

PROCÉDURE TECHNIQUE D'INSTALLATION

Chatbot Intelligent IA Déconnectée - Cas d'usage inspirants

Version : 1.0

Date : Juillet 2025

Auteur : Erick Mormin - EKM Conseils

Projet : IA Déconnectée - Vidéo 3

SOMMAIRE

1. Prérequis système
2. Installation de l'environnement
3. Création du chatbot éducatif
4. Déploiement chatbot agricole
5. Configuration assistant santé
6. Mise en place gardien culturel
7. Template personnalisable
8. Tests et validation
9. Dépannage
10. Maintenance et évolution

PRÉREQUIS SYSTÈME

Configuration minimale recommandée

- **RAM :** 4 Go minimum (8 Go idéal)
- **Stockage :** 16 Go disponibles
- **Processeur :** ARM64 ou x86_64
- **Système :** Ubuntu 22.04 LTS / Debian 12 / Raspberry Pi OS

Outils requis

- **Python :** Version 3.10 ou supérieure
- **pip :** Gestionnaire de paquets Python
- **Connexion Internet :** Uniquement pour l'installation initiale

Vérification préalable

Vérifier Python

```
python3 --version
```

```
# Vérifier pip  
pip3 --version
```

```
# Vérifier espace disque  
df -h
```

INSTALLATION DE L'ENVIRONNEMENT

Étape 1 : Mise à jour du système

```
sudo apt update && sudo apt upgrade -y
```

Étape 2 : Installation des dépendances

```
# Installation Python et pip  
sudo apt install python3 python3-pip -y
```

```
# Installation Streamlit  
pip3 install streamlit
```

```
# Vérification installation  
streamlit --version
```

Étape 3 : Test de validation environnement

```
# Créer fichier de test  
cat > test_env.py << 'EOF'  
  
import streamlit as st  
  
print("✅ Streamlit importé avec succès")  
  
print("🔗 Environnement prêt pour IA déconnectée")  
  
EOF
```

```
# Exécuter test  
python3 test_env.py
```

Résultat attendu :

```
✅ Streamlit importé avec succès
```

 Environnement prêt pour IA déconnectée

CRÉATION DU CHATBOT ÉDUCATIF

Étape 1 : Créer la base de connaissances éducative

```
cat > base_education.py << 'EOF'

# Base de connaissances éducative française

responses_education = {

    "mathématiques": [

        "Les mathématiques aident à résoudre des problèmes quotidiens.",

        "Pour calculer une surface : longueur × largeur.",

        "Les fractions représentent des parts d'un tout.",

        "Le théorème de Pythagore :  $a^2 + b^2 = c^2$  pour un triangle rectangle."

    ],

    "sciences": [

        "L'eau bout à 100°C au niveau de la mer.",

        "Les plantes ont besoin de lumière pour la photosynthèse.",

        "Le cycle de l'eau : évaporation, condensation, précipitation.",

        "La gravité fait tomber les objets vers le sol."

    ],

    "histoire madagascar": [

        "Madagascar fut peuplée il y a environ 2000 ans.",

        "Les royaumes malgaches dominaient l'île avant la colonisation.",

        "L'indépendance fut proclamée le 26 juin 1960.",

        "Antananarivo est la capitale depuis le royaume merina."

    ],

    "français": [

        "Un nom désigne une personne, un animal ou une chose.",

        "Le verbe exprime une action ou un état.",

        "L'adjectif qualifie le nom.",

        "Une phrase commence par une majuscule et finit par un point."

    ],

    "géographie": [
```

```

    "Madagascar est la 4ème plus grande île du monde.",
    "Le climat tropical domine dans le Nord.",
    "Les hauts plateaux occupent le centre de l'île.",
    "L'océan Indien entoure Madagascar."
]
}

def get_reponse_education(question):
    question_lower = question.lower()

    for sujet, reponses in responses_education.items():
        if sujet in question_lower:
            import random
            return random.choice(reponses)

    # Réponse par défaut personnalisée
    return f"Question intéressante sur '{question}'. Peux-tu préciser ta matière ? (mathématiques, sciences, histoire, français, géographie)"

EOF

```

Étape 2 : Créer l'interface éducative

```

cat > chatbot_education.py << 'EOF'

import streamlit as st

import random

from base_education import responses_education, get_reponse_education

# Configuration page
st.set_page_config(
    page_title="Assistant Éducatif Local",
    page_icon="🎓",
    layout="wide"
)

```

En-tête

```
st.title("🎓 Assistant Éducatif Local")
```

```
st.write("*Aide aux devoirs - Fonctionne sans Internet*")
```

```
st.markdown("---")
```

Sidebar avec informations

```
st.sidebar.header("📖 Matières disponibles")
```

```
st.sidebar.write("- 📊 Mathématiques")
```

```
st.sidebar.write("- 🧪 Sciences")
```

```
st.sidebar.write("- 📖 Histoire Madagascar")
```

```
st.sidebar.write("- 📝 Français")
```

```
st.sidebar.write("- 📖 Géographie")
```

```
st.sidebar.markdown("---")
```

```
st.sidebar.info("💡 Pose ta question dans ta matière préférée !")
```

Interface principale

```
col1, col2 = st.columns([2, 1])
```

with col1:

Sélection matière

```
matiere = st.selectbox(
```

```
    "Choisis ta matière :",
```

```
    ["Mathématiques", "Sciences", "Histoire Madagascar", "Français", "Géographie"],
```

```
    index=0
```

```
)
```

