

SmartIrrig - Système Intelligent de Gestion d'Irrigation par Simulation IoT

Sous-titre : Plateforme de simulation et optimisation d'irrigation agricole basée sur l'IA

Table des Matières

1. [Présentation du Projet](#)
 2. [Objectifs Pédagogiques](#)
 3. [Spécifications Fonctionnelles](#)
 4. [Architecture Technique](#)
 5. [Intelligence Artificielle & Machine Learning](#)
 6. [Planning Détailé \(12 Semaines\)](#)
 7. [Organisation de l'Équipe](#)
 8. [Stack Technique](#)
 9. [Livrables](#)
 10. [Critères d'Évaluation](#)
-

1. Présentation du Projet {#présentation}

Contexte

L'agriculture consomme environ **70% de l'eau douce mondiale**. Une irrigation inefficace entraîne :

- Gaspillage d'eau (30-60% de pertes)
- Coûts énergétiques élevés
- Stress hydrique des cultures
- Impact environnemental négatif
- Baisse de rendement

SmartIrrig propose une solution intelligente de gestion d'irrigation basée sur l'IoT et l'IA pour optimiser la consommation d'eau.

Vision

Développer une plateforme web de **simulation virtuelle** d'un système d'irrigation intelligent qui :

- Simule des capteurs IoT en temps réel
- Utilise des règles avancées multi-facteurs

- Applique du Machine Learning pour prédire les besoins en eau
- Optimise automatiquement l'irrigation
- Fournit des visualisations et analyses détaillées

Public Cible

- **Agriculteurs** : Gestion et surveillance des parcelles
- **Agronomes** : Analyse et optimisation
- **Gestionnaires** : Supervision multi-sites
- **Chercheurs** : Tests de stratégies d'irrigation

Type de Projet

Simulation 100% virtuelle :

- Aucun matériel IoT physique requis
 - Capteurs simulés algorithmiquement
 - Données générées de manière réaliste
 - Environnement web accessible partout
 - Focus sur l'intelligence logicielle
-

🎓 2. Objectifs Pédagogiques {#objectifs}

Compétences Techniques

IoT & Systèmes Embarqués (Virtuel)

- Compréhension des protocoles IoT (MQTT, WebSocket)
- Simulation de capteurs et actionneurs
- Gestion de flux de données temps réel
- Architecture pub/sub

Intelligence Artificielle & Machine Learning

- Algorithmes de prédiction (régression)
- Apprentissage supervisé
- Feature engineering
- Modèles de séries temporelles
- TensorFlow.js ou scikit-learn

Data Science

- Analyse de données temporelles
- Visualisations interactives (graphiques temps réel)
- Statistiques descriptives et prédictives
- Détection d'anomalies

Développement Full Stack

- Architecture temps réel (WebSocket)
- Dashboards interactifs
- Base de données time-series
- API REST + Real-time

Compétences Transversales

- Modélisation de systèmes complexes
- Pensée algorithmique
- Optimisation multi-critères
- Gestion de projet IoT
- Documentation technique

Compétences Métier

- Agronomie de base (besoins en eau des cultures)
- Optimisation des ressources
- Gestion environnementale
- Décision data-driven

3. Spécifications Fonctionnelles {#specifications}

3.1 Capteurs Virtuels Simulés

Capteur d'Humidité du Sol (Priorité 1)

- Type : Hygromètre capacitif
- Plage : 0% - 100%
- Profondeurs : 10cm, 30cm, 60cm
- Fréquence : Toutes les 5 minutes
- Précision simulée : ±2%
- Comportement : Décroît naturellement, augmente lors d'irrigation

Capteur de Température (Priorité 1)

- Type : Thermomètre numérique
- Plage : -10 °C à 50 °C
- Localisations : Sol, Air ambiant
- Fréquence : Toutes les 5 minutes
- Variations : Cycle jour/nuit, saisons

Capteur de Luminosité (Priorité 1)

