile Operating System Market Share Worldwide**. GlobalStats. mar. 2020. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide>. Acesso em: 20/04/2020.

[32] FAKHROUTDINOV, Kirill. UML Component Diagrams. **Uml-diagram.org**, [*s. l.*], [2009?]. Disponível em: <https://www.uml-diagrams.org/component-diagrams.html>. Data de acesso em: 26/11/2020.

[33] JABBAR, Haider K.; KHAN, Rafiqul Z. **Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study).** Aligarh Muslin University. Disponível em <https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf>. Acesso em: 03/09/2020

[34] **Repositório do grupo, usado para armazenar imagens de treinamento e os códigos produzidos e utilizados.** Disponível em <https://gitlab.com/vMarcelino/physiotherapy-platform/>. Acesso em: 01/09/2020.

[35] **Documentação do Keras para Early Stopping.** Disponível em <https://keras.io/api/callbacks/early_stopping/>. Data de acesso: 21/09/2020.

[36] MASTERS, D.; LUSCHI, C. **Revisiting Small Batch Training for Deep Neural Networks.** Disponível em: <https://arxiv.org/abs/1804.07612>. Data de acesso: 21/09/2020.

[37] BENGIO, Y. **Practical recommendations for gradient-based training of deep architectures.** Disponível em: <https://arxiv.org/abs/1206.5533>. Data de acesso: 21/09/2020.

[38] RUDER, S. **An overview of gradient descent optimization algorithms.** Disponível em: <https://arxiv.org/abs/1609.04747>. Data de acesso: 21/09/2020.

[39] **Weights and Biases.** Disponível em <https://www.wandb.com/>. Data de acesso: 02/12/2020.

[40] COMPUTER HOPE. **Plugin**. Computer Hope, [*s. l.*], 6 fev. 2020. Disponível em: <https://www.computerhope.com/jargon/p/plugin.htm>. Data de acesso: 02/11/2020.

[41] Unity. **Plug-ins**. Unity User Manual: Scripting, [*s. l.*], 19 mar. 2018. Disponível em: <https://docs.unity3d.com/Manual/Plugins.html>. Data de acesso: 02/11/2020.

[42] Unity. **Packages**. Unity User Manual, [*s. l.*], 19 mar. 2018. Disponível em: <https://docs.unity3d.com/Manual/PackagesList.html>. Data de acesso: 02/11/2020.

[43] Unity. **Unity Asset Store**. [*S. l.*]. Disponível em: <https://assetstore.unity.com/> Data de acesso: 02/11/2020.

[44] IBUKURO, Koki. **TensorFlow Lite Unity Samples**. Disponível em: <https://github.com/asus4/tf-lite-unity-sample>. Data de acesso: 22/11/2020.

[45] HASSAN, Ahmed Shamin. **Observer vs Pub-Sub Pattern**. Medium, [*s. l.*], 29 out. 2017. Disponível em: <https://medium.com/towards-artificial-intelligence/observer-pattern-vs-pub-sub-pattern-7f467bcf5fe>. Data de acesso: 23/11/2020.

[46] ELHADAD, Michael. **Software Engineering - Fall 1999: Lecture 10: Event Models**. [*S. l.*: *s. n.*], 1999. Disponível em: <https://www.cs.bgu.ac.il/~elhadad/se/events.html>. Data de acesso em: 23/11/2020.

[47] FFMPEG. **Documentation**. [S. l.]. Disponível em: <https://ffmpeg.org/download.html>. Data de acesso: 23/10/2020.

[48] Unity. **UnityWebRequest**. Unity User Manual: Scripting API, [*s. l.*], 24 jun. 2020. Disponível em: <https://docs.unity3d.com/ScriptReference/Networking.UnityW

ebRequest.html>. Data de acesso: 02/11/2020.

[49] **Figma**. Disponível em <https://www.figma.com/>. Data de acesso: 22/11/2020.

