UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

DANILO CÉSAR SILVA SOARES
Orientadora: Profª. Dra. Andrea Gomes Campos
Coorientador: Msc. Breno Nunes de Sena Keller

IMPLEMENTAÇÃO DE ARQUITETURA MODULAR PARA INTEGRAÇÃO DE CLASSIFICADOR NO APLICATIVO CITOFOCUS

Ouro Preto, MG
2025

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

DANILO CÉSAR SILVA SOARES

IMPLEMENTAÇÃO DE ARQUITETURA MODULAR PARA INTEGRAÇÃO DE CLASSIFICADOR NO APLICATIVO CITOFOCUS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profª. Dra. Andrea Gomes Campos
Coorientador: Msc. Breno Nunes de Sena Keller

Ouro Preto, MG
2025

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S676i Soares, Danilo César Silva.
Implementação de arquitetura modular para integração de classificador no aplicativo Citofocus. [manuscrito] / Danilo César Silva Soares. - 2025.
48 f.: il.: color., tab..

Orientadora: Profa. Dra. Andrea Gomes Campos.
Coorientador: Me. Breno Nunes de Sena Keller.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Inteligência artificial - Aplicações médicas. 2. Aplicativos móveis. 3. Visão por computador. 4. Citopatologia. I. Campos, Andrea Gomes. II. Keller, Breno Nunes de Sena. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.8:616-091.8

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Danilo César Silva Soares

Implementação de Arquitetura Modular para Integração de Classificador no Aplicativo CitoFocus

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 2 de Abril de 2025.

Membros da banca

Andrea Gomes Campos (Orientadora) - Doutora - Universidade Federal de Ouro Preto
Breno Nunes de Sena Keller (Coorientador) - Mestre - Universidade Federal de Ouro Preto
Fernanda Sumika Hojo de Souza (Examinadora) - Doutora - Universidade Federal de Ouro Preto
Reinaldo Silva Fortes (Examinador) - Doutor - Universidade Federal de Ouro Preto

Andrea Gomes Campos Bianchi, Orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 2/04/2025.

Documento assinado eletronicamente por **Andrea Gomes Campos**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/04/2025, às 18:14, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0886269** e o código CRC **78B4145D**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Dedico esse trabalho à todos os amigos, professores, principalmente os envolvidos no trabalho, e familiares, especialmente meus pais. Agradeço a todos por me apoiaram durante a jornada de estudos e aprendizado e a grande contribuição dessas pessoas na minha formação, pessoal e profissional.

Resumo

Este trabalho apresenta o desenvolvimento e a evolução do CitoFocus, um aplicativo voltado ao apoio diagnóstico em citopatologia do colo do útero. O objetivo principal é facilitar a troca de informações entre citopatologistas por meio do compartilhamento seguro de imagens e dados relevantes, promovendo a colaboração entre especialistas. Inicialmente concebido para permitir discussões clínicas com base em casos reais, o aplicativo foi aprimorado com uma reestruturação completa do *backend*, o que garantiu maior modularidade, escalabilidade e desempenho. Além disso, foi integrada uma solução baseada em inteligência artificial capaz de classificar imagens e identificar possíveis lesões, fornecendo resultados visuais e explicativos acessíveis tanto ao autor do caso quanto aos demais usuários que interagirem com ele. As melhorias implementadas tornaram o CitoFocus uma ferramenta mais eficiente, colaborativa e tecnológica, potencializando o suporte à análise diagnóstica e contribuindo para a melhoria da qualidade do rastreamento do câncer de colo do útero.

Palavras-chave: Aplicativo. Classificador. *Back-End*. Arquitetura. Câncer de colo de útero. Citopatologia.

Abstract

This work presents the development and evolution of CitoFocus, an application designed to support diagnostic processes in cervical cytopathology. The main objective is to facilitate information exchange among cytopathologists through the secure sharing of images and relevant clinical data, encouraging collaboration between specialists. Initially created to enable case-based clinical discussions, the app has undergone a complete backend restructuring, resulting in improved modularity, scalability, and system performance. Furthermore, an artificial intelligence-based classifier was integrated to analyze images and identify potential lesions, providing visual outputs and explanatory summaries accessible to both the case author and users who interact with the case. These enhancements have made CitoFocus a more efficient, collaborative, and technologically advanced tool, strengthening diagnostic support and contributing to improved cervical cancer screening quality.

Keywords: Application. Classifier. Back-End. Architecture. Cervical cancer. Cytopathology.

Lista de Ilustrações

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
UFOP	Universidade Federal de Ouro Preto
CEA	*Cytopathologist Eye Assistant*
APIs	*Application Programming Interface* - Interface de programação de aplicações
DTO	*Data Transfer Object* - Objeto de transferência de dados
MVC	*Model-View-Controller*
ML	*Machine Learning* - Aprendizado de máquina
CNN	*Convolutional Neural network* - Redes Neurais Convolucionais
YOLO	*You Only Look Once*
SR	*Softmax Regression*
SVM	*Support Vector Machine*
GEDT	*GentleBoost Ensemble of Decision Tree*
ID	identity - identidade

Sumário

1 Introdução

Segundo World Health Organization (2024), o câncer cervical é o quarto câncer mais comum em mulheres no mundo. Além disso, esse câncer apresenta as maiores taxas de ocorrência de novos casos e de mortalidade em países subdesenvolvidos e em desenvolvimento. Já no Brasil, é o terceiro tumor maligno mais frequente em mulheres e ocupa a quarta posição entre as principais causas de mortalidade por câncer em mulheres no país, conforme dados do Instituto Nacional do Câncer (2022).

O estágio pré-câncer, aquele onde a doença ainda não desenvolveu, raramente causa sintomas, por isso seguir os protocolos de rastreamentos é importante, visto que a identificação nesse estágio resulta em tratamentos eficazes na grande maioria dos casos. Essa etapa é caraterizada por exames, onde se coleta amostras e profissionais citopatologistas fazem as análises e o diagnóstico.

Conforme World Health Organization (2024), a desigualdade de acesso ao serviço de triagem é um fator determinante para a maior ou menor incidência do câncer. O exame Papanicolaou convencional – ou esfregaço cervical, coleta das células por meio da raspagem da superfície do colo - é o meio mais comum nos países de média e baixa renda para diagnóstico da doença. Cada lâmina do esfregaço possui milhares de células e devido a isso a etapa da análise em laboratório é repetitiva, exaustiva e depende diretamente da experiência do profissional, características que tornam o processo difícil. Dificuldades essas que muitas vezes acabam resultando em erros nos diagnósticos que contribuem para o aumento da incidência desse tumor, no caso de falsos-negativos, e consequências negativas nas vidas dos pacientes em casos de falsos-positivos, (REZENDE; BIANCHI; CARNEIRO, 2021).

Para minimizar as falhas cometidas, os citopatologistas buscam ativamente oportunidades de pesquisa e troca de conhecimentos com seus pares, participando de discussões e estudos de caso que enriquecem suas experiências e contribuem para diagnósticos mais precisos. Para permitir essa comunicação, é muito comum que eles utilizem ferramentas de comunicação, como mídias sociais e aplicativos de mensagens.

Nesse contexto, foi desenvolvido o CitoFocus, (GUIMARÃES, 2021; KELLER et al., 2021), um aplicativo *mobile* que apoia os citopatologistas permitindo que eles compartilhem casos em ambiente especializado. Ao postar informações sobre o caso que estão trabalhando os profissionais criam discussões com seus pares, possibilitando a troca de informações e solução de dúvidas.

O aplicativo permite que os usuários criem estudos de caso, interagindo em quadros que estão sendo analisados por colegas por meio de votos e justificativas, auxiliando o autor na análise de seu caso e os demais profissionais em discussões elucidativas.

Como citado, uma das formas de interação no aplicativo ocorre por meio dos votos nos casos. Nesse contexto, este trabalho promoveu a integração de um classificador — algoritmo ou método de inteligência artificial treinado para classificar objetos automaticamente — ao aplicativo, permitindo que ele analisasse as imagens e interagisse com o sistema por meio de votos automáticos. Para isso, foi necessária uma reestruturação do *back-end* do aplicativo, viabilizando a coleta e o uso dos dados gerados pelas interações dos profissionais, os quais foram utilizados para treinar o classificador. Assim, o sistema passa a contar com uma nova forma de resposta, complementar à dos usuários reais.

1.1 Justificativa

O CitoFocus originalmente foi desenvolvido com o propósito de permitir a colaboração à distância entre citopatologistas, facilitando o compartilhamento de informações e a construção de análises de casos em um ambiente profissional. Os usuários citopatologistas podem criar casos, inserindo imagens e outras informações que permitam a análise e que não comprometam a privacidade do paciente. Também podem participar das votações nos casos criados por seus pares, escolhendo uma opção e fornecendo uma justificativa.

Ainda no contexto de proporcionar um ambiente adequado para discussões entre os especialistas, a ideia de ampliá-lo adicionando a funcionalidade de gerar uma classificação automática usando inteligência artificial vai ao encontro de adicionar mais uma opinião sobre o problema. Entretanto, acrescentar um classificador na arquitetura já existente é um desafio, do ponto de vista da ferramenta, e do uso de IA em dispositivo móvel. Desse modo, nossa proposta é ter uma arquitetura modular que permita a integração de um classificador ao aplicativo CitoFocus. Com o desenvolvimento dessa ferramenta teremos mais uma opinião sobre o caso, que poderá apoiar a análise dos profissionais.

1.2 Objetivos

O objetivo deste trabalho é propor e implementar uma arquitetura modular que permita integrar o classificador proposto por Diniz et al. (2022) no CitoFocus sem causar impacto no funcionamento do aplicativo.

1.2.1 Objetivos Específicos

Este trabalho também tem como objetivos específicos:

- Desenvolver o *back-end*;
- Modularizar a arquitetura do *back-end*;

- Implementar o classificador proposto por Diniz et al. (2022) em um formato consumível pelo aplicativo;
- Integrar o classificador no aplicativo;
- Analisar o desempenho do sistema e realizar testes.

1.3 Organização do Trabalho

O Capítulo 2 fornece uma revisão da literatura e discute trabalhos relevantes relacionados ao tema. O Capítulo 3 detalha a metodologia de desenvolvimento do projeto, enquanto o Capítulo 4 apresenta os resultados obtidos. Finalmente, o Capítulo 5 oferece as conclusões do estudo.

2 Revisão Bibliográfica

Este Capítulo apresenta a fundamentação teórica deste trabalho, com a finalidade de contextualizar a pesquisa através de conceitos teóricos e soluções publicadas por outros pesquisadores da área. Na Seção 2.1, são introduzidos os conceitos fundamentais que irão sustentar o desenvolvimento desta pesquisa. Já na Seção 2.2, são descritos estudos relacionados ao tema, que ajudam a entender o problema e suas possíveis soluções.

2.1 Fundamentação Teórica

Nesta Seção, é apresentada e discutida a fundamentação teórica e as principais ideias utilizadas no desenvolvimento do trabalho.

2.1.1 Citopatologia, Pré-Neoplasias e Neoplasias

> O exame citopatológico é um teste realizado para detectar alterações nas células do colo do útero que possam predizer a presença de lesões precursoras do câncer ou do próprio câncer. É a principal estratégia para detectar lesões precocemente. A técnica de coleta adequada e no momento e condições oportunas garante um espécime de melhor qualidade e fornece resultados mais confiáveis. (FUNDAÇÃO OSWALDO CRUZ (2023))

As chamadas lesões precursoras do câncer são as lesões pré-neoplásicas, isto é, lesões que indicam a possibilidade do surgimento de neoplasias malignas.

Em relação a essas lesões, ou neoplasias, são tumores que cuja origem ocorre pelo crescimento anormal das células, de acordo com Reisner (2015). É importante lembrar que tais neoplasias podem ser tanto malignas quanto benignas, por isso a importância de identificar essa lesão, e assim, mais importante ainda a detecção de lesões pré-neoplásicas, pois a partir delas é possível começar a fase de tratamento antes que o câncer se estabeleça no organismo da paciente.

A identificação na fase em que a doença ainda não se estabeleceu, a fase pré-neoplásica, e seu posterior tratamento aumenta significativamente as chances de cura, (World Health Organization, 2024). Isso ocorre, pois as alterações celulares ainda não se tornaram o câncer propriamente dito, que é estágio mais grave da doença. É importante ressaltar que no estágio onde o tumor maligno ainda não se estabeleceu, raramente o paciente é acometido por sintomas. Por isso a importância em seguir os protocolos médicos de exames que aumentam as chances de cura dessa doença.

2.1.2 Exame de Papanicolaou

O exame de Papanicolaou é realizado com o intuito de identificar alterações nas células do colo do útero, sendo esta a principal estratégia para identificar lesões precursoras do câncer de colo de útero. O teste de Papanicolaou convencional, também conhecido como esfregaço cervicovaginal, ocorre com a coleta da amostra de células a partir da raspagem do colo de útero e posterior análise das células em laboratório, sendo um método barato, não invasivo, eficaz e simples de realizar, segundo (MAMMAS; DA; SACHAN et al., 2012, 2018 apud REZENDE; BIANCHI; CARNEIRO, 2021, p. 2).

