HydroGuard: Sistema Inteligente de Monitoramento e Alerta de Enchentes em Rios

> Este projeto faz parte do curso de **Inteligência Artificial** da FIAP - Online 2024. Este repositório é a **Global Solution - Protech the Future**.

- **Repositório do Projeto:** GitHub

Objetivo do Projeto

O **HydroGuard** é uma solução digital inovadora desenvolvida para enfrentar o crescente desafio das enchentes em rios, um dos eventos naturais extremos mais impactantes no Brasil e no mundo. Nosso principal objetivo é criar um sistema capaz de **prever, monitorar e mitigar os impactos de enchentes**, fornecendo alertas precoces para comunidades ribeirinhas e autoridades. Utilizando dados reais e as mais recentes tecnologias de Inteligência Artificial e IoT, buscamos transformar a resposta a desastres, tornando-a mais rápida, inteligente e eficaz.

Nossa Escolha: Por Que HydroGuard?

A escolha do **HydroGuard** foi baseada em uma análise criteriosa das necessidades do desafio Global Solution 2025.1 e das capacidades da nossa equipe. Optamos por esta proposta pelas seguintes razões fundamentais:

- **Impacto Direto e Tangível:** Enchentes fluviais afetam milhões, causam perdas de vidas e enormes prejuízos. Uma previsão e alerta eficazes salvam vidas e bens, o que confere ao projeto uma relevância social imediata e inegável.
- **Alinhamento Perfeito com Requisitos Técnicos:**
  - **Sensores ESP32 e IoT:** O monitoramento do nível do rio e da precipitação via sensores é a espinha dorsal do sistema, permitindo uma integração **natural e essencial** do ESP32, cumprindo um dos requisitos obrigatórios do projeto de forma orgânica.
  - **Machine Learning/Redes Neurais em Python:** A previsão de enchentes é um problema clássico de séries temporais, ideal para aplicação de modelos de Machine Learning (como Regressão, Random Forest ou até Redes Neurais Recorrentes/Convolucionais), garantindo o cumprimento do outro requisito obrigatório do projeto.
- **Disponibilidade e Facilidade de Dados:** Há uma vasta quantidade de dados hidrológicos e meteorológicos gratuitos e públicos no Brasil (ANA, SGB, CEMADEN), estruturados em séries temporais, o que facilita enormemente a coleta, limpeza e preparação para o treinamento do modelo de ML, otimizando nosso tempo de desenvolvimento.
- **Escopo Viável para MVP:** Ao focar no monitoramento e alerta (e não no controle de fluxo, por exemplo), conseguimos delimitar um MVP robusto e funcional que pode ser entregue dentro do prazo apertado, mantendo a complexidade gerenciável.
- **Foco em Dados Reais:** Mesmo com a flexibilização do `disasterscharter.org`, o HydroGuard se baseia em dados reais de rios e chuvas, mantendo a essência do desafio.

Escopo do MVP (Produto Mínimo Viável)

Para esta fase da Global Solution, o **HydroGuard** será apresentado como uma Prova de Conceito (PoC) funcional com as seguintes características:

1. **Monitoramento Simulado de Sensores:**
    - Utilização de um **ESP32 simulado no Wokwi** para representar sensores de nível de água (ultrassônico) e pluviômetro (chuva acumulada).
    - O ESP32 enviará dados simulados (nível do rio e precipitação) para um script Python via comunicação serial.
2. **Coleta e Pré-processamento de Dados Históricos:**
    - Dados reais de nível do rio e precipitação para o **Rio Meia Ponte (Goiás)** serão coletados de fontes como o sistema HidroWeb da ANA.
    - Esses dados serão limpos, integrados e utilizados para a engenharia de features, preparando-os para o treinamento do modelo de Machine Learning.
3. **Modelo de Previsão de Enchentes (ML em Python):**
    - Desenvolvimento de um modelo de Machine Learning (ex: Random Forest Regressor ou Rede Neural Densa simples) em Python.
    - O modelo será treinado com os dados históricos para prever o nível do rio em um futuro próximo (ex: próximas 1-2 horas).
4. **Sistema de Alerta Precoce:**
    - O script Python receberá os dados "em tempo real" do ESP32 simulado.
    - Utilizará o modelo treinado para prever o nível do rio.
    - Se o nível previsto ultrapassar um limiar de "enchente iminente", o sistema emitirá um alerta visual (ex: LED no ESP32 simulado) e uma mensagem no console.
5. **Documentação e Demonstração:**
    - Entrega de um PDF detalhado sobre o projeto.
    - Vídeo de demonstração prática mostrando a interação do ESP32 simulado com o sistema Python e o acionamento do alerta.