[50] Unity. **Introdução ao Space Shooter.** Disponível em: <https://learn.unity.com/tutorial/introduction-to-space-shooter#>. Data de acesso: 02/12/2020

APÊNDICE A - Formatação de Imagens

Disponível em: <https://gitlab.com/vMarcelino/physiotherapy-platform/> [34]

Função

Converter imagens ao formato esperado para o treinamento da rede neural (128x128 pixels, escala de cinza).

Modo de Uso

1. Instalar as dependências com `pip install -r rescale_requirements.txt`
2. Criar uma pasta denominada 'source', no mesmo diretório deste ***notebook***, caso ainda não exista.
3. Dentro de `./source/`, criar uma pasta para cada classe a ser usada no treinamento, nomeando-as de acordo com a ***label*** em questão.
4. Dentro de cada pasta `./source/***<label_da_classe>***/`, adicionar as fotos correspondentes.
5. Executar todo o ***notebook***.
6. Os arquivos convertidos serão gerados em `./destination`.

Exemplo de Estrutura Final

```
.
├── ...
├── rescale.py
├── rescale_requirements.txt
├── source
│   ├── closed
│   │   ├── image0.jpeg
│   │   ├── image1.jpeg
│   │   └── image2.jpeg
│   └── open
│       ├── image3.jpeg
│       ├── image4.jpeg
│       └── image5.jpeg
└── destination
    ├── closed
    │   ├── _image0.jpg
    │   ├── _image1.jpg
    │   └── _image2.jpg
```

```
    └──── open
        ├──── _image3.jpg
        ├──── _image4.jpg
        └──── _image5.jpg
```

Primeiro, são definidos os nomes das pastas usados para entrada e saída da conversão de imagens.

```
source_folder = 'source'
destination_folder = 'destination'
```

Para cada classe, cria uma pasta correspondente no diretório de destino e, lá, adiciona as imagens convertidas.

```
from PIL import Image
import os
import pathlib

for folder in os.listdir(source_folder):
   if os.path.isdir(os.path.join(source_folder, folder)):
       # Ensures the destination folder exists.
       pathlib.Path(os.path.join(destination_folder,
folder)).mkdir(parents=True, exist_ok=True)

       for filename in os.listdir(os.path.join(source_folder,
folder)):
           file_path = os.path.join(source_folder, folder, filename)
           if os.path.isfile(file_path):
               # Converts image.
               img = Image.open(file_path).convert('L')  # Luma only
(grayscale).
               resized_image = img.resize((128, 128))

               # Saves to the destination folder.
               filename, _ = os.path.splitext(filename)
               filename = '_' + filename + '.jpg'
               resized_image.save(os.path.join(destination_folder,
folder, filename))
```

APÊNDICE B - Treinamento da Rede Neural

Disponível em: <https://gitlab.com/vMarcelino/physiotherapy-platform/> [26]

Função

Treinar o modelo para categorização de imagens, de acordo com as classes fornecidas.

Modo de Uso

1. Instalar o python3-dev com `sudo apt install python3-dev`.
2. Instalar as dependências com `pip install -r train_requirements.txt`.
3. Cadastrar-se em Weights & Biases (https://www.wandb.com/) e fazer o login pelo terminal.
4. (Opcional) Para usar a GPU, é necessário instalar os pacotes da NVIDIA (https://www.tensorflow.org/install/gpu#software_requirements)
5. Executar todo o ***notebook***

Primeiro, são feitos os *imports* dos módulos necessários. Aqui, também inicializa-se o Weights & Biases, *dashboard* a ser usado para monitorar o aprendizado do modelo.

```
import os
import sys
import argparse
import pandas as pd
import numpy as np
import random

from model import MobileNetv2

import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Conv2D, Reshape, Activation
from tensorflow.keras.models import Model

import wandb
from wandb.keras import WandbCallback
```

É necessário definir os metadados que guiarão o treinamento. São quatro valores a serem definidos:

- classes: É a quantidade de classes a serem treinadas. Cada classe é uma opção de saída, um valor possível a ser previsto pela rede. No caso, temos duas opções, 'mão aberta' ou 'mão fechada' e, por isso, o valor usado é 2
- image_size: O tamanho de cada imagem quadrada a ser usada como entrada. Como optamos por 128x128 pixels, o valor usado é 128
- batch_size: O tamanho de cada lote de treinamento, indo de 1 até o tamanho total do dataset. Como indicado por pesquisa, foi usado o valor 32
- epochs: O número de vezes que o treinamento deve percorrer o dataset completo. Por recomendação, foi usado um valor alto (10000), em conjunto ao Early-Stop, técnica que encerra o treinamento antes de finalizar todas as épocas caso a curva de aprendizado pare de expressar melhoras nos resultados.

```
classes = 2  # The number of classes of dataset.
image_size = 128  # The image size of train sample.
batch_size = 32  # The number of train samples per batch.
epochs = 10000  # The number of train iterations.
```

A tarefa seguinte é estruturar o dataset, dividindo-o em 2 grupos: imagens de treinamento (90%) e de validação (10%). Cada imagem pode receber distorção, zoom, rotação, deslocamento, espelhamento e/ou inversão, para aumentar a diversidade do dataset.

```
train_data_directory = 'destination'

def image_inverter(image):
    # Has a 50% chance to invert the image, in order to
    # diversify the dataset.
    return 1 - image if random.choice((True, False)) else image

train_datagen = ImageDataGenerator(
   rescale=1. / 255,
   shear_range=0.2,
   zoom_range=0.2,
   rotation_range=180,
   width_shift_range=0.2,
   height_shift_range=0.2,
   horizontal_flip=True,
   vertical_flip=True,
   validation_split=0.1,
   preprocessing_function=image_inverter
)
```

```
train_generator = train_datagen.flow_from_directory(
    train_data_directory,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_data_directory,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',
    subset='validation'
)
```

Agora, o modelo é inicializado com a estrutura do MobileNetV2.

Optou-se por utilizar o Adam (*Adaptive Moment Estimation*) como algoritmo de otimização, para atualizar iterativamente os pesos da rede durante o treinamento, por ser normalmente a melhor opção para esse tipo de rede.

Como dito anteriormente, será usada a estratégia de *Early Stop*, monitorando a taxa de erro nas predições de validação, e com paciência de 300 épocas (espera até 300 épocas sem melhorias na acurácia para encerrar o treinamento).

A taxa de erro (*loss*) é calculada utilizando o *Categorical Cross-Entropy*, indicado para problemas de categorização.

```
wandb.init(
    project="physiotherapy-platform",
    config=dict(MobileNetv2_alpha=0.3)
)

model = MobileNetv2(
    (image_size, image_size, 1),
    classes,
    wandb.config.MobileNetv2_alpha
)

optimizer = Adam()
```

```
earlystop = EarlyStopping(
    monitor='loss',
    patience=300,
    verbose=2,
    mode='min'
)

model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy']
)

history = model.fit(train_generator,
                    validation_data=validation_generator,
                    steps_per_epoch=None,
                    validation_steps=None,
                    epochs=epochs,
                    callbacks=[earlystop, WandbCallback()])
```

Por fim, o modelo é salvo e convertido para o formato *tflite*, mais apropriado ao uso em celulares.

```
if not os.path.exists('model'):
    os.makedirs('model')

df = pd.DataFrame.from_dict(history.history)
df.to_csv('model/history.csv', encoding='utf-8', index=False)
model.save('model/model.h5')
model.save('model/saved')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TF Lite model.
with tf.io.gfile.GFile('model/model.tflite', 'wb') as f:
    f.write(tflite_model)

with tf.io.gfile.GFile(
    os.path.join(wandb.run.dir, 'model.tflite'), 'wb'
) as f:
    f.write(tflite_model)

model.save(os.path.join(wandb.run.dir, "model.h5"))
model.save(os.path.join(wandb.run.dir, "SavedModel"))
```

APÊNDICE C - Acesso à Câmera e Inferência

Para ilustrar o procedimento de criação de um jogo com acesso à câmera e capacidade de inferência, será criado um novo projeto no *Unity*. O primeiro passo é obter os *plug-ins* do TensorFlow Lite, para que se tenha acesso às suas funções e seja possível criar o interpretador baseado no modelo treinado previamente.