Zone de question

```
question = st.text_area(
```

```

"Pose ta question :",
placeholder="Exemple : Comment calculer l'aire d'un rectangle ?",
height=100
)

# Bouton de soumission
if st.button("🔍 Obtenir de l'aide", type="primary"):
    if question.strip():
        with st.spinner("😞 Je réfléchis..."):
            reponse = get_reponse_education(question)

            st.success("📖 Explication :")
            st.write(reponse)
            st.info("💡 Conseil : Demande à ton enseignant pour approfondir le sujet !")
        else:
            st.warning("⚠️ Pose d'abord ta question !")

with col2:
    st.subheader("🏆 Conseil du jour")
    conseil_jour = random.choice([
        "La lecture développe ton vocabulaire !",
        "Pratique les mathématiques chaque jour.",
        "Pose des questions, c'est important !",
        "L'histoire nous aide à comprendre le présent.",
        "Les sciences expliquent le monde qui nous entoure."
    ])
    st.info(conseil_jour)

    st.subheader("📊 Statistiques")
    st.metric("Questions disponibles", "20+")
    st.metric("Matières couvertes", "5")

```

```
st.metric("Langue", "Français")
```

```
# Footer
```

```
st.markdown("---")
```

```
st.markdown("**🏠 Assistant éducatif développé pour l'IA déconnectée - Adapté aux besoins locaux**")
```

```
EOF
```

Étape 3 : Test du chatbot éducatif

```
# Lancer l'application
```

```
streamlit run chatbot_education.py
```

URL d'accès : <http://localhost:8501>

DÉPLOIEMENT CHATBOT AGRICOLE

Étape 1 : Base de connaissances agricole

```
cat > base_agricole.py << 'EOF'
```

```
# Conseils agricoles adaptés à Madagascar
```

```
conseils_agricole = {
```

```
    "riz": {
```

```
        "plantation": "Période optimale : octobre-décembre selon les régions. Préparation des pépinières en septembre.",
```

```
        "maladies": [
```

```
            "Pyriculariose : taches brunes sur les feuilles. Traitement : variétés résistantes.",
```

```
            "Helminthosporiose : dessèchement des feuilles. Prévention : rotation des cultures."
```

```
        ],
```

```
        "conseils": "Rotation avec légumineuses recommandée. Drainage adapté essentiel. Espacement 20x20 cm."
```

```
    },
```

```
    "manioc": {
```

```
        "plantation": "Début saison des pluies (octobre-novembre). Boutures de 20 cm minimum.",
```

```
        "maladies": [
```

```
            "Mosaïque : jaunissement des feuilles. Solution : utiliser boutures saines.",
```

```
            "Bactériose : flétrissement brutal. Prévention : éviter blessures lors plantation."
```

```
        ],
```

```

    "conseils": "Espacement 1m x 1m. Buttage après 3 mois. Récolte après 8-12 mois."
},
"haricot": {
    "plantation": "Deux saisons : mars-avril ou septembre-octobre. Semis direct recommandé.",
    "maladies": [
        "Rouille : pustules orangées sur feuilles. Traitement : éviter arrosage feuillage.",
        "Anthracnose : taches noires sur gousses. Prévention : semences saines."
    ],
    "conseils": "Rotation obligatoire. Tuteurage pour variétés grimpantes. Récolte à 90 jours."
},
"maïs": {
    "plantation": "Octobre-décembre avec premières pluies. Densité : 50 000 plants/ha.",
    "maladies": [
        "Striure : rayures sur feuilles. Traitement : variétés tolérantes.",
        "Charbon : tumeurs noires sur épis. Prévention : rotation 3 ans minimum."
    ],
    "conseils": "Buttage indispensable. Apport fumier avant semis. Récolte à maturité complète."
},
"tomate": {
    "plantation": "Saison fraîche : avril-août. Pépinière puis repiquage à 6 semaines.",
    "maladies": [
        "Mildiou : taches brunes humides. Traitement : aération, éviter humidité.",
        "Flétrissement : plant qui sèche. Prévention : rotation, sol bien drainé."
    ],
    "conseils": "Tuteurage obligatoire. Arrosage au pied uniquement. Paillage recommandé."
}
}

def get_conseil_agricole(culture, type_conseil):
    culture_lower = culture.lower()

```



```
if culture_lower in conseils_agricole:
    data = conseils_agricole[culture_lower]
```

```
if "calendrier" in type_conseil.lower():
    return f"📅 {data['plantation']}"
elif "maladie" in type_conseil.lower():
    import random
    return f"🐛 {random.choice(data['maladies'])}"
elif "pratique" in type_conseil.lower():
    return f"💡 {data['conseils']}"
```

```
return f"Conseil non disponible pour {culture}. Cultures disponibles : riz, manioc, haricot, maïs, tomate."
```

```
EOF
```

