🚀 PaxNet AI - Démarrage Rapide pour Développeurs

TL;DR : Commencez à coder en 15 minutes

🎯 L'Essentiel en 30 Secondes

PaxNet AI = IA Open Source pour la **PAIX** + **PROTECTION CATASTROPHES**
Règle #1 : Toute action passe par ETHOS (garde-fou éthique)
Règle #2 : Chaque feature sert les DEUX missions
Règle #3 : L'humain décide toujours, l'IA propose

⚡ Setup Express (15 min)

1. Cloner et Installer

```bash
# Clone
git clone https://github.com/paxnet-ai/core.git
cd paxnet-ai

# Setup automatique
make setup-dev

# Lancer ETHOS (OBLIGATOIRE)
make run-ethos

# Vérifier que tout fonctionne
make test-minimal
```

2. Structure Simplifiée

```
paxnet-ai/
├── services/
│   ├── ethos/       # 🔴 PRIORITÉ 1 - Éthique
│   ├── nexus/       # 🟡 PRIORITÉ 2 - Facilitateur
│   ├── chronos/     # 🟢 PRIORITÉ 3 - Prédictions
│   └── phoenix/     # 🟠 PRIORITÉ 4 - Urgences
├── shared/          # Code partagé
├── docker/          # Containers dev
└── docs/            # Documentation
```

3. Votre Premier Code

```python
# services/ethos/quickstart.py
from ethos.core import EthicalValidator

# Exemple : Valider une action double mission
validator = EthicalValidator()

# Cas 1 : Médiation de conflit
peace_action = {
    "type": "mediation",
    "target": "community_dispute",
    "has_consent": True,
    "is_transparent": True
}

# Cas 2 : Évacuation d'urgence
disaster_action = {
    "type": "evacuation",
    "severity": "critical",
    "affected_people": 5000,
    "respects_dignity": True
}

# ETHOS valide les DEUX
print(validator.validate(peace_action))    # ✅ Approved
print(validator.validate(disaster_action)) # ✅ Approved
```

📋 Checklist Première Semaine

Jour 1 : Comprendre

- [ ] Lire ce guide (15 min)
- [ ] Explorer Architecture Double Mission (30 min)
- [ ] Rejoindre Discord #dev-general

Jour 2 : Setup

- [ ] Installer environnement local
- [ ] Faire tourner ETHOS
- [ ] Passer les tests de base

Jour 3 : Première Contribution

- [ ] Prendre une issue `good-first-issue`
- [ ] Coder une petite feature

- [ ] Soumettre première PR

Jour 4-5 : Intégration

- [ ] Code review avec l'équipe
- [ ] Comprendre la double mission
- [ ] Planifier contribution suivante

🎯 Priorités de Développement (Ordre STRICT)

🔴 Phase 1 : ETHOS D'abord (Mois 1-2)

RIEN ne fonctionne sans garde-fous éthiques

```python
# Priorité ABSOLUE : Validation éthique
class EthosCore:
    def validate(self, action):
        # 4 checks minimaux obligatoires
        return all([
            self.check_human_consent(action),
            self.check_transparency(action),
            self.check_privacy(action),
            self.check_dual_mission(action)
        ])
```

À livrer :

- API validation (POST /validate)
- Dashboard transparence
- Logs publics
- Tests 95%+

🟡 Phase 2 : NEXUS + Trinité (Mois 3-4)

Le cerveau qui coordonne

```typescript
// NEXUS : Toujours proposer, jamais imposer
async proposeSolutions(crisis: Crisis): Promise<Solution[]> {
    const options = await this.generateOptions(crisis);
    const validated = await ethos.validateAll(options);

    return validated.map(opt => ({
        ...opt,
        decision_by: 'HUMAN_REQUIRED' // TOUJOURS
    }));
}
```

🟢 Phase 3 : CHRONOS (Mois 5-6)

Prédire pour prévenir

```python
# Double prédiction
def predict_risks(region):
    conflict_risk = self.predict_social_tensions(region)
    disaster_risk = self.predict_natural_hazards(region)

    # Retourner les DEUX
    return {
        'conflict_probability': conflict_risk,
        'disaster_probability': disaster_risk,
        'recommended_action': self.prioritize(conflict_risk, disaster_risk)
    }
```

