Voici un plan détaillé et complet de la **Phase 2 : Configuration de l’environnement de développement** en incluant toutes les étapes nécessaires jusqu’à la création d’un dossier bien structuré. Ce plan prend en compte le fait que le chatbot doit gérer **50 000 utilisateurs**, ce qui nécessite une infrastructure robuste et scalable.

**🔷 Phase 2 : Configuration de l’environnement de développement (15+ tâches)**

L’objectif ici est de mettre en place un environnement stable, sécurisé et bien documenté pour assurer un développement fluide et évolutif.

**1. Création des comptes et récupération des accès API**

📌 **But :** Obtenir toutes les clés API et accès nécessaires.  
✅ **Tâches :**

* **Créer un compte OpenAI** et récupérer la **clé API GPT-4**.
* **Créer un compte Twilio** (pour WhatsApp), **Meta Developer** (pour Messenger), **Slack API**, etc.
* **Créer un compte Firebase** (si authentification Google est prévue).
* **Créer un compte AWS/GCP/Azure** pour l'hébergement.
* **Stocker les clés API** dans un **fichier .env** pour éviter les fuites.

🔧 **Commandes à exécuter (Exemple de fichier .env)**

OPENAI\_API\_KEY=sk-xxxxxxxxxxxxxxxxxxxx

TWILIO\_API\_KEY=xxxxxxxxxxxxxxxxxxxxxx

DATABASE\_URL=postgresql://user:password@localhost/dbname

JWT\_SECRET\_KEY=super\_secret\_key

**2. Installation et configuration de Python et FastAPI (Backend)**

📌 **But :** Préparer l’environnement Python et installer les dépendances backend.  
✅ **Tâches :**

* Installer **Python 3.10+** pour assurer la compatibilité avec FastAPI.
* Installer **pyenv** et **venv** pour gérer les versions de Python.
* Créer un **environnement virtuel** pour isoler les dépendances.
* Installer **FastAPI, Uvicorn, SQLAlchemy et d'autres bibliothèques essentielles**.

🔧 **Commandes à exécuter :**

# Installer pyenv et gérer la version de Python

curl https://pyenv.run | bash

pyenv install 3.10

pyenv virtualenv 3.10 chatbot-env

# Activer l'environnement virtuel et installer les dépendances

python -m venv env

source env/bin/activate # Sur Windows : env\Scripts\activate

pip install fastapi uvicorn openai sqlalchemy psycopg2 python-dotenv passlib jwt

**3. Installation et configuration du frontend avec Next.js**

📌 **But :** Mettre en place un frontend performant et optimisé pour le SEO.  
✅ **Tâches :**

* Installer **Node.js 18+** et **npm**.
* Créer un projet Next.js avec TypeScript pour un meilleur typage.
* Installer les dépendances essentielles (TailwindCSS, Axios, Zustand).
* Configurer les variables d’environnement dans next.config.js.

🔧 **Commandes à exécuter :**

# Installer Node.js (si pas encore installé)

nvm install 18

nvm use 18

# Créer un projet Next.js avec TypeScript

npx create-next-app@latest chatbot-frontend --typescript

# Installer les dépendances frontend

cd chatbot-frontend

npm install axios tailwindcss postcss autoprefixer zustand dotenv

**4. Configuration du repository GitHub**

📌 **But :** Versionner le projet proprement et structurer les branches de développement.  
✅ **Tâches :**

* **Créer un repository GitHub privé** pour le projet.
* **Initialiser Git et créer un .gitignore** pour éviter de pousser des fichiers sensibles.
* **Créer des branches** : main, dev, feature/chatbot, etc.
* **Mettre en place des hooks Git** pour vérifier la qualité du code avant chaque commit.

🔧 **Commandes à exécuter :**

# Initialiser le projet

git init

git remote add origin https://github.com/username/chatbot.git

# Créer un fichier .gitignore

echo "env/

\*.log

\_\_pycache\_\_/

node\_modules/

" > .gitignore

# Créer et passer sur la branche de développement

git checkout -b dev

git add .

git commit -m "Initial commit"

git push origin dev

**5. Mise en place d’un serveur local pour tester l’API**

📌 **But :** Pouvoir tester le backend rapidement sans attendre le déploiement.  
✅ **Tâches :**

* Installer **Postman** ou **Insomnia** pour tester les endpoints.
* Lancer un serveur FastAPI en mode développement.
* Tester un premier endpoint basique.

🔧 **Commandes à exécuter :**

# Lancer le serveur FastAPI en local

uvicorn main:app --reload

📌 **Vérifier en ouvrant :**  
<http://127.0.0.1:8000/docs> (Documentation auto-générée par FastAPI)

**6. Conteneurisation avec Docker**

📌 **But :** Assurer la portabilité et faciliter le déploiement.  
✅ **Tâches :**

* Installer Docker et Docker Compose.
* Écrire un Dockerfile et docker-compose.yml pour un déploiement rapide.

🔧 **Exemple de Dockerfile :**

FROM python:3.10

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

**7. Configuration de la base de données**

📌 **But :** Préparer la base pour stocker les utilisateurs et logs de conversations.  
✅ **Tâches :**

* Choisir PostgreSQL pour la robustesse.
* Installer PostgreSQL et configurer un premier schéma.
* Écrire les modèles SQLAlchemy pour gérer les utilisateurs.

🔧 **Exemple de connexion à PostgreSQL :**

from sqlalchemy import create\_engine

DATABASE\_URL = "postgresql://user:password@localhost/chatbotdb"

engine = create\_engine(DATABASE\_URL)

**8. Sécurisation des accès**

📌 **But :** Éviter les failles de sécurité dès le début du développement.  
✅ **Tâches :**

* Configurer les **tokens JWT** pour l’authentification.
* Vérifier que les clés API ne sont jamais exposées.
* Ajouter des **CORS headers** pour limiter les accès.

**9. Automatisation et CI/CD**

📌 **But :** Automatiser les tests et le déploiement.  
✅ **Tâches :**

* Mettre en place **GitHub Actions** pour tester et déployer.
* Automatiser les tests avec **pytest** côté backend.
* Vérifier le format du code avec **Prettier (JS) et Black (Python)**.

**10. Documentation et création du dossier final**

📌 **But :** Structurer le projet pour une maintenance facilitée.  
✅ **Tâches :**

* Rédiger un README.md clair avec l’architecture du projet.
* Détailler les commandes utiles (makefile pour automatiser les tâches).
* Organiser le dossier final.

🔧 **Structure du projet final :**

chatbot-project/

│── backend/

│ ├── main.py

│ ├── database.py

│ ├── models.py

│ ├── requirements.txt

│── frontend/

│ ├── pages/

│ ├── components/

│ ├── package.json

│── docker/

│ ├── Dockerfile

│ ├── docker-compose.yml

│── .gitignore

│── README.md

Avec ce plan, l’environnement est prêt pour démarrer **le développement du chatbot capable de gérer 50 000 utilisateurs** de manière efficace ! 🚀