Estes *plug-ins* estão disponibilizados dentro de um *package* na plataforma *OpenUPM* e pode ser baixado diretamente do site (https://openupm.com/packages/com.github.asus4.tflite/). Após finalizar o *download*, basta importar o pacote para o novo projeto e reiniciá-lo.

A próxima etapa é obter o acesso à câmera. Para isso, será criado um "GameObject" vazio na cena, chamado "ImageClassification" e, a ele, adicionado um novo *script*, nomeado "ImageClassificationScript", apresentado, em partes, a seguir.

```
public class ImageClassificationScript : MonoBehaviour {
    [SerializeField] RawImage cameraDisplay;
    private WebCamDevice[] devices;
    private WebCamDevice chosenCamera;
    private WebCamTexture cameraTexture;

    void Start() {
        StartCamera();
    }

    void StartCamera() {
        devices = WebCamTexture.devices;
        chosenCamera = devices[0];
        foreach (var device in devices) {
            if (device.isFrontFacing) chosenCamera = device;
        }
        cameraTexture = new WebCamTexture(chosenCamera.name, 128, 128, 60);
        cameraTexture.Play();
        cameraDisplay.texture = cameraTexture;
    }
}
```

Assim que o objeto é instanciado no jogo, é executado o código responsável por definir o dispositivo padrão de captura de imagem que será usado e renderizar em tempo real sua saída em um textura dentro do ambiente do jogo. Este processo é realizado por meio do código apresentado anteriormente, fazendo o uso da classe "WebCamTexture", pertencente às funcionalidades *core* do motor de jogo.

Figura 12: Demonstração da imagem da câmera renderizada.

Fonte: Produção própria em projeto Unity.

O passo seguinte é adicionar o código responsável por criar um interpretador e executar a inferência, com base em um frame capturado na câmera. Primeiramente, deve ser desenvolvido o código que inicializa o interpretador do TensorFlow Lite com o modelo gerado a partir do treinamento da rede neural. Para isso, o arquivo do modelo - no caso, chamado “hand_gesture_model.tflite” - deve ser adicionado ao projeto e referenciado com uma variável do *script*. Em seguida, cria-se uma função - “StartInterpreter()”, que será executada na instanciação do objeto e fará uso das bibliotecas do TensorFlow Lite, obtidas nos *plug-ins*. A função cria um interpretador que utilizará duas *threads* e carrega o arquivo do modelo. Também é necessário ajustar o tensor para ficar de acordo com o tamanho das entradas (128x128).

```
void StartInterpreter() {
        var options = new InterpreterOptions()
        {
            threads = 2,
            useNNAPI = false,
        };
        interpreter = new Interpreter(FileUtil.LoadFile(fileName), options);
        interpreter.ResizeInputTensor(0, new int[] { 1, 128, 128, 1 });
        interpreter.AllocateTensors();
}
```

Uma vez criado o interpretador, é preciso chamá-lo para executar a inferência e, consequentemente, classificar o gesto, o que será feito por meio da função “Invoke()”. A função pode ser dividida em três partes. No início, é feito um pré-processamento do *frame* capturado, na qual são retirados os componentes ‘RGB’ da imagem via o método ‘grayscale()’, para que fique de acordo com os dados usados para o treinamento da rede neural. Em seguida, o interpretador que foi criado no passo anterior é chamado e, por fim, seu resultado é avaliado.

```
void Invoke(WebCamTexture texture) {
        isProcessing = true;
```

```
        Color[] pixels = texture.GetPixels();
        for (int i = 0; i < 128; i++) {
            for (int j = 0; j < 128; j++) {
                int W = (int) (texture.width * ((float)j / 128));
                int H = (int) (texture.height * ((float)i / 128));
                inputs[i, j] = pixels[H * texture.width + W].grayscale;
            }
        }

        interpreter.SetInputTensorData(0, inputs);
        interpreter.Invoke();
        interpreter.GetOutputTensorData(0, outputs);

        if (outputs[0] > 0.5) {
            outputTextDisplay.text = (
                "FECHADA: " + outputs[0] * 100).ToString() + "%");
        } else {
            outputTextDisplay.text = (
                "ABERTA: " + ((1 - outputs[0]) * 100).ToString() + "%");
        }
        isProcessing = false;
}
```