Cada lâmina enviada para análise possui milhares de células a serem analisadas pelos profissionais via inspeção visual, cenário este que alinhado ao fato de o trabalho ser repetitivo, exaustivo e depender da experiência de cada citopatologista que tornam o processo de exame difícil e delicado. Devido a esses características que diversos estudos tentam diminuir a ocorrência de cenários onde temos falsos-positivos e falsos-negativos no diagnóstico. Os primeiros causam impactos negativos no bem estar dos pacientes, como ansiedade, sofrimento e tratamentos excessivos desnecessários e o segundo pode levar a detecção tardia que diminui as chances de cura da doença.

É extremamente importante ressaltar a necessidade da paciente sempre continuar a fazer o exame nas datas previstas segundo orientação médica, para que qualquer alteração seja identificada e o tratamento precoce realizado, que aumenta enormemente as chances de cura, segundo World Health Organization (2024).

2.1.3 Aplicativo CitoFocus

CitoFocus é um aplicativo móvel desenvolvido para permitir a colaboração e estudo entre profissionais citopatologistas em um ambiente apropriado para essas discussões, (GUIMARÃES, 2021; KELLER et al., 2021). A colaboração entre os profissionais ocorre por meio da criação de casos no aplicativo, fornecendo imagens das células que o profissional deseja esclarecer dúvidas e outras informações que permita a análise, como idade do paciente e descrição com outras informações relevantes. Por fim, é pedido que o usuário selecione as opções que estarão disponíveis para voto, sendo pré-definido uma máximo de 4 opções, e no final é adicionado a opção “Outros” automaticamente, a Figura 2.1 mostra a tela inicial do aplicativo.

O apoio fornecido pela ferramenta é a construção de um ambiente adequado para as discussões de casos analisados no exame de Papanicolaou. A plataforma proporciona a colaboração entre profissionais especializados por meio do compartilhamento de informações, experiência e estudos de casos, objetivando auxiliar no esclarecimento de dúvidas que podem surgir durante a análise clínica de casos.

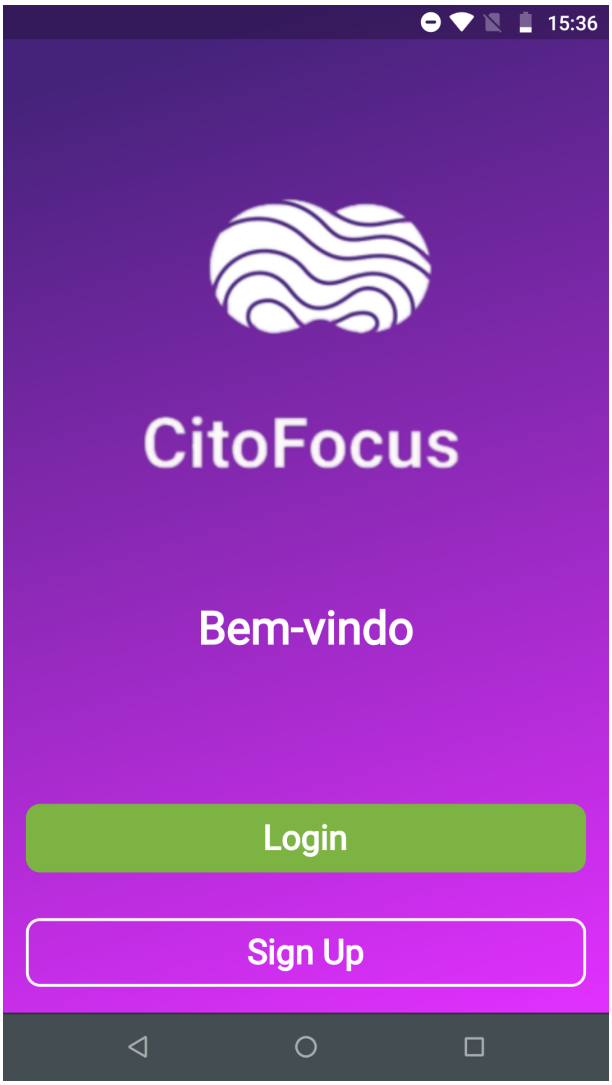
Sobre a colaboração entre os profissionais especializados, é necessário que eles comprovem sua especialização na plataforma para ser permitido que criem casos e participem das

votações. Caso contrário suas contas serão do perfil visitante - perfil definido na criação da conta e que não precisa comprovar formação - que permite somente visualização dos casos na plataforma.

Com a postagem de um caso por um usuário, outros usuários da mesma categoria podem votar e justificar seu voto. Essa interação é permitida enquanto o caso estiver aberto, visto que seu autor pode inserir uma data de encerramento, caso não deseje o padrão de 10 dias duração.

Com o encerramento dos casos eles ainda continuam armazenados na plataforma. Com essa característica aliada ao fato de existir a conta do perfil visitante, estudantes da área podem se cadastrar na plataforma para utilizarem o aplicativo. Porém, esse uso se limita a estudos de casos reais e a visualizar as justificativas dos profissionais.

Figura 2.1 – Tela inicial do aplicativo CitoFocus.



2.1.4 NestJS

O *NestJS* é um *framework* para *Node.js*[1] que facilita o desenvolvimento de aplicações do lado do servidor, serviço *back-end*. Ele utiliza *TypeScript* como sua linguagem principal, mas também é compatível com *JavaScript*.

Entre suas características mais notáveis estão uma arquitetura escalável, que permite organizar projetos complexos de maneira eficiente, ampla integração com diversos sistemas de banco de dados, e um robusto suporte para a criação de microsserviços. Para manter essas características o *framework* recomenda a utilização de entidades na estruturação do código, pois com a separação, cada regra é definida em sua respectiva entidade resultando em maior separação das funções.

2.1.5 Arquitetura Hexagonal

A arquitetura hexagonal, também conhecida como Arquitetura de Portas e Adaptadores, visa criar um sistema desacoplado e flexível, facilitando a manutenção e a inclusão de novas funcionalidades, exemplo da arquitetura na Figura 2.2. O conceito central é isolar o núcleo da aplicação (lógica de negócios) das interações externas, como bancos de dados, APIs (*Application Programming Interface* - Interface de programação de aplicações), interfaces de usuário ou serviços externos. Isso é feito através de portas, que definem como o núcleo se comunica com o exterior, e adaptadores, que implementam essas interações de forma específica.

Com isso, a aplicação se torna independente de tecnologias externas, tornando mais fácil a troca de implementações (como substituir um banco de dados ou interface) sem impactar a lógica central. Além disso, a arquitetura favorece o desenvolvimento orientado a testes, já que permite testar o núcleo da aplicação sem dependências externas.

Segue abaixo a definição dos principais componentes:

- Núcleo: é a parte central da aplicação, onde contém todas as regras que são independentes das implementações externas.
- Portas: interfaces que definem como o núcleo interage com os serviços externos (*front-end* e banco de dados). É a definição das regras dessa interação.
- Adaptadores: implementação das portas que permite a interação com serviços externos.
- DTO (*Data Transfer Object*): objetos de transferência de dados, são classes que são definidas para serem utilizadas durante a transferência de dados entre camadas da aplicação, definindo os dados a serem transferidos e a validação da transferência.

[1] https://nodejs.org/en/ Acessado em: 23/08/2024

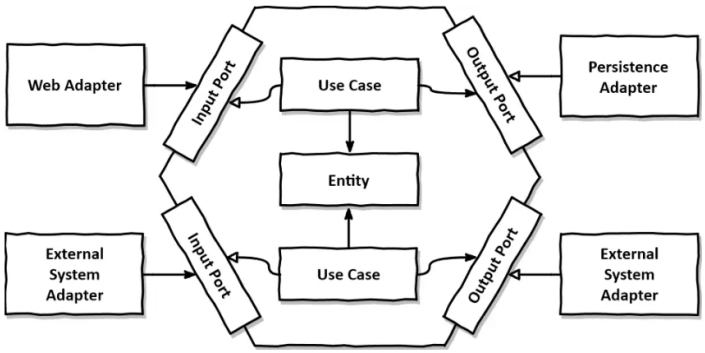


Figura 2.2 – Imagem que representa o exemplo de uma arquitetura hexagonal com suas camadas de comunicação. Retirado de Hombergs (2019)

2.1.6 Arquitetura MVC

A arquitetura MVC (*Model-View-Controller*) é um padrão de design amplamente utilizado no desenvolvimento de software, especialmente em aplicações *web*. Ela divide a aplicação em três componentes principais:

- *Model*: responsável pela lógica de dados, regras de negócios e comunicação com o banco de dados. O *model* gerencia o estado da aplicação e reage a solicitações de alteração ou consulta dos dados.
- *View*: encarregada da apresentação dos dados ao usuário. A *view* exibe as informações do *model* de forma visual, como páginas *web*, e não contém lógica de negócios.
- *Controller*: atua como intermediário entre o *model* e a *view*. Ele recebe as interações do usuário, processa as solicitações (consultando ou alterando o *model*) e atualiza a *view* conforme necessário.

Essa separação de responsabilidades facilita a manutenção, permite o desenvolvimento paralelo e favorece a reutilização de código.

Ambas as arquiteturas apresentadas — Hexagonal e MVC — foram consideradas na estruturação do sistema desenvolvido. No Capítulo 3, será detalhado como esses conceitos foram aplicados na prática, orientando as decisões de implementação e organização do código.

2.1.7 Modularização

De acordo com Gado (2021) modularização é o processo em que separamos as funções de um sistema em pequenas partes que funcionam de forma independente. O objetivo ao desagrupar essas rotinas é proporcionar maior flexibilidade durante o desenvolvimento do sistema ou inclusão de novas funcionalidades, tanto nos que já foram publicados quanto os que ainda estão em fase de desenvolvimento.

Gomes, Romano e Dias (2023) salientam a importância da modularização para facilitar a entrega de novas funcionalidades, ou incrementos, de uma aplicação, destacando o cenário de aplicações para dispositivos móveis. Os autores informam que, ao aplicar a estratégia de desagrupar a aplicação em pequenos contextos, reduzimos o tempo de desenvolvimento, o tempo de submissão das alterações para o cenário de produção - cenário após os testes em que o aplicativo já está disponível para os usuários utilizarem - permitindo o aumento de resultados.

Cada contexto que for desagrupado será responsável por uma função específica, de forma que elas possam ser desenvolvidas separadamente, sem que uma prejudique o desenvolvimento da outra. Nesse contexto, em caso de melhorias ou correções de falhas, não seria necessário trocar todo o conjunto, e os desenvolvedores atuariam somente no elemento de interesse.

Em cenários de desenvolvimento, podemos ter mais de um desenvolvedor atuando sem que esperem o outro finalizar uma rotina para testar o seu desenvolvimento, resultando nessa etapa ser mais objetiva e clara. Outro cenário interessante é a submissão para testes, o desagrupamento permite que partes sejam testadas enquanto outras ainda não foram concluídas.

2.1.8 *Machine and Deep Learning*

O conceito de inteligência artificial é amplo e se refere a algoritmos projetados para simular a inteligência humana. Já *Machine Learning* (ML), aprendizado de máquina na tradução literal, trata-se de uma área da inteligência artificial que se dedica ao desenvolvimento de sistemas capazes de aprender, identificar padrões e tomar decisões sem muita intervenção humana, (MITCHELL, 1999 apud FERNANDES; FILHO, 2019, p. 2). E *Deep Learning*, ou aprendizado profundo, se refere a um ramo de ML, que utiliza de redes neurais artificiais profundas para aprender a partir dos dados disponibilizados, (LECUN; BENGIO; HINTON, 2015 apud LUDERMIR, 2021, p. 5).

As redes neurais são algoritmos em camadas inspirados no cérebro humano no qual os dados são processados em diferentes etapas com diferentes objetivos. Inicialmente temos a etapa de treinamento da rede, onde são fornecidos dados já classificados para se ter controle de como ela está sendo treinada. Após essa etapa temos a fases de testes, no qual um novo conjunto de dados também classificado, é processado pela rede sendo verificado se a rede foi treinada corretamente ou a necessidade de ajustes. Logo após essas fases, a rede é capaz de classificar novos dados de forma autônoma, Holdsworth e Scapicchio (2024).

Ainda sobre as redes neurais, temos as redes neurais convolucionais (CNN) que é um tipo de rede neural profunda projetada para o processamento de dados em forma de grade, como pode ser visto na Figura 2.3.

Dentro do contexto descrito, temos os classificadores, que podem ser *Deep Learning* (utilizando CNNs profundas), ou não. Os classificadores são algoritmos de ML que são aplicados com o intuito de atribuir uma classificação, ou etiqueta, a um dado avaliado. Um classificador

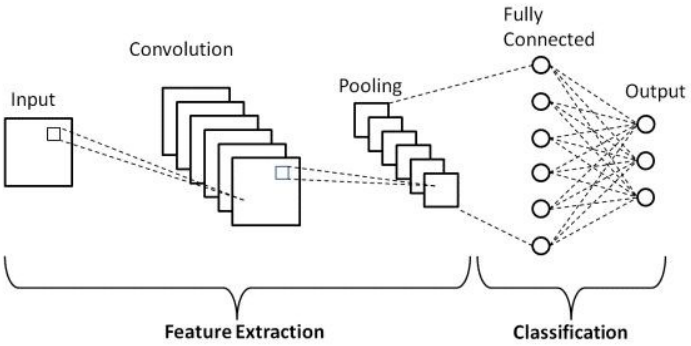


Figura 2.3 – Imagem que representa o exemplo do funcionamento de uma CNN. Retirado de Couto (2023)

atribui uma classe a um elemento avaliado conforme seus processos. É importante destacar que, alguns algoritmos somente realizam a classificação, sendo necessário outro algoritmo para extrair o material necessário para essa tarefa.