Étape 2 : Interface agricole

```
cat > chatbot_agricole.py << 'EOF'
```

```
import streamlit as st
```

```
import random
```

```
from base_agricole import conseils_agricole, get_conseil_agricole
```

```
# Configuration
```

```
st.set_page_config(
    page_title="Conseiller Agricole Local",
    page_icon="🌱",
    layout="wide"
)
```

```
# En-tête
```

```
st.title("🌱 Conseiller Agricole Local")
```

```
st.write("*Conseils adaptés au climat malgache - Fonctionne sans Internet*")
```

```
st.markdown("---")
```

```
# Sidebar
```

```
st.sidebar.header("🌱 Cultures disponibles")
```

```
for culture in conseils_agricole.keys():
```

```
    st.sidebar.write(f"🌱 {culture.capitalize()}")
```

```
st.sidebar.markdown("---")
```

```
st.sidebar.header("📅 Calendrier général")
```

```
st.sidebar.write("**Saison des pluies **: Oct-Mars")
```

```
st.sidebar.write("**Saison sèche **: Avr-Sept")
```

```
st.sidebar.write("**Cyclones **: Déc-Mars")
```

```
# Interface principale
```

```
col1, col2 = st.columns([2, 1])
```

```
with col1:
```

```
    st.subheader("🎯 Votre consultation agricole")
```

```
# Sélection culture
```

```
culture = st.selectbox(
```

```
    "Quelle culture vous intéresse ?",
```

```
    ["Riz", "Manioc", "Haricot", "Maïs", "Tomate"],
```

```
    index=0
```

```
)
```

```
# Type de conseil
```

```
type_conseil = st.radio(
```

```
    "Type de conseil recherché :",
```

```
    ["📅 Calendrier plantation", "🐛 Problème maladie", "💡 Bonnes pratiques"],
```

```
    index=0
```

```
)
```

```

# Question libre (optionnel)
question_libre = st.text_area(
    "Question spécifique (optionnel) :",
    placeholder="Décrivez votre problème ou question...",
    height=80
)

if st.button("🌱 Obtenir conseil", type="primary"):
    with st.spinner("😓 Analyse en cours..."):
        conseil = get_conseil_agricole(culture, type_conseil)

    st.success("🌱 Conseil personnalisé :")
    st.write(conseil)

    if question_libre:
        st.info(f"📝 Votre question : {question_libre}")
        st.write("💬 Pour une réponse spécifique, consultez votre technicien agricole local.")

with col2:
    st.subheader("☀️ Météo agricole")
    st.info("Consultez les prévisions avant plantation")

    st.subheader("📊 Rendements moyens/ha")
    rendements = {
        "Riz": "3-4 tonnes",
        "Manioc": "15-20 tonnes",
        "Haricot": "1-1.5 tonne",
        "Maïs": "2-3 tonnes",
        "Tomate": "25-30 tonnes"
    }

    for culture, rendement in rendements.items():

```

```

    st.metric(f"{culture}", rendement)

# Section upload photo (simulation)
st.markdown("---")
st.subheader("📷 Diagnostic par photo")
uploaded_file = st.file_uploader(
    "Téléchargez une photo de votre plante (optionnel)",
    type=['jpg', 'jpeg', 'png']
)

if uploaded_file:
    st.image(uploaded_file, caption="Photo téléchargée", width=300)
    st.info("🔍 Pour un diagnostic précis, consultez votre agent agricole avec cette photo.")

# Footer
st.markdown("---")
st.markdown("*🌿 Conseiller agricole développé pour l'IA déconnectée - Adapté à Madagascar*")
EOF

```

📁 CONFIGURATION ASSISTANT SANTÉ

Étape 1 : Base de connaissances santé préventive

```

cat > base_sante.py << 'EOF'

# Informations de prévention santé validées
sante_prevention = {
    "paludisme": {
        "prevention": [
            "Utiliser une moustiquaire imprégnée chaque nuit",
            "Éliminer toutes les eaux stagnantes autour de la maison",
            "Porter des vêtements couvrants le soir",
            "Utiliser répulsifs anti-moustiques naturels (citronnelle)"
        ],
        "symptomes": "Fièvre, maux de tête, courbatures, frissons",
    }
}

```

```

    "action": "Consulter IMMÉDIATEMENT le centre de santé le plus proche"
  },
  "diarrhee": {
    "prevention": [
      "Boire uniquement de l'eau potable (bouillie ou traitée)",
      "Se laver les mains avant de manger et après les toilettes",
      "Cuire suffisamment tous les aliments",
      "Conserver les aliments au frais et couverts"
    ],
    "symptomes": "Selles liquides fréquentes, déshydratation, crampes",
    "action": "Réhydratation orale immédiate + consultation si persistant"
  },
  "hygiene": {
    "conseils": [
      "Lavage des mains aux 5 moments clés (OMS)",
      "Latrine à plus de 30m des points d'eau",
      "Conservation des aliments dans des récipients fermés",
      "Nettoyage quotidien des surfaces de préparation"
    ]
  },
  "grossesse": {
    "conseils": [
      "Consultation prénatale mensuelle obligatoire",
      "Prise d'acide folique et fer selon prescription",
      "Vaccination antitétanique à jour",
      "Éviter alcool, tabac et automédication"
    ],
    "urgences": "Saignements, maux de tête violents, vision trouble → URGENCE"
  },
  "enfant": {
    "conseils": [

```

```

        "Allaitement maternel exclusif 6 premiers mois",
        "Vaccinations selon calendrier national",
        "Surveillance croissance avec carnet de santé",
        "Vitamine A tous les 6 mois jusqu'à 5 ans"
    ],
    "urgences": "Fièvre haute, difficultés respiratoires, convulsions → URGENCE"
}
}

```

```
def get_info_sante(categorie):
```

```

    categorie_clean = categorie.lower().replace("✂", "").replace("💧", "").replace("👩",
    "").replace("👶", "").strip()

```

```
    if "paludisme" in categorie_clean:
```

```
        info = sante_prevention["paludisme"]
```

```

        return f"✂ **Prévention paludisme : **\n" + "\n".join([f"• {p}" for p in info["prevention"]]) +
        f"\n\n🚨 **Symptômes : ** {info['symptomes']}\n🏠 **Action : ** {info['action']}"

```

```
    elif "hygiene" in categorie_clean or "eau" in categorie_clean:
```

```

        conseils = sante_prevention["hygiene"]["conseils"] +
        sante_prevention["diarrhee"]["prevention"]

