Projet : Plateforme d'Intelligent Météo & Agricole

Composer des équipes de 3 personnes pour proposer des solutions, mettre à dispo le code source capture solution finale et des cvs, quizz LMS, et choisir un créneau de prez en ligne pour les solutions les mieux

Contexte et Problème

L'agriculture en Afrique (et ailleurs) est fortement dépendante de la météo.
Or, les agriculteurs font souvent face à :

- **Imprévisibilité des pluies** → irrigation mal planifiée.
- **Sécheresses et chaleurs extrêmes** → pertes de récoltes.
- **Maladies liées au climat** (champignons, parasites).
- **Manque d'informations fiables en temps réel** → décisions basées sur l'intuition plutôt que sur des données.

Les solutions existantes sont souvent :

- soit trop générales (prévisions météo pour toute une ville, pas adaptées au champ),
- soit peu accessibles (technologie complexe, langues étrangères, besoin d'Internet stable).

Objectif du Projet

Créer une **plateforme intelligente d'aide à la décision agricole** qui combine :

1. **Prévisions météo locales** adaptées aux champs agricoles.
2. **Modèles prédictifs intelligents** (pluie, sécheresse, irrigation, maladies).
3. **Application simple et accessible** (mobile/web + SMS/WhatsApp).

Le but : **aider les agriculteurs à mieux planifier semis, arrosage et récoltes pour améliorer rendement et réduire les pertes.**

Architecture du Projet (3 grandes briques)

1. Data Engineering – Pipeline de Données (ETL)

Explication simple :
C'est le "cœur" qui **récupère les données, les nettoie et les stocke**.

Sources de données :

- **Météo (temps réel et prévisions)** → API OpenWeather One Call 3.0.
- **Agriculture mondiale** → FAO (FAOSTAT), Copernicus (humidité des sols, végétation).
- **Agriculture locale** → données des coopératives, instituts de recherche (ISRA, CIRAD), capteurs IoT si disponibles.

Processus ETL :

- **E**xtract → récupérer données météo/agro (APIs, fichiers).
- **T**ransform → uniformiser unités (°C, mm pluie, % humidité), enrichir avec cultures locales.
- **L**oad → stocker dans une base PostgreSQL/TimescaleDB.

Outils : Airflow (automatisation), Python (nettoyage), PostgreSQL (stockage).

2. Data Science & MLOps – Modèles de Prédiction

Explication simple :
On utilise les données pour **créer des modèles capables de prédire et alerter**.

Cas d'usage pour agriculteurs :

- Prédire la pluie → éviter d'arroser pour rien.
- Anticiper sécheresse → planifier irrigation.
- Détecter risque de maladies (ex. chaleur + humidité = champignons).
- Estimer besoins en eau → recommandations personnalisées.

Comment ça marche ?

- Les données météo et agricoles sont transformées en **indicateurs (features)**.
- Les modèles ML apprennent des **tendances passées** pour prédire l'avenir.

- Exemple : si sol = sec + météo annonce chaleur → modèle prédit besoin d'irrigation.

Outils : scikit-learn, XGBoost, Prophet (séries temporelles), MLflow (gestion des modèles), FastAPI (déploiement API), Docker (conteneurs).

3. Développement logiciel – Application interactive, UX UI Design

Explication simple :
C'est **l'interface visible par l'agriculteur**.

Fonctionnalités :

- Tableau de bord météo agricole (température, humidité, pluie).
- Graphiques simples → évolution pluie, irrigation recommandée.
- Notifications SMS/WhatsApp → alertes météo (pluie, sécheresse).
- Conseils agricoles personnalisés selon type de culture.
- Interface multilingue (français + langues locales).

Outils :

- Backend → FastAPI.
- Frontend → React (web) ou Flutter (mobile).
- Visualisation → Plotly.
- Notifications → Twilio (SMS/WhatsApp).