Nesse sentido, o YOLO descrita na Seção 2.1.8.1, realiza a detecção - necessário para o contexto de redes que classificam imagens -, extração e classificação, sendo um exemplo de um classificador *Deep Learning*.

2.1.8.1 Algoritmo *YOLO (You Only Look Once)*

YOLO, acrônimo para *You Only Look Once* (em português Você Só Olha Uma Vez), é um método de detecção de objetos em imagens e vídeos cujo nome carrega sua principal característica, a detecção e classificação passando a imagem pelo método somente uma vez.

O algoritmo YOLO (You Only Look Once), proposto por Redmon et al. (2016), inovou significativamente os métodos de detecção de objetos em imagens e vídeos ao reformular o processo tradicional. Em vez de depender de propostas de regiões de interesse e aplicar classificadores separadamente — abordagem comum nos métodos anteriores, porém computacionalmente custosa —, o YOLO divide a imagem de entrada em uma grade de células. Cada célula é responsável por detectar objetos, estimar sua localização e classificar sua categoria, tudo em uma única etapa. Segundo Terven, Córdova-Esparza e Romero-González (2023), essa abordagem representou uma mudança de paradigma na área, pois reduziu consideravelmente o tempo de processamento mantendo uma boa acurácia, o que a tornou uma das técnicas mais influentes da última década.

O YOLO utiliza CNNs profundas que realizam a tarefa descrita de forma eficiente e rápida, resultando em um método veloz e de qualidade. Ele caracteriza-se por ser uma ferramenta inovadora poderosa, que possui grande velocidade e eficiência na detecção de objetos em tempo real. Atualmente é utilizada em diversas áreas, desde a casos em que se analisa imagens a outros em que é utilizada em análise de vídeos para segurança e veículos autônomos.

Após a publicação da primeira versão, diversas versões com aprimoramentos foram

desenvolvidas tanto pelos autores originais quanto por outros pesquisadores. Durante a escrita deste trabalho, a versão mais recente identificada é o YOLOv10. Além disso, cada versão possui três opções de tamanhos diferentes, sendo a "s", "m" e "l", ordenadas de forma crescente de tamanho e de potência de classificação, visto que a maior possui maior precisão.

Nesse contexto de tamanho, as versões menores, principalmente a "s", caracteriza-se por ser menor e mais rápida. A versão "s" foi projetada para ambientes com recursos limitados ou onde a velocidade de detecção e classificação é a prioridade sobre a máxima precisão. Apesar disso, é importante citar que ela apresenta grande precisão, sendo ligeiramente menor quando comparada com as versões "m" e "l".

2.1.8.2 Terminologias

Nessa subseção será descrito termos relacionados a ML que foram utilizadas ao longo deste trabalho.

- *Overfitting*: na tradução literal é ajuste excessivo. É o termo usado quando o modelo se adapta muito bem aos dados de treinamento e começa a captar dados muito específicos e desnecessários desse conjunto de dados. Nesse caso, o modelo não se torna generalizado o que resulta numa baixa precisão na classificação dos dados de teste e até mesmo incapaz de classificar os dados que se deseja, isto é, dados que não foram usados para treino;

- Precisão: é a métrica utilizada para avaliar a exatidão das classificações realizadas pelo classificador. É definido pela proporção entre os verdadeiros positivos entre todas as classificações positivas realizadas pelo modelo, Equação 2.1;

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos(TP)}}{\text{Verdadeiros Positivos(TP)} + \text{Falsos Positivos(FP)}} \tag{2.1}$$

- Sensibilidade: métrica utilizada para avaliar se o classificador é capaz de classificar corretamente dentro de um determinado grupo. Se ele classificou todos os positivos, por exemplo, corretamente ou se existe algum falso-positivo nesse grupo classificado, Equação 2.2;

$$\text{Sensibilidade} = \frac{\text{Verdadeiros Positivos(TP)}}{\text{Verdadeiros Positivos(TP)} + \text{Falsos Negativos(FN)}} \tag{2.2}$$

- Especificidade: métrica utilizada para avaliar se o modelo identificou corretamente todos os negativos. A proporção entre os negativos classificados em relação a todos os negativos do conjunto de dados, Equação 2.3;

$$\text{Especificidade} = \frac{\text{Verdadeiros Negativos(TN)}}{\text{Verdadeiros Negativos(TN)} + \text{Falsos Positivos(FP)}} \tag{2.3}$$

- *F1-Score*: é uma métrica que combina precisão e sensibilidade, como exemplificado na Equação 2.4. Ela é útil em cenários em que há desbalanceamento de classes (desbalanceamento do *dataset*), pois a precisão pode ser tendenciosa.

$$F1\text{-}Score = \frac{\text{Precisão} * \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}} \tag{2.4}$$

  Interpretação:

  1. F1-Score tendendo a 1: indica que o modelo tem boa precisão e sensibilidade.
  2. F1-Score tendendo a 0: indica que o modelo falhou nas classificações.

- Desbalanceamento do *dataset*: um dataset (conjunto de dados) desbalanceado é quando as classes a serem classificadas não estão igualmente representadas no conjunto. Por exemplo, no caso de classificação entre positivo e negativo, temos um conjunto de dados com 100 imagens, porém somente 10 representam o conjunto negativo.

2.2 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos relacionados, sejam eles trabalhos da literatura ou tecnologias similares ao proposto neste trabalho. Além disso, as tecnologias aqui apresentados serão comparados com a tecnologia apresentada no trabalho.

2.2.1 Algoritmos de detecção e classificação de lesões

Para minimizar as falhas destacadas anteriormente dos métodos de rastreamento de lesões, métodos automáticos de leitura de exames de Papanicolaou têm sido desenvolvidos, com o intuito de apoiar os especialistas melhorando o controle de qualidade dos exames e aumentando a detecção de anormalidades nas células.

Um desses trabalhos é tratado por Rehman et al. (2020) que abordam a criação de um sistema automático de triagem para a detecção do câncer, ou de anomalias que indicam os estágios iniciais da doença. O sistema utiliza uma CNN, que é treinada utilizando a base de dados Herlev - banco de imagens de células cervicais extraídas utilizando esfregaço Papanicolaou e classificadas por especialistas - por meio do aprendizado por transferência. Essa técnica é utilizada para reaproveitar um modelo em cenários onde não temos dados suficientes para treiná-lo e que as tarefas são similares. A técnica se caracteriza pelo pré-treinamento em um modelo com muitos dados e em seguida treinado no modelo específico da tarefa.

Com o modelo já treinado, a próxima etapa é a classificação final das amostras celulares. Para isso são propostos três classificadores diferentes: *Softmax Regression* (SR), *Support Vector Machine* (SVM) e *GentleBoost Ensemble of Decision Tree* (GEDT). A performance do sistema é avaliada em dois protocolos de teste diferentes, um de 2 classes (normais e anormais) e outro

de 7 classes (de normais até anormais, com a evolução do grau de severidade) , no sistema de classificação Herlev.

A análise da performance dos resultados demonstrou que o sistema proposto apresenta uma performance superior aos seus antecessores em diversas condições de teste, alcançando uma precisão de 99,6% e sensibilidade de 99,3% para o cenário de duas classes e precisão de 98,85% e sensibilidade de 98,8% para o cenário de sete classes. Essa melhoria na precisão do diagnóstico, se utilizada a ferramenta, diminuiria os erros de diagnósticos associados ao teste PAP convencional. Dessa forma, teríamos um dispositivo eficiente e confiável para a triagem em massa de células para identificar lesões cancerígenas ou pré-cancerígenas.

Já os autores, Rezende, Bianchi e Carneiro (2021) investigaram a viabilidade da implementação de métodos automáticos para a detecção precoce de lesões pré-neoplásicas a partir de outros trabalhos. Os métodos automáticos, como os sistemas ThinPrep Imaging System, FocalPoint GS Imaging System e CytoProcessor, foram avaliados em estudos que analisaram mais de 1,3 milhão de lâminas citopatológicas. Esses sistemas demonstraram desempenhos diagnósticos iguais ou superiores ao método tradicional, reduzindo significativamente os falso-negativos.

Apesar desse desempenho, o uso desses métodos automáticos ainda não é amplamente difundido em laboratórios de citopatologia, principalmente em países de baixa e média renda, devido aos elevados custos. Além destes, nem métodos mais avançados do próprio exame de Papanicolaou, como o método em meio líquido, que diminuem as falhas de diagnósticos não são comuns nesses países também devido aos elevados custos de utilização. Dessa forma, estudos adicionais são necessários para explorar o uso de tecnologias apoiando a citologia convencional para auxiliar na expansão da sua implementação. Com a contínua evolução tecnológica e a redução de custos, a automação promete melhorar significativamente a qualidade dos exames de câncer cervical e facilitando o acesso a essas ferramentas, contribuindo para a redução das taxas de mortalidade associadas a essa doença.

Nesse cenário de tecnologias que apoiam os citopatologistas, Diniz et al. (2022) discutem o desenvolvimento do *Cytopathologist Eye Assistant* (CEA), uma ferramenta que utiliza algoritmos de *deep learning* para auxiliar no rastreamento e classificação de células a partir de imagens de esfregaços de Papanicolaou. O CEA utiliza o modelo YOLOv5s para detecção e a família de redes EfficientNets para classificação e uma interface simples para permitir a interação do usuário.

Em relação ao modelo utilizado, o YOLOv5s, possui uma versão leve da arquitetura YOLO. Embora ainda carregue o nome da família original, o YOLOv5 não foi desenvolvido pela equipe responsável pelas versões iniciais do algoritmo, mas sim pela Ultralytics (2020). Apesar disso, o modelo trouxe melhorias significativas em desempenho, praticidade de uso e velocidade de inferência, o que o tornou amplamente adotado em aplicações práticas.

Quanto ao resultados, o CEA se mostrou promissor, com precisão de 97,5% e sensibilidade

de 99,2%, sendo ambos elevados, demonstrando sua capacidade de identificar e classificar células cervicais de maneira eficaz e eficiente. Além disso, obteve *F1-Score* de 98,3%.

A pesquisa destaca as dificuldades da análise tradicional dos esfregaços de Papanicolaou sofridas pelos especialistas, como fadiga e alta carga de trabalho, salientando a importância de se explorar tecnologias para apoio aos especialistas. A ferramenta CEA foi testada em uma base de dados de citologia convencional, comum em países subdesenvolvidos devido ao seu menor custo. A avaliação pelos citopatologistas revelou o potencial do CEA em servir como uma valiosa ferramenta de apoio, fornecendo suporte ao profissional e melhorando a padronização dos diagnósticos.

O estudo conclui que o CEA pode significativamente melhorar a qualidade dos diagnósticos de câncer cervical, oferecendo suporte aos citopatologistas na detecção e classificação de células pré-neoplásicas. A integração da tecnologia em rotinas de análise pode não apenas reduzir os falsos-positivos e falso-negativos, mas também aumentar a precisão e eficiência dos exames, contribuindo para um diagnóstico mais confiável e precoce do câncer de colo de útero.

Sobre os classificadores analisados, podemos perceber o quanto eles podem auxiliar os profissionais durante a construção do diagnóstico, com base na precisão e sensibilidade dessas ferramentas em detectar e classificar. Além disso, os estudos também afirmam que o desenvolvimento de novas tecnologias como as apresentadas é uma excelente maneira de permitir o acesso a métodos com menos falhas no diagnósticos e com custos baixos, se tornando acessível para os países que mais sofrem com a doença.

Apesar disso, é necessário um método eficiente de utilização dessas ferramentas, visto que não é todos os laboratórios que tem equipamentos robustos para executar todos esses métodos, ou são todos os profissionais que terão acesso a treinamento para usá-los.

2.2.2 Integração de algoritmos de detecção e classificação em dispositivos móveis

Apesar da grande quantidade de algoritmos de detecção e classificação de imagens usando redes CNNs e CNNs profundas, ainda temos alguns desafios quando o sistema a ser utilizado é um dispositivo móvel, logo, analisamos trabalhos que trataram esse problema.

Iftikhar et al. (2024) apresentam um trabalho sobre a utilização CNNs para auxiliar no controle de doenças no cultivo agrícola. O foco do trabalho é a detecção precoce e classificação de doenças, em culturas como maçã, milho e batata, por meio da análise das folhas das culturas. Apesar de não se tratar da classificação de doenças em seres humanos um dos pontos importantes do artigo é a integração do modelo proposto com um aplicativo móvel, permitindo praticidade na utilização da ferramenta o que resulta no controle rápido de pragas.

O aplicativo proposto, foi desenvolvido somente para dispositivos Android, utilizando o

TensorFlow, biblioteca que permite a criação de modelos de ML. Dessa forma, os agricultores podem fotografar as folhas das plantas e submetê-las ao aplicativo, que irá processar as imagens e fornecer um diagnóstico, independentemente de onde estiver. Ainda sobre o carregamento de imagens, o aplicativo também permite o carregamento de uma imagem já salva no dispositivo para a classificação. Além da identificação da doença, o aplicativo fornece informações que auxilia o produtor rural na tomada de decisão.

O desenvolvimento do aplicativo móvel visa a criação de uma ferramenta de detecção de doenças mais acessível, com o intuito de diminuir a infestação de doenças e melhorar a produção dos trabalhadores. Nesse caso, ela é destinada aos produtores em áreas remotas, e se aplica àqueles que não tem possibilidade de acesso a recursos que melhorem sua produção, devido a custos e outros fatores.