O resultado do interpretador é uma porcentagem indicando a chance de a imagem representar uma mão fechada. De acordo com esse valor, optou-se por indicar com um texto na tela do jogo (“outputTextDisplay”) o estado da mão. Assim, é possível validar a integração da rede neural ao projeto *Unity*.

Após adicionar o objeto de texto à cena principal e atribuí-lo à variável “outputTextDisplay” do *script* de classificação de imagem, o jogo pode ser executado e o funcionamento do interpretador validado.

Figura 13: Execução do jogo após a adição de *“ImageClassificationScript”*.

Fonte: Produção própria em projeto Unity.

APÊNDICE D - Sistema de Eventos Customizados

O projeto desenvolvido no ‘Apêndice C’ será usado como base para a criação do sistema de eventos. Como mencionado no capítulo “Projeto e Implementação” deste documento, o *package* a ser criado deve conter três eventos: detecção de mão - para criar a interface de controle baseada nos gestos -, requisição de login e requisição de *upload* de vídeo. Porém, a demonstração será feita apenas para a detecção de gestos, já que o processo dos demais é análogo.

O primeiro passo é criar a entidade controladora dos eventos (*broker*), que fará a interface entre os *Publishers* e os *Subscribers*. Para isso, deve ser criado um novo objeto de jogo vazio na cena, chamado “EventsManager” e, a ele, será atribuído uma nova classe, “EventsManager”, apresentada a seguir.

```
public class EventsManager : MonoBehaviour {
    public static EventsManager instance;
    //...

    public event Action<int, Boolean> MoveHandTrigger;
    public void OnHandMovementTrigger(int instanceId, Boolean isOpenHand) {
        MoveHandTrigger?.Invoke(instanceId, isOpenHand);
    }
}
```

No código, a classe “EventsManager” representa o *broker* e, para cada evento, possui dois componentes: uma variável pública que representa o evento e um método a ser chamado para invocar os *Subscribers* do evento.

A variável “MoveHandTrigger” e seus parâmetros são a mensagem que será enviada de um *Publihser* aos *Subscribers* no acionamento de “OnHandMovementTrigger(), que indica o disparo de um evento, onde a linha “MoveHandTrigger?.Invoke(instanceId, isOpenHand);” é responsável por acionar o *subscriber*.

Com o evento definido no *broker*, é necessário criar o código para seu disparo, o que será feito no *script* de classificação de gestos. No método “Invoke()” do “ImageClassificationScript”, existe uma condição que analisa a saída do interpretador e, de acordo com o seu valor, define se a imagem contém uma mão aberta ou fechada. Este é o local ideal para adicionar o disparo do evento e configurar as mensagens corretamente.

```
if (outputs[0] > 0.5) {
```

```
   EventsManager.instance.OnHandMovementTrigger(gameObject.GetInstanceID(),
false);
} else {
  EventsManager.instance.OnHandMovementTrigger(gameObject.GetInstanceID(),
true);
}
```

A linha “EventsManager.instance.OnHandMovementTrigger()” é responsável por acessar o *broker* e executar a função “OnHandMovementTrigger()”, que, como dito antes, acionará os *Subscribers*. Dessa forma, o disparo do evento está criado.