- Type : Photorésistance LDR
- Plage : 0 - 100,000 lux
- Fréquence : Toutes les 10 minutes
- Simulation : Cycle solaire réaliste

- Impact : Évapotranspiration

Pluviomètre (Priorité 2)

- Type : Capteur de pluie
- Mesure : mm de précipitation
- Fréquence : Temps réel lors d'événements
- Simulation : Événements météo aléatoires

Anémomètre (Vent) (Priorité 2)

- Type : Capteur de vitesse du vent
- Plage : 0 - 100 km/h
- Impact : Évaporation accélérée
- Fréquence : Toutes les 15 minutes

Capteur NPK (Nutriments) (Priorité 3 - Bonus)

- Mesure : Azote, Phosphore, Potassium
- Usage : Optimisation fertigation

Débitmètre d'Eau (Priorité 1)

- Mesure : Litres/minute
- Total : Consommation cumulée
- Alertes : Fuites détectées

3.2 Actionneurs Virtuels

Électrovanne d'Irrigation

- États : Ouvert / Fermé
- Contrôle : Manuel / Automatique
- Débit : Configurable (L/min)
- Zones : Multiple (A, B, C...)

Pompe à Eau

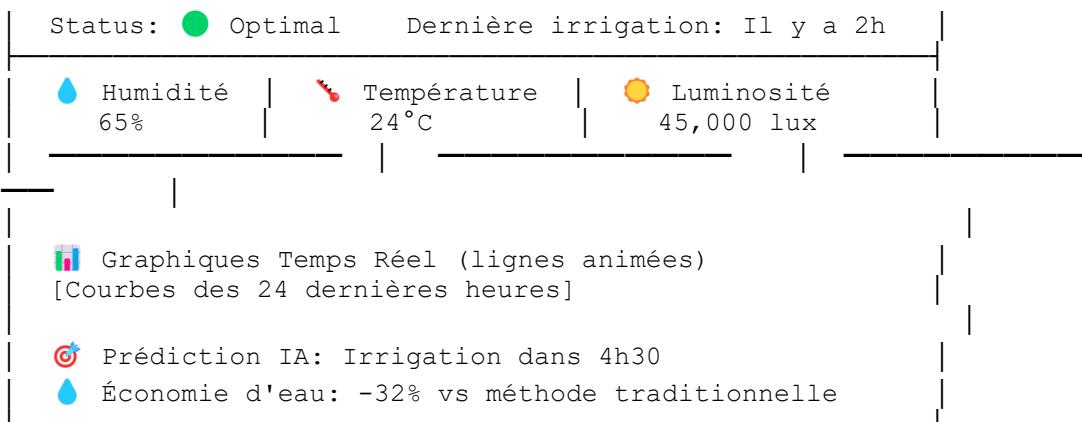
- États : ON / OFF
- Puissance : Configurable (W)
- Pression : Simulée (bar)

3.3 Fonctionnalités Essentielles (MUST HAVE)

Tableau de Bord Temps Réel

Vue Principale

 SmartIrrig - Zone A



Widgets en Temps Réel

- Jauge animée pour chaque capteur
- Graphiques live (Chart.js/Recharts)
- Indicateurs de statut colorés
- Alertes contextuelles
- Prédiction IA affichée
- Consommation d'eau en direct

🤖 Système de Règles Avancées

Moteur de Décision Multi-Facteurs

```
// Exemple de règle complexe
IF (humidité_sol < seuil_critique)
    AND (température > 25°C)
    AND (luminosité > 30000 lux)
    AND (vent < 15 km/h)
    AND (probabilité_pluie < 20%)
    AND (heure BETWEEN 6h-10h OR 18h-22h)
THEN
    déclencher_irrigation(durée: calculée_par_IA)
```

