CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ROGÉRIO DE CARVALHO CAJÁ

APLICATIVO MÓVEL PARA CRIAÇÃO E VALIDAÇÃO DE MODELOS DE RECONHECIMENTO DE PADRÕES

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Leonardo Henrique Raiz

FRANCA/SP
2025

APLICATIVO MÓVEL PARA CRIAÇÃO E VALIDAÇÃO DE MODELOS DE RECONHECIMENTO DE PADRÕES

ROGÉRIO DE CARVALHO CAJÁ

RESUMO

O avanço da Inteligência Artificial (IA) tem ampliado sua presença em diversas áreas, mas o processo de criação e treinamento de modelos ainda é restrito a ambientes especializados. Este trabalho propõe o desenvolvimento de um aplicativo móvel capaz de simular a criação, o treinamento e a validação de modelos de reconhecimento de padrões em ambiente totalmente local e offline.

O sistema, denominado *AI Model Trainer*, foi desenvolvido com *React Native* e o framework *Expo* para execução multiplataforma, utilizando *SQLite* para persistência local de dados. O aplicativo permite ao usuário criar projetos, adicionar conjuntos de dados rotulados, simular o processo de treinamento com visualização de progresso e testar os modelos com novas entradas.

A proposta tem caráter educacional, visando democratizar o acesso ao aprendizado prático em IA, eliminando a necessidade de infraestrutura em nuvem ou recursos computacionais avançados. O projeto combina autonomia técnica, usabilidade e valor pedagógico, oferecendo uma plataforma intuitiva para estudantes e entusiastas compreenderem os princípios básicos de aprendizado de máquina e reconhecimento de padrões.

Palavras-chave: Inteligência Artificial. Aprendizado de Máquina. Reconhecimento de Padrões. *React Native*. Aplicativo Móvel. Sistema Offline.

ABSTRACT

Title: *Mobile Application for Creation and Validation of Pattern Recognition Models*

The exponential growth of artificial intelligence (AI) has made it present in many areas of society, but the process of creating and training AI models remains complex and limited to specialized environments. This project proposes the development of a mobile application capable of simulating the creation, training, and validation of pattern recognition models in a completely local and offline environment.

The system, named AI Model Trainer, was developed using React Native with the Expo framework for cross-platform implementation and SQLite for local data persistence. The application allows users to create projects, add labeled datasets, simulate training with progress visualization, and test the models with new inputs.

This educational approach aims to democratize access to AI experimentation by enabling practical learning without the need for cloud services or advanced infrastructure. The project combines technical autonomy, usability, and pedagogical value, providing an intuitive platform for students and enthusiasts to understand the basic principles of machine learning and pattern recognition.

***Keywords**: Artificial Intelligence; Machine Learning; Pattern Recognition; React Native; Mobile Application; Offline System.*

1 INTRODUÇÃO

Nas últimas décadas, a inteligência artificial (IA) tem avançado de maneira exponencial, tornando-se presente em praticamente todos os setores da sociedade. Modelos de aprendizado de máquina vêm sendo aplicados em tarefas que vão desde o reconhecimento facial até a análise preditiva de dados corporativos. No entanto, a criação e o treinamento desses modelos ainda permanecem, em grande parte, restritos a ambientes altamente técnicos, exigindo infraestrutura robusta, conhecimentos especializados e conexão constante com a nuvem.

Essa barreira técnica tem afastado estudantes, pesquisadores iniciantes e pequenos desenvolvedores do processo de experimentação e aprendizado prático em IA. Mesmo para tarefas simples — como classificar imagens, identificar padrões em textos ou validar pequenas bases de dados — o caminho tradicional envolve ferramentas complexas, configurações demoradas e custos com serviços em nuvem. Em muitos casos, o simples desejo de “treinar um modelo” transforma-se em um processo frustrante e inacessível.

Foi nesse cenário que surgiu a motivação para o desenvolvimento deste projeto: um aplicativo móvel capaz de criar, treinar e testar modelos de reconhecimento de padrões de forma totalmente local e offline. A proposta nasceu da necessidade de democratizar o acesso à experimentação em IA, proporcionando uma experiência acessível, intuitiva e independente de conexões externas.

Mais do que uma ferramenta, o aplicativo busca ser um ambiente de aprendizado e exploração, permitindo que qualquer usuário — com ou sem conhecimento avançado — possa criar seus próprios modelos personalizados de classificação.

O aplicativo foi construído com base em tecnologias híbridas, utilizando *React Native* com o framework *Expo* para o desenvolvimento multiplataforma e o *SQLite* como banco de dados local, garantindo armazenamento e persistência de dados mesmo em modo offline. Essa abordagem possibilita que o usuário mantenha seus projetos, dados de treinamento e resultados diretamente no dispositivo, sem a necessidade de servidores externos ou dependência de *APIs* de terceiros.

Além de oferecer uma interface simples e didática, o sistema simula o processo de treinamento e avaliação de modelos, permitindo que o usuário acompanhe o progresso, visualize métricas de acurácia e valide seus resultados de maneira prática. A aplicação se posiciona como uma solução educacional e exploratória, que pode ser utilizada tanto em contextos acadêmicos — para introdução aos conceitos de aprendizado de máquina — quanto em experimentos pessoais ou protótipos rápidos.

A "dor" que este projeto busca resolver, portanto, é a distância entre o interesse em IA e a complexidade das ferramentas disponíveis. Ao remover barreiras técnicas e permitir que o processo de treinamento ocorra de forma simples e local, o aplicativo contribui para tornar o aprendizado de inteligência artificial mais acessível, interativo e inclusivo.

Nos capítulos seguintes, serão detalhados os objetivos, a fundamentação teórica e a metodologia aplicada para o desenvolvimento do sistema, bem como a modelagem de seus processos, casos de uso, requisitos funcionais e não funcionais, e as tecnologias que sustentam sua arquitetura.

1.1 TERMO DE ABERTURA DE PROJETO (TAP)

Segundo o site da Artia (2020): "o termo de abertura do projeto (TAP) é um documento que formaliza o início de um projeto, confere autoridade ao gerente de projeto e agrupa todas as informações necessárias para a execução das atividades envolvidas". O TAP serviu como guia norteador durante toda a execução, assegurando que as decisões técnicas e metodológicas estivessem alinhadas ao propósito educacional do *AI Model Trainer*, bem como aos princípios de Engenharia de Software adotados neste trabalho.

O documento completo encontra-se disponível no Apêndice A deste relatório.

1.2 JUSTIFICATIVA

O avanço das tecnologias de inteligência artificial tem transformado de forma significativa a maneira como as pessoas interagem com sistemas digitais, consomem informação e tomam decisões. No entanto, apesar de sua crescente popularização, a criação e o treinamento de modelos de IA ainda são processos altamente técnicos, demandando conhecimentos específicos em linguagens de programação, bibliotecas especializadas e ambientes complexos de desenvolvimento.

Essa realidade cria uma barreira de entrada considerável para estudantes e profissionais que desejam aprender, testar e compreender como funcionam os modelos de aprendizado de máquina. Na prática, o ensino e a experimentação em IA costumam ficar restritos a ambientes acadêmicos ou corporativos, onde há infraestrutura adequada e suporte técnico disponível. Assim, a oportunidade de aprendizado prático é limitada e, muitas vezes, inacessível a quem busca compreender os fundamentos da área de maneira autônoma.

A justificativa para o desenvolvimento deste projeto está justamente na necessidade de democratizar o acesso à experimentação em inteligência artificial, por meio de uma ferramenta simples, intuitiva e portátil. O aplicativo proposto oferece um ambiente de criação, treinamento e validação de modelos de reconhecimento de padrões totalmente local, eliminando a necessidade de servidores externos ou conexão com a internet. Essa abordagem permite que qualquer pessoa, mesmo sem experiência prévia, possa compreender de forma prática os conceitos por trás de um modelo de classificação.