```

```
        return f"💧 **Hygiène eau/aliments : **\n" + "\n".join([f"• {c}" for c in conseils])
```

```
    elif "mère" in categorie_clean or "grossesse" in categorie_clean:
```

```
        info = sante_prevention["grossesse"]
```

```

        return f"👩 **Santé mère-enfant : **\n" + "\n".join([f"• {c}" for c in info["conseils"]]) + f"\n\n🚨
        **Urgences : ** {info['urgences']}"

```

```
    return "Information de santé non disponible. Consultez votre centre de santé."
```

```
EOF
```

Étape 2 : Interface santé sécurisée

```
cat > chatbot_sante.py << 'EOF'
```

```

import streamlit as st

from base_sante import sante_prevention, get_info_sante

# Configuration

st.set_page_config(

    page_title="Assistant Santé Communautaire",

    page_icon="🏠",

    layout="wide"

)


# AVERTISSEMENT OBLIGATOIRE

st.error("⚠️ IMPORTANT : Cette IA donne des informations générales de prévention, PAS de diagnostic médical")

st.warning("🏠 En cas de symptômes ou urgence : consultez immédiatement un professionnel de santé")


st.title("🏠 Assistant Santé Communautaire")

st.write("**Information préventive - Fonctionne sans Internet**")

st.markdown("---")


# Sidebar avec contacts d'urgence

st.sidebar.header("🚒 Contacts d'urgence")

st.sidebar.error("**SAMU : ** 15")

st.sidebar.error("**Pompiers : ** 18")

st.sidebar.error("**Police : ** 17")


st.sidebar.markdown("---")

st.sidebar.header("🏠 Centres de santé proches")

st.sidebar.write("- Centre Ambatondrazaka : 5km")

st.sidebar.write("- Hôpital Moramanga : 45km")

st.sidebar.write("- Dispensaire Andasibe : 12km")


st.sidebar.markdown("---")

```

```
st.sidebar.info("☎️ **Numéro vert santé : ** 910 (gratuit)")
```

```
# Interface principale
```

```
col1, col2 = st.columns([2, 1])
```

```
with col1:
```

```
    st.subheader("📄 Informations de prévention disponibles")
```

```
    categorie = st.selectbox(
```

```
        "Information recherchée :",
```

```
        [
```

```
            "🌿 Prévention paludisme",
```

```
            "💧 Hygiène eau/aliments",
```

```
            "👩 Santé mère-enfant",
```

```
            "👶 Santé de l'enfant",
```

```
            "🧼 Hygiène générale"
```

```
        ],
```

```
        index=0
```

```
    )
```

```
if st.button("📄 Obtenir information", type="primary"):
```

```
    with st.spinner("📄 Recherche d'informations..."):
```

```
        info = get_info_sante(categorie)
```

```
    st.info("📄 Information préventive :")
```

```
    st.markdown(info)
```

```
    st.warning("🚨 Rappel : En cas de symptômes, consultez le centre de santé le plus proche")
```

```
with col2:
```

```
    st.subheader("📊 Statistiques santé")
```



```

st.metric("Centres de santé", "3 à proximité")

st.metric("Vaccinations", "Gratuit -18 ans")

st.metric("Consultations", "Tarif social")


st.subheader("📅 Calendrier vaccinal")

st.write("***0-2 mois : ** BCG, Hépatite B")

st.write("***2-4-6 mois : ** DTC, Polio")

st.write("***9 mois : ** Rougeole")

st.write("***Femmes : ** Antitétanique")


# Section médicaments

st.markdown("---")

st.subheader("💊 Compréhension notice médicament")


notice_text = st.text_area(
    "Copiez ici le texte de votre notice médicament :",
    placeholder="Collez le texte de la notice que vous ne comprenez pas...",
    height=100
)


if notice_text and st.button("📖 Simplifier la notice"):
    st.info("📖 **Conseils pour comprendre : **")
    st.write("• Respectez la posologie indiquée")
    st.write("• Vérifiez les contre-indications")
    st.write("• Lisez les effets secondaires")
    st.write("• Conservez selon les instructions")

    st.warning("⚠️ Pour toute question sur votre traitement, consultez votre médecin ou pharmacien")


# Footer sécurisé

st.markdown("---")

```

st.error("🔒 **Disclaimer :** Cet assistant ne remplace jamais une consultation médicale")

st.markdown("**🏥 Assistant santé développé pour l'IA déconnectée - Information préventive uniquement**")

EOF

🏠 MISE EN PLACE GARDIEN CULTUREL

Étape 1 : Base patrimoine malgache

```
cat > base_culture.py << 'EOF'
```

```
# Patrimoine culturel malgache
```

```
patrimoine_malgache = {
```

```
    "proverbes": {
```

```
        "marina": {
```

```
            "malgache": "Ny marina tsy mba maty",
```

```
            "français": "La vérité ne meurt jamais",
```

```
            "explication": "Proverbe sur l'importance et la permanence de la vérité"
```

```
        },
```

```
        "fihavanana": {
```

```
            "malgache": "Ny fihavanana no harem-pirenena",
```

```
            "français": "La solidarité est la richesse de la nation",
```

```
            "explication": "Valeur fondamentale malgache de l'entraide mutuelle"
```

```
        },
```

```
        "vintana": {
```

```
            "malgache": "Ny vintana tsy azo atao",
```

```
            "français": "Le destin ne peut être forcé",
```

```
            "explication": "Acceptation respectueuse du cours des événements"
```

```
        }
```

```
    },
```

```
    "contes": {
```

```
        "ikoto_imahaka": {
```

```
            "titre": "Ikoto sy Imahaka",
```

```
            "resume": "Conte populaire sur deux frères aux caractères opposés",
```

```

    "morale": "L'intelligence et la gentillesse valent mieux que la force",
    "personnages": "Ikoto (malin), Imahaka (costaud mais bête)"
  },
  "andriambahoaka": {
    "titre": "Andriambahoaka",
    "resume": "Prince qui découvre la vraie richesse parmi son peuple",
    "morale": "La vraie richesse vient de la générosité et du service",
    "personnages": "Prince Andriambahoaka, villageois"
  }
},