Iftikhar et al. (2024) também abordam os desafios e as limitações na implementação do sistema, como variáveis que podem impactar na detecção da doença, como a qualidade da imagem ou questões ambientais. Além disso, também cita a necessidade de otimizar o modelo proposto, permitindo que ele opere em dispositivos com menor processamento.

Os autores concluem que a capacidade de capturar e processar imagens em tempo real diretamente do campo é um avanço significativo na gestão de doenças agrícolas, que ajuda a reduzir perdas de produção e a melhorar a eficiência das práticas agrícolas.

Nesse sentido, ao compararmos com o proposto nesse trabalho percebemos a importância do trabalho de Iftikhar et al. (2024) que obteve resultados positivos em ambientes com mais limitações dos que os que são esperados do público de utilização do CitoFocus, visto que são profissionais que atuam em laboratórios. Além disso, podemos correlacionar também visto que o aplicativo proposto além de identificar a doença também fornece informações que auxilia na tomada de decisão, um cenário bem parecido com o CitoFocus, que permite a comunicação entre profissionais objetivando a troca de informações para auxiliar na tomada de decisão na análise do caso.

Já Monsur et al. (2017) tratam sobre o desenvolvimento de um sistema para dispositivos móveis para detecção de câncer de colo de útero. O aplicativo permite o carregamento de imagens já salvas no dispositivo, ou pode ocorrer a captura das imagens no momento da sua execução e logo em seguida as imagens selecionadas são enviadas ao classificador, a Figura 2.4 mostra o fluxo de funcionamento do aplicativo. O trabalho discute a importância de métodos acessíveis para o diagnóstico precoce dessa doença, principalmente em regiões com menores recursos.

O foco principal do estudo está na criação de um aplicativo Android que possui algoritmos de processamento de imagens, como filtragem, segmentação e detecção de bordas, para que seja feita a análise das imagens carregadas por meio do dispositivo móvel, permitindo que os profissionais utilizem a ferramenta em diferentes localidades e situações.

O processo de análise inclui a remoção de ruídos, conversão para tons de cinza e segmen-

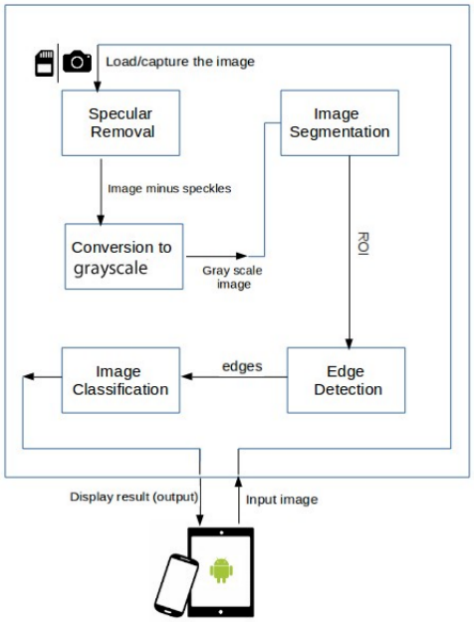


Figura 2.4 – Imagem que representa o fluxograma de Funcionamento. Retirado de Monsur et al. (2017)

tação da região de interesse. Após essas fases, o algoritmo de detecção é aplicado para identificar possíveis anormalidades que podem indicar a presença de lesões causadoras da doença ou ela mesma em casos mais avançados. A última etapa do processo, após a identificação de locais de interesse, é aplicar o classificador para que seja catalogado e gerado a resposta para análise do usuário.

Os resultados obtidos com o sistema foram promissores, apresentando uma especificidade de 79% e uma sensibilidade de 83%, valores que indicam uma boa capacidade de detecção do sistema, principalmente se levarmos em conta que foi desenvolvido para dispositivos com poucos recursos. O estudo também discute as vantagens de usar a plataforma *Android* principalmente devido à sua ampla base de usuários, o que facilita a distribuição do aplicativo.

O artigo conclui que o uso de dispositivos móveis para análise de imagens médicas tem um grande potencial para melhorar os programas de triagem de câncer cervical e sugere que futuros trabalhos explorem a expansão do sistema para outras plataformas e o uso de bases de dados mais amplas para aprimorar os modelos de análise e classificação, demonstrando o quão importante é uma boa base de dados para a construção de uma ferramenta de qualidade para apoio aos especialistas.

Buiu, Dănăilă e Răduţă (2020) discutem o desenvolvimento de um sistema para análise e classificação de imagens de exames de colposcopia – exame que permite a visualização do colo de útero - com o intuito de identificar lesões precursoras do câncer de colo de útero, ou a própria doença. O trabalho foca em utilizar redes convolucionais, especificamente o modelo MobileNetV2, arquitetura de rede projetada para ser eficiente em dispositivos móveis e embarcados.

O trabalho inova, quando comparado aos estudos anteriores a sua publicação, ao criar

um aplicativo móvel voltado para áreas de baixa renda, onde o acesso a equipamentos médicos avançados não é comum. A ferramenta desenvolvida permite uma análise preliminar de imagens digitais do colo do útero com o intuito de identificar lesões precursoras do câncer.

Os resultados mostram que o uso do modelo MobileNetV2, especialmente em um conjunto de redes, pode alcançar uma alta precisão na classificação de lesões, sugerindo que essa tecnologia tem potencial no apoio a diagnósticos rápidos e precisos. Além disso, o foco na utilização de um aplicativo móvel destaca a importância da ferramenta em cenários do mundo real, visto que atenderá comunidades onde recursos mais avançados é de difícil acesso.

Os autores consideraram os resultados bastante promissores ao aplicar o conjunto de rede MobileNetV2 para a análise. O método implementado alcançou uma precisão de 83,33% na classificação de quatro classes (normal; lesão de baixo grau; lesão de alto grau; câncer) e 91,66% na classificação binária (normal e lesão/câncer), demonstrando potencial para se tornar uma ferramenta de apoio aos especialistas. Além disso, o estudo apresentou soluções para evitar o *overfitting* e superou desafios como o desbalanceamento do *dataset*. Por fim, eles afirmam a necessidade de melhorias antes de sua aplicação em ambientes clínicos.

Referente aos trabalhos citados acima, temos a proposta de aplicativos que podem ser utilizados em diferentes áreas, característica essa semelhante ao que é tratado no nosso trabalho. Com exceção do Iftikhar et al. (2024), Monsur et al. (2017) e Buiu, Dănăilă e Răduţă (2020) tratam da detecção de doenças em humanos e mostram bons resultados e possibilidade de auxiliar os profissionais na construção do diagnóstico. Apesar disso, quando comparamos aos modelos da seção 2.2.1 eles possuem espaço para melhorias em suas precisões e sensibilidades. Essa diferença na precisão e sensibilidade pode ser atribuída principalmente ao fato das tecnologias de classificação utilizadas possuírem limitações, principalmente a usada por Buiu, Dănăilă e Răduţă (2020), que utilizam um classificador embarcado.

3 Desenvolvimento

Neste Capítulo será descrito o que foi desenvolvido, relatando as decisões tomadas e suas explicações. Conforme comentado anteriormente, o CitoFocus é um projeto cujo objetivo é permitir a colaboração remota entre citopatologistas durante a construção de laudos citopatológicos, porém, para o objetivo desse trabalho modificações foram necessárias no *back-end* para que ele atendesse aos requisitos estabelecidos para o aplicativo e permitisse a implantação do classificador no *back-end*.

3.1 Arquitetura do *back-end*

Para o desenvolvimento do novo *back-end*, foi escolhido o *framework NestJS*, uma vez que ele oferece uma estrutura robusta baseada em *TypeScript*, favorecendo a escalabilidade, reutilização de código e aplicação de boas práticas como injeção de dependência e modularização. Tais características foram determinantes para a adoção do *NestJS*, especialmente por sua aderência à arquitetura proposta para este sistema.

Como foi detalhado na Seção 2.1.4, o *NestJS*S permite que a lógica de negócio permaneça isolada de implementações externas, facilitando futuras alterações em serviços ou tecnologias de suporte — como banco de dados ou sistemas de autenticação — com mínimas mudanças no código principal. Por exemplo, a substituição do banco de dados exige apenas a troca de configurações específicas na camada de serviço e ajustes nos repositórios das entidades.

A arquitetura adotada no *back-end* foi baseada na arquitetura hexagonal, que preconiza a separação clara entre o núcleo da aplicação e suas interfaces externas, promovendo testabilidade, manutenção e flexibilidade. Além disso, foram aplicados alguns princípios do padrão MVC (Model-View-Controller) para organizar a estrutura interna dos módulos.

Essa abordagem arquitetural, combinada com os recursos do *NestJS*, assegura que o núcleo da aplicação — a lógica de negócios — seja mantido inalterado, mesmo com a eventual substituição de tecnologias externas, o que contribui significativamente para a longevidade e adaptabilidade do sistema.

Quanto ao segundo modelo, o MVC, utilizamos principalmente os termos e ideias, como *controller*, responsável por organizar todas as rotas - como o *back-end* se comunica com o *front-end* - e em outras nomeações de arquivos e diretórios, para melhor adequação aos cenários de implementação.

À vista disso, para a arquitetura hexagonal temos as interfaces que definem como a aplicação interage com o *front-end*, as regras de interação (portas), como podemos ver na Figura 3.1, e as implementações que permitem essa interação (adaptadores), no caso da aplicação são o

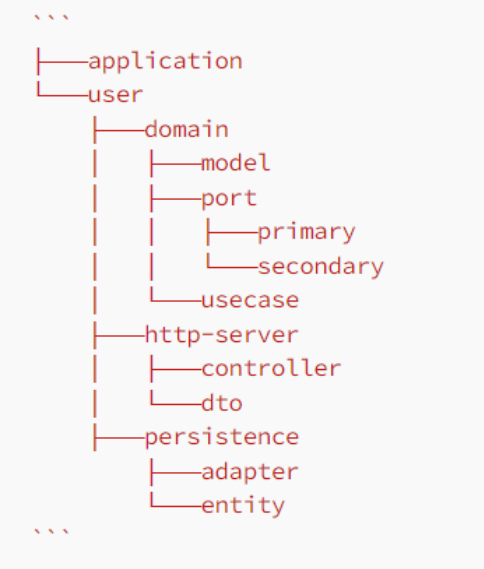


Figura 3.1 – Exemplo de organização de um projeto *NestJs* com arquitetura hexagonal. Retirado de Rerick (2023)

service e o *controller*, respectivamente.

Segundo essa mesma arquitetura, temos a implementação que permite a comunicação com o banco de dados, a qual nós chamamos de *repository*. É importante frisar que, a arquitetura hexagonal organiza sua estrutura utilizando o conceito de núcleo da aplicação, portas e adaptadores.

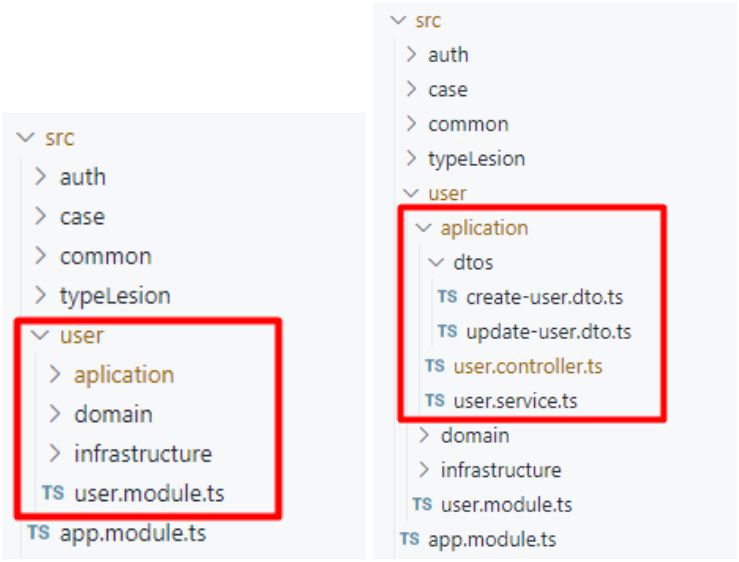
Em contrapartida, nos temos respectivamente o *domain* que é o diretório que armazena regras de negócio, que não mudam independentemente das tecnologias externas que o serviço se relaciona. O *service* que é o arquivo que define como as rotas irão interagir com o *front-end* do aplicativo e o *controller* arquivo que define as rotas que serão utilizadas pelo *front-end* para busca e envio de informações.

Nossa implementação, quanto a organização segue a estrutura estabelecida pelo modelo definido acima. Como exemplo temos a entidade *user* apresentada na Figura 3.2a.

O objeto *user* é a entidade onde é definido os usuários, suas regras e atributos. Na Figura 3.3 temos a definição de todos os atributos de usuário, e se eles são obrigatórios ou não (a presença do "?" indica a não obrigatoriedade). É por meio dessa entidade que é possível cadastrar, editar, mudar o tipo de conta e realizar a busca de informações relativas a ele, devido a tudo que definimos nela.

O diretório *application* define toda a regra de comunicação com o *front-end* da aplicação, sendo assim ali temos o *service*, o *controller* e o DTO para a entidade usuário, a Figura 3.2b destaca esses componentes.

