Link Video: https://youtu.be/-Mz7iqoxxyc

Link GitHub: https://github.com/FelipeLivino/GlobalSolution_fase4_fiap/tree/master

Nome do Projeto: Global Solution (GS1) - FIAP

Nome do Grupo: Rumo ao NEXT

Integrantes:

- Felipe Livino dos Santos (RM 563187)
- Daniel Veiga Rodrigues de Faria (RM 561410)
- Tomas Haru Sakugawa Becker (RM 564147)
- Daniel Tavares de Lima Freitas (RM 562625)
- Gabriel Konno Carrozza (RM 564468)

---

1. Introdução

Em um cenário agrícola cada vez mais suscetível a eventos adversos, como incêndios e variações climáticas extremas, torna-se crucial o desenvolvimento de soluções tecnológicas eficientes para o monitoramento e a prevenção de riscos. O presente projeto, propõe um sistema integrado de monitoramento de risco agrícola, com o objetivo de prevenir e detectar incidentes de forma proativa. A solução inova ao combinar três pilares tecnológicos fundamentais: **visão computacional**, capaz de identificar focos de incêndio em estágios iniciais; **telemetria de sensores**, para o acompanhamento contínuo de condições ambientais críticas como temperatura e fumaça; e um robusto sistema de **armazenamento e processamento de dados**.

Para concretizar essa proposta, foram desenvolvidos e treinados **dois modelos distintos de Inteligência Artificial**. O primeiro, baseado na arquitetura **ResNet18**, é dedicado à análise de imagens para a detecção precoce de incêndios. O segundo modelo, um classificador **Random Forest**, foi treinado para interpretar dados provenientes de sensores ambientais, avaliando o nível de risco com base em parâmetros como temperatura e concentração de fumaça. A comunicação e o processamento dos dados são orquestrados por um conjunto de APIs RESTful. Uma API principal, desenvolvida em **FastAPI (Python)**, é responsável por receber e processar os dados, utilizando os modelos de IA para gerar alertas em tempo real sobre riscos de incêndio e níveis de perigo detectados pelos sensores. Uma segunda API, construída em **Node.js (Express)**, atua como uma ponte para receber os dados simulados dos sensores através da plataforma **Wokwi**, representando o componente de hardware embarcado do sistema. Todos os dados coletados e processados são **integrados e persistidos em um banco de dados MySQL**, permitindo a análise histórica e a tomada de decisões mais informadas. A Figura 1 ilustra a arquitetura geral da solução proposta.

2. Desenvolvimento

2.1 Arquitetura da Solução

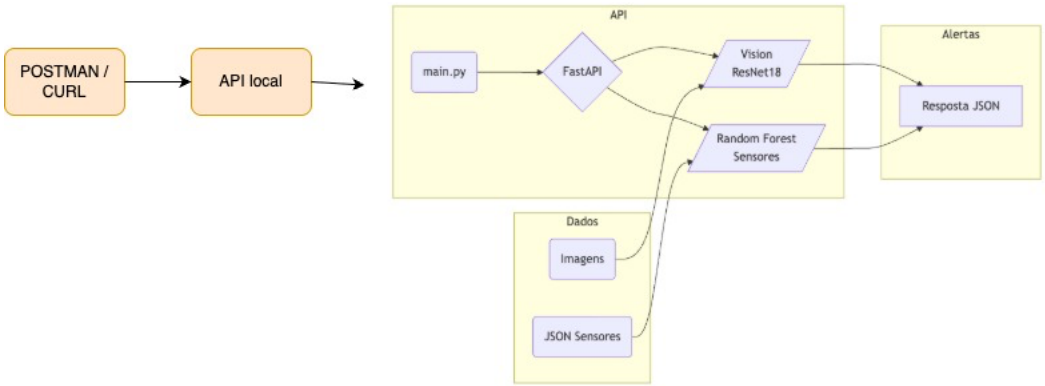
A arquitetura do projeto é dividida em três frentes principais:

- **Visão Computacional:** Utiliza um modelo de rede neural **ResNet18**, que foi treinado com imagens rotuladas para classificar uma área como "Incêndio" ou "Sem incêndio".
- **Telemetria de Sensores (ESP32):** Emprega um classificador **Random Forest** para analisar dados de sensores (temperatura e fumaça). O modelo opera sobre features normalizadas por meio de StandardScaler e classifica o risco em quatro níveis: `NORMAL`, `ATENCAO`, `ALTO` ou `ALERTA MAXIMO`.
- **APIs e Banco de Dados:**
    - Uma **API em Python (FastAPI)** é responsável por receber os dados, processá-los com os modelos de Machine Learning e retornar as predições. Os modelos (`.pth` para imagens e `.joblib` para sensores) são carregados na inicialização da API pelo script `src/model/main_model.py`.
    - Uma **API em Node.js (Express)** foi desenvolvida para receber os dados enviados pelo simulador Wokwi.
    - Um banco de dados **MySQL** é utilizado para armazenar as informações dos sensores e suas respectivas geolocalizações.

