



# Chatbot Desktop - Application Professionnelle

Application de bureau professionnel pour interagir avec des API OpenAI compatibles, avec interface graphique moderne, streaming en temps réel, et coloration syntaxique avancée.

## 🌟 Fonctionnalités

### 🎯 Cœur de l'Application

- ✓ **Streaming en temps réel** - Réponses progressives via QThread
- ✓ **SSL Bypass** - Support des certificats auto-signés (serveurs entreprise)
- ✓ **Multi-conversations** - Gestion illimitée de conversations simultanées
- ✓ **Base de données SQLite** - Persistance locale des conversations
- ✓ **Export JSON/Markdown** - Export sélectif ou complet

### 🎨 Interface Utilisateur

- ✓ **Sidebar avec sélection multiple** - Shift+Clic pour sélection par lot
- ✓ **QWebEngineView** - Rendu HTML avec Highlight.js
- ✓ **Coloration syntaxique** - 10+ langages supportés
- ✓ **Boutons "Copier"** - Sur chaque bloc de code
- ✓ **Scroll intelligent** - Auto-scroll vers la dernière question
- ✓ **Personnalisation** - Couleurs de code configurables

### 🔧 Technique

- ✓ **Logging avancé** - Mode DEBUG activable via CLI
- ✓ **Architecture MVC** - Séparation claire UI/Business/Data
- ✓ **Signaux/Slots PyQt6** - Communication asynchrone propre
- ✓ **Gestion d'erreurs robuste** - Try/catch et logging complet



## Prérequis

- **Python 3.8+**
- **Système d'exploitation** : Windows / macOS / Linux



## Installation

### 1. Cloner ou télécharger le projet

```
bash
```

```
git clone <repo_url>
cd chatbot_desktop
```

## 2. Créer un environnement virtuel (recommandé)

```
bash

# Windows
python -m venv venv
venv\Scripts\activate

# Linux/macOS
python3 -m venv venv
source venv/bin/activate
```

## 3. Installer les dépendances

```
bash

pip install -r requirements.txt
```

## Lancement

### Mode Normal

```
bash

python main.py
```

### Mode Debug (avec logs console)

```
bash

python main.py --debug
```

### Avec base de données personnalisée

```
bash

python main.py --db /path/to/custom.db
```

### Afficher l'aide

```
bash

python main.py --help
```

## Configuration

### Premier Lancement

1. **Ouvrir les Paramètres** : Menu `Paramètres` → `Configuration...` (Ctrl+,)

2. **Onglet Connexion** :

- Entrer votre **Clé API**
- Configurer l'**URL de base** (défaut : `https://api.openai.com/v1`)
- Choisir le **Modèle** (ex: `gpt-4`, `gpt-3.5-turbo`)
- **Décocher "Vérification SSL"** si serveur auto-signé

3. **Tester la connexion** avec le bouton `Tester la connexion`

4. **Onglet Apparence Code** :

- Personnaliser les couleurs de syntaxe
- Prévisualiser en temps réel
- Réinitialiser aux valeurs par défaut si besoin

5. **Enregistrer**

### Serveurs Auto-signés (Entreprise)

Pour les serveurs avec certificats auto-signés :

1. Dans les paramètres, **décocher** `Activer la vérification SSL`
2. Cette configuration utilise `httpx.Client(verify=False)`
3. ⚠ **Sécurité** : N'utilisez cette option que pour des serveurs de confiance

## Utilisation

### Créer une Conversation

- **Bouton** : `+ Nouvelle` dans la sidebar
- **Raccourci** : `Ctrl+N`

### Envoyer un Message

1. Taper le message dans la zone de saisie
2. **Entrée** = Envoyer
3. **Shift+Entrée** = Nouvelle ligne

### Sélection Multiple

- **Shift+Clic** sur les conversations

- **Shift+Flèches** pour navigation
- **Suppr** pour supprimer la sélection

## Export

1. Menu **Fichier** → **Exporter...** (Ctrl+E)
2. Choisir :
  - **Sélection actuelle** (si conversations sélectionnées)
  - **Toutes les conversations**
3. Format : **JSON** ou **Markdown**
4. Sauvegarder