```

```

"traditions": {
  "famadihana": {
    "nom": "Famadihana",
    "description": "Cérémonie de retournement des morts",
    "periode": "Juillet à septembre (saison sèche)",
    "signification": "Renouvellement du lien avec les ancêtres"
  },
  "saotra": {
    "nom": "Saotra",
    "description": "Offrande et prière aux ancêtres",
    "occasion": "Événements importants, demandes de bénédiction",
    "signification": "Respect et communication avec les razana"
  },
  "kabary": {
    "nom": "Kabary",
    "description": "Art oratoire traditionnel malgache",
    "usage": "Cérémonies officielles, mariages, réconciliations",
    "signification": "Éloquence et sagesse dans la parole"
  }
},

```

```

"artisanat": {
    "lamba": {
        "nom": "Lamba",
        "description": "Tissu traditionnel en soie sauvage",
        "origine": "Hauts plateaux, élevage vers à soie",
        "usage": "Vêtement de cérémonie, linceul"
    },
    "vannerie": {
        "nom": "Vannerie raphia",
        "description": "Tressage de raphia et bambou",
        "produits": "Paniers, chapeaux, nattes",
        "regions": "Côte Est principalement"
    }
}
}

```

```

def get_info_culturelle(categorie, element=None):
    if "proverbe" in categorie.lower():
        if element and element in patrimoine_malgache["proverbes"]:
            p = patrimoine_malgache["proverbes"][element]
            return f"""Malgache : ** {p['malgache']} \n Français : ** {p['français']} \n Explication : **
{p['explication']} """
        else:
            import random
            p = random.choice(list(patrimoine_malgache["proverbes"].values()))
            return f"""Malgache : ** {p['malgache']} \n Français : ** {p['français']} \n Explication : **
{p['explication']} """

    elif "conte" in categorie.lower():
        if element and element in patrimoine_malgache["contes"]:
            c = patrimoine_malgache["contes"][element]

```

```

        return f"***{c['titre']}**\n{c['resume']}\n**Morale : ** {c['morale']}\n**Personnages : **\n{c['personnages']}"
    else:
        import random

        c = random.choice(list(patrimoine_malgache["contes"].values()))

        return f"***{c['titre']}**\n{c['resume']}\n**Morale : ** {c['morale']}"

elif "tradition" in categorie.lower():
    if element and element in patrimoine_malgache["traditions"]:
        t = patrimoine_malgache["traditions"][element]

        return f"***{t['nom']}**\n{t['description']}\n**Signification : ** {t['signification']}"
    else:
        traditions = list(patrimoine_malgache["traditions"].values())

        return "\n\n".join([f"***{t['nom']}**\n{t['description']}" for t in traditions[:2]])

return "Information culturelle non disponible."

# Fonction traduction simple (simulation)
def traduire_simple(phrase, direction="fr_to_mg"):
    traductions_base = {
        "bonjour": "salama",
        "merci": "misaotra",
        "au revoir": "veloma",
        "oui": "eny",
        "non": "tsia",
        "eau": "rano",
        "nourriture": "sakafo",
        "maison": "trano",
        "famille": "fianakaviana",
        "enfant": "zaza",
        "école": "sekoly"
    }

```

```
}
```

```
phrase_lower = phrase.lower()
```

```
if direction == "fr_to_mg":
```

```
    for fr, mg in traductions_base.items():
```

```
        if fr in phrase_lower:
```

```
            return phrase_lower.replace(fr, mg)
```

```
else:
```

```
    for fr, mg in traductions_base.items():
```

```
        if mg in phrase_lower:
```

```
            return phrase_lower.replace(mg, fr)
```

```
    return f"Traduction non disponible pour: {phrase}"
```

```
EOF
```