Além do aspecto educacional, o projeto também tem uma relevância técnica significativa. Ao adotar tecnologias híbridas como *React Native* e *Expo*, é possível entregar uma aplicação multiplataforma, de baixo custo e fácil manutenção. O uso do *SQLite* como banco de dados local reforça o propósito de autonomia do aplicativo, garantindo persistência dos dados sem dependência de conexões externas.

Outro ponto importante é a simplicidade do ciclo de desenvolvimento e uso, que o torna ideal para ambientes de ensino técnico e superior, como disciplinas de inteligência artificial, programação ou sistemas embarcados. O aplicativo também pode ser explorado por professores, pesquisadores e desenvolvedores independentes como ferramenta de prototipagem rápida de conceitos.

Em síntese, o projeto é justificado por sua utilidade educacional, sua contribuição à inclusão tecnológica e sua viabilidade técnica, ao combinar tecnologias modernas de desenvolvimento móvel com uma proposta pedagógica voltada à compreensão prática de modelos de IA. Ao simplificar processos que antes dependiam de recursos avançados, o aplicativo se propõe a ser um ponto de partida acessível para o aprendizado e a experimentação no campo da inteligência artificial.

1.3 FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento do aplicativo *AI Model Trainer* fundamenta-se em conceitos e tecnologias relacionadas à inteligência artificial, aprendizado de máquina, reconhecimento de padrões, armazenamento local de dados e desenvolvimento móvel multiplataforma. Nesta seção, são apresentados os principais fundamentos teóricos que embasam o projeto.

1.3.1 Inteligência Artificial e Reconhecimento de Padrões

A Inteligência Artificial (IA) é um campo da ciência da computação voltado à criação de sistemas capazes de realizar tarefas que, tradicionalmente, exigiriam inteligência humana — como reconhecimento de voz, visão computacional, raciocínio lógico e tomada de decisão.

De acordo com *Russell* e *Norvig* (2016), a IA busca desenvolver agentes autônomos que percebem o ambiente e tomam ações que maximizam suas chances de sucesso em determinado objetivo.

Dentro desse contexto, o reconhecimento de padrões é uma das áreas mais importantes da IA. Ele envolve o processamento e a análise de dados (como imagens, sons ou textos) para identificar características recorrentes e classificar informações com base em exemplos anteriores.

No caso deste projeto, o reconhecimento de padrões é aplicado de forma educacional e simulada, permitindo que o usuário compreenda os conceitos de classificação supervisionada — ou seja, o treinamento de um modelo com exemplos rotulados que ensinam a máquina a reconhecer novas entradas.

1.3.2 Aprendizado de Máquina

O Aprendizado de Máquina (*Machine Learning*) é uma subárea da IA responsável por desenvolver algoritmos que aprendem a partir de dados e melhoram seu desempenho com o tempo, sem serem explicitamente programados para isso.

Segundo *Mitchell* (1997), um sistema aprende se seu desempenho em uma tarefa T, medido por uma métrica P, melhora com a experiência E.

Na prática, o aprendizado de máquina envolve três etapas principais:

- coleta e preparação dos dados de treinamento (entrada e rótulo);
- treinamento do modelo, ajustando seus parâmetros internos para minimizar erros;
- teste e validação, avaliando o desempenho do modelo com novos dados.

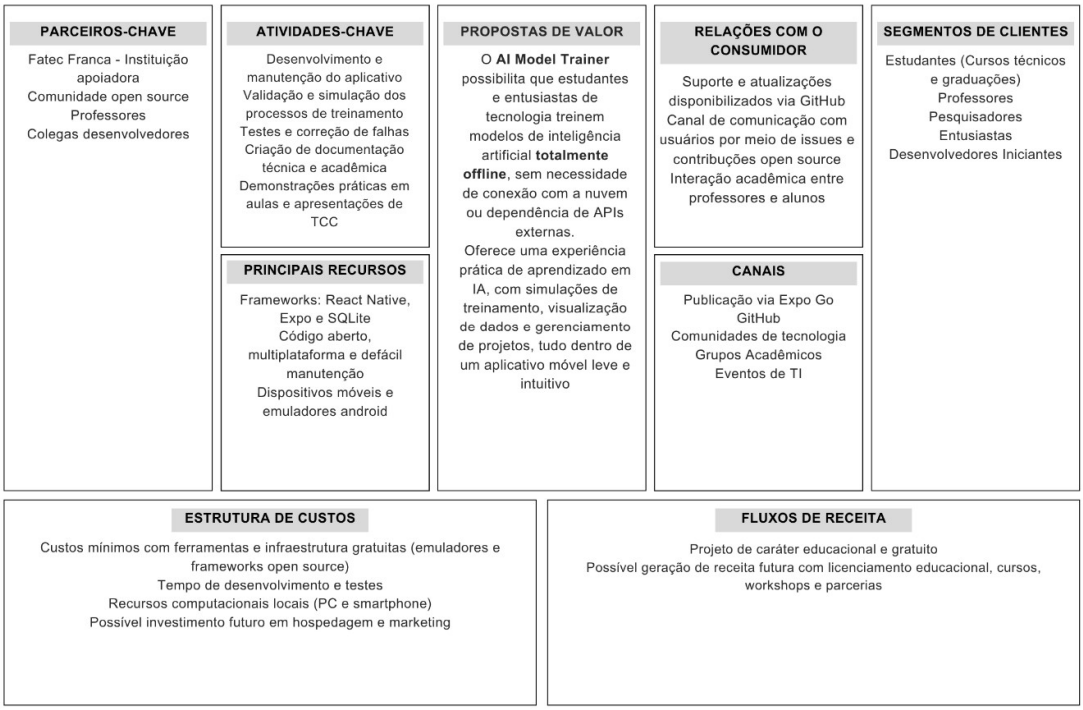
O aplicativo proposto simula esse processo em ambiente local, permitindo ao usuário adicionar dados rotulados (imagens ou textos), acompanhar o progresso de treinamento e visualizar métricas de acurácia.

Embora o modelo não realize um treinamento real com redes neurais ou algoritmos de regressão, a simulação é baseada em etapas reais de machine learning, proporcionando uma experiência educativa e visualmente compreensível.

2 CANVAS DE NEGÓCIO

Quadro 1 – Canvas de negócio

AI MODEL TRAINER



3 LEVANTAMENTO DE REQUISITOS

3.1 ELICITAÇÃO E ESPECIFICAÇÃO DOS REQUISITOS

De acordo com o site da DEVMEDIA (2014), “Elicitação de requisitos é uma fase do projeto onde são extraídas informações do cliente sobre o que ele deseja que seja construído. É a fase em que o profissional de TI entende a necessidade do cliente e o orienta. É o momento de conversa com o usuário, de sentimento sobre o que este espera que seja entregue a ele. Na elicitação de requisitos são percebidas as necessidades do sistema e as características que esse sistema deve ter.”. É na etapa de elicitação de requisitos que traduzimos os entregáveis para o cliente, ou seja: entendemos o que o cliente precisa/quer e especificamos, em linguagem técnica, como o sistema/software a ser desenvolvido atenderá essas necessidades.

A elicitação de requisitos do *AI Model Trainer* foi conduzida com base em uma análise exploratória do contexto educacional de cursos de tecnologia e nas demandas observadas durante disciplinas que envolvem aprendizado de máquina e desenvolvimento de software.