Facteurs de Décision

1. **Primaires** : Humidité du sol (3 niveaux)
2. **Environnementaux** : Température, vent, pluie
3. **Temporels** : Heure, saison, historique
4. **Culturaux** : Type de culture, stade de croissance
5. **Économiques** : Coût énergétique, tarification eau

Modes de Fonctionnement

- **Manuel** : Contrôle total utilisateur
- **Programmé** : Horaires fixes
- **Semi-automatique** : Règles + validation manuelle
- **Intelligent** : IA complète avec ML

Machine Learning & Prédictions

Modèle 1 : Prédiction de l'Humidité Future

```
# Entrées (features):
- Humidité actuelle (3 profondeurs)
- Température (24h historique)
- Luminosité
- Vent
- Pluie récente
- Irrigation passée
- Saison
- Type de sol

# Sortie:
- Humidité prévue à +1h, +3h, +6h, +12h, +24h

# Algorithme: Régression (Random Forest ou LSTM)
```

Modèle 2 : Recommandation d'Irrigation Optimale

```
# Entrées:
- État actuel tous capteurs
- Prévisions météo simulées
- Historique 7 jours
- Type de culture

# Sortie:
- Meilleur moment d'irrigation (timestamp)
- Durée optimale (minutes)
- Volume d'eau (litres)
- Économie estimée (%)
```

Algorithme: Classification + Régression

Modèle 3 : Détection d'Anomalies

- Fuites détectées (débit anormal)
- Capteurs defectueux (valeurs aberrantes)
- Gaspillage d'eau
- Sous-irrigation
- Sur-irrigation

Entraînement du Modèle

- Données historiques générées (simulation de 1 an)
- 80% train / 20% test
- Validation croisée
- Métriques : RMSE, MAE, R²
- Ré-entraînement périodique avec nouvelles données

Historique et Analytics

Visualisations

- Graphiques multi-séries temporelles
- Heatmaps de consommation
- Comparaisons périodes
- Corrélations entre variables
- Graphiques de tendances

Périodes Disponibles

- Dernières 24 heures (granularité: 5 min)
- 7 derniers jours (granularité: 1h)
- 30 derniers jours (granularité: 1 jour)
- 1 an (granularité: 1 semaine)
- Comparaison années

Export de Données

- CSV : Données brutes
- Excel : Rapports formatés
- PDF : Graphiques + statistiques
- JSON : API data

Gestion Multi-Zones

Zones/Parcelles

Zone A: Tomates (2 hectares)
└ Capteurs: 4 hygromètres, 2 thermomètres
└ Irrigation: 2 électrovannes
└ Statut: Optimal 

Zone B: Maïs (5 hectares)
└ Capteurs: 6 hygromètres, 3 thermomètres
└ Irrigation: 4 électrovannes
└ Statut: Attention 

Zone C: Légumes variés (1 hectare)
└ Capteurs: 2 hygromètres, 1 thermomètre
└ Irrigation: 1 électrovanne
└ Statut: Critique 

Vue Cartographique

- Plan de la ferme (Canvas ou SVG)
- Zones colorées selon statut
- Placement des capteurs
- Réseau d'irrigation visualisé

Alertes et Notifications

Types d'Alertes

- Critique : Humidité < 20% (irrigation urgente)
- Attention : Humidité < 40%
- Info : Irrigation planifiée dans 1h
- Anomalie : Capteur défaillant
- Fuite : Débit anormal détecté
- Météo : Pluie prévue, irrigation annulée

Canaux de Notification

- In-app (temps réel)
 - Email (résumé quotidien)
 - Sons/vibrations (optionnel)
 - Dashboard dédié
-

3.4 Fonctionnalités Avancées (SHOULD HAVE)

Planificateur Intelligent

- Calendrier d'irrigation optimisé
- Intégration météo simulée
- Ajustement automatique selon conditions
- Scénarios "et si..." (simulation)

Analyse Coûts/Bénéfices

- Consommation d'eau (m³)
- Coût énergétique (kWh)
- Économies vs méthode traditionnelle
- ROI du système intelligent