Este último define, quais dados são transferidos entre os componentes da aplicação (*front-end* e *back-end*), o tipo do dado, e se a informação é obrigatória de ser enviada e a validação



(a) Figura exibindo a organização do código com destaque para a entidade usuário (região delimitada em vermelho).

(b) Figura exibindo a organização do código com foco no diretório *application*(região delimitada em vermelho) da entidade usuário.

Figura 3.2 – Figuras exibindo a organização do código destacando a organização de diferentes componentes da entidade usuário após aplicação da arquitetura proposta.

da transferência (se os dados estão sendo transferidos), como pode ser visto na Figura 3.4. Esse modelo de projeto auxilia em diversos cenários, pois permite criar uma classe com seus atributos definidos, facilitando a construção de um código limpo e coeso. Pois sempre lidamos com essa classe ao trabalhar com as rotas no *back-end*, ao invés de estabelecer a cada definição de rotas os dados que são transferidos nela, como pode ser visto na Figura 3.4. A Figura 3.5 é um exemplo da utilização do DTO, e exemplifica o que foi citado acima, principalmente na questão de um código limpo e de fácil entendimento.

A partir da Figura 3.2b podemos ver a existência de dois DTOs. Eles foram implementados pois o cenário de uso são diferentes, um define todos os atributos, as informações que serão transferidas para a criação de usuário e o outro é utilizada para definir as informações para a atualização dos dados de usuário.

Ainda na Figura 3.2a temos o diretório referente ao *domain* que define a regra de negócio da entidade. Essa regra define como o objeto deve ser tratado e não é impactada pelas diferentes implementações de outros componentes, bancos de dados ou *frameworks* do *front-end*. Nela definimos todos os atributos da entidade e algumas regras que devem ser seguidas.

Um exemplo é a regra para usuário, caso seja o cadastro de um novo usuário ele é cadastrado como visitante, e somente quando ele comprovar sua especialização na área de citopatologia que ele passa a ter uma conta do tipo citopatologista. Essa regra permite a definição das permissões de cada usuário da plataforma, sendo controlado com essa definição de tipo de

```
export class User {
  id: string;
  type: UserType;
  name: string;
  email: string;
  password: string;
  linkedInLink?: string;
  lattesLink?: string;
  //Fields for Citopatologists
  profession?: string;
  specializationCertificate?: string;
  conclusionDate?: Date;
  hasExperience?: boolean;
  experienceCertificate?: string;

  private constructor(params: InstanceUserParams) {
    const {
      id,
      type,
      name,
      email,
      password,
      linkedInLink,
      lattesLink,
      profession,
      specializationCertificate,
      conclusionDate,
      hasExperience,
      experienceCertificate,
    } = params;

    this.id = id;
    this.type = type;
    this.name = name;
    this.email = email;
    this.password = password;
    this.linkedInLink = linkedInLink;
    this.lattesLink = lattesLink;
    this.profession = profession;
    this.specializationCertificate = specializationCertificate;
    this.conclusionDate = conclusionDate;
    this.hasExperience = hasExperience;
    this.experienceCertificate = experienceCertificate;
  }
```

Figura 3.3 – Trecho da definição da classe usuário para a entidade usuário e do construtor desse objeto. Podemos visualizar todos os atributos definidos para o objeto usuário, sendo definido os atributos obrigatórios e os opcionais (presença do "?").

usuário como cada um pode interagir na plataforma.

É dentro do *domain* que as lógicas de negócios relacionadas a entidade são definidas. É ele que determina o correto funcionamento e permite que os dados sejam salvos no banco ou sejam enviados para o *back-end*.

Além dessa, temos o diretório *infrastructure*, que é onde definimos as regras de interação com o banco de dados. Dentro desse último diretório citado, temos a definição do *repository*, Figura 3.6, que é semelhante as rotas, somente que se comunicam com o local em que armazena-se os dados da aplicação, o banco de dados. Caso seja decidido a mudança do banco, as alterações serão realizadas no arquivo da Figura 3.6 para cada entidade, e no arquivo *module* do aplicativo, Figura 3.7.

Em relação ao *module*, cada entidade possui o seu, que é responsável por organizar e encapsular todas as regras citadas acima, facilitando a organização e reutilização de código, pois cada entidade e suas regras são definidas e vinculadas a um *module*, e quando necessário

```
export class CreateUserDto {

  @IsString()
  @IsOptional()
  type: string;

  @IsString()
  @IsNotEmpty()
  name: string;

  @IsEmail()
  @IsNotEmpty()
  email: string;

  @IsString()
  @IsNotEmpty()
  password: string;

  @IsString()
  @IsOptional()
  profilePicture?: string;

  @IsString()
  @IsOptional()
  linkedInLink?: string;
```

```
  @IsString()
  @IsOptional()
  lattesLink?: string;

  @IsString()
  @IsOptional()
  profession?: string;

  @IsString()
  @IsOptional()
  specializationCertificate?: string;

  @IsDate()
  @IsOptional()
  conclusionDate?: Date;

  @IsBoolean()
  @IsOptional()
  hasExperience?: boolean;

  @IsString()
  @IsOptional()
  experienceCertificate?: string;
}
```

(a) (b)

Figura 3.4 – Exemplo de implementação de DTO de criação de usuário. Nele podemos ver todos os dados para a entidade usuário, sendo os obrigatórios com a *tag "@IsNotEmpty()"* e os opcionais com a *tag "@Optional()"*.

```
@Post()
@UsePipes(ValidationPipe)
create(@Body() createUserDto: CreateUserDto) {
  return this.userService.create(createUserDto);
}
```

Figura 3.5 – Exemplo de uma rota para usuário, sendo essa a rota para a criação de usuário.

precisamos somente acessar essas definições ao invés de duplicar o código.

Isso também auxilia na modularização do sistema, pois podemos alterar uma entidade adicionando novas regras e funções sem a necessidade de alterar outra entidade. Por fim, o aplicativo possui o seu próprio *module*, que como os de cada entidade, possui as mesmas responsabilidades - nesse caso ele também encapsula todas as entidades do serviço - e a de configurar a interação com o banco de dados.

É interessante citar que, com a separação em entidades e a criação do *module* organizando as lógicas para cada entidade, o *module* do aplicativo torna-se significativamente mais organizado e claro e o *back-end* mais modularizado devido a essa separação de funções.

Um exemplo disso é que, caso seja necessária uma alteração em alguma das entidades, podemos mexer na entidade específica e na regra específica que deve ser alterada (comunicação com o banco, lógica do serviço ou rotas para o *front-end*) sem a necessidade de alterar as outras entidades. Em outro cenário, caso seja necessário retirar uma entidade *back-end*, o código não sofre impacto, pois é preciso somente retirá-la do *module* da aplicação. As entidades restantes

```
export class MongoUserRepository implements UserRepository {
  constructor(@InjectModel('User') private readonly userModel: Model<User>) {}

  async create(user: User): Promise<User> {
    return await this.userModel.create(user);
  }
  async findAll(page: number, limit: number): Promise<User[]> {
    return await this.userModel
      .find({})
      .limit(limit)
      .skip((page - 1) * limit);
  }

  async findOneId(id: string): Promise<User> {
    return await this.userModel.findOne({ id: id });
  }

  async findOneEmail(email: string): Promise<User> {
    return await this.userModel.findOne({ email: email });
  }

  async update(id: string, updateUserParams: UpdateUserParams): Promise<void> {
    await this.userModel.updateOne(
      { id },
      {
        ...updateUserParams,
      },
    );
    return;
  }
  async remove(id: string): Promise<void> {
    await this.userModel.remove({ id: id });
    return;
  }
}
```

Figura 3.6 – Exemplo de implementação da interação com o banco de dados. Essa é a implementação para a entidade usuário.

```
@Module({
  imports: [
    ConfigModule.forRoot(),
    MongooseModule.forRoot(`mongodb+srv://${process.env.MONGO_INITDB_ROOT_USERNAME}:
      ${process.env.MONGO_INITDB_ROOT_PASSWORD}
      @citofocus.vwxffom.mongodb.net/?retryWrites=true&w=majority`,
    ),
    AuthModule,
    UserModule,
    CaseModule,
    TypeLesionModule,
  ],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

Figura 3.7 – Arquivo *module* do *back-end* após aplicação da arquitetura proposta. Sendo possível visualizar a definição da conexão com o banco de dados (região delimitadda em vermelho) e logo abaixo as entidades que o aplicativo está consumindo.

somente sofrem alteração nessa situação caso estejam utilizando essa entidade.

3.2 Integração do Classificador

Devido à modularização do sistema, para integrar o modelo treinado do *YOLOv5* ao código existente, as modificações se limitaram somente a entidade *Case*. Essa entidade é onde temos a definição do caso, suas regras e atributos, Figura 3.8. Por meio dessa entidade que realizamos o gerenciamento do caso, sua criação, edição e manipulação dos dados, como computação dos

votos.

```
21  private constructor(params: CreateCaseParams) {
22    const {
23      id,
24      images,
25      options,
26      authorId,
27      authorName,
28      description,
29      age,
30      dateLimit,
31      vote,
32      comments,
33      results,
34      imagesYolo,
35      descYolo,
36    } = params;
37
38    this.id = id;
39    this.images = images;
40    this.options = options;
41    this.authorId = authorId;
42    this.authorName = authorName;
43    this.description = description;
44    this.age = age;
45    this.dateLimit = dateLimit;
46    this.vote = vote;
47    this.comments = comments;
48    this.results = results;
49    this.imagesYolo = imagesYolo;
50    this.descYolo = descYolo;
51  }
```

Figura 3.8 – Trecho da definição da classe *case*, para a entidade usuário.

As principais modificações e implementações ocorreram no *service* da entidade, garantindo que as imagens enviadas fossem classificadas automaticamente pelo modelo antes de serem armazenadas. As outras modificações se limitaram a adicionar nas definições do objeto caso as variáveis *imagesYolo*, linha 34 e 49 da Figura 3.8, que armazena o id das imagens processadas pelo classificador, e *descYolo*, linha 35 e 50 da Figura 3.8, armazena o texto de saída do classificador, pode ser visto nas Figuras 3.9 e 3.10, que informa quantos lesões foram identificadas como positivas e negativas, Figura 3.11,

```
id : "91238849-533c-4ab2-935f-6c32beff2859"
▾ images : Array (1)
    0: "19u4eBtz3gdaJguyzjJwwNJUMzU4xnrF5"
▸ options : Array (4)
  authorId : "a2389975-aab7-4425-9bd6-51f32f8dacb6"
  authorName : "Danilo Demonstração2"
  description : "91"
  age : 91
  dateLimit : 2025-03-29T00:55:00.000+00:00
▸ vote : Array (empty)
▸ comments : Array (empty)
▸ results : Array (4)
▾ imagesYolo : Array (1)
    0: "1GGe8lw4SGwpY0QtJx4z8jFAPMgpERNIJ"
  descYolo : "13 Lesioneds, 15 Negatives"
```

Figura 3.9 – Figura tirada do banco de dados mostrando o armazenamento de um caso com uma imagem fornecida e processada e um texto de saída do classificador.

No *service* foi criada a função *sendToClassifier*, responsável por processar e classificar as imagens enviadas. O fluxo dessa função começa com o salvamento da imagem recebida, que é

```
id : "a10443a4-d0be-4fc0-85f4-575dd688ad85"
▾ images : Array (3)
    0: "1w2wsoaWemTrV-QrHs1eJbDvGRHY1K4lH"
    1: "1FwidE5bC6Y7BpNY4xCrj4MxT0yuVugop"
    2: "1sKdr45ll8r739IYq3drvnNUH7-55ewuI"
▸ options : Array (5)
  authorId : "a2389975-aab7-4425-9bd6-51f32f8dacb6"
  authorName : "Danilo Demonstração2"
  description : "939495"
  age : 93
  dateLimit : 2025-03-27T01:00:00.000+00:00
▸ vote : Array (empty)
▸ comments : Array (empty)
▸ results : Array (5)
▾ imagesYolo : Array (3)
    0: "1OXYTkeCdxbmyxkAiPJZHIlUWZ0njS4B4"
    1: "1ua-JSWncREO3eAPD5_dm_nJKZFAPRVxD"
    2: "1hbIXNyLXjftxn1QCanLPRqHWxk_5aN3i"
  descYolo : "13 Lesioneds, 15 Negatives, 3 Lesioneds, 34 Negatives, 65 Lesioneds, 9…"
```


Figura 3.10 – Figura tirada do banco de dados mostrando o armazenamento de um caso com várias imagens fornecidas e processadas e vários textos de saída do classificador.

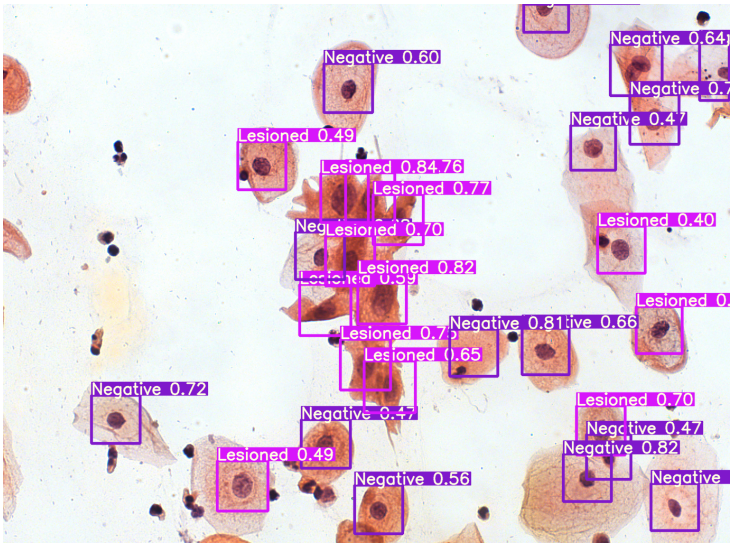


Figura 3.11 – Imagem de amostra gerada após o processamento pelo classificador.

armazenada temporariamente em um diretório local, diretório *uploads*, com um nome formatado que identifica a imagem. O nome para salvar a imagem é uma associação entre o *ID* do caso e o número da imagem naquele caso. Em seguida, foram definidos os caminhos necessários para o funcionamento do *YOLOv5*, incluindo o *script* de detecção (*detect.py*), os pesos do modelo (nomeado como *best.pt*), a imagem a ser processada e o diretório onde os resultados seriam armazenados, a Listagem 3.1 mostra o exemplo de um comando gerado, na mesma ordem descrita anteriormente.

Listagem 3.1 – as