Arquitetura do Projeto

- **Machine Learning**
    - Carregar dados históricos reais (Ana HidroWeb) para treinar o modelo de previsão de enchentes.
    - Treinar modelo de previsão do nível máximo do rio do dia seguinte.
    - Salvar modelo treinado.
- **Banco de Dados**
    - Armazenar dados dos estações de monitoramento, trechos de rio, tipos de estações, etc.
    - Armazenar dados dos sensores, tipo de sensores, etc.
    - Armazenar dados do modelo treinado e métricas do modelo treinado.
    - Armazenar dados de previsão de enchentes.
    - Armazenar dados dos alertas.
- **Programa 1 - Collector Sender**: Coleta de Dados (ESP32) e Envio via MQTT

    - Medir nível do rio e precipitação.
    - Enviar dados via MQTT para o Programa 2.

- **Programa 2 - Listener Saver**: Recepção via MQTT e Armazenamento de dados no PostgreSQL
    - Receber dados do ESP32 via MQTT.
    - Salvar dados em banco de dados.

- **Programa 3 - Predictor Notifier**: Previsão e Alerta (executado regularmente)
    - Carregar dados do banco de dados.
    - Utilizar modelo treinado para prever o nível máximo do rio no dia seguinte.
    - Enviar alerta por e-mail se a previsão exceder X metros.

- **Programa 4 - Simple Alert Logger**: Recebimento de alertas via webhook e Exibição no console (demonstrativo)
    - Receber alertas via webhook.
    - Exibir alertas no console.

- **Programa 5 - Interactive Dashboard**: Dashboard Streamlit
    - Visualização da última previsão registrada (nível do rio e risco previsto).
    - Exibição dos últimos alertas gerados, com data, severidade e mensagem.
    - Tabela interativa com histórico completo de alertas, acessível via painel expansível.
    - Desenvolvido com Streamlit para visualização em tempo real dos dados salvos no banco.

Como Rodar o Projeto (MVP)

Requisitos:

- Git
- Python 3+ e pip
- Docker e Docker Compose

Passo a Passo:

1. **Clone o Repositório:**

```
git clone https://github.com/brunoconterato/gs-disaster.git
cd gs-disaster
```

2. **Ative o ambiente virtual:**

```
source .venv/bin/activate # linux
.venv/Scripts/activate # windows
```

3. **Instale o PyTorch e o Pytorch Lightning:**

```
python -m pip install lightning
```

Nota: O Pytorch Lightning já instala o Pytorch. Qualquer problema, reinstale o PyTorch manualmente conforme as instruções do site oficial: PyTorch.

4. **Instale as Dependências Python:**

```
pip install -r requirements.txt
```

5. **Crie o arquivo .env:**

```
cp .env.example .env
```

6. **Adicione as variáveis de ambiente:**

```
POSTGRES_HOST="localhost"
POSTGRES_PORT="5432"
POSTGRES_USER="user"
POSTGRES_PASSWORD="password"
POSTGRES_DB="hydroguard"

ALERT_LOGGER_WEBHOOK_HOST="0.0.0.0"
ALERT_LOGGER_WEBHOOK_PORT="8000"

MQTT_BROKER="your-mqtt-broker.com"
MQTT_PORT="8883"
MQTT_USER="your-mqtt-user"
MQTT_PASSWORD="your-mqtt-password"
```

7. **Execute o Docker Compose para iniciar o banco de dados:**

```
docker-compose up -d
```