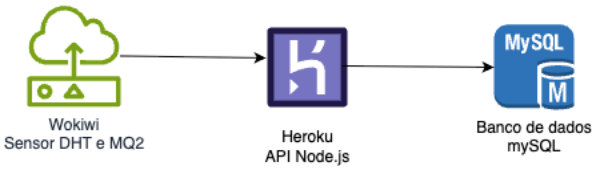
No Caderno do jupyter, contém outros treinamentos de modelos, porém, partimos do ponto de escolher os modelos com maior acurácia.

Os modelos foram treinados localmente, pois o Google Colab free, não comportou o treinamento das imagens. Adicionamos a opção do modelo ser executado em diversos devices como CUDA (Placas de videio da NVIDIA), MPS (disponível em dispositivos Apple) e CPU.

Abaixo o gráfico de arquitetura da aplicação que faz o uso do modelo de IA:



Abaixo o modelo de conexão dos sensores com a API no servidor Heroku:



- Os dados gerados pelo wokwi, foram usados para o treinamento do modelo de dados dos sensores

2.2 Justificativas

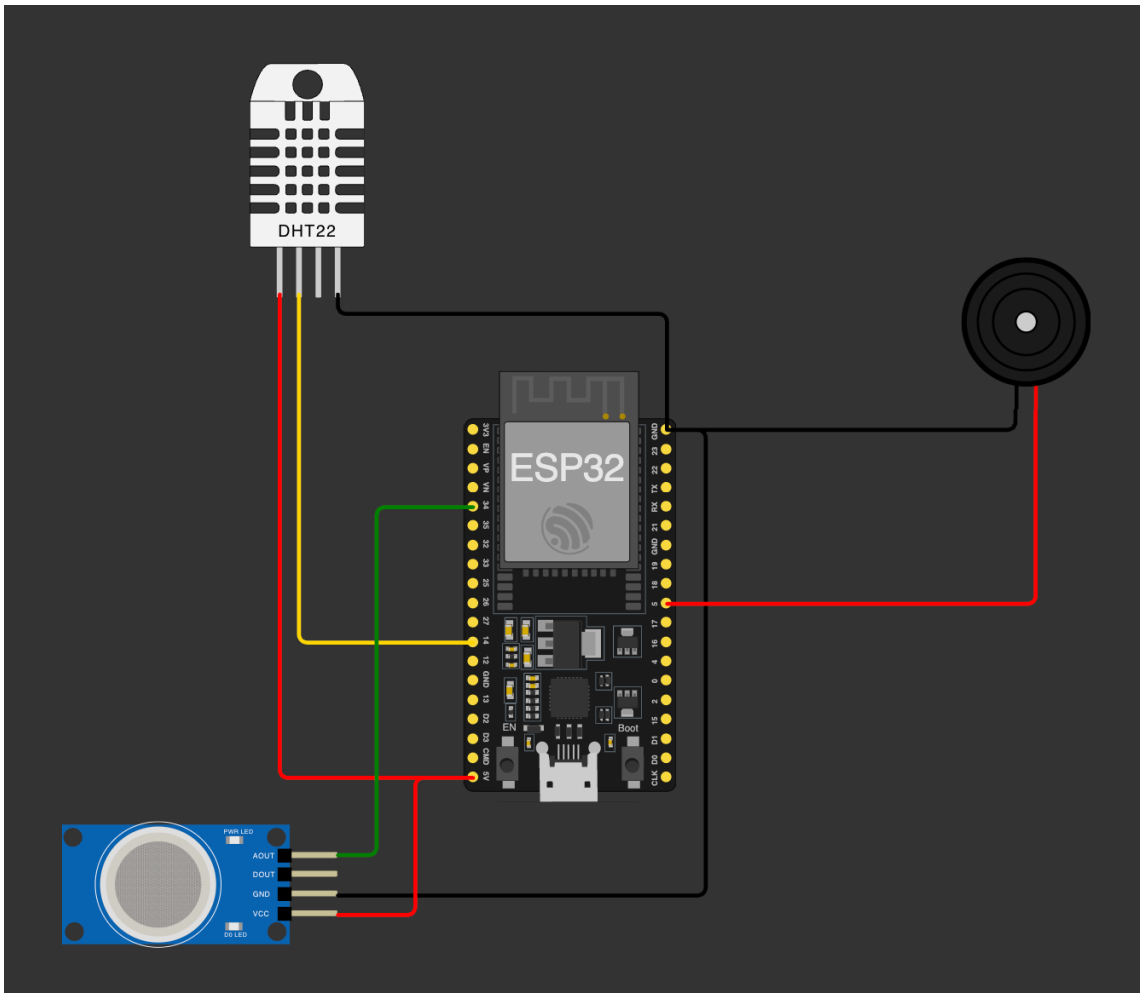
A escolha das tecnologias buscou eficiência e precisão para o monitoramento agrícola:

- **ResNet18:** É um modelo de visão computacional reconhecido por seu bom equilíbrio entre profundidade (e, portanto, capacidade de aprendizado) e eficiência computacional, sendo adequado para tarefas de classificação de imagens como a detecção de incêndios.
- **Random Forest:** É um algoritmo robusto e eficaz para tarefas de classificação com dados tabulares, como as leituras de sensores, sendo menos propenso a overfitting e fornecendo boa performance com features normalizadas.
- **FastAPI e Node.js:** O uso de FastAPI para a API de predição aproveita sua alta performance e a documentação automática (`/docs`), enquanto o Node.js é eficaz para receber e processar rapidamente os dados de I/O-bound do simulador Wokwi.
- **Wokwi:** Permite a simulação de hardware (ESP32 e sensores) sem a necessidade de componentes físicos, facilitando o desenvolvimento e os testes do sistema embarcado.

2.3 Circuitos

O hardware do projeto foi simulado na plataforma Wokwi utilizando um microcontrolador ESP32 conectado a dois sensores principais para a coleta de dados ambientais.

- **Componentes do Circuito:**
  - **Sensor DHT22:** Conectado ao pino 14 do ESP32, mede temperatura e umidade. Requer um resistor de pull-up de 10 kΩ entre os pinos DATA e 5V.
  - **Sensor de Gás/Fumaça MQ2:** Conectado ao pino 34 (um canal de conversor analógico-digital), simula a detecção de fumaça. Para garantir leituras precisas, o simulador aguarda 10 segundos para o aquecimento do sensor.
- **Arquitetura do Circuito:**



2.4 Códigos e Funcionalidades

A solução é composta por um simulador de circuito e duas APIs principais, com funcionalidades para predição e recebimento de dados.

- **Simulador (Wokwi):**
  - Coleta dados de temperatura e fumaça dos sensores DHT22 e MQ2 a cada 4 segundos.
  - Formata os dados coletados em um payload JSON.
  - Envia esses dados via HTTP POST para uma API externa (a aplicação em Node.js hospedada no Heroku).
- **API de Predição (FastAPI):**
  - **Endpoint `/prediction/img`:**
    - **Método:** POST
    - **Função:** Recebe o upload de uma imagem no formato JPEG (`Form-Data`).
    - **Resposta:** Retorna a classificação da imagem, como `{"classe": "INCENDIO", "prob": 0.98}`.
  - **Endpoint `/prediction/sensor`:**
    - **Método:** POST
    - **Função:** Recebe um lote de leituras de sensores em formato JSON (ex: `{"dados": [[27.5, 1], [30.2, 0]]}`).

        - **Resposta:** Retorna o nível de risco agregado, como `{"risco": "ALTO"}`.
        - **OBS:** Essa API pode receber diversos dados de momentos diferentes ou de sensores diferentes que ira retornar o conjunto de dados, evitando o uso excessivo da API
    - **Documentação:** A API possui documentação interativa gerada automaticamente e disponível nos endpoints `/docs` e `/redoc`.
- **API de Recebimento (Node.js):**
    - **Função:** Receber os dados enviados pelo simulador Wokwi.
    - **Hospedagem:** O guia de instalação detalha o processo de deploy da aplicação no Heroku, incluindo a configuração do banco de dados JawsDB MySQL e o uso da Heroku CLI para o deploy via Git.

2.5 Caderno de treinamento

- O repositório no GitHub contém notebooks com os treinamentos completos dos modelos de imagem e sensores.

3. Resultados Esperados

O sistema foi projetado para entregar os seguintes resultados:

- **Detecção de Incêndio:** Classificar corretamente imagens de plantações, identificando a presença ou ausência de incêndios com um grau de probabilidade. O resultado esperado para uma imagem com fogo é um JSON contendo a classe "INCENDIO".
- **Análise de Risco Ambiental:** Processar dados contínuos de sensores de temperatura e fumaça para gerar um nível de risco em tempo real (`NORMAL`, `ATENCAO`, `ALTO`, `ALERTA MAXIMO`), permitindo ações preventivas.
- **Comunicação Integrada:** Garantir que o hardware simulado (ESP32) colete e envie os dados com sucesso para a API em Node.js , que por sua vez os armazena no banco de dados.
- **Acessibilidade e Testes:** Fornecer endpoints de API funcionais e documentados para que os resultados possam ser consultados de forma programática. A disponibilidade de testes via Postman e cURL garante a verificação do funcionamento.

4. Conclusões

O documento detalha a construção de uma solução completa e multifacetada para o monitoramento de riscos agrícolas. A combinação de visão computacional, telemetria de sensores e uma arquitetura de APIs robustas demonstra uma abordagem moderna e eficaz para resolver um problema prático e de grande relevância. A documentação, a simulação e a integração entre os componentes demonstram maturidade técnica e viabilidade de implementação real.