```
python "H:\yolov5\detect.py" --weights "H:\yolov5\weights\best.pt"
   ↪ --name H:\cito_focus_api\dist\10a0a402-e27a-43a4-a475-
   ↪ dcebf4970fa4_0 --img-size 1376 --source H:\cito_focus_api\
   ↪ dist\uploads\10a0a402-e27a-43a4-a475-
   ↪ dcebf4970fa4_0_classificada.jpg --agnostic-nms --save-txt
```

Para a execução do modelo, utilizou-se a função *exec*, importada do módulo *childprocess*

do Node.js Documentation (2025). Essa função permite a execução de comandos do sistema operacional diretamente a partir do código *TypeScript*, possibilitando a automação de tarefas externas ao ambiente do *Node.js*. No contexto da integração, *exec* foi usada para chamar o interpretador Python e executar o *script detect.py*, responsável por analisar as imagens, detectar e classificar as regiões que possuem alguma lesão. O comando construído inclui diversos parâmetros, como o caminho para os pesos do modelo (*best.pt*), o tamanho das imagens processadas e a opção que salva os resultados em um arquivo de texto, conforme ilustrado no bloco 3.1.

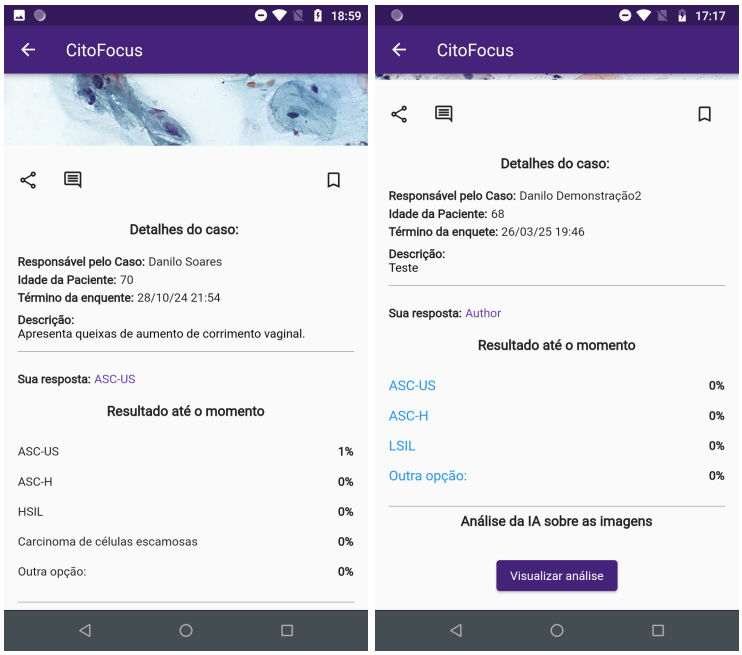
Esse processo garante a execução correta do código em Python, permitindo que a classificação das imagens ocorra automaticamente. Além disso, a saída gerada pelo *exec* é capturada e analisada, extraindo informações relevantes, como a presença de lesões ou a confirmação de imagens negativas. Essa abordagem permite que a função *sendToClassifier* interprete os resultados do modelo de maneira estruturada, extraindo dados úteis para o fluxo de trabalho do sistema. Após a extração da classificação, a imagem original temporária é removida, e após o armazenamento da versão processada, a pasta de processamento dessa última imagem também é removida.

Com essa implementação, a função *sendToClassifier* foi integrada à função *create*, que já existia e é responsável por criar novos casos dentro do sistema. Agora, sempre que um novo caso é criado, o sistema percorre todas as imagens enviadas, chamando a função *sendToClassifier* para processar e classificar cada uma delas. Os resultados obtidos são então coletados e, juntamente com as imagens originais enviadas pelo usuário, suas versões processadas e os respectivos IDs, armazenados no banco de dados e vinculados ao caso correspondente. Com essa automação, o sistema passou a identificar possíveis lesões de maneira eficiente e organizar os resultados para facilitar futuras análises e acessos aos dados.

Além dos aspectos técnicos da integração do classificador, é fundamental destacar as questões relacionadas à segurança, privacidade e ética envolvidas no uso de imagens médicas e inteligência artificial. As imagens utilizadas são anonimizadas desde o início do processamento, assegurando a proteção de informações sensíveis dos pacientes. Todo o fluxo de dados é realizado em ambiente controlado, com uso de diretórios temporários e exclusão automática dos arquivos após o processamento, evitando o armazenamento indevido e não necessário de dados, mantendo somente aqueles necessários para análise e interação dos profissionais. Além disso, os resultados produzidos pelo modelo de detecção não substituem o julgamento clínico dos especialistas, sendo apresentados como um recurso complementar para apoio à decisão. Essa estratégia visa garantir não apenas a eficiência técnica, mas também o respeito aos princípios éticos que regem a prática médica, como a confidencialidade, a responsabilidade profissional e a equidade no acesso à informação.

3.3 Exibição dos Resultados da Classificação de Imagens

Com o objetivo de aprimorar a usabilidade, a organização das informações dentro do aplicativo e permitir a visualização das novas informações, foram implementadas modificações na estrutura da interface. Inicialmente, a aplicação contava com uma única tela para visualização das informações do caso, Figura 3.12a, que apresentava todas as informações e funcionalidades disponíveis.



(a) Antiga tela de visualização do caso com voto do usuário.

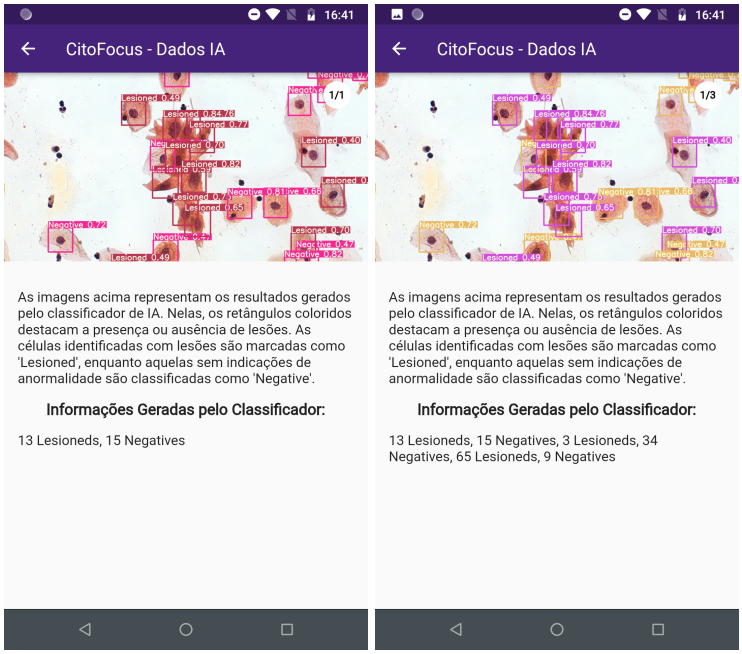
(b) Atual tela de visualização do caso com voto do usuário.

Figura 3.12 – Telas de resultado do caso, sendo a 3.12a a tela antiga e 3.12b a nova tela.

Na versão revisada da interface, Figura 3.12b, foi introduzida uma nova seção contendo um botão de navegação. Essa adição possibilita um acesso mais intuitivo e estruturado às funcionalidades do sistema, melhorando a experiência do usuário. O botão recém-incorporado conduz o usuário a uma nova tela, que podem ser vistas nas Figura 3.13a ou 3.13b, que foi projetada para exibir informações complementares de maneira mais organizada e acessível.

Essa reformulação tem como principal objetivo otimizar o fluxo de navegação dentro da aplicação, garantindo que o usuário possa acessar os recursos de forma mais eficiente. Além disso, a separação das funcionalidades em telas distintas foi pensada para manter a experiência original, evitando alterações abruptas no fluxo já estabelecido. Dessa forma, a interface permanece intuitiva e familiar para os usuários, ao mesmo tempo em que reduz a sobrecarga de informações na tela principal, tornando a interação com o sistema mais fluida e organizada.

Além disso, a nova seção de navegação, que permite acessar a próxima tela, não é visível quando o usuário não votou no caso, (Figuras 3.14a e 3.14a). Ela se torna visível apenas após o usuário registrar seu voto (Figuras 3.15a e 3.15a). Para o autor do caso, essa seção aparece assim



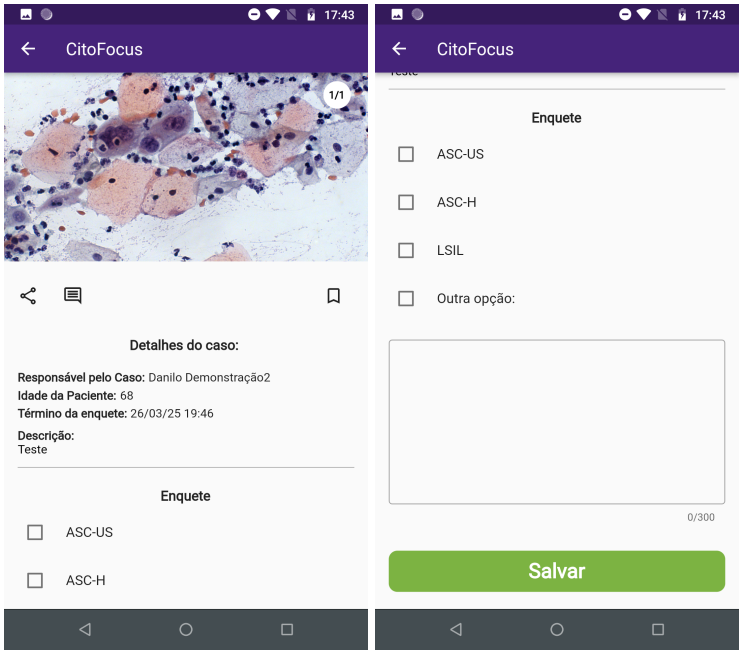
(a) Tela de visualização dos dados gerados pelo classificador contendo uma imagem.

(b) Tela de visualização dos dados gerados pelo classificador contendo três imagens.

Figura 3.13 – As telas são dois comportamentos diferentes da tela de visualização dos dados gerados pelo classificador, contendo somente uma imagem e um texto de saída e contendo várias imagens e vários textos de saída.

que ele salva o caso e o classificador finaliza o processamento das imagens.

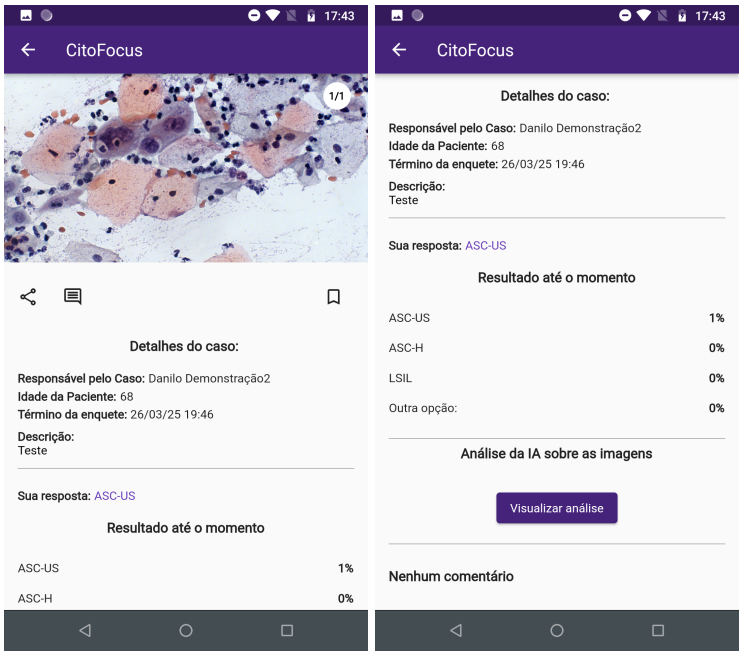
Inicialmente, a ideia era integrar o classificador como um voto adicional, mas optamos por disponibilizar os dados gerados pelo classificador a todos os usuários, garantindo que todos tenham acesso às mesmas informações. Isso permite que as discussões ocorram de forma equilibrada, sem que um usuário possua informações exclusivas que outros não têm, enriquecendo a análise coletiva e favorecendo a troca de conhecimentos entre os citopatologistas.



(a) Tela de visualização do caso antes do usuário votar.

(b) Tela de visualização do caso antes do usuário votar.

Figura 3.14 – Tela de visualização do caso antes de o usuário votar. Podemos visualizar os dados fornecidos pelo autor, o espaço para votação e espaço onde é possível inserir a justificativa.



(a) Tela de visualização do caso após o usuário votar.

(b) Tela de visualização do caso após o usuário votar.

Figura 3.15 – Tela de visualização do caso após o usuário votar. Podemos visualizar os dados fornecidos pelo autor, a computação dos votos até o momento, o botão para navegação para a página de visualização dos dados gerados pelo classificador e espaço onde é possível visualizar as justificativas.

4 Resultados

Neste Capítulo será apresentado, interpretado e analisado os resultados alcançados por esse trabalho.

Como citado no Capítulo 3, além das modificações no *back-end* foi necessária uma pequena modificação no *front-end* da aplicação para permitir a visualização dos dados do classificador.

Durante a implementação do *back-end* foi escolhido o modelo de projeto em que utiliza o DTO. Como resultado da escolha da implementação dessa classe, o DTO, temos o código das rotas, *controller* e também as regras dessas rotas, o *service*, mais organizado e modular.

Adotamos a abordagem de implementar dois tipos diferentes de DTOs, um para cadastro ou criação de alguma entidade e outro para atualização. Isso tornou o código ainda mais limpo e de fácil entendimento, pois os atributos que são transferidos e sua obrigatoriedade são definidos dentro dessa classe, e não ao longo da declaração das rotas onde esses objetos são utilizados.

A Figura 4.1 mostra duas rotas implementadas para usuário e a Figura 4.2 mostra a implementação da rota de criação de usuário, o *service*. Esse é o padrão que foi seguido para a implementação de todas as rotas do *back-end*.