Étape 2 : Interface gardien culturel

```
cat > chatbot_culture.py << 'EOF'
```

```
import streamlit as st
```

```
import random
```

```
from base_culture import patrimoine_malgache, get_info_culturelle, traduire_simple
```

```
# Configuration
```

```
st.set_page_config(
```

```
    page_title="Gardien du Patrimoine Malgache",
```

```
    page_icon="🏛️",
```

```
    layout="wide"
```

```
)
```

```
# En-tête culturel
```

```
st.title("🏛️ Gardien du Patrimoine Malgache")
```

```
st.write("**Préservation et transmission culturelle - Fonctionne sans Internet**")
```

```
st.markdown("----")
```

```
# Sidebar culturelle
```

```
st.sidebar.header("🏠 Patrimoine disponible")
```

```
st.sidebar.write("- 💬 Proverbes traditionnels")
```

```
st.sidebar.write("- 📖 Contes populaires")
```

```
st.sidebar.write("- 🏡 Traditions ancestrales")
```

```
st.sidebar.write("- 🧶 Artisanat local")
```

```
st.sidebar.markdown("---")
```

```
st.sidebar.header("🌐 Langues")
```

```
st.sidebar.write("- MG Malgache (officiel)")
```

```
st.sidebar.write("- FR Français (officiel)")
```

```
st.sidebar.write("- 🔄 Traduction basique")
```

```
# Interface principale avec onglets
```

```
tabs = st.tabs(["📖 Contes", "💬 Proverbes", "🏠 Traditions", "🧶 Artisanat", "🔄 Traduction"])
```

```
with tabs[0]:
```

```
    st.subheader("📖 Contes traditionnels malgaches")
```

```
    conte_choisi = st.selectbox(
```

```
        "Choisis un conte :",
```

```
        ["Ikoto sy Imahaka", "Andriambahoaka", "Conte aléatoire"]
```

```
    )
```

```
    if st.button("📖 Découvrir le conte", key="conte"):
```

```
        if conte_choisi == "Conte aléatoire":
```

```
            info = get_info_culturelle("conte")
```

```
        else:
```

```
            key = conte_choisi.lower().replace(" ", "_").replace("sy", "imahaka")
```

```

info = get_info_culturelle("conte", key)

st.success("🗨️ Histoire traditionnelle :")

st.markdown(info)

st.info("📖 Ces contes transmettent les valeurs malgaches de génération en génération")

with tabs[1]:

    st.subheader("🌟 Proverbes malgaches")

    if st.button("🎲 Proverbe aléatoire", key="proverbe"):

        proverbe_info = get_info_culturelle("proverbe")

        st.success("🗨️ Ohabolana (Proverbe) :")

        st.markdown(proverbe_info)

    st.markdown("----")

    st.write("***🌟 Proverbes populaires disponibles :***")

    for key, p in patrimoine_malgache["proverbes"].items():

        with st.expander(f"🗨️ {p['malgache']}"):

            st.write(f"***Français :*** {p['français']}")

            st.write(f"***Explication :*** {p['explication']}")

with tabs[2]:

    st.subheader("🏠 Traditions ancestrales")

    tradition = st.selectbox(

        "Sélectionne une tradition :",

        ["Famadihana", "Saotra", "Kabary", "Toutes les traditions"]

    )

    if st.button("🗨️ En savoir plus", key="tradition"):

```



```

if tradition == "Toutes les traditions":
    info = get_info_culturelle("tradition")
else:
    info = get_info_culturelle("tradition", tradition.lower())

st.success("🏛️ Tradition malgache :")
st.markdown(info)
st.info("🌟 Ces traditions maintiennent le lien avec les ancêtres (razana)")

```

with tabs[3]:

```
st.subheader("🧶 Artisanat traditionnel")
```

```
col1, col2 = st.columns(2)
```

with col1:

```

st.write("** 🧶 Lamba (Soie sauvage)**")
st.write("- Tissu cérémoniel traditionnel")
st.write("- Élevage de vers à soie")
st.write("- Symbole de prestige")

```

with col2:

```

st.write("** 🧺 Vannerie Raphia**")
st.write("- Tressage bambou et raphia")
st.write("- Paniers, chapeaux, nattes")
st.write("- Savoir-faire ancestral")

```

```
if st.button("👉 Plus d'informations artisanat"):
```

```

    st.info("🧡 L'artisanat malgache reflète la richesse des ressources naturelles et l'habileté des artisans locaux")

```

with tabs[4]:

```
st.subheader("🔄 Traduction Français ↔ Malgache")
```

```
direction = st.radio(
    "Direction de traduction :",
    ["FR → MG Français vers Malgache", "MG → FR Malgache vers Français"]
)
```

```
phrase = st.text_input(
    "Tapez votre phrase :",
    placeholder="Exemple: bonjour, merci, au revoir..."
)
```

```
if phrase and st.button("🔄 Traduire"):
    if "Français vers Malgache" in direction:
        traduction = traduire_simple(phrase, "fr_to_mg")
    else:
        traduction = traduire_simple(phrase, "mg_to_fr")
```

```
st.success("📄 Traduction :")
st.write(f"***Résultat : ** {traduction}")
st.info("💡 Traduction basique - Pour du vocabulaire avancé, consultez un locuteur natif")
```

```
# Section interactive
```

```
st.markdown("---")
```

```
st.subheader("🎯 Quiz culturel interactif")
```

```
if st.button("🎲 Question surprise"):
    questions = [
        {
            "question": "Que signifie 'Ny fihavanana no harem-pirenena' ?",
            "reponse": "La solidarité est la richesse de la nation",
```

```

        "explication": "Valeur fondamentale malgache"
    },
    {
        "question": "Qu'est-ce que le Famadihana ?",
        "reponse": "Cérémonie de retournement des morts",
        "explication": "Tradition de lien avec les ancêtres"
    },
    {
        "question": "Comment dit-on merci en malgache ?",
        "reponse": "Misaoitra",
        "explication": "Expression de gratitude courante"
    }
]

```

```

quiz = random.choice(questions)
st.question = st.empty()
st.question.info(f"🔍 **Question :** {quiz['question']}")

```

```

if st.button("Voir la réponse", key="reponse_quiz"):
    st.success(f"✅ **Réponse :** {quiz['reponse']}")
    st.write(f"📖 **Explication :** {quiz['explication']}")

```

Footer culturel

```
st.markdown("---")
```

```
st.markdown(f"🏠 Gardien culturel développé pour l'IA déconnectée - Préservation du patrimoine malgache")
```

```
st.markdown(f"🌟 'Ny razana no loharano' - Les ancêtres sont la source")
```