O processo teve caráter iterativo e qualitativo, buscando compreender as necessidades de estudantes e docentes que enfrentam dificuldades para realizar experimentos de Inteligência Artificial sem acesso a infraestrutura de nuvem ou ferramentas pagas. Além disso, seguiu-se o princípio de refinamento progressivo, onde os requisitos foram detalhados à medida que novas funcionalidades eram implementadas e testadas, garantindo alinhamento contínuo entre a proposta educacional e a viabilidade técnica.

Inicialmente, foi feito um levantamento informal junto a colegas e professores, por meio de conversas orientadas e observação de uso de ferramentas didáticas existentes. Essa investigação permitiu identificar as principais limitações: dificuldade de configurar ambientes de aprendizado, dependência de internet, e ausência de soluções didáticas que simulassem o processo de treinamento de IA de forma simples e visual.

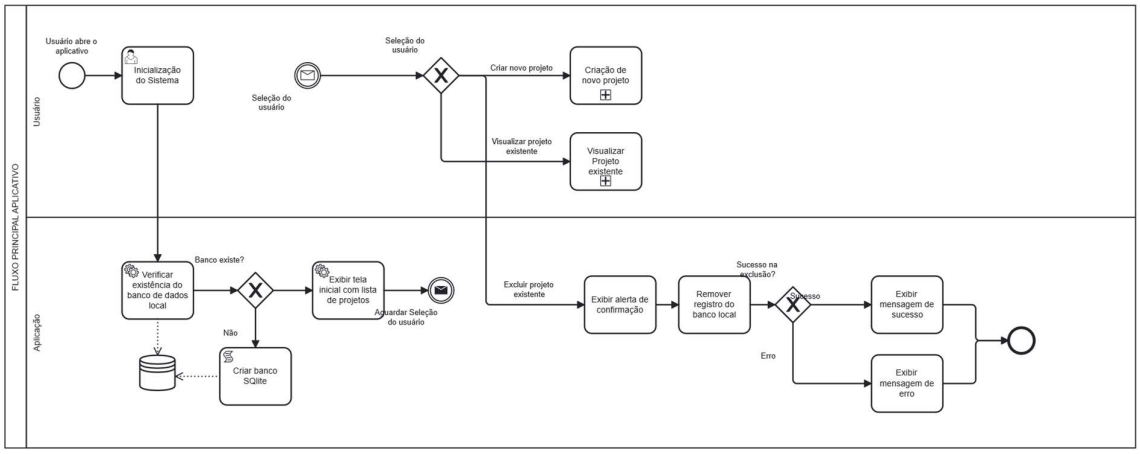
Com base nessas informações, foram elaborados requisitos funcionais e não funcionais do sistema, posteriormente organizados em tabela e validados ao longo do desenvolvimento.

3.2 DIAGRAMA BPMN (VISÃO DE PROCESSO)

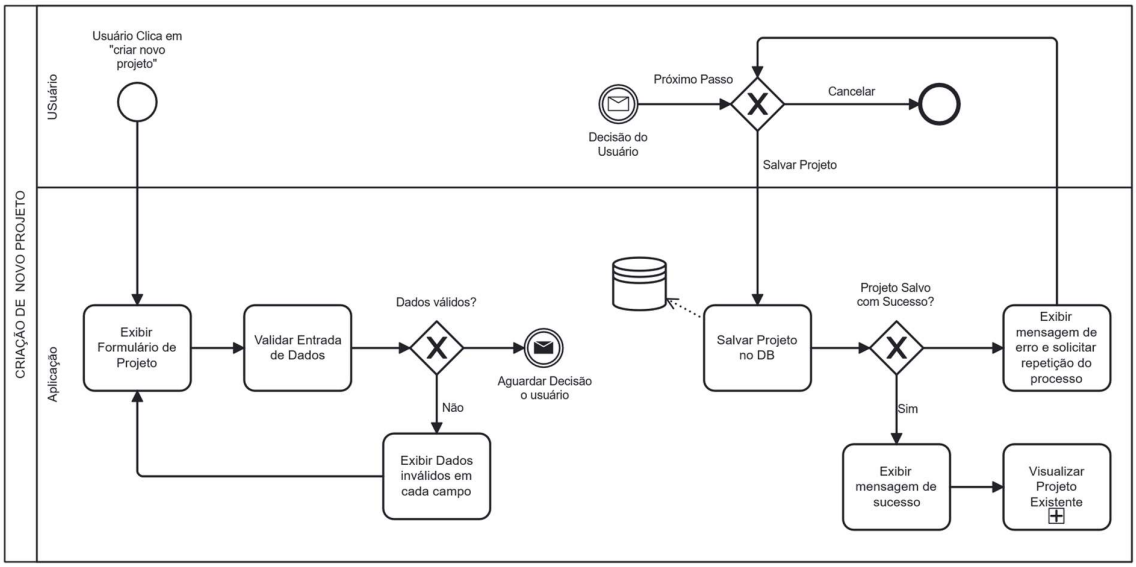
Segundo Pedro (2024), “o BPMN é a ferramenta de notações de modelos de processos de negócios utilizada na metodologia BPM, desenvolvida pela BPMI. Trata se de um diagrama formado por raias e fluxogramas que simplificam a visualização e entendimento dos processos de um negócio.” O dito fluxograma é um poderoso aliado para mapear processos e definir pontos de decisões, ações, envios, consultas etc. É o processo do cliente em formato de desenho compreensível por qualquer pessoa que não faça parte do meio.

O diagrama BPMN (*Business Process Model and Notation*) será utilizado para representar o fluxo de ações do usuário dentro do aplicativo, desde a criação de um novo projeto até a execução dos testes de modelo.

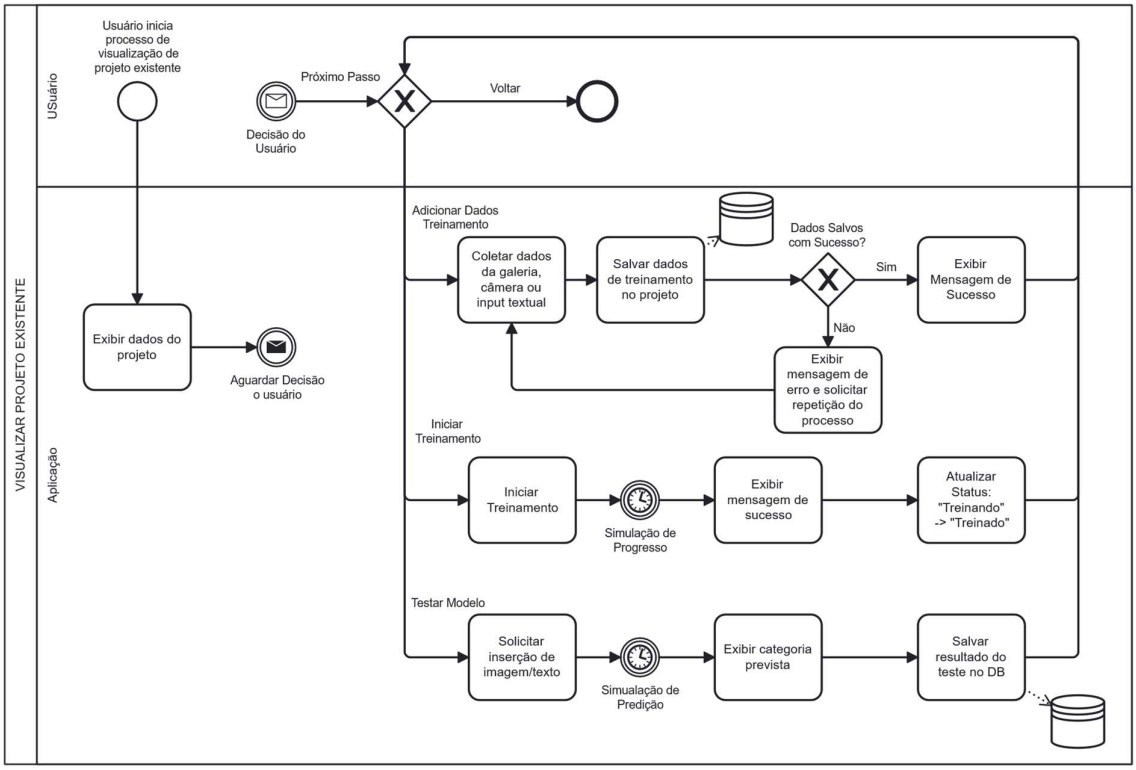
Quadro 2 – BPMN – Fluxo Principal Aplicativo