Architecture technique simplifiée

Sources données OpenWeather,

1. ETL : Airflow + PostgreSQL

2. ML Models : scikit-learn

3. API : FastAPI + Docker

4. Application | Alertes SMS | (Web/Mobile) | WhatsApp/SMS

Plan de mise en œuvre (Milestones)

1. **Phase 1 – Collecte et intégration données (2 jours)**
    a. Connexion API OpenWeather.
    b. Intégration FAO + Copernicus.
    c. Stockage PostgreSQL.
2. **Phase 2 – Prototypage modèles (2 jours)**
    a. Prédiction pluie/sécheresse.
    b. Détection risques maladies.
    c. Validation sur données locales.
3. **Phase 3 – Développement app (5 jours)**
    a. API backend (FastAPI).
    b. Interface web/mobile (React ou Flutter).
    c. Visualisation des prédictions.
4. **Phase 4 – Déploiement & MLOps (3 jours)**
    a. Dockerisation des modèles.
    b. Mise en place CI/CD (GitHub Actions).
    c. Monitoring (Grafana).
5. **Phase 5 – Test terrain pilote (3 mois)**
    a. Collaboration avec coopérative locale.
    b. Collecte retours utilisateurs.
    c. Ajustement modèles/app.

Impact attendu

- Réduction de la consommation d'eau (meilleure irrigation).
- Augmentation des rendements agricoles (semis optimisés).
- Réduction des pertes dues aux aléas climatiques (alertes précoces).
- Accessibilité → même sans Internet stable grâce aux SMS.
- Contribution au développement durable et à la sécurité alimentaire.

Résumé simple :
Ce projet crée une **chaîne complète** → **données météo + agricoles** → **modèles intelligents** → **application accessible** → **impact concret sur les pratiques agricoles**.

Instructions générales

Livrables attendus (dépôt GitHub obligatoire)

Chaque équipe doit créer un **repository GitHub** dès le premier jour et y déposer tous les livrables :

1. **Code source complet** avec un README.md expliquant :
    a. l'architecture,
    b. l'installation,
    c. les instructions de lancement.
2. **Captures d'écran ou vidéo démo** (dossier /docs).
3. **CV des membres** (dossier /team).
4. **Résultats du quiz LMS** (preuve de validation individuelle, dossier /lms).
5. **Présentation finale (slides ou PDF)** (dossier /presentation).

⚠ **Aucun livrable ne sera accepté en dehors de GitHub.**

Le lien du repo devra être communiqué au jury avant la soutenance.

Déroulement du projet

Durée totale : **2 semaines**

Phase 1 – Semaine 1

- **Mise en place technique** :
    - Pipeline ETL (collecte + intégration des données).
    - Prototype de modèles prédictifs (pluie, sécheresse, maladies).
- **Suivi avec mentor (Jour 7)** :
    - Rendez-vous obligatoire en ligne via **Google Meet**.
    - Des **boucles/salles (breakout rooms)** seront créées pour accueillir chaque équipe.
    - Le mentor passera dans chaque salle (20-30 minutes/équipe) pour :
        - évaluer l'avancement,
        - répondre aux questions,
        - donner des recommandations.

Phase 2 – Semaine 2

- Développement de l’application (backend + interface).
- Dockerisation et mise en place CI/CD.
- Finalisation du repo GitHub (code, docs, démo).
- Préparation de la présentation finale.

Phase finale – Jour 14

- **Dépôt GitHub final obligatoire.**
- Présentation en ligne (15 min de pitch + 5 min Q&A).

Accompagnement

- Chaque équipe est suivie par un **mentor DataBeez**.
- Les questions sont posées via le meeting (ou communauté).
- Un **feedback intermédiaire** est donné à mi-parcours (Jour 7).

Nous vous remercions pour votre intérêt et sommes impatients de découvrir vos réalisations.

Bien cordialement,
L'équipe de recrutement Hack2Hire