EOF

TEMPLATE PERSONNALISABLE

Étape 1 : Template de base adaptable

```
cat > template_personnalisable.py << 'EOF'
```

```

import streamlit as st

import random


# =====

# CONFIGURATION - À PERSONNALISER

# =====


# Titre et description de votre assistant

TITRE_ASSISTANT = "🎯 Votre Assistant Local Personnalisé"

DESCRIPTION = "Adaptez ce titre et cette description à votre cas d'usage"

ICONE_PAGE = "🎯"


# Votre domaine d'expertise (À COMPLÉTER)

VOTRE_BASE_CONNAISSANCES = {
    "categorie_exemple_1": {
        "mot_cle_1": [
            "Première réponse possible pour ce mot-clé",
            "Deuxième réponse alternative",
            "Troisième réponse variante"
        ],
        "mot_cle_2": [
            "Réponse pour le deuxième mot-clé",
            "Autre réponse possible"
        ]
    },
    "categorie_exemple_2": {
        "mot_cle_3": [
            "Réponse pour la catégorie 2",
            "Réponse alternative catégorie 2"
        ]
    }
}

```

```

# AJOUTEZ VOS PROPRES CATÉGORIES ICI
}

# Options pour interface (À ADAPTER)
OPTIONS_CATEGORIES = list(VOTRE_BASE_CONNAISSANCES.keys())

# =====

# LOGIQUE DE RÉPONSE - À ADAPTER
# =====

def generer_reponse_personnalisee(question, categorie_selectionnee):
    """
    Fonction principale de génération de réponses
    Adaptez cette logique selon vos besoins
    """
    question_lower = question.lower()

    # Recherche dans la catégorie sélectionnée
    if categorie_selectionnee in VOTRE_BASE_CONNAISSANCES:
        for mot_cle, reponses in VOTRE_BASE_CONNAISSANCES[categorie_selectionnee].items():
            if mot_cle.lower() in question_lower:
                return random.choice(reponses)

    # Recherche globale si pas trouvé dans catégorie
    for categorie, contenus in VOTRE_BASE_CONNAISSANCES.items():
        for mot_cle, reponses in contenus.items():
            if mot_cle.lower() in question_lower:
                return f"[Trouvé dans {categorie}] " + random.choice(reponses)

    # Réponse par défaut personnalisable

```

```
    return f"Question intéressante sur '{question}'. Pouvez-vous préciser ou reformuler ? Les domaines disponibles sont : {' '.join(OPTIONS_CATEGORIES)}"
```

```
# =====
```

```
# INTERFACE STREAMLIT - À PERSONNALISER
```

```
# =====
```

```
def main():
```

```
    # Configuration page
```

```
    st.set_page_config(
        page_title=TITRE_ASSISTANT,
        page_icon=ICONE_PAGE,
        layout="wide"
    )
```

```
    # En-tête principal
```

```
    st.title(TITRE_ASSISTANT)
    st.write(f"*{DESCRIPTION}*")
    st.markdown("---")
```

```
    # Sidebar personnalisable
```

```
    st.sidebar.header("📖 Domaines disponibles")
    for categorie in OPTIONS_CATEGORIES:
        st.sidebar.write(f"- 📄 {categorie.replace('_', ' ').title()}")
```

```
    st.sidebar.markdown("----")
```

```
    st.sidebar.info("💡 Personnalisez cette sidebar selon vos besoins !")
```

```
    # Interface principale
```

```
    col1, col2 = st.columns([2, 1])
```

with col1:

```
st.subheader("💬 Votre consultation")
```

```
# Sélection catégorie
```

```
categorie = st.selectbox(
    "Choisissez votre domaine :",
    OPTIONS_CATEGORIES,
    format_func=lambda x: x.replace('_', ' ').title()
)
```

```
# Zone de question
```

```
question = st.text_area(
    "Posez votre question :",
    placeholder="Décrivez votre besoin ou question...",
    height=100
)
```

```
# Bouton principal
```

```
if st.button("👉 Obtenir réponse", type="primary"):
    if question.strip():
        with st.spinner("😓 Analyse en cours..."):
            reponse = generer_reponse_personnalisee(question, categorie)

            st.success("💡 Réponse personnalisée :")
            st.write(reponse)

            st.info("📝 Conseil : Affinez votre question si besoin !")
    else:
        st.warning("⚠️ Veuillez d'abord poser votre question !")
```

with col2:

```
st.subheader("📊 Informations")
```

```

st.metric("Catégories", len(OPTIONS_CATEGORIES))

st.metric("Réponses", sum(len(contenus) for contenus in
VOTRE_BASE_CONNAISSANCES.values()))

st.metric("Statut", "Hors ligne )

# Zone personnalisable

st.subheader(" À personnaliser")

st.info("Ajoutez ici vos propres métriques, conseils ou informations contextuelles")


# Section optionnelle avancée

with st.expander(" Options avancées"):

    st.write("**  Personnalisations possibles :**")

    st.write("- Ajouter upload de fichiers")

    st.write("- Intégrer images/photos")

    st.write("- Créer système de feedback")

    st.write("- Ajouter export des réponses")

    st.write("- Implémenter historique")


# Footer personnalisable

st.markdown("---")

st.markdown("**  Assistant développé avec l'approche IA Déconnectée - Adaptez selon vos besoins**")

```



```

# =====

# GUIDE D'ADAPTATION

# =====

def afficher_guide_adaptation():

    st.title("📖 Guide d'adaptation du template")

    st.markdown("""
## 🎯 Comment personnaliser ce template

### 1. **Modifiez les constantes de configuration**

```python
TITRE_ASSISTANT = "👨‍⚕️ Assistant Médical Local" # Votre titre
DESCRIPTION = "Conseils de santé communautaire" # Votre description
ICONE_PAGE = "📖" # Votre icône
```

### 2. **Créez votre base de connaissances**

```python
VOTRE_BASE_CONNAISSANCES = {
 "votre_domaine": {
 "mot_cle_important": [
 "Votre première réponse experte",
 "Votre deuxième réponse alternative"
]
 }
}
```

### 3. **Adaptez la logique de réponse**

Modifiez `generer_reponse_personnalisee()` selon vos besoins spécifiques.