8. **(Primeira vez) Execute o script para inicializar o banco de dados:**

8.1 Execute o script de inicialização para criar as tabelas: `bash python db/scripts/init_db.py`

```
Isso criará todas as tabelas conforme definidas em `models.py`.
```

8.2 **Popule o banco com dados de exemplo** Execute:

```
```bash
python db/scripts/populate_db.py
```

Esse script insere um rio, um trecho, tipos de estação e sensor, três estações de monitoramento e nove sensores, conforme os dados reais do Rio Meia Ponte (Goiás).
```

9. **Execute a Simulação do ESP32 no Wokwi:**
    - Abra o projeto ESP32 no Wokwi: Wokwi Project
    - Inicie a simulação (play button).
    - Manipule os sliders para simular o nível da água e a precipitação.
10. **Execute o Listener Saver:**

```
python listener_saver/main.py
```

Esse programa ficará ouvindo os dados do ESP32 e salvando no banco de dados.

11. **Execute o Simple Alert Logger para ouvir os alertas:**

```
python simple_alert_logger/main.py
```

Esse problema ficará ouvindo os alertas e printando no console.

12. **Ative o cronjob para executar o Predictor Notifier (regularmente):**

Para ativar o preditor e notificador de alertas, adicione o cronjob:

```
./predictor_notifier/cron-manager.sh add minutely # a cada 1 minuto (demonstração)
./predictor_notifier/cron-manager.sh add daily # diariamente às 00:00
```

Para esquecer o cronjob, execute:

```
./predictor_notifier/cron-manager.sh remove minutely # remove o cronjob de cada minuto (demonstração)
./predictor_notifier/cron-manager.sh remove daily # remove o cronjob de diariamente às 00:00
```

13. **Abra o Dashboard Interativo:**

    Com o banco populado e os alertas sendo gerados, execute:

    ```
    streamlit run dashboard/app.py
    ```

    Isso abrirá o dashboard no navegador em:

    ```
    http://localhost:8501
    ```

    O painel exibirá:

    - A última previsão de enchente registrada
    - Os alertas gerados (recentes e histórico)
    - Tabela interativa em `st.expander` com todos os dados dos alertas

    Para testar a funcionalidade, você pode disparar um alerta simulado com:

    ```
    python predictor_notifier/simulate_alert.py
    ```

Documentação

- Hardware: Collector Sender
- Software: Listener Saver
- Banco de Dados
    - Modelo de Dados

Próximos Passos e Melhorias Futuras

O MVP do HydroGuard é um ponto de partida. Para futuras iterações, pretendemos explorar:

- **Modelos de ML Mais Avançados:** Implementação de Redes Neurais Recorrentes (RNNs/LSTMs) para aprimorar a previsão de séries temporais, inspiradas na tese de referência.
- **Computação em Nuvem:** Deploy do sistema de monitoramento e ML em plataformas de nuvem para escalabilidade.
- **Alerta Multi-canal:** Criação de um webhook para receber alertas e enviar via SMS ou e-mail para autoridades e população.
- **Validação com Dados Reais:** Testes em cenários reais com estações de monitoramento.
- **Dashboards Interativos:** Desenvolvimento de uma interface gráfica para visualização em tempo real e configuração de alertas.

Tecnologias Utilizadas

Categoria	Ferramentas
Linguagem	Python 3.9+
Manipulação de Dados	Pandas, NumPy
Visualização	Matplotlib
Aprendizado Profundo	PyTorch
Pré-processamento	Scikit-learn (StandardScaler)
Ambiente	Jupyter Notebook, CUDA (GPU)
IoT/Hardware	ESP32, Wokwi (simulação)
Comunicação	PySerial
Banco de Dados	PostgreSQL
Containerização	Docker, Docker Compose
Webhook	FastAPI, Uvicorn

Equipe

Membros (Grupo 46)

- Amandha Nery (RM560030)
- Bruno Conterato (RM561048)
- Gustavo Castro (RM560831)
- Kild Fernandes (RM560615)
- Luis Emidio (RM559976)

Professores

- Tutor: Leonardo Ruiz Orabona
- Coordenador: André Godoi

Desenvolvido com paixão e inteligência para um futuro mais seguro.