## Coloration Syntaxique

### Langages Supportés

- Python
- JavaScript / TypeScript
- Bash / Shell
- PowerShell
- Java
- JSON
- HTML / CSS
- PHP
- Perl
- SQL
- C / C++
- C#
- Ruby
- Go
- Rust

### Personnalisation

Dans **Paramètres** → **Apparence Code** :

- **Commentaires** : 

- **Mots-clés :** 
- **Chaînes :** 
- **Nombres :** 
- **Fonctions :** 

## Structure du Projet

```

chatbot_desktop/
├── main.py                # Point d'entrée
├── requirements.txt       # Dépendances
├── chatbot.db             # Base de données (auto-généré)
├── core/
│   ├── logger.py         # Système de logging
│   ├── database.py       # Gestionnaire SQLite
│   ├── api_client.py     # Client OpenAI (SSL bypass)
│   ├── settings_manager.py # QSettings wrapper
│   ├── export_manager.py # Export JSON/MD
│   └── main_controller.py # Contrôleur principal
├── ui/
│   ├── main_window.py    # Fenêtre principale
│   ├── sidebar_widget.py # Historique conversations
│   ├── chat_widget.py    # Zone de chat (WebEngine)
│   ├── input_widget.py   # Zone de saisie
│   └── settings_dialog.py # Dialogue paramètres
├── workers/
│   └── api_worker.py      # Thread streaming API
└── utils/
    ├── html_generator.py  # Génération HTML
    ├── code_parser.py     # Détection code
    └── css_generator.py   # CSS personnalisé

```

## Logs en Mode Debug

Lorsque lancé avec `--debug`, l'application affiche :

[CONFIG] État de la configuration  
[API] Requêtes et chunks reçus  
[PARSER] Blocs de code détectés  
[DATABASE] Opérations SQL  
[EXPORT] Fichiers exportés  
[ERREUR] Stack traces complètes

## Dépannage

### L'application ne démarre pas

```
bash

# Vérifier l'installation
pip install --upgrade -r requirements.txt

# Lancer en mode debug
python main.py --debug
```

### Erreur de connexion API

1. Vérifier la clé API
2. Tester l'URL avec curl : `curl -I https://api.openai.com/v1`
3. Si certificat auto-signé : décocher `Vérification SSL`

### Pas de coloration syntaxique

- Vérifier les balises de code : ``python
- Internet requis pour CDN Highlight.js
- Consulter les logs : `--debug`

## Raccourcis Clavier

Raccourci	Action
<code>Ctrl+N</code>	Nouvelle conversation
<code>Ctrl+E</code>	Exporter
<code>Ctrl+,</code>	Paramètres
<code>Ctrl+Q</code>	Quitter
<code>Entrée</code>	Envoyer message
<code>Shift+Entrée</code>	Nouvelle ligne
<code>Suppr</code>	Supprimer sélection

## Sécurité

- Les clés API sont stockées dans `QSettings` (registre Windows / fichiers config Linux/macOS)
- **SSL Bypass** : À utiliser uniquement pour serveurs de confiance
- Les conversations sont stockées en clair dans SQLite local

## Contribution

Cette application est un exemple complet d'architecture professionnelle PyQt6.

### Points d'Extension

- Ajouter d'autres modèles IA
- Implémenter des plugins
- Support de fichiers (images, PDFs)
- Mode collaboratif multi-utilisateurs

## Licence

À définir selon vos besoins.

## Support

Pour toute question :

1. Consulter les logs en mode `--debug`
2. Vérifier la configuration dans `Paramètres`
3. Tester la connexion API

---

**Version** : 1.0.0

**Framework** : PyQt6

**Compatibilité API** : OpenAI Compatible