```
@Post()
@UsePipes(ValidationPipe)
create(@Body() createUserDto: CreateUserDto) {
  return this.userService.create(createUserDto);
}

@Get()
@UsePipes(ValidationPipe)
findAll(@Query('page') page: number, @Query('limit') limit: number) {
  return this.userService.findAll(page, limit);
}
```

Figura 4.1 – Exemplo de rotas para entidade usuário. A primeira é a rota para criação de usuário e a segunda a rota para buscar a lista de usuários.

A primeira rota apresentada é a de criação de usuário. Esse exemplo e o mostrado na Figura 4.2 é o resultado da utilização do DTO. Esses objetos permitiriam a criação de um código organizado e de fácil entendimento, o que facilitou na modularização, escalabilidade e correções exigidas, como pode ser visto nessas Figuras.

A modularização do código é reforçada pela tecnologia utilizada na implementação do *back-end*, o *NestJs*, mencionada no Capítulo 3. Isso se deve à sua estrutura baseada na separação de entidades, que posteriormente são vinculadas em um único arquivo, o *module* do *back-end*.

A Figura 4.3 exemplifica o *module* do *back-end*, onde temos a chamada de todas as entidades, nesse caso são quatro entidades que estão dentro da região delimitada vermelho. Além

```
async create(createUserDto: CreateUserDto): Promise<User> {
  const {
    name,
    email,
    password,
    linkedInLink = "",
    lattesLink = "",
    profession = "",
    specializationCertificate = "",
    conclusionDate = null,
    hasExperience = null,
    experienceCertificate = "",
  } = createUserDto;

  const type = UserType.Visitor;

  const user = User.create({
    id : uuidv4(),
    type,
    name,
    email,
    password: bcrypt.hashSync(password, 10),
    linkedInLink,
    lattesLink,
    profession,
    specializationCertificate,
    conclusionDate,
    hasExperience,
    experienceCertificate,
  });
```

Figura 4.2 – Exemplo de implementação de rotas, sendo a implementação da rota de criação de usuário. Podemos visualização o quanto o uso do DTO torna o código mais limpo ao definir as funções.

disso, essa figura também mostra outro detalhe citado em relação a modularização, que é a facilidade em se trocar os serviços externos, pois é nesse arquivo a definição do banco de dados a ser utilizado. Para trocar esse serviço trocamos o código que está dentro da região delimitada em azul e a regra do *infrastructure* na Figura 3.2a.

```
@Module({
  imports: [
    ConfigModule.forRoot(),
    MongooseModule.forRoot(`mongodb+srv://${process.env.MONGO_INITDB_ROOT_USERNAME}
    :${process.env.MONGO_INITDB_ROOT_PASSWORD}
    @citofocus.vwxffom.mongodb.net/?retryWrites=true&w=majority`,
    ),
    AuthModule,
    UserModule,
    CaseModule,
    TypeLesionModule,
  ],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

Figura 4.3 – Arquivo *module* do *back-end*. Sendo possível visualizar a definição da conexão com o banco de dados e as entidades que o aplicativo está consumindo.

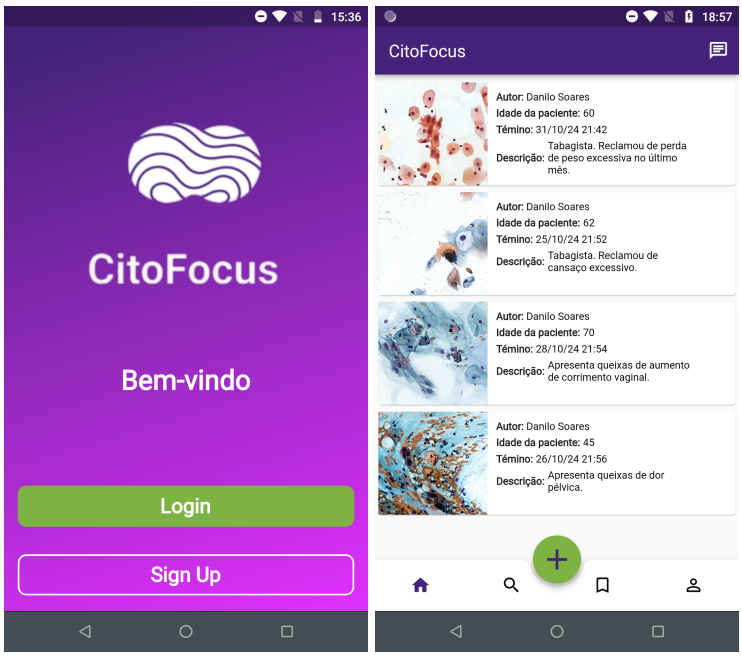
Quanto às modificações para integrar o classificador ao *back-end*, elas se resumiram a criar duas novas funções no arquivo *service*. São as funções *sendToClassifier* e *executeCommand*, como podem ser vistas na Figura 4.4 e também foi necessário chamar a primeira função citada dentro da função *create*, que também pode ser vista na Figura 4.4.

Além disso, foi necessário adicionar novos atributos ao objeto *case*, especificamente um

vetor para armazenar as imagens classificadas e um campo para o texto gerado pelo classificador, conforme ilustrado na Figura 3.8. Graças ao modelo, ao *framework* e às arquiteturas adotadas no desenvolvimento do *back-end*, obtivemos um serviço altamente desacoplado, garantindo que eventuais modificações sejam restritas aos componentes diretamente envolvidos, sem impactar o funcionamento das demais funcionalidades.

```
35  async create(createCaseDto: CreateCaseDto, @UploadedFiles() imageUpload: Express.Multer.File[]): Promise<Case> {···
95  }
96
97  async uploadFile(@UploadedFiles() image, filePath: string, nomeImg: string) {···
149 }
150
151 async sendToClassifier(imageBuffer: Buffer, fileName: string): Promise<{ classification: string; fileId: string; }>{···
201 }
202
203 private async executeCommand(command: string): Promise<string> {···
221 }
```

Figura 4.4 – Funções responsáveis pela criação do caso, *upload* das imagens, e processamento das imagens pelo classificador, respectivamente. As duas últimas funções são novas, enquanto que a primeira foi modificada para incluir a etapa de processamento das imagens.



(a) Tela inicial do aplicativo, sendo possível navegar para a tela de cadastro ou de login

(b) Tela inicial após acesso no aplicativo, apresenta a listagem de casos.

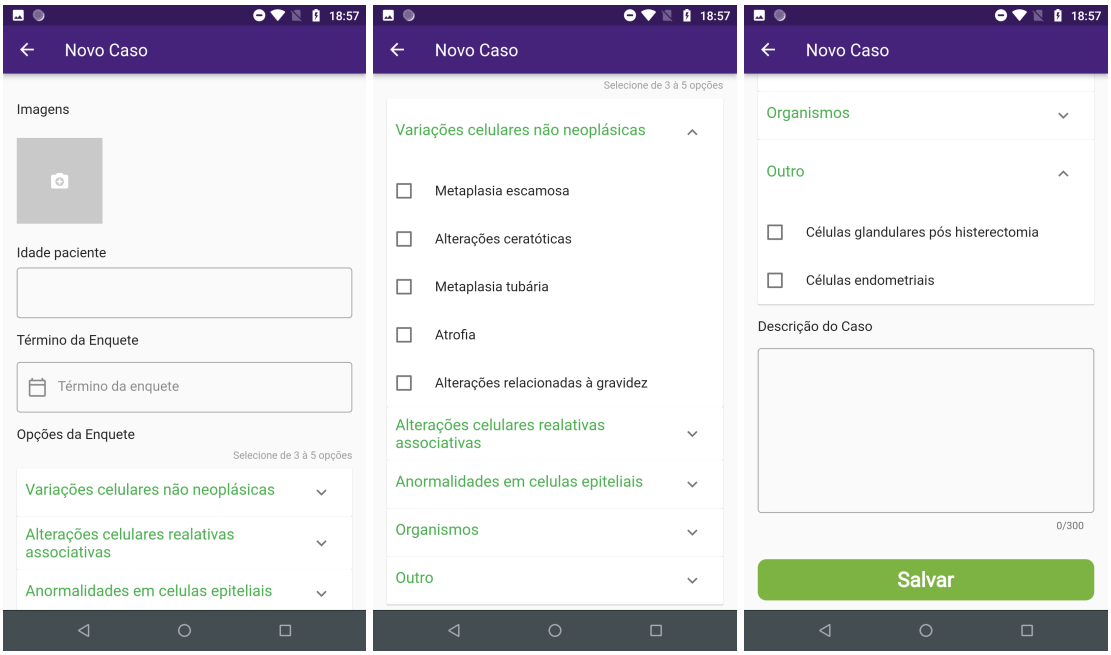
Figura 4.5 – Telas iniciais.

Para apresentar os dados gerados na aplicação sem impactar o fluxo original, foi criada uma nova tela dedicada à visualização dos resultados. Os testes indicam que essa abordagem integrou as novas informações de forma intuitiva, mantendo a experiência do usuário consistente, sem interferir no fluxo já estabelecido. Além disso, ao restringir a exibição dessa seção apenas após a votação, a aplicação garantiu que os dados fossem apresentados no momento mais adequado, servindo como um suporte adicional à análise dos usuários sem interferir no objetivo original.

Seguem algumas figuras apresentando o funcionamento do aplicativo que mesmo após o desenvolvimento do novo *back-end* manteve o fluxo de informações. A Figura 4.5a apresenta a tela de inicial do aplicativo. Logo após a verificação das credenciais temos a Figura 4.5b que mostra a tela onde temos todos os casos disponíveis.

As Figuras 4.6 mostram a tela de criação de caso. Na primeira figura inicialmente temos a opção de escolha das imagens do caso, a opção abre um *pop-up* para o usuário selecionar se irá para a câmera tirar a foto ou se vai para a galeria buscar a imagem já armazenada no dispositivo móvel. Além disso, possui o campo para a idade e a opção para selecionar a data de encerramento da enquete, Figura 4.6a, que temos por padrão 10 dias.

A Figura 4.6b, mostra as opções que o usuário pode selecionar para o caso. Existe os tipos de lesões que são abertas para que sejam selecionadas as lesões daquele tipo. E a Figura 4.6c mostra onde o usuário insere uma descrição para o caso.

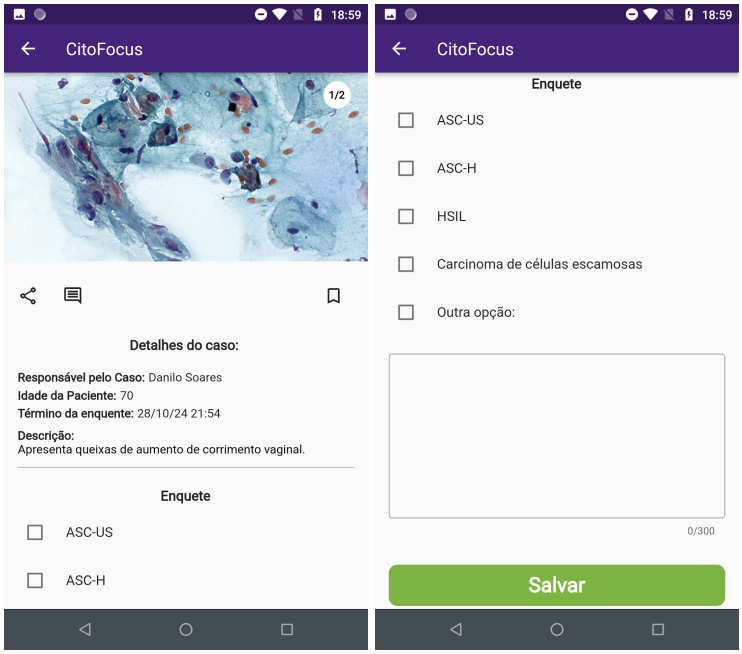


(a) (b) (c)

Figura 4.6 – Tela de criação dos casos.