🟠 Phase 4 : PHOENIX (Mois 7-8)

Quand ça devient critique

Performance cibles :

- Latence : < 100ms
- Disponibilité : 99.99%
- Scalabilité : 1M req/s

🔵 Phase 5+ : Le Reste Peut Attendre

- HARMONY : Communication avancée
- ATLAS : Mobilisation masse
- EMPATHIA, GAIA, MELODY : R&D future

💻 Stack Technique Rapide

Par Service (Recommandé)

Service	Language	Framework	Priorité IA
ETHOS	Python	FastAPI	SHAP, Fairlearn
NEXUS	TypeScript	NestJS	Transformers
CHRONOS	Python	FastAPI	TensorFlow, Prophet
PHOENIX	Go	Gin	- (Performance pure)

IA Open Source Essentielles

```python
# Top 5 pour commencer
essential_ai = {
    'tensorflow': '2.15+',      # Prédictions
    'transformers': '4.36+',    # NLP multilingue
    'scikit-learn': '1.3+',     # ML classique
    'prophet': '1.1+',          # Séries temporelles
    'shap': '0.44+'             # Explicabilité
}
```

🛑 Erreurs à NE PAS Faire

❌ JAMAIS

```python
# 1. Contourner ETHOS
def quick_action(crisis):
    return execute_immediately(crisis)  # INTERDIT !

# 2. Décider à la place de l'humain
def ai_decides(options):
    return options[0]  # L'IA ne décide JAMAIS seule

# 3. Ignorer une des missions
def handle_emergency(event):
    if event.type == 'disaster':
        return handle_disaster(event)
    # Et la paix ? INCOMPLET !
```

✅ TOUJOURS

```python
# 1. Valider avec ETHOS
def safe_action(crisis):
    if ethos.validate(crisis).approved:
        return propose_to_human(crisis)

# 2. Proposer multiple options
def generate_proposals(crisis):
    return [option1, option2, option3]  # Choix humain

# 3. Servir les DEUX missions
def dual_response(event):
    peace_response = handle_conflict_prevention(event)
    disaster_response = handle_disaster_prep(event)
    return combine_responses(peace_response, disaster_response)
```

📞 Obtenir de l'Aide

Discord Channels

- **#dev-help** : Questions techniques
- **#ethos-validation** : Questions éthiques
- **#code-review** : Revues de code
- **#dev-général** : Discussions

Documentation

- Architecture Complète
- Guide Contribution
- Principes ETHOS
- API References

Contacts Directs

- **Tech Lead** : tech-lead@paxnet-ai.org
- **ETHOS Questions** : ethos@paxnet-ai.org
- **Urgences Prod** : incidents@paxnet-ai.org

🎯 Votre Mission Cette Semaine

1. **Setup** : Environnement fonctionnel
2. **Comprendre** : Double mission (Paix + Catastrophes)

3. **Coder** : Une petite feature qui sert les DEUX
4. **Tester** : Validation ETHOS obligatoire
5. **Partager** : PR avec la communauté

💡 Tips de Pro

Développement Efficace

bash

```bash
# Alias utiles (.bashrc)
alias pax='cd ~/paxnet-ai'
alias ethos='make run-ethos'
alias paxtest='make test-watch'

# Git hooks automatiques
git config core.hooksPath .githooks
```

Debug Rapide

python

```python
# Toujours logger les décisions éthiques
import structlog
logger = structlog.get_logger()

def process_crisis(crisis):
    logger.info("crisis_received",
                type=crisis.type,
                severity=crisis.severity,
                dual_mission=True)

    validation = ethos.validate(crisis)
    logger.info("ethos_validation",
                approved=validation.approved,
                violations=validation.violations)
```

🏁 C'est Parti !

3 Commandes pour Commencer :

```bash
git clone https://github.com/paxnet-ai/core.git
cd paxnet-ai && make setup-dev
make run-ethos
```

1 Règle à Retenir :

> "Chaque ligne de code protège l'humanité sur DEUX fronts"

Prochaine Étape :

👉 Prenez une issue `good-first-issue`

Bienvenue dans l'équipe PaxNet AI ! Ensemble, construisons la paix ET protégeons des catastrophes.

🌍✨