A detecção do evento do lado de um *Subscriber* exige duas ações: configurá-lo para receber a mensagem do *broker* e definir a ação a ser tomada após o acionamento. Para isso, é necessário fazer uma alteração na função “Start()” do *script* que escutará o evento, para configurar o objeto ao qual ele está conectado para ouvir as mensagens geradas pelo disparo de “MoveHandTrigger”.

```
void Start() {
    EventsManager.instance.MoveHandTrigger += changeSize;
}
```

Ao escutar o evento, a função “changeSize” será acionada. Neste exemplo, a função altera o tamanho do objeto dependendo da mensagem recebida. Ao adicionar um objeto na cena em formato de cubo à cena e atribuir o *script* de mudança de tamanho a ele, é possível validar o funcionamento do sistema de eventos.

Figura 14: Demonstração do sistema de eventos para a detecção de gestos.

Fonte: Produção própria em projeto Unity.

APÊNDICE E - Captura de Imagens e Comunicação com Servidor WEB

A adição das funcionalidades de captura de imagens e comunicação com servidor WEB a um projeto *Unity* serão baseadas no projeto resultante do procedimento realizado no Apêndice D deste documento, por já constar um sistema de eventos e a capacidade de analisar *frames* da câmera.

O primeiro passo é o armazenamento de quadros. Como as imagens serão posteriormente enviadas ao servidor utilizando o *UnityWebRequest*, é importante analisar esta ferramenta para decidir o formato em que as imagens ficarão salvas. Ao analisar sua documentação, percebe-se que a classe de comunicação HTTP do *Unity* possui facilidade em enviar parâmetros de texto. Portanto, o formato selecionado para guardar os *frames* é uma lista de *strings*, na qual cada termo representa a imagem convertida em uma *string* em *Base64*.

Assim, cada quadro deverá passar por um pré-processamento, no qual é convertido em dados binários e, posteriormente, em *Base64*. Este processo ocorrerá no método “AddToFrameList()”, que receberá como parâmetro a mesma imagem passada ao interpretador.

```
void AddToFrameList(WebCamTexture webcamTexture) {
    Texture2D texture = new Texture2D(webcamTexture.width,
webcamTexture.height, TextureFormat.ARGB32, false);

    // Save the image to the Texture2D
    texture.SetPixels(webcamTexture.GetPixels());
    texture.Apply();

    // Save the Base64 string on the list
    string base64Tex = System.Convert.ToBase64String(texture.EncodeToPNG());
    framesList.Add(base64Tex);
    framesRecorded++;
}
```

A chamada desta função deve ocorrer sempre que for desejado armazenar um *frame* para pertencer ao vídeo. Para isso, foi adicionada uma parametrização que permite ao desenvolvedor de jogos customizar a partir do menu de inspeção quais serão os momentos em que um quadro é salvo. Os parâmetros indicam qual o *frame* inicial a partir do qual será iniciada a captura (“InitialFrame”) e quantos *frames* serão salvos (“FramesToRecord”). Dessa forma, o código de “Update()” do *script* de classificação de imagens foi atualizado para chamar “AddToFrameList()”, considerando tais parâmetros.

```
 void Update() {
    if (!isProcessing && cameraTexture)
        Invoke(cameraTexture);

    if (frameCounter >= initialFrame &&
        framesRecorded < framesToRecord &&
        frameCounter % sampleRate == 0)
    {
        AddToFrameList(cameraTexture);
    }

    if (framesRecorded >= framesToRecord)
        frameCounter++;
}
```