Simulation Météo

- Génération de patterns météo réalistes
- Saisons simulées
- Événements extrêmes (canicule, sécheresse)
- Impact sur irrigation

Module Expérimental

- Tester différentes stratégies
- Comparer algorithmes ML
- A/B testing de règles
- Sandbox de simulation

Interface Responsive

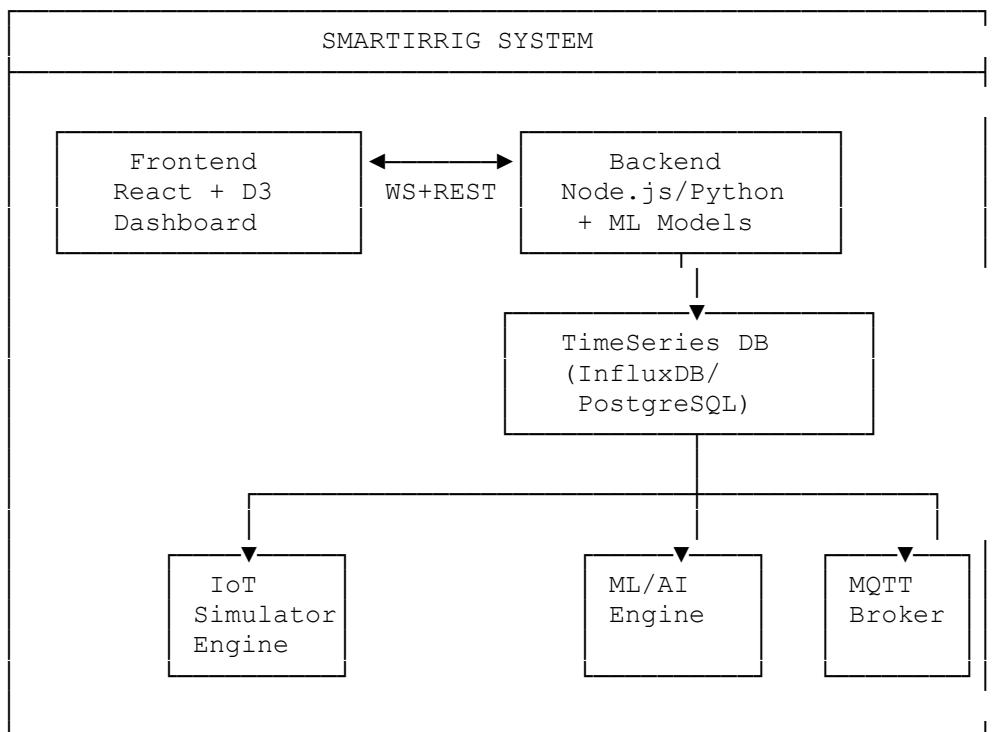
- Design adaptatif (mobile/tablet/desktop)
 - PWA (Progressive Web App)
 - Mode hors-ligne (cache local)
-

3.5 Fonctionnalités Bonus (NICE TO HAVE)

- 🎮 Mode "Jeu" : Gestion de ferme gamifiée
 - 🌐 API publique pour intégrations
 - 🔊 Commandes vocales (Web Speech API)
 - 🤝 Collaboration multi-utilisateurs temps réel
 - 📸 Reconnaissance d'images (santé des plantes)
 - 🌧️ Intégration API météo réelle
 - 🏆 Système de badges/achievements
 - 📚 Module de formation/tutoriels
-

4. Architecture Technique {#architecture}

4.1 Architecture Globale



4.2 Flux de Données Temps Réel

Simulation Loop (toutes les 5 secondes) :

1. IoT Simulator génère nouvelles valeurs
 - ↳ Algorithmes réalistes (courbes sinus, bruit gaussien, événements aléatoires)

