🚀 Feuille de Route Développement PaxNet AI

Guide Complet pour les Développeurs - Double Mission

🗓️ Vue d'Ensemble : 18 Mois vers le MVP Double Mission

```
mermaid

gantt
    title Roadmap PaxNet AI - Phases Principales
    dateFormat  YYYY-MM-DD

    section Phase 1 - Fondations
    ETHOS Core           :crit, 2025-01-01, 60d
    Infrastructure Base  :crit, 2025-01-15, 45d

    section Phase 2 - Trinité
    NEXUS                :active, 2025-03-01, 60d
    MEMORIA              :2025-03-15, 60d
    AURORA               :2025-04-01, 45d

    section Phase 3 - Opérationnel
    HARMONY              :2025-05-15, 60d
    CHRONOS              :2025-06-01, 75d

    section Phase 4 - Urgence
    PHOENIX              :2025-08-15, 60d
    ATLAS                :2025-09-01, 60d

    section Phase 5 - Recherche
    EMPATHIA             :2025-11-01, 60d
    GAIA & MELODY        :2025-12-01, 60d
```

🎯 PHASE 1 : FONDATIONS CRITIQUES (Mois 1-2)

"Sans éthique, pas de confiance. Sans infrastructure, pas de scalabilité."

🔴 PRIORITÉ ABSOLUE : ETHOS (Semaines 1-8)

Pourquoi d'abord ? Sans garde-fous éthiques, tout le reste est dangereux.

Stack Technique ETHOS

yaml

```yaml
backend:
  language: Python 3.11+
  framework: FastAPI
  database: PostgreSQL + Redis

ai_libraries:
  - SHAP  # Explicabilité obligatoire
  - Fairlearn  # Détection biais
  - AIF360  # Métriques équité

monitoring:
  - Prometheus + Grafana
  - OpenTelemetry
  - ELK Stack pour logs
```

Fonctionnalités Minimales v0.1

python

```python
# ethos/core/validator.py
class EthicalValidator:
    """MVP : Validation basique mais robuste"""

    def validate_action(self, action: Action) -> ValidationResult:
        checks = [
            self.check_human_consent(action),
            self.check_transparency(action),
            self.check_privacy(action),
            self.check_non_discrimination(action)
        ]

        if all(checks):
            return ValidationResult(approved=True, log_publicly=True)
        else:
            return ValidationResult(
                approved=False,
                violations=self.get_violations(checks),
                require_human_review=True
            )
```

Livrables Semaine 8

- [ ] API REST validation éthique
- [ ] Dashboard transparence publique

- [ ] Système de veto distribué
- [ ] Logs immutables (blockchain light)
- [ ] Tests couverture 95%+

🟡 Infrastructure de Base (Semaines 3-8, parallèle)

Architecture Microservices

```yaml
orchestration:
  platform: Kubernetes (K8s)
  service_mesh: Istio

messaging:
  queue: RabbitMQ
  stream: Apache Kafka

storage:
  sql: PostgreSQL (sharded)
  nosql: MongoDB
  cache: Redis Cluster
  files: MinIO (S3 compatible)

observability:
  metrics: Prometheus + Grafana
  traces: Jaeger
  logs: Fluentd + Elasticsearch
```

Setup Développement Local

```bash
# Makefile racine
setup-dev:
  docker-compose -f docker/dev.yml up -d
  ./scripts/init-db.sh
  ./scripts/seed-test-data.sh

run-ethos:
  cd services/ethos && python -m uvicorn app:app --reload

test-all:
  pytest --cov=services --cov-report=html
```

🎯 PHASE 2 : TRINITÉ COLLABORATIVE (Mois 3-4)

"Le cœur qui fait battre PaxNet"

🟡 NEXUS - Le Facilitateur (Semaines 9-16)

Dépend de : ETHOS fonctionnel

Stack Technique

```yaml
backend:
  language: TypeScript
  framework: NestJS
  runtime: Node.js 20+

ai_integration:
  - Hugging Face Transformers
  - LangChain pour orchestration
  - OpenAI API (optionnel)
```

MVP Fonctionnalités

```typescript
// nexus/src/facilitator/facilitator.service.ts
export class FacilitatorService {
  async proposeSolutions(crisis: Crisis): Promise<Proposal[]> {
    // 1. Analyser le type de crise
    const analysis = await this.analyzeCrisis(crisis);

    // 2. Générer options selon double mission
    const options = crisis.type === CrisisType.CONFLICT
      ? await this.generatePeaceOptions(analysis)
      : await this.generateDisasterOptions(analysis);

    // 3. Validation ETHOS obligatoire
    const validated = await this.ethosClient.validateBatch(options);

    // 4. Retourner TOUJOURS multiple choix
    return validated.map(opt => ({
      ...opt,
      finalDecision: 'HUMAN_REQUIRED',
      nexusRole: 'ADVISOR_ONLY'
    }));
  }
}
```