Com a captura de imagens finalizada, o próximo passo é implementar a comunicação com servidor WEB usando o *UnityWebRequest*. Para isso, deve ser criado um novo *script*, chamado “ApiController” que será o responsável pelas requisições. Como explicado no Apêndice E deste documento, existem dois eventos relativos à comunicação via Web: um para login e um para o *upload* de imagens. Assim, o “ApiController” deve ser tratado com um *Subscriber* a tais eventos, e a escuta aos seus disparos deve ser configurada no método “Start()”. Além disso, para evitar atrasos no jogo, os métodos que executam a requisição devem ser desenvolvidos como corrotinas, um tipo diferente de função em C#, que não podem ser chamados diretamente pela infraestrutura de eventos, o que implica na criação de um método extra que será chamado pelo evento e acionará a função que realiza a requisição. A estrutura básica do *script* é apresentada a seguir.

```
public class ApiController : MonoBehaviour {
    public string uploadImagesUrl = "http://127.0.0.1:5000/uploadImages";
    public string loginUrl = "http://127.0.0.1:5000/login";
    public GameObject imageClassGameObject;

    void Start() {
        EventsManager.instance.UploadImagesTrigger += requestImageUpload;
        EventsManager.instance.LoginTrigger += requestLogin;
    }

    public void requestLogin(int id, string username, string password, bool
remember_login) {
        StartCoroutine(Login(username, password, remember_login, loginUrl));
    }

    public static IEnumerator Login(string username, string password, bool
remember_login, string url) {
        //...
```

```
    }

    public void requestImageUpload(int id) {
        ImageClassificationScript imageClassification =
imageClassGameObject.GetComponent<ImageClassificationScript>();
        List<string> framesList = imageClassification.framesList;
        StartCoroutine(UploadImages(framesList, uploadImagesUrl));
    }

    public static IEnumerator UploadImages(List<string> frames, string url) {
       //...
    }
}
```

Os métodos “requestLogin()” e “requestImageUpload()” são acionados pelo evento e, por sua vez, iniciam as corrotinas “Login()” e “UploadImages()”. Outro ponto relevante do esqueleto acima é a obtenção da lista de *frames* do objeto de classificação de imagem, o que significa que “ApiController” deve ter uma referência a este objeto.

Para implementar as requisições, como já dito, será feito o uso do *UnityWebRequest*. O envio do login será feito por meio de uma requisição com um corpo no formato JSON. Por limitações da ferramenta do *Unity*, tal requisição deve ser feita utilizando o método PUT do padrão REST. Os campos enviados no corpo são o login, senha e uma *flag* para manter o usuário logado. Seu código está apresentado a seguir.

```
public static IEnumerator Login(string username, string password, bool
remember_login, string url) {
    LoginRequest requestBody = new LoginRequest();
    requestBody.email = username;
    requestBody.password = password;
    requestBody.remember_login = remember_login;

    string requestString = JsonUtility.ToJson(requestBody) ?? "";
    UnityWebRequest www = UnityWebRequest.Put(url, requestString);
    www.SetRequestHeader("Content-Type", "application/json");
    yield return www.SendWebRequest();
}
```

Para a construção do corpo da requisição, foi criada a classe “LoginRequest”, que detalha os campos que devem ser utilizados com o login. Após instanciar um objeto dessa classe utilizando os parâmetros enviado pelo evento, ele é transformado em uma *string* JSON e a requisição é enviada.

Já o envio das imagens não pode ser feito em JSON por limitações do *Unity* e, portanto, deve-se utilizar a estrutura de *forms*, adicionando um campo para cada *string* que representa um *frame* convertido em *Base64*. Com essa estrutura, não existe a mesma limitação do JSON, então ela pode ser enviada utilizando o método POST.

```
public static IEnumerator UploadImages(List<string> frames, string url) {
    WWWForm form = new WWWForm();
    for (int i = 0; i < frames.Count; i++)
    {
        form.AddField("frame" + i, frames[i]);
    }

    UnityWebRequest www = UnityWebRequest.Post(url, form);
    yield return www.SendWebRequest();
}
```

O último passo é criar os *Publishers* que dispararão os eventos “UploadImagesTrigger” e “LoginTrigger”. Isso é feito por meio da adição de duas linhas de código nos métodos que os lançarão, por exemplo, um método executado ao pressionar um botão da interface. A linha necessária para o login é:

```
EventsManager.instance.OnLoginTrigger(gameObject.GetInstanceID(), login,
password, remember_login);
```

Nela, “login” e “password” são *strings* e “remember_login” é uma booleana. A linha necessária para o *upload* de imagens é:

```
EventsManager.instance.OnUploadImagesTrigger(gameObject.GetInstanceID());
```