2. Données publiées via MQTT/WebSocket
 - ↳ Topic: smartirrig/zone-A/sensors

3. Backend reçoit et stocke en TimeSeries DB
 - ↳ Indexation temporelle optimisée

4. Moteur de règles analyse les données
 - ↳ Évalue conditions multi-facteurs

5. Modèle ML fait prédictions
 - ↳ Humidité future + Recommandations

6. Décision d'irrigation prise
 - ↳ Automatique ou suggérée à l'utilisateur

7. Si irrigation: Actionneurs activés
 - ↳ Électrovannes ouvertes (virtuellement)

8. État mis à jour et diffusé au Frontend
 - ↳ WebSocket push vers dashboard

9. Dashboard se rafraîchit en temps réel
 - ↳ Graphiques animés, métriques actualisées

4.3 Simulateur IoT (Algorithmes)

Simulation d'Humidité du Sol

```
class HumiditySensor {
  constructor() {
    this.value = 60; // Initial: 60%
    this.evaporationRate = 0.5; // %/heure
  }

  simulate(deltaTime, temperature, sunlight, wind, isIrrigating,
  isRaining) {
    // Évaporation naturelle
    let evaporation = this.evaporationRate * (deltaTime / 3600);

    // Facteurs multiplicateurs
    evaporation *= (1 + (temperature - 20) * 0.02); // Température
    evaporation *= (1 + sunlight / 100000); // Soleil
    evaporation *= (1 + wind / 50); // Vent

    this.value -= evaporation;

    // Irrigation: +10% par minute
    if (isIrrigating) {
      this.value += 10 * (deltaTime / 60);
    }

    // Pluie: variable
    if (isRaining) {
      this.value += Math.random() * 5 * (deltaTime / 60);
    }
  }
}
```

```

        }

        // Limites
        this.value = Math.max(0, Math.min(100, this.value));

        // Bruit réaliste
        this.value += (Math.random() - 0.5) * 0.5;

        return Math.round(this.value * 10) / 10;
    }
}

```

Simulation de Température

```

class TemperatureSensor {
    simulate(hour, season) {
        // Température de base selon saison
        const baseTemp = {
            spring: 18, summer: 28, autumn: 15, winter: 8
        }[season];

        // Variation jour/nuit (courbe sinusoïdale)
        const dailyVariation = 8 * Math.sin((hour - 6) * Math.PI / 12);

        // Bruit aléatoire
        const noise = (Math.random() - 0.5) * 2;

        return baseTemp + dailyVariation + noise;
    }
}

```

Simulation de Luminosité

```

class LightSensor {
    simulate(hour) {
        if (hour < 6 || hour > 20) return 0; // Nuit

        // Courbe en cloche pour le jour
        const peakHour = 13;
        const maxLux = 100000;
        const curve = Math.exp(-Math.pow(hour - peakHour, 2) / 18);

        return maxLux * curve * (0.8 + Math.random() * 0.2);
    }
}

```

4.4 Base de Données

TimeSeries avec PostgreSQL + TimescaleDB

```

-- Extension TimescaleDB pour séries temporelles
CREATE EXTENSION IF NOT EXISTS timescaledb;

-- Table des mesures de capteurs
CREATE TABLE sensor_data (
    time TIMESTAMPTZ NOT NULL,
    zone_id INTEGER NOT NULL,
    sensor_type VARCHAR(50) NOT NULL,