🟡 MEMORIA - La Mémoire (Semaines 10-17)

Dépend de : Infrastructure storage

Stack Technique

```yaml
backend:
  language: Rust
  framework: Actix-web
  database: TimescaleDB + S3

ml_pipeline:
  - Apache Spark pour analyse
  - MLflow pour versioning modèles
  - DVC pour data versioning
```

🟡 AURORA - L'Amplificateur (Semaines 13-18)

Dépend de : NEXUS + MEMORIA

Stack Technique

```yaml
backend:
  language: Go
  framework: Gin
  patterns: Event-driven + CQRS

integration:
  - GraphQL Federation
  - gRPC pour inter-service
  - WebSockets pour real-time
```

🎯 PHASE 3 : SYSTÈMES OPÉRATIONNELS (Mois 5-7)

"Les muscles de PaxNet"

🟢 HARMONY - Médiation & Communication (Semaines 19-26)

Dépend de : Trinité fonctionnelle

Stack IA Prioritaire

python

```python
# Double usage critique
nlp_stack = {
    'base_models': [
        'facebook/nllb-200-3.3B',  # Traduction 200 langues
        'microsoft/deberta-v3-base',  # Sentiment analysis
        'cardiffnlp/twitter-roberta-base-emotion'  # Émotions
    ],
    'specialized': {
        'mediation': 'custom-bert-mediation',  # À entraîner
        'crisis': 'disaster-bert-large'  # Pré-entraîné
    }
}
```

Cas d'Usage Double Mission

python

```python
class HarmonyOrchestrator:
    def handle_communication(self, context: Context) -> Response:
        if context.is_conflict_mediation:
            return self.mediate_dialogue(
                parties=context.parties,
                cultural_context=context.culture,
                emotion_level=self.analyze_emotions(context)
            )
        elif context.is_evacuation:
            return self.broadcast_emergency(
                languages=context.affected_languages,
                severity=context.threat_level,
                tone='calm_authoritative'
            )
```

🟢 CHRONOS - Prédiction Duale (Semaines 21-30)

Dépend de : MEMORIA pour historique

Stack ML/DL Critique

```yaml
time_series:
  - Prophet (Meta) pour patterns
  - LSTM/GRU pour séquences complexes
  - ARIMA pour baseline

catastrophes:
  - TensorFlow pour séismes
  - PyTorch pour météo extrême
  - Scikit-learn pour risques composites

conflits:
  - BERT pour analyse médias sociaux
  - Graph Neural Networks pour réseaux
  - XGBoost pour indicateurs économiques
```

🎯 PHASE 4 : SYSTÈMES D'URGENCE (Mois 8-10)

"Quand chaque seconde compte"

🔴 PHOENIX - Intervention Rapide (Semaines 31-38)

Dépend de : CHRONOS pour alertes

Architecture Haute Performance

```go
// Optimisé pour latence minimale
type EmergencyResponse struct {
    responseTime  time.Duration  // Target: <100ms
    availability  float64        // Target: 99.99%
    concurrent    int            // Target: 100k req/s
}

func (p *Phoenix) HandleEmergency(alert Alert) Response {
    // Circuit breaker pour résilience
    return p.circuitBreaker.Execute(func() Response {
        // Parallel processing
        ch := make(chan Task, 1000)

        // Décisions en <100ms
        go p.assessSituation(alert, ch)
        go p.mobilizeResources(alert, ch)
        go p.notifyResponders(alert, ch)

        return p.coordinateResponse(ch)
    })
}
```

🟠 ATLAS - Mobilisation Citoyenne (Semaines 33-40)

Dépend de : HARMONY pour communication

Stack Mobile + Web

```yaml
mobile:
  framework: React Native
  offline: SQLite + Sync

web:
  frontend: Next.js 14+
  maps: Leaflet + OpenStreetMap
  realtime: Socket.io

matching_algorithm:
  - KNN pour proximité
  - Skills matching ML
  - Availability prediction
```

🎯 PHASE 5 : RECHERCHE EXPLORATOIRE (Mois 11-12+)

"L'innovation pour demain"

🔵 EMPATHIA - IA Émotionnelle

Peut attendre : Pas critique pour MVP

```python
# R&D : Multimodal emotion AI
future_stack = {
    'facial': 'OpenCV + FER',
    'voice': 'Whisper + Prosody',
    'text': 'RoBERTa-emotion',
    'fusion': 'Custom multimodal transformer'
}
```

🔵 GAIA - Environnement

Peut attendre : Enrichissement futur

🔵 MELODY - Thérapie Sonore

Peut attendre : Feature avancée

📊 Métriques de Succès par Phase

Phase 1 : Fondations (2 mois)

- ✅ ETHOS valide 1000 décisions/seconde
- ✅ Infrastructure supporte 10k users concurrent
- ✅ 99.9% uptime en dev

Phase 2 : Trinité (2 mois)

- ✅ NEXUS génère solutions en <500ms
- ✅ MEMORIA stocke 1TB données/mois
- ✅ AURORA amplifie 100 connexions/s