```

4. ****Personnalisez l'interface****

- Changez les couleurs et mise en page
- Ajoutez vos métriques spécifiques
- Intégrez vos fonctionnalités uniques

5. ****Testez et itérez****

- Testez avec vos utilisateurs cibles
- Collectez les questions fréquentes
- Enrichissez progressivement votre base

""")

=====

LANCEMENT APPLICATION

=====

if __name__ == "__main__":

 # Décommentez la ligne suivante pour afficher le guide

 # afficher_guide_adaptation()

 # Application principale

 main()

EOF

Étape 2 : Exemple d'adaptation - Commerce local

cat > exemple_commerce.py << 'EOF'

import streamlit as st

import random

Assistant pour commerce local - Exemple d'adaptation

TITRE_ASSISTANT = "🛒 Assistant Commerce Local"

DESCRIPTION = "Conseils gestion et comptabilité - Fonctionne sans Internet"

```

base_commerce = {
  "comptabilite": {
    "tva": [
      "TVA à 20% : Prix HT × 1,20 = Prix TTC",
      "Pour déduire la TVA : Prix TTC ÷ 1,20 = Prix HT",
      "TVA collectée - TVA déductible = TVA à payer"
    ],
    "benefice": [
      "Bénéfice = Chiffre d'affaires - Charges",
      "Marge brute = (Prix vente - Prix achat) ÷ Prix vente × 100",
      "Seuil rentabilité = Charges fixes ÷ Taux marge"
    ],
    "tresorerie": [
      "Cash-flow = Recettes - Dépenses sur période",
      "Surveillez les délais clients vs fournisseurs",
      "Prévision trésorerie indispensable"
    ]
  },
  "gestion": {
    "stock": [
      "Stock optimal = (Vente moyenne × délai) + Stock sécurité",
      "Rotation stock = Coût ventes annuel ÷ Stock moyen",
      "Inventaire physique au minimum 1 fois/an"
    ],
    "client": [
      "Fidéliser coûte 5x moins cher que conquérir",
      "Carte fidélité simple = +15% chiffre d'affaires",
      "Écoutez les réclamations = opportunités"
    ],
    "fournisseur": [

```

```

        "Négociez paiement 30 jours minimum",
        "Diversifiez vos sources d'approvisionnement",
        "Comparez prix mais aussi service"
    ]
},
"marketing": {
    "promotion": [
        "Promotion 20% = besoin +25% volume pour même CA",
        "Période creuse = meilleur moment promotions",
        "Promo sur produits à forte marge uniquement"
    ],
    "vitrine": [
        "Changez vitrine toutes les 2 semaines",
        "Éclairage LED = -60% consommation",
        "Prix visibles et lisibles obligatoire"
    ]
}
}

def conseils_commerce(question, domaine):
    question_lower = question.lower()

    if domaine in base_commerce:
        for sujet, conseils in base_commerce[domaine].items():
            if sujet in question_lower:
                return random.choice(conseils)

    return f"Conseil commercial : Pour '{question}', précisez votre domaine (comptabilité, gestion, marketing)"

# Interface

```

```

st.set_page_config(page_title="Assistant Commerce", page_icon="🏪")
st.title("🏪 Assistant Commerce Local")
st.write("*Conseils gestion TPE/PME - Sans Internet*")

domaine = st.selectbox("Domaine :", ["comptabilite", "gestion", "marketing"])
question = st.text_area("Votre question commerce :")

if st.button("💡 Conseil personnalisé"):
    if question:
        conseil = conseils_commerce(question, domaine)
        st.success("🎯 Conseil professionnel :")
        st.write(conseil)
        st.info("📊 Pour un accompagnement personnalisé, consultez un expert-comptable")

st.sidebar.header("📊 Ratios utiles")
st.sidebar.metric("CA/employé", "150K€/an")
st.sidebar.metric("Marge brute min", "30%")
st.sidebar.metric("Rotation stock", "6-12x/an")
EOF

```

✅ TESTS ET VALIDATION

Test de fonctionnement global

```

# Script de test automatisé
cat > test_complet.py << 'EOF'

#!/usr/bin/env python3

import subprocess
import time
import requests
import sys

```

```
print("🔧 Test complet IA Déconnectée - Cas d'usage")
print("=" * 50)
```

```
def test_streamlit_install():
    """Test installation Streamlit"""
    try:
        import streamlit

        print("✅ Streamlit installé correctement")

        return True
    except ImportError:
        print("❌ Streamlit non installé")

        return False
```

```
def test_bases_donnees():
    """Test bases de connaissances"""
    tests = [
        ("base_education.py", "responses_education"),
        ("base_agricole.py", "conseils_agricole"),
        ("base_sante.py", "sante_prevention"),
        ("base_culture.py", "patrimoine_malgache")
    ]

    succes = 0

    for fichier, variable in tests:
        try:
            with open(fichier, 'r') as f:
                contenu = f.read()

                if variable in contenu:
                    print(f"✅ {fichier} : Base de données présente")

                    succes += 1
        except:
            else:
```

```

        print(f"❌ {fichier} : Variable {variable} manquante")
    except FileNotFoundError:
        print(f"❌ {fichier} : Fichier non trouvé")

    return succes == len(tests)

def test_interfaces():
    """Test fichiers interfaces"""
    interfaces = [
        "chatbot_education.py",
        "chatbot_agricole.py",
        "chatbot_sante.py",
        "