```

```

    sensor_id VARCHAR(100) NOT NULL,
    value DOUBLE PRECISION NOT NULL,
    unit VARCHAR(20),
    metadata JSONB
);

-- Convertir en hypertable (TimescaleDB)
SELECT create_hypertable('sensor_data', 'time');

-- Index pour performances
CREATE INDEX idx_zone_sensor ON sensor_data (zone_id, sensor_type,
time DESC);

-- Table des événements d'irrigation
CREATE TABLE irrigation_events (
    id SERIAL PRIMARY KEY,
    zone_id INTEGER NOT NULL,
    start_time TIMESTAMPTZ NOT NULL,
    end_time TIMESTAMPTZ,
    duration_minutes INTEGER,
    water_volume_liters DOUBLE PRECISION,
    mode VARCHAR(50), -- 'manual', 'scheduled', 'automatic', 'ai'
    triggered_by VARCHAR(100),
    reason TEXT
);

-- Table des prédictions ML
CREATE TABLE ml_predictions (
    id SERIAL PRIMARY KEY,
    zone_id INTEGER NOT NULL,
    prediction_time TIMESTAMPTZ NOT NULL,
    target_time TIMESTAMPTZ NOT NULL, -- Quand la prédition
    s'applique
    predicted_humidity DOUBLE PRECISION,
    confidence DOUBLE PRECISION,
    model_version VARCHAR(50),
    features JSONB
);

-- Table des zones/parcelles
CREATE TABLE zones (
    id SERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    area_hectares DOUBLE PRECISION,
    crop_type VARCHAR(100),
    soil_type VARCHAR(100),
    coordinates JSONB, -- GeoJSON
    optimal_humidity_min DOUBLE PRECISION DEFAULT 40,
    optimal_humidity_max DOUBLE PRECISION DEFAULT 70,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table des alertes
CREATE TABLE alerts (
    id SERIAL PRIMARY KEY,
    zone_id INTEGER REFERENCES zones(id),
    alert_type VARCHAR(50) NOT NULL,
    severity VARCHAR(20), -- 'info', 'warning', 'critical'
    message TEXT NOT NULL,
    is_resolved BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMPTZ DEFAULT NOW(),

```

```

        resolved_at TIMESTAMPTZ
    ) ;

-- Agrégations automatiques (TimescaleDB)
-- Moyennes horaires
CREATE MATERIALIZED VIEW sensor_data_hourly
WITH (timescaledb.continuous) AS
SELECT
    time_bucket('1 hour', time) AS bucket,
    zone_id,
    sensor_type,
    AVG(value) as avg_value,
    MIN(value) as min_value,
    MAX(value) as max_value,
    COUNT(*) as num_readings
FROM sensor_data
GROUP BY bucket, zone_id, sensor_type;

-- Rafraîchissement automatique
SELECT add_continuous_aggregate_policy('sensor_data_hourly',
    start_offset => INTERVAL '3 hours',
    end_offset => INTERVAL '1 hour',
    schedule_interval => INTERVAL '1 hour');

```

4.5 API REST Endpoints

Capteurs et Données

GET /api/sensors	# Liste des capteurs actifs
GET /api/sensors/:id/current	# Valeur actuelle
GET /api/sensors/:id/history to, interval)	# Historique (filtres: from,
GET /api/zones/:id/sensors	# Tous capteurs d'une zone
POST /api/simulator/start	# Démarrer simulation
POST /api/simulator/stop	# Arrêter simulation
PUT /api/simulator/config simulation	# Configurer paramètres

Irrigation

GET /api/irrigation/status	# État actuel irrigation
POST /api/irrigation/start	# Démarrer manuellement
POST /api/irrigation/stop	# Arrêter
GET /api/irrigation/history	# Historique événements
GET /api/irrigation/schedule	# Planning
POST /api/irrigation/schedule	# Créer planification

Machine Learning

GET /api/ml/predictions/:zoneId	# Prédictions futures
POST /api/ml/predict	# Demander nouvelle prédiction
GET /api/ml/recommendations	# Recommandations IA
GET /api/ml/model-metrics	# Performance du modèle
POST /api/ml/retrain	# Re-entraîner modèle

Analytics

GET /api/analytics/dashboard	# Données dashboard
------------------------------	---------------------

```

GET      /api/analytics/water-usage          # Consommation d'eau
GET      /api/analytics/efficiency           # Efficacité irrigation
GET      /api/analytics/compare              # Comparaison périodes
POST     /api/analytics/export               # Export données