As Figuras 4.8, 4.7a e 4.7b mostram um caso criado em que o usuário ainda não votou.

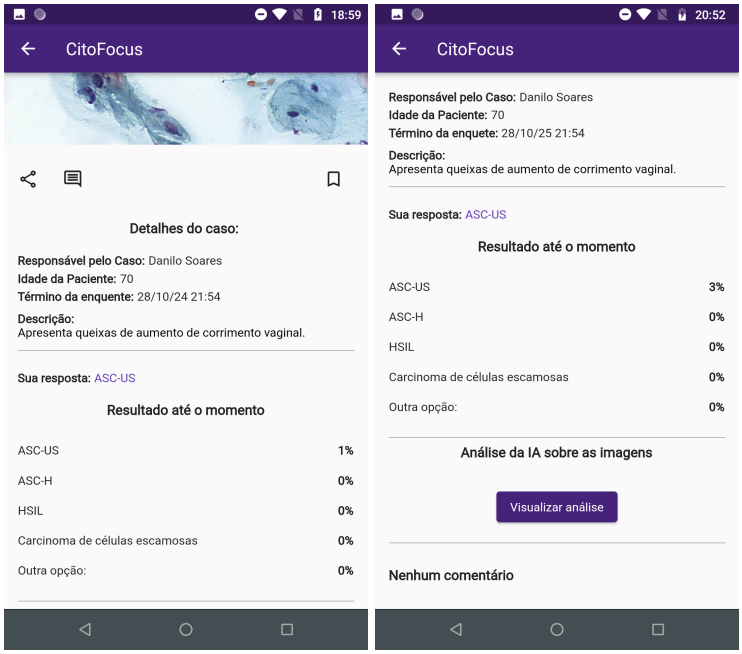
Por fim, as Figuras 4.8a e 4.8b apresentam a tela carregada com o resultado até o momento, exibida ao usuário quando ele entra em um caso que já foi votado. Essa tela mantém o fluxo original da aplicação, sem impactar a experiência do usuário.



(a) Tela de visualização do caso sem voto do usuário.

(b) Tela de visualização do caso sem voto do usuário.

Figura 4.7 – Tela de visualização de caso, sendo possível votar e um espaço para adicionar uma justificativa.



(a) Tela de visualização do caso com voto do usuário, sendo possível visualizar os dados fornecidos para análise do caso.

(b) Tela de visualização do caso com voto do usuário com botão de navegação para tela de dados gerados pelo classificador visível.

Figura 4.8 – Tela de visualização de caso com votos já computados.

5 Considerações Finais

5.1 Conclusão

O CitoFocus foi desenvolvido como uma solução para apoiar os profissionais da área da citologia na tomada de decisões, visando aprimorar a colaboração entre citopatologistas no diagnóstico de lesões. Seu objetivo é reduzir erros e ruídos causados por plataformas não especializadas para a interação desses profissionais, promovendo assim a troca de informações de forma precisa. Além disso, a adoção dessa plataforma oferece um ambiente semelhante a uma biblioteca de armazenamento de discussões de casos citopatológicos.

Esse trabalho propõe o desenvolvimento de uma arquitetura que permita integrar um classificador no aplicativo, mantendo seu funcionamento original e de forma que o classificador possa interagir com a aplicação fornecendo mais informações para os usuários se apoiarem. O classificador integrado na plataforma foi elaborado por Diniz et al. (2022).

De acordo com o que foi proposto por esse trabalho, temos a nova implementação do *back-end* utilizando a arquitetura proposta. Com esse desenvolvimento, temos essa camada da aplicação modularizada, sendo assim cada pequeno módulo da aplicação fica responsável por si mesmo, garantindo maior organização e escalabilidade. Essa modularização facilita futuras atualizações, manutenção e aprimoramentos sem comprometer o funcionamento geral do sistema.

Com a nova implementação do *back-end* utilizando a arquitetura, temos uma organização que permite sua fácil manutenção, atualização e integração do classificador. Isso reforça a flexibilidade do sistema para que outras versões do classificador possam ser incorporadas sem impactos significativos na aplicação.

Também temos as modificações que permitem a integração do *back-end* com o classificador, garantindo que as imagens enviadas pelos usuários sejam processadas pelo classificador e que nosso serviço capture todas as informações relevantes geradas por ele. Essa integração evidencia a viabilidade de um aplicativo que incorpora um classificador automático para auxiliar profissionais, ampliando as possibilidades de apoio à análise citopatológica.

Além disso, o aplicativo continua funcionando corretamente, demonstrando que todas as alterações estruturais não impactaram negativamente seu fluxo de informações e usabilidade. Como mostrado nas Figuras presentes no Capítulo 4 e nas Figuras 3.12a e 3.12b, a adição de uma nova tela para visualização dos dados gerados não comprometeu o funcionamento e o fluxo original, assegurando uma experiência contínua e intuitiva para os usuários como era originalmente.

A ideia inicial era que o classificador atuasse como um voto adicional. No entanto,

conforme detalhado no Capítulo 3, essa implementação não foi viável. Em vez disso, foi possível integrá-lo para fornecer novas informações que auxiliam na análise dos usuários, alinhando-se ao objetivo original da aplicação: apoiar citopatologistas na análise de casos. Com isso, o sistema não apenas preserva sua funcionalidade, mas também se consolida como uma ferramenta de inteligência artificial aplicada à citologia, permitindo que profissionais utilizem a tecnologia para aprimorar diagnósticos e tomadas de decisão.

5.2 Trabalhos Futuros

Como parte dos próximos passos, planejamos testar o aplicativo com profissionais da saúde, especialmente os especialistas da área, para avaliar sua usabilidade, precisão e impacto na prática clínica. Esses testes permitirão identificar pontos fortes e possíveis melhorias na interface, nos algoritmos de análise de imagens e na integração com fluxos de trabalho existentes.

Além disso, realizaremos um estudo de receptividade para compreender a experiência dos usuários, analisando fatores como facilidade de uso, tempo de resposta e confiabilidade dos resultados. O *feedback* coletado será essencial para ajustes e refinamentos, garantindo que a ferramenta atenda às necessidades reais dos profissionais e contribua de forma eficiente para a tomada de decisão.

Referências

BUIU, C.; DĂNĂILĂ, V.-R.; RĂDUŢĂ, C. N. Mobilenetv2 ensemble for cervical precancerous lesions classification. *Processes*, MDPI, v. 8, n. 5, p. 595, 2020.

COUTO, G. *O Poder das Redes Neurais Convolucionais: Um Guia Prático para Construir sua CNN Básica*. 2023. Linkedin. Último acesso 15 de setembro de 2024. Disponível em: <https://pt.linkedin.com/pulse/o-poder-das-redes-neurais-convolucionais-um-guia-pr%C3%A1tico-couto-j7omf>.

DINIZ, D. N.; KELLER, B. N. S.; REZENDE, M. T.; BIANCHI, A. G. C.; CARNEIRO, C. M.; OLIVEIRA, R. R. e. R.; LUZ, E. J. S.; USHIZIMA, D. M.; MEDEIROS, F. N. S. de; SOUZA, M. J. F. A cytopathologist eye assistant for cell screening. *AppliedMath*, v. 2, n. 4, p. 659–674, 2022. ISSN 2673-9909. Disponível em: <https://www.mdpi.com/2673-9909/2/4/38>.

FERNANDES, F. T.; FILHO, A. D. P. C. Perspectivas do uso de mineração de dados e aprendizado de máquina em saúde e segurança no trabalho. *Revista Brasileira de Saúde Ocupacional*, SciELO Brasil, v. 44, p. e13, 2019.

FUNDAÇÃO OSWALDO CRUZ. *Instituto Nacional de Saúde da Mulher, da Criança e do Adolescente Fernandes Figueira. Portal de Boas Práticas em Saúde da Mulher, da Criança e do Adolescente. Postagens: Coleta e Indicações para o Exame Citopatológico do Colo Uterino*. 2023. Rio de Janeiro, 25 mai. 2023. Disponível em: <https://portaldeboaspraticas.iff.fiocruz.br/atencao-mulher/coleta-e-indicacoes-para-o-exame-citopatologico-do-colo-uterino/>. Acessado em: 01 out. 2024.

GADO, W. *Modularização: fFunções e Procedimentos*. 2021. TreinaWeb. Último acesso 29 de julho de 2024. Disponível em: <https://www.treinaweb.com.br/blog/modularizacao-funcoes-e-procedimentos>.

GOMES, M. F. d. S. L.; ROMANO, S. M. V.; DIAS, J. C. Modularização de aplicativos ios. *Revista Processando o Saber*, v. 15, p. 01–15, 2023.

GUIMARÃES, T. D.

Desenvolvimento de um aplicativo para colaboração entre citopatologistas. — Faculdade de Ciência da Computação, Universidade Federal de Ouro Preto, 2021. Último acesso 05 de setembro de 2024.

HOLDSWORTH, J.; SCAPICCHIO, M. *O que é deep learning?* 2024. IBM. Último acesso 29 de julho de 2024. Disponível em: <https://www.ibm.com/br-pt/topics/deep-learning>.

HOMBERGS, T. *Hexagonal Architecture with Java and Spring*. 2019. Reflectoring. Último acesso 09 de setembro de 2024. Disponível em: <https://reflectoring.io/spring-hexagonal/>.

IFTIKHAR, M.; KANDHRO, I. A.; KAUSAR, N.; KEHAR, A.; UDDIN, M.; DANDOUSH, A. Plant disease management: a fine-tuned enhanced cnn approach with mobile app integration for early detection and classification. *Artificial Intelligence Review*, Springer, v. 57, n. 7, p. 1–29, 2024.

Instituto Nacional do Câncer. *Conceito e Magnitude*. 2022. Instituto Nacional do Câncer. Último acesso 29 de julho de 2024. Disponível em: <https://www.gov.br/inca/pt-br/assuntos/gestor-e-profissional-de-saude/controle-do-cancer-do-colo-do-utero/conceito-e-magnitude>.

KELLER, B.; GUIMARAES, T. D.; MALAQUIAS, P. I. d. S.; FERREIRA, G. M. d. S.; RESENDE, M. T.; CARNEIRO, C. M.; BIANCHI, A. G. Citofocus: Uma plataforma para colaboração e aprendizado em citopatologia. In: SBC. *Anais do XXI Simpósio Brasileiro de Computação Aplicada à Saúde*. [S.l.], 2021. p. 404–409.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, SciELO Brasil, v. 35, p. 85–94, 2021.

MAMMAS, I.; DA, S. George n. papanicolaou (1883–1962): Fifty years after the death of a great doctor. *Scientist and Humanitarian. J BUON*, v. 17, n. 1, p. 180–4, 2012.

MITCHELL, T. M. Machine learning and data mining. *Communications of the ACM*, ACM New York, NY, USA, v. 42, n. 11, p. 30–36, 1999.

MONSUR, S. A.; ADESHINA, S. A.; SUD, S.; SOBOYEJO, W. O. A mobile-based image analysis system for cervical cancer detection. In: IEEE. *2017 13th International Conference on Electronics, Computer and Computation (ICECCO)*. [S.l.], 2017. p. 1–5.

Node.js Documentation. *child_process.exec(command[, options][, callback])*. [S.l.], 2025. Acessado em: 11 fev. 2025. Disponível em: <https://nodejs.org/api/child_process.html#child_processexeccommand-options-callback>.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

REHMAN, A.-u.; ALI, N.; TAJ, I.; SAJID, M.; KARIMOV, K. S. An automatic mass screening system for cervical cancer detection based on convolutional neural network. *Mathematical Problems in Engineering*, v. 2020, n. 1, p. 4864835, 2020. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/4864835>.

REISNER, H. *Patologia: Uma Abordagem por Estudos de Casos*. Elsevier, 2015. 15 p. Consulta feita no Google Livros. Disponível em: <https://books.google.com.br/books?id=cx_hCgAAQBAJ&pg=PA21&hl=pt-BR&source=gbs_toc_r&cad=2#v=onepage&q&f=false>.

RERICK, E. *Arquitetura Hexagonal com NestJS*. 2023. Medium. Último acesso 09 de setembro de 2024. Disponível em: <https://medium.com/@eduardorerick/arquitetura-hexagonal-com-nestjs-54f5e87bb210>.

REZENDE, M. T.; BIANCHI, A. G.; CARNEIRO, C. M. Cervical cancer: Automation of pap test screening. *Diagnostic cytopathology*, Wiley Online Library, v. 49, n. 4, p. 559–574, 2021.

SACHAN, P. L.; SINGH, M.; PATEL, M. L.; SACHAN, R. A study on cervical cancer screening using pap smear test and clinical correlation. *Asia-Pacific journal of oncology nursing*, Elsevier, v. 5, n. 3, p. 337–341, 2018.

TERVEN, J.; CÓRDOVA-ESPARZA, D.-M.; ROMERO-GONZÁLEZ, J.-A. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, MDPI, v. 5, n. 4, p. 1680–1716, 2023.

ULTRALYTICS. *YOLOv5*. 2020. <https://github.com/ultralytics/yolov5>. Accessed: 2025-06-29.

World Health Organization. *Cervical cancer*. 2024. World health Organization. Último acesso 03 de setembro de 2024. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/cervical-cancer?gad_source=1>.