Quadro 3 – BPMN – Criação de Novo Projeto



Quadro 4 – BPMN – Visualizar Projeto Existente



3.3 REQUISITOS FUNCIONAIS

Segundo o site Casa do Desenvolvedor (2023), “Os requisitos funcionais são aqueles que visam atingir a solução dos problemas do usuário. Desse modo, eles trabalham diretamente no objetivo para o qual uma solução foi escrita.”. Traduzindo: trata-se dos requisitos que especificarão o que o sistema deverá fazer/providenciar/prover. Eles visam atender as necessidades dos usuários em forma de funções no sistema.

Tabela 1 – Requisitos Funcionais

ID	Categoria	Descrição do Requisito	Prioridade
RF01	Evidente	O sistema deve permitir criar novos projetos de treinamento com nome e tipo.	Alta
RF02	Evidente	O sistema deve listar todos os projetos cadastrados no banco local.	Alta
RF03	Evidente	O sistema deve permitir a edição e exclusão de projetos existentes.	Média

ID	Categoria	Descrição do Requisito	Prioridade
RF04	Evidente	O usuário deve poder adicionar amostras de dados (imagem ou texto) com rótulos personalizados.	Alta
RF05	Evidente	O sistema deve simular o processo de treinamento exibindo progresso e acurácia estimada.	Alta
RF06	Evidente	O sistema deve permitir testes de modelo, com entrada de novas amostras e exibição de resultados.	Alta
RF07	Oculto	O aplicativo deve salvar automaticamente todas as alterações no banco local *SQLite*.	Alta
RF08	Evidente	O sistema deve permitir operação completa sem conexão com a internet.	Alta

3.4 REQUISITOS NÃO FUNCIONAIS

Ainda no site Casa do Desenvolvedor (2023), “os requisitos não funcionais são premissas essenciais para as execuções das funções definidas pelos requisitos funcionais.” Ou seja, eles determinam como o sistema executará as funções definidas nos requisitos funcionais

Tabela 2 – Requisitos Não Funcionais

ID	Categoria	Descrição do Requisito	Prioridade
RNF01	Compatibilidade	O aplicativo deve ser multiplataforma, executando em dispositivos *Android* e *iOS*.	Alta
RNF02	Usabilidade	O sistema deve possuir interface simples, responsiva e intuitiva.	Alta
RNF03	Performance	O banco de dados local deve suportar pelo menos 100 registros sem degradação perceptível.	Média
RNF04	Performance	O tempo de resposta entre ações do usuário e retorno visual não deve ultrapassar 1 segundo.	Média
RNF05	Confiabilidade	O sistema deve garantir a integridade dos dados armazenados no *SQLite*.	Alta

RNF06	Manutenibilidade	O aplicativo deve seguir boas práticas de codificação e modularização.	Média

3.5 CASOS DE USO

Segundo o site DevMedia, “Um caso de uso é uma sequência de interações entre o ator (alguém ou algo que interage com o sistema) e o sistema, que acontece de forma atômica, na perspectiva do ator. Isso significa que quando criamos um caso de uso apenas nos preocupamos com “o que” o sistema deve fazer e não com “o como” deve fazer.” Em outras palavras, é uma esquematização das interações entre os atores e o sistema que ajuda a contextualizar o que o sistema deve fazer de acordo com as ações dos atores.

A seguir, são apresentados os principais casos de uso que descrevem o comportamento funcional do sistema do ponto de vista do usuário.

Tabela 3 – Casos de uso

Caso de Uso	Descrição	Atores	Fluxo Principal
UC01 – Criar Projeto	Permite ao usuário criar um novo projeto no aplicativo, informando nome e tipo (imagem ou texto).	Usuário	1. O usuário acessa a tela inicial 2.Seleciona “Novo Projeto” 3. Informa os dados 4. O sistema salva o projeto no banco local.
UC02 – Listar Projetos	Exibe a lista de projetos cadastrados no banco de dados local.	Usuário	1. O sistema acessa o banco SQLite 2. Retorna todos os projetos armazenados 3. Exibe os resultados na interface.
UC03 – Editar ou Excluir Projeto	Permite modificar informações de um projeto ou removê-lo permanentemente.	Usuário	1. O usuário seleciona um projeto 2. Escolhe editar ou excluir 3. O sistema

Caso de Uso	Descrição	Atores	Fluxo Principal
			atualiza ou remove o registro no SQLite.
UC04 – Adicionar Dados de Treinamento	O usuário adiciona amostras de imagem ou texto com seus respectivos rótulos.	Usuário	1. O usuário abre o projeto 2. Adiciona as amostras 3. O sistema salva os dados localmente.
UC05 – Simular Treinamento	Executa o processo de treinamento simulado, calculando progresso e acurácia.	Usuário / Sistema	1. O usuário inicia o treinamento 2. O sistema simula iterações 3. Exibe a acurácia final e atualiza o status do modelo.
UC06 – Realizar Teste de Modelo	Permite inserir novas amostras e verificar o resultado do modelo simulado.	Usuário	1. O usuário informa uma entrada 2. O sistema compara com os padrões aprendidos 3. Retorna a previsão simulada.
UC07 – Persistir Dados Localmente	Garante que todas as informações sejam salvas e recuperadas do banco local.	Sistema	1. Cada operação de criação ou edição grava no SQLite 2. O sistema recupera dados sempre que o app é reaberto.

3.6 DIAGRAMA DE CLASSES

O Diagrama de Classes é um dos principais artefatos da modelagem orientada a objetos, permitindo representar a estrutura estática do sistema por meio de suas classes, atributos, métodos e relacionamentos.