Phase 3 : Opérationnel (3 mois)

- ✅ HARMONY traduit 50 langues temps réel
- ✅ CHRONOS prédit avec 85% précision

Phase 4 : Urgence (3 mois)

- ✅ PHOENIX répond en <100ms
- ✅ ATLAS mobilise 10k volontaires/heure

🛠️ Outils de Développement Essentiels

IDE & Extensions

```yaml
vscode_extensions:
  - Python: ms-python.python
  - Rust: rust-lang.rust-analyzer
  - Go: golang.go
  - Docker: ms-azuretools.vscode-docker
  - K8s: ms-kubernetes-tools.vscode-kubernetes-tools

productivity:
  - GitHub Copilot
  - Thunder Client (API testing)
  - GitLens
```

CI/CD Pipeline

yaml

```yaml
# .github/workflows/main.yml
name: PaxNet CI/CD

on: [push, pull_request]

jobs:
  ethos-check:
    runs-on: ubuntu-latest
    steps:
      - name: Validate Ethical Compliance
        run: ./scripts/ethos-validate-all.sh

  test:
    needs: ethos-check
    strategy:
      matrix:
        service: [ethos, nexus, memoria, aurora]
    steps:
      - name: Test ${{ matrix.service }}
        run: make test-${{ matrix.service }}

  deploy:
    if: github.ref == 'refs/heads/main'
    needs: test
    steps:
      - name: Deploy to Staging
        run: kubectl apply -k k8s/staging/
```

📚 Documentation Obligatoire

Par Service

markdown

```
services/
├── {service}/
│   ├── README.md          # Vue d'ensemble
│   ├── API.md             # Endpoints
│   ├── ARCHITECTURE.md    # Décisions techniques
│   ├── ETHICS.md          # Considérations éthiques
│   └── EXAMPLES.md        # Cas d'usage double mission
```

Standards de Code

- **Python** : Black + isort + mypy
- **TypeScript** : ESLint + Prettier
- **Go** : gofmt + golangci-lint
- **Rust** : rustfmt + clippy

🚨 Points d'Attention Critiques

1. Ne JAMAIS contourner ETHOS

```python
# ❌ INTERDIT
def quick_decision(crisis):
    return execute_action(crisis)  # Sans validation

# ✅ OBLIGATOIRE
def safe_decision(crisis):
    validation = ethos.validate(crisis)
    if validation.approved:
        return execute_action(crisis)
    else:
        return request_human_intervention(crisis)
```

2. Double Mission = Double Test

```python
# Chaque feature DOIT être testée pour les 2 missions
def test_alert_system():
    # Test conflit
    assert alert.works_for(CrisisType.CONFLICT)

    # Test catastrophe
    assert alert.works_for(CrisisType.DISASTER)

    # Test basculement
    assert alert.can_switch_priority_fast()
```

3. Performance Critique

- Latence max évacuation : 100ms
- Disponibilité minimale : 99.99%
- Scalabilité : 1M users simultanés

🎓 Formation Équipe

Semaine 1 : Onboarding

- Vision PaxNet + Double Mission
- Architecture globale
- Standards ETHOS

Semaine 2 : Stack Technique

- Setup environnement
- Premiers commits
- Review process

Ongoing : Montée en Compétence

- Weekly tech talks
- Pair programming
- Code reviews constructives

🗓️ Planning Détaillé Premier Trimestre

Janvier 2025

- **S1-2** : Setup équipe, environnements
- **S3-4** : ETHOS core development

Février 2025

- **S5-6** : ETHOS finalization + tests
- **S7-8** : Infrastructure production-ready

Mars 2025

- **S9-10** : NEXUS development
- **S11-12** : MEMORIA parallel dev

💡 Conseils aux Développeurs

1. **Commencez Simple**

"MVP d'abord, optimisation ensuite"

2. **Testez la Double Mission**

"Si ça ne marche que pour la paix OU les catastrophes, c'est incomplet"

3. Documentation = Code

"Code non documenté = dette technique"

4. Éthique First

"En cas de doute, consultez ETHOS"

5. Fail Fast, Learn Faster

"Les erreurs en dev sauvent des vies en prod"

🏁 Prochaines Actions Immédiates

Pour les Tech Leads

- [ ] Valider architecture avec l'équipe
- [ ] Assigner ownership par service
- [ ] Créer boards Jira/GitHub Projects

Pour les Développeurs

- [ ] Fork repo principal
- [ ] Setup environnement local
- [ ] Prendre une issue "good-first-issue"

Pour les DevOps

- [ ] Provisionner clusters K8s
- [ ] Setup CI/CD pipelines
- [ ] Configurer monitoring

"Construisons ensemble un système qui protège l'humanité sur tous les fronts, une ligne de code à la fois."

Contact : tech-lead@paxnet-ai.org
Discord : #dev-general
Dernière MAJ : Janvier 2025