```

Zones et Configuration

```

GET      /api/zones                         # Liste zones
POST     /api/zones                         # Créer zone
PUT      /api/zones/:id                     # Modifier zone
DELETE   /api/zones/:id                     # Supprimer zone
GET      /api/zones/:id/stats               # Statistiques zone

```

Alertes

```

GET      /api/alerts                          # Toutes alertes
GET      /api/alerts/active                  # Alertes non résolues
PUT      /api/alerts/:id/resolve            # Marquer résolue
POST     /api/alerts/settings               # Configurer seuils

```

WebSocket (Temps Réel)

```

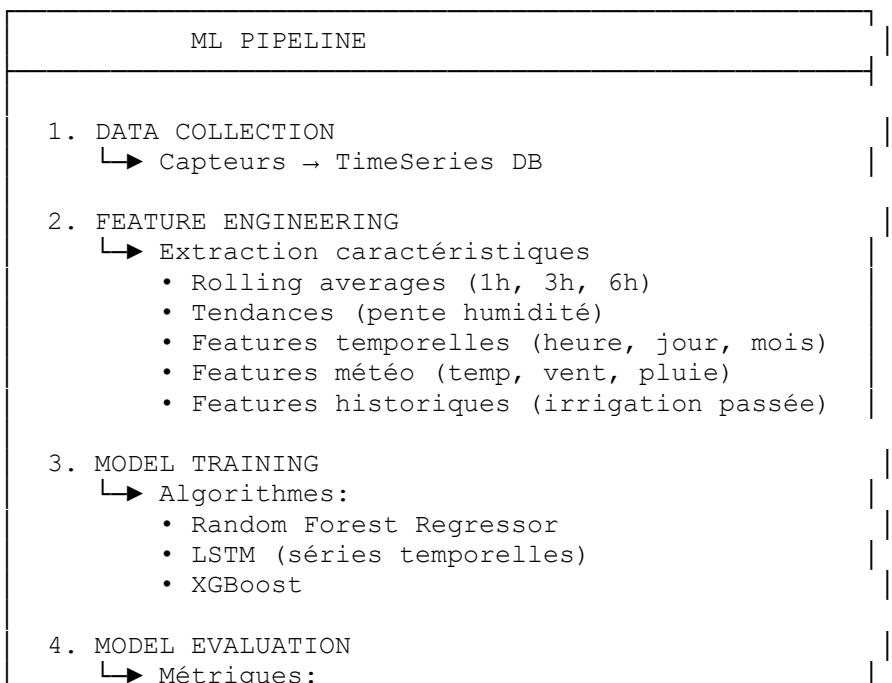
WS       /ws/sensors                       # Stream données capteurs
WS       /ws/zones/:id                     # Stream zone spécifique
WS       /ws/alerts                         # Stream alertes
WS       /ws/predictions                    # Stream prédictions ML

```



5. Intelligence Artificielle & Machine Learning {#ia-ml}

5.1 Architecture ML



- RMSE, MAE, R²
 - Cross-validation
5. DEPLOYMENT
 ↳ API endpoint pour prédictions
6. MONITORING
 ↳ Drift detection
 ↳ Auto-retrain si performance baisse

5.2 Modèle 1: Prédiction d'Humidité

Objectif : Prédire l'humidité du sol dans les prochaines heures

```
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

class HumidityPredictor:
    def __init__(self):
        self.model = RandomForestRegressor(
            n_estimators=100,
            max_depth=20,
            random_state=42
        )

    def prepare_features(self, data):
        """
        Features engineering
        """
        features = []

        # 1. Humidité actuelle (3 profondeurs)
        features.append(data['humidity_10cm'])
        features.append(data['humidity_30cm'])
        features.append(data['humidity_60cm'])

        # 2. Moyennes glissantes
        features.append(data['humidity_rolling_1h'])
        features.append(data['humidity_rolling_3h'])
        features.append(data['humidity_rolling']
```