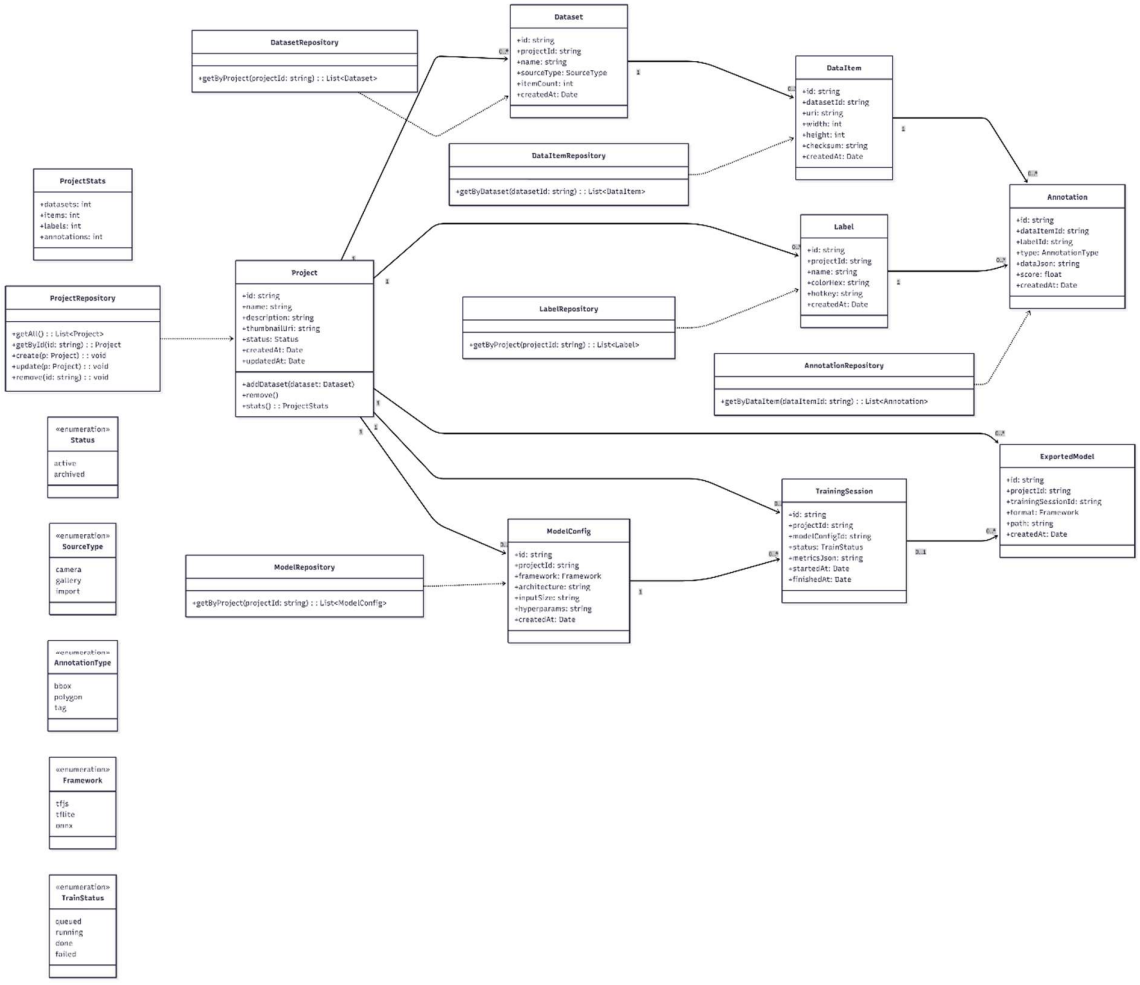
No contexto do AI Model Trainer, ele evidencia a relação entre entidades fundamentais, como *Project*, *Dataset*, *Label*, *Annotation* e *ModelConfig*, refletindo a organização dos dados que sustentam as operações do aplicativo. Essa estrutura facilita a compreensão da persistência e manipulação das informações locais no banco *SQLite*, além de apoiar a modularidade do código.

Segundo *Sommerville* (2019), a modelagem de classes é essencial para a clareza arquitetural e a manutenção de sistemas orientados a objetos, pois reduz ambiguidades e permite a evolução controlada do software.

Pressman (2016) complementa que diagramas de classes bem definidos funcionam como uma ponte entre a análise e o design, auxiliando no refinamento dos requisitos e na comunicação entre equipe técnica e *stakeholders*.

Dessa forma, o diagrama apresentado cumpre papel central na documentação técnica do projeto, oferecendo uma visão abrangente da estrutura de dados e dos relacionamentos que sustentam as funcionalidades descritas nos requisitos funcionais.

Quadro 5 – Diagrama de Classes



4 FERRAMENTAS E MÉTODOS

4.1 FERRAMENTAS UTILIZADAS

As principais ferramentas e dependências utilizadas no desenvolvimento foram:

Tabela 4 – Ferramentas utilizadas

Ferramenta/Biblioteca	Versão Utilizada	Descrição
React Native	0.79.5	*Framework JavaScript* para criação de aplicativos móveis multiplataforma.
Expo	54.0.13	Plataforma que simplifica o desenvolvimento e distribuição de aplicativos *React Native*.

Ferramenta/Biblioteca	Versão Utilizada	Descrição
Expo Router	5.1.4	Biblioteca de roteamento baseada em arquivos, facilitando a organização das telas.
***SQLite* (expo-*sqlite*)**	16.0.8	Banco de dados relacional embutido que permite o armazenamento local.
Expo *Image Picker* / *Camera*	17.0.8	Módulos para seleção e captura de imagens utilizadas no treinamento.
TypeScript	5.8.3	Linguagem com tipagem estática opcional, oferecendo maior segurança e clareza ao código.
Visual Studio Code	1.95	Ambiente de desenvolvimento principal, com suporte a formatação e depuração.
Android Studio* / *Expo Go	2024.1.1/ 2.31.3	Ferramentas utilizadas para teste e execução em ambiente móvel.

Essas tecnologias foram escolhidas por oferecerem robustez, documentação ampla e suporte ativo da comunidade, além de atenderem ao objetivo de um aplicativo leve, autônomo e multiplataforma.

4.1.1 Armazenamento Local e Banco de Dados *SQLite*

A persistência dos dados é um dos pilares do sistema. Para isso, o aplicativo utiliza o *SQLite*, um banco de dados relacional embutido que opera localmente dentro do dispositivo móvel, sem necessidade de conexão com servidores externos.

O *SQLite* é amplamente utilizado em aplicações móveis devido à sua leveza, rapidez e simplicidade de integração. Ele permite armazenar informações em tabelas relacionais e realizar consultas SQL completas, mesmo em ambientes offline.

No contexto deste projeto, o *SQLite* é responsável por armazenar os projetos criados pelo usuário, dados de treinamento (imagens e textos) e resultados dos testes realizados, garantindo que todo o funcionamento do aplicativo seja independente da internet.

A escolha do *SQLite* foi motivada por três fatores principais:

- autonomia e portabilidade: os dados são mantidos localmente no dispositivo, permitindo uso em qualquer lugar;
- baixo consumo de recursos: ideal para dispositivos móveis com hardware limitado;
- facilidade de integração com *React Native*, por meio de bibliotecas já consolidadas na comunidade.

4.1.2 Desenvolvimento Multiplataforma com *React Native* e *Expo*

Para o desenvolvimento do aplicativo, foi adotado o framework *React Native*, criado pela Meta (Facebook), que permite o desenvolvimento de aplicativos móveis multiplataforma utilizando *JavaScript* e *React*.

Com essa tecnologia, é possível compilar o mesmo código para Android e iOS, mantendo desempenho nativo e reduzindo significativamente o tempo e o custo de desenvolvimento.

O projeto também faz uso do Expo, uma plataforma que simplifica o ciclo de desenvolvimento e distribuição de aplicativos *React Native*, oferecendo um ambiente integrado com ferramentas de build, depuração, simulação e publicação.

O Expo fornece módulos prontos para uso, como *expo-image-picker* (seleção de imagens), *expo-camera* (captura de fotos) e *expo-sqlite* (integração com banco de dados local).

A escolha dessas tecnologias foi motivada por fatores técnicos e práticos:

- multiplataforma nativa, garantindo compatibilidade entre Android e iOS;
- agilidade no desenvolvimento, com *hot reload* e bibliotecas integradas;
- suporte ativo da comunidade e documentação robusta;
- foco em acessibilidade, permitindo que o projeto seja mantido e expandido mesmo por desenvolvedores iniciantes.

4.1.3 Experiência Offline e Usabilidade

Um dos diferenciais do aplicativo é a sua capacidade de operar totalmente offline. Essa característica foi projetada para permitir o uso em ambientes com baixa conectividade — como salas de aula, laboratórios ou regiões rurais —, onde o acesso à internet pode ser limitado.

Além disso, a interface do usuário (UI) foi desenvolvida com foco na simplicidade e na didática, priorizando componentes visuais claros, navegação intuitiva e

feedback constante durante as ações do usuário (como criação, treinamento e teste de modelos).

Essa combinação de tecnologia, acessibilidade e design pedagógico reflete o propósito principal do projeto: tornar a inteligência artificial tangível e compreensível a qualquer pessoa.

4.1.4 Estrutura de Pastas e Organização do Código

O projeto foi estruturado com base em boas práticas de modularização e separação de responsabilidades, conforme ilustrado abaixo:

Quadro 6 – Estrutura de Pastas

```
📂 ai-model-trainer/
 ├ 📂 app/          → Telas e rotas principais do aplicativo
 ├ 📂 components/   → Componentes reutilizáveis (cards, botões, loaders)
 ├ 📂 services/     → Serviços de acesso ao banco de dados e simulações
 ├ 📂 utils/        → Funções auxiliares (validações, formatações)
 ├ 📂 assets/       → Recursos estáticos (ícones, imagens, fontes)
 ├ 📜 App.tsx       → Ponto de entrada principal do aplicativo
 ├ 📜 package.json  → Configuração do projeto e dependências
 └ 📜 tsconfig.json → Configurações de compilação TypeScript
```

Essa organização modular facilita a manutenção, legibilidade e expansão futura do sistema, mantendo um fluxo claro entre interface, lógica e persistência de dados.

4.1.5 Arquitetura do Sistema

A arquitetura do aplicativo segue o padrão de três camadas:

- camada de Apresentação: responsável pela interface gráfica e interação com o usuário;
- camada de Lógica de Negócio: implementa as regras de funcionamento e simulação de treinamento;
- camada de Dados: gerencia o armazenamento e recuperação das informações no banco local *SQLite*.

Essa estrutura garante independência entre as partes do sistema, permitindo manutenção e evolução sem afetar a estabilidade geral do aplicativo.

4.1.6 Considerações sobre a Metodologia

A metodologia adotada priorizou o desenvolvimento iterativo e incremental, com foco em resultados tangíveis e ajustes contínuos. Essa estratégia permitiu testar e validar o sistema à medida que as funcionalidades eram implementadas, garantindo uma entrega final estável, funcional e coerente com os objetivos educacionais e técnicos do projeto.

5 METODOLOGIA DE DESENVOLVIMENTO

O desenvolvimento do aplicativo *AI Model Trainer* seguiu uma abordagem iterativa e incremental, com foco na simplicidade, portabilidade e experiência de uso. Essa metodologia possibilitou a criação de um sistema funcional e didático, que per mite ao usuário compreender o ciclo de aprendizado de máquina sem necessidade de infraestrutura complexa ou dependência de serviços externos.

5.1 ETAPAS DE DESENVOLVIMENTO

O projeto foi dividido em três etapas principais: análise e planejamento, implementação e testes e validação. Cada uma dessas etapas foi essencial para garantir o alinhamento entre os objetivos pedagógicos e as restrições técnicas do projeto.

5.1.1 Análise e Planejamento

Nesta etapa inicial, foram definidos o escopo do sistema, os tipos de usuários, os requisitos funcionais e não funcionais, além da arquitetura base. Também foram avaliadas alternativas tecnológicas para o desenvolvimento multiplataforma e armazenamento local.

Após a análise, optou-se pela combinação de *React Native* com *Expo* no *front-end* e *SQLite* como banco de dados local, garantindo simplicidade, compatibilidade entre sistemas operacionais e funcionamento *offline*.

5.1.2 Implementação

A etapa de implementação compreendeu o desenvolvimento de toda a estrutura do aplicativo, desde a criação das telas até a configuração do banco de dados e lógica de simulação de aprendizado.

Durante o desenvolvimento, o design da interface foi elaborado diretamente no código, com base em princípios de usabilidade e clareza visual, sem necessidade de ferra mentas de prototipação.

Os principais módulos implementados foram:

- gerenciamento de projetos: criação, edição e exclusão de projetos no banco local;
- gerenciamento de dados de treinamento: adição de amostras de texto ou imagem com rótulos definidos pelo usuário;
- simulação de treinamento: atualização progressiva do modelo com cálculo de acurácia simulada;
- testes de modelo: execução de testes com dados novos e exibição de resultados preditivos;
- persistência local: integração completa com o *SQLite*;
- interface e navegação: criação das telas e rotas utilizando *Expo Router* e *React Navigation*.

Essa fase representou o núcleo do trabalho, sendo conduzida de forma incremental, com verificações contínuas e ajustes conforme o funcionamento esperado das telas.

5.1.3 Testes e Validação

Na etapa final, o aplicativo foi testado em dispositivos Android e emuladores, verificando o comportamento das funcionalidades principais e a consistência dos da dos armazenados. Foram realizados testes de fluxo completo — desde a criação de um projeto até a execução de testes — além de simulações offline para confirmar a autonomia do sistema. Os resultados mostraram estabilidade, boa performance e uma experiência de uso intuitiva, condizente com o objetivo educacional do projeto.

6 RESULTADOS E DISCUSSÃO

Após a conclusão das etapas de desenvolvimento e testes, o aplicativo *AI Model Trainer* apresentou resultados satisfatórios em relação aos objetivos estabelecidos no projeto. O sistema se mostrou funcional, estável e didático, atingindo seu propósito de demonstrar, de forma prática e acessível, o funcionamento básico de um processo de treinamento e teste de modelos de aprendizado de máquina.

6.1 RESULTADOS OBTIDOS

Os resultados alcançados podem ser agrupados em três principais dimensões: funcionalidade, usabilidade e desempenho.

6.1.1 Funcionalidade

Todas as funcionalidades previstas nos requisitos funcionais foram implementadas com sucesso.

O aplicativo permite:

- criar e gerenciar projetos de aprendizado;
- adicionar amostras de dados de texto ou imagem;
- simular o treinamento do modelo com feedback de progresso e acurácia;
- realizar testes com novas entradas;
- salvar e recuperar dados localmente no *SQLite*, sem necessidade de internet.

Durante os testes, o sistema manteve comportamento consistente mesmo em cenários de uso intensivo, com múltiplos projetos e diversas amostras cadastradas.

6.1.2 Usabilidade

A interface foi projetada para ser intuitiva e pedagógica, utilizando cores, ícones e mensagens claras para guiar o usuário. A navegação entre as telas é fluida, e o fluxo de ações — criar, treinar, testar — segue uma sequência lógica e natural.

Usuários que participaram de testes informais relataram facilidade de uso e compreensão intuitiva do funcionamento do aplicativo, mesmo sem conhecimento prévio em programação ou inteligência artificial.

6.1.3 Desempenho

O desempenho do sistema foi avaliado a partir do tempo de resposta entre ações e do consumo de recursos locais. Em dispositivos intermediários (Android 10 e superiores), o aplicativo manteve tempo médio de resposta inferior a 500ms entre comandos e atualizações de interface, sem travamentos ou lentidão perceptível.

O uso de *SQLite* e renderização otimizada com componentes *React Native* assegurou leveza, mesmo com grandes volumes de dados armazenados localmente.

6.2 TESTES REALIZADOS

Os testes foram realizados em ambiente local, abrangendo:

Tabela 5 – Testes Realizados

Tipo de Teste	Descrição	Resultado
Teste Funcional	Verificação do correto funcionamento das operações CRUD (criar, ler, atualizar e excluir projetos e amostras).	Todas as operações executadas com sucesso.
Teste de Simulação de Treinamento	Avaliação do cálculo de progresso e acurácia.	Simulações executadas corretamente, com resultados consistentes.
Teste de Persistência de Dados	Fechamento e reabertura do app para verificar armazenamento no *SQLite*.	Dados mantidos sem perda.
Teste Offline	Execução completa do app sem acesso à internet.	Todas as funcionalidades operaram normalmente.
Teste de Usabilidade	Uso do aplicativo por diferentes perfis de usuários (iniciante e técnico).	Relatos de fácil compreensão e fluidez.

Esses testes demonstraram que o aplicativo atende integralmente às necessidades funcionais e não funcionais especificadas na modelagem.

6.3 DISCUSSÃO DOS RESULTADOS

O *AI Model Trainer* provou ser uma ferramenta eficaz para demonstrar, em ambiente controlado e sem dependências externas, os conceitos fundamentais de aprendizado de máquina. A adoção de uma arquitetura local com *SQLite* e de um modelo de simulação, em vez de integração com APIs externas, garantiu independência e estabilidade ao sistema.

A abordagem didática e visual adotada na interface permite que estudantes e curiosos compreendam, de forma interativa, como dados são utilizados em processos de treinamento e teste de modelos de IA. Além disso, o uso de tecnologias como *React Native* e *Expo* viabilizou o desenvolvimento multiplataforma com uma base de código única, simplificando a manutenção e futuras expansões.

6.4 LIMITAÇÕES DO SISTEMA

Apesar dos resultados positivos, algumas limitações foram identificadas:

- ausência de aprendizado real: o treinamento é apenas simulado, não havendo execução de algoritmos de aprendizado de máquina efetivos;
- dependência de armazenamento local: a ausência de sincronização em nuvem impede o uso simultâneo entre dispositivos;
- interface estática: embora funcional, o design pode ser aprimorado para incluir elementos visuais mais interativos, como gráficos de desempenho;
- falta de suporte para grandes volumes de dados: o *SQLite*, apesar de leve, não é ideal para aplicações com alto volume de amostras ou imagens de grande porte.

Mesmo com essas limitações, o sistema cumpre seu papel principal como ferramenta de demonstração conceitual e educacional, sendo uma base sólida para futuras evoluções.

6.5 CONSIDERAÇÕES FINAIS SOBRE OS RESULTADOS

Os resultados obtidos demonstram que o *AI Model Trainer* alcançou seu objetivo de forma satisfatória: um aplicativo leve, multiplataforma, offline e didático, capaz de simular o ciclo básico de aprendizado de máquina de forma compreensível e acessível.

O projeto se consolida como uma prova de conceito educacional, com potencial para evoluir em direção a um ambiente de experimentação prática com modelos reais, mantendo a simplicidade e clareza que o caracterizam.

7 CONSIDERAÇÕES FINAIS

O desenvolvimento do aplicativo *AI Model Trainer* representou um exercício prático de aplicação dos conceitos de engenharia de software, inteligência artificial e desenvolvimento mobile multiplataforma. A proposta surgiu da necessidade de criar uma ferramenta acessível e educacional, capaz de demonstrar o funcionamento do processo de aprendizado de máquina sem depender de infraestrutura complexa, conexões em nuvem ou bibliotecas externas.

Ao longo do projeto, foi possível projetar e implementar uma solução funcional, leve e autônoma, que cumpre o papel de simular o ciclo de vida de um modelo de IA

— desde a coleta de dados até o teste de previsões — em um ambiente completamente offline.

O sistema atende aos objetivos propostos:

- Funciona de maneira autônoma, utilizando *SQLite* como armazenamento local;
- Permite visualizar e compreender a lógica de aprendizado supervisionado de forma didática;
- Opera em múltiplas plataformas, graças à estrutura baseada em *React Native* com *Expo*;
- Mantém foco educacional, favorecendo o aprendizado prático de estudantes e curiosos sobre inteligência artificial.

A arquitetura modular e a metodologia incremental empregada permitiram uma construção progressiva, com validações contínuas e testes iterativos, resultando em um aplicativo estável e coerente com os requisitos definidos.

Além do aspecto técnico, o projeto reforçou a importância de conciliar simplicidade e funcionalidade, demonstrando que conceitos complexos como aprendizado de máquina podem ser explorados de maneira intuitiva, sem necessidade de dependências externas ou processamento em nuvem.

O projeto *AI Model Trainer* demonstrou que é possível aproximar os conceitos de inteligência artificial e ciência de dados do público geral por meio de soluções simples, bem estruturadas e acessíveis. O processo de desenvolvimento proporcionou aprendizado significativo sobre modelagem de sistemas, persistência de dados, design de interface e boas práticas de desenvolvimento mobile.

Mais do que um aplicativo, o trabalho representa um ponto de partida para o desenvolvimento de soluções educacionais que democratizam o entendimento da IA — mostrando que, com criatividade e planejamento, é possível ensinar conceitos complexos de forma leve, interativa e sem depender de grandes estruturas computacionais.

7.1 CONTRIBUIÇÕES DO PROJETO

O *AI Model Trainer* oferece três contribuições principais:

- didática e acessibilidade: o aplicativo traduz conceitos teóricos de IA em uma interface prática e visual, tornando o aprendizado mais tangível;

- independência tecnológica: ao eliminar a necessidade de integrações externas, o projeto funciona em qualquer ambiente móvel compatível com *Expo*;
- base para expansão: a estrutura modular e o uso de tecnologias abertas permitem que o projeto evolua facilmente para versões mais avançadas.

Essas contribuições reforçam o valor do projeto não apenas como produto técnico, mas também como recurso educacional e demonstrativo.

7.2 TRABALHOS FUTUROS

Apesar dos resultados alcançados, o projeto abre espaço para diversas evoluções, tanto técnicas quanto conceituais.

Entre as possibilidades de continuação, destacam-se:

- implementação de aprendizado real: incorporar bibliotecas como *TensorFlow Lite* ou *PyTorch Mobile* para realizar treinamento e inferência reais, mantendo a execução offline;
- sincronização em nuvem: integrar o sistema com serviços como *Firebase* ou *MongoDB Atlas*, permitindo compartilhamento e backup dos projetos;
- visualização de métricas: criar gráficos e dashboards com evolução de acurácia, perda e desempenho;
- modo interativo educacional: incluir tutoriais, desafios e trilhas de aprendizado guiado dentro do aplicativo;
- suporte multiplataforma ampliado: expandir o funcionamento para desktop (via Electron) e web, aproveitando a compatibilidade do *React Native*.

Essas melhorias visam transformar o protótipo conceitual em uma ferramenta robusta de ensino e experimentação prática com aprendizado de máquina.

7.3 DIFICULDADES E APRENDIZADOS

Durante o desenvolvimento do *AI Model Trainer*, diversas dificuldades técnicas e conceituais foram enfrentadas, proporcionando aprendizados relevantes em relação ao uso de tecnologias híbridas e à simulação de processos de inteligência artificial em dispositivos móveis.

A primeira dificuldade consistiu em simular o processo de aprendizado de máquina de maneira coerente e compreensível, sem recorrer a bibliotecas de IA reais. Foi necessário projetar uma lógica que reproduzisse o comportamento esperado de

um modelo em treinamento — apresentando progresso, acurácia e validação — de forma didática e sem processamento intensivo. Essa decisão exigiu a criação de um mecanismo próprio de cálculo de acurácia simulada, ajustado às amostras fornecidas pelo usuário, garantindo uma experiência visualmente fiel a um processo real de aprendizado supervisionado.

Outra dificuldade importante foi o gerenciamento do estado e da persistência dos dados no banco local *SQLite*, dentro do ambiente *React Native*. A integração entre as camadas de interface, lógica e armazenamento demandou atenção à consistência das transações, especialmente ao atualizar ou excluir registros. O desafio maior foi garantir que as alterações realizadas nas telas fossem imediatamente refletidas no banco, sem travamentos ou perdas de dados, mantendo a fluidez da aplicação.

O uso do framework Expo também trouxe limitações pontuais. Embora simplifique a execução multiplataforma, algumas bibliotecas nativas, como as relacionadas à manipulação de arquivos e imagens, apresentaram restrições em versões específicas do Android. Isso exigiu adaptações no código e testes repetidos para assegurar compatibilidade em diferentes dispositivos.

Além disso, a gestão de estado global da aplicação se mostrou um ponto crítico. O uso de *hooks* do *React Native* e funções assíncronas para atualização dos dados requereu um cuidado especial para evitar inconsistências de renderização, principalmente durante a simulação de treinamento, onde múltiplos estados são alterados simultaneamente.

Por fim, uma dificuldade adicional esteve na definição de uma interface didática e intuitiva, capaz de comunicar conceitos técnicos de IA de forma simples. Esse processo reforçou a importância da prototipagem direta no código, testando fluxos com usuários e ajustando textos, cores e feedbacks visuais até atingir clareza e acessibilidade adequadas.

Como aprendizados principais, destacam-se:

- a importância de planejar o ciclo de vida de dados locais em aplicações móveis;
- a necessidade de compreender os limites das ferramentas híbridas e seus impactos na performance;
- e o valor de traduzir processos complexos em experiências simples e educativas.

Essas experiências consolidaram o domínio sobre *React Native*, *Expo* e *SQLite*, além de reforçar a capacidade de transformar conceitos abstratos de inteligência artificial em soluções práticas e acessíveis.

REFERÊNCIAS

Expo Documentation. ***Expo.dev***, 2025. Disponível em: <https://docs.expo.dev>. Acesso em: 9 ago. 2025.

MITCHELL, T. M. **Machine Learning*. New York***: McGraw-Hill, 1997.

PRESSMAN, R. S. **Engenharia de Software: uma abordagem profissional**. 8. ed. Porto Alegre: AMGH, 2016.

React Native Documentation. ***Meta Platforms***, 2025. Disponível em: <https://reactnative.dev>. Acesso em: 9 ago. 2025.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4th. ed. Upper Saddle River: Pearson, 2016.

SQLITE.ORG. *SQLite Documentation.* 2024. Disponível em: <https://www.sqlite.org/docs.html>. Acesso em: 9 ago. 2025.
SOMMERVILLE, I. *Engenharia de Software.* 10. ed. São Paulo: Pearson, 2019.

Termo de abertura do projeto: o que é e como montar o seu. [S. l.], 20 fev. 2020. Disponível em: <https://artia.com/blog/termo-de-abertura-do-projeto>. Acesso em: 5 nov. 2024.

PEDRO, João. **BPMN: o que é, como funciona e como criar na prática.** [S. l.], 22 jul. 2024. Disponível em: <https://www.nomus.com.br/blog-industrial/bpmn>. Acesso em: 5 nov. 2024.

Elicitação de Requisitos: Levantamento de requisitos e técnicas de Elicitação. [S. l.], 2014. Disponível em: <https://www.devmedia.com.br/elicitacao-de-requisitos levantamento-de-requisitos-e-tecnicas-de-elicitacao/31872>. Acesso em: 5 nov. 2024.

Requisitos funcionais e não funcionais: o que são e como identificar? [S. l.], 24 fev. 2023. Disponível em: <https://blog.casadodesenvolvedor.com.br/requisitos funcionais-e-nao-funcionais>. Acesso em: 5 nov. 2024.
em:

Especificação de Casos de Uso: Engenharia de Software 32. [S. l.], 2011. Disponível em <https://www.devmedia.com.br/especificacao-de-casos-de-uso engenharia-de-software-32/19012>. Acesso em: 7 nov. 2024.

APÊNDICE A – TERMO DE ABERTURA DE PROJETO

Curso: Análise e Desenvolvimento de Sistemas – Fatec Franca

Aluno: Rogério de Carvalho Cajá

Título: *AI Model Trainer* – Aplicativo Educacional para Treinamento de Modelos de IA Offline

Orientador(a): Prof. Me. Leonardo Henrique Raiz

CONTEXTO E JUSTIFICATIVA

A aprendizagem prática em Inteligência Artificial é limitada por custos, dependência de infraestrutura e dificuldade de acesso a ferramentas didáticas. O projeto *AI Model Trainer* propõe um aplicativo educacional que simula o processo de treinamento de modelos de IA de forma 100% offline, permitindo a estudantes e professores compreenderem conceitos de machine learning diretamente em dispositivos móveis.

OBJETIVO GERAL

Desenvolver um aplicativo móvel multiplataforma que permita criar, treinar e testar modelos de reconhecimento de padrões de forma totalmente local e offline, utilizando tecnologias acessíveis e de fácil manutenção. O sistema visa democratizar o acesso à experimentação em inteligência artificial, oferecendo uma ferramenta intuitiva, educativa e funcional, que possibilite o aprendizado prático sobre classificação de dados sem a necessidade de infraestrutura em nuvem.

OBJETIVOS ESPECÍFICOS

- implementar um banco de dados local em *SQLite*, responsável por armazenar projetos, amostras de dados e resultados de testes, garantindo persistência mesmo sem conexão com a internet;
- desenvolver uma interface responsiva e intuitiva, utilizando o framework Expo e a biblioteca *React Native*, para oferecer uma experiência fluida tanto em dispositivos *Android* quanto *iOS*;
- criar módulos para diferentes tipos de reconhecimento, como classificação de imagens e classificação de textos, permitindo que o usuário selecione o tipo de projeto conforme sua necessidade;

- simular o processo de treinamento de modelos, apresentando ao usuário o progresso do treinamento, métricas de acurácia e status visual do modelo;
- disponibilizar recursos de teste dos modelos treinados, permitindo inserir novas amostras e visualizar as previsões simuladas;
- garantir o funcionamento totalmente offline, evitando dependência de APIs externas, conexões de rede ou serviços em nuvem;
- fornecer um ambiente didático, que auxilie estudantes e entusiastas na compreensão dos princípios de aprendizado de máquina e reconhecimento de padrões;
- aplicar boas práticas de desenvolvimento, com separação de camadas, reutilização de componentes e código documentado, visando manutenibilidade e expansão futura.

ESCOPO

O projeto contempla:

- Modelagem de dados e arquitetura local;
- Implementação funcional do aplicativo;
- Documentação técnica e apresentação acadêmica.

Não contempla: integrações com serviços em nuvem ou APIs externas.

STAKEHOLDERS PRINCIPAIS

Papel	Responsável
Aluno Desenvolvedor	Rogério Cajá
Orientador(a)	Prof. Me. Leonardo Henrique Raiz
Usuários Finais	Estudantes e professores de TI
Instituição	Fatec Franca

PRINCIPAIS RISCOS

- Falhas de compatibilidade entre versões de bibliotecas;
- Desempenho limitado em dispositivos de baixo custo;
- Atrasos no cronograma por sobrecarga acadêmica.

CRONOGRAMA RESUMIDO

Etapa	Período
Planejamento e estudo	Ago/2025
Modelagem e estruturação	Set/2025
Desenvolvimento e testes	Out/2025
Documentação e entrega final	Nov/2025

APROVAÇÃO

Aluno: Rogério de Carvalho Cajá

Orientador(a): Prof. Me. Leonardo Henrique Raiz

Coordenador(a): Prof. Me. Leonardo Henrique Raiz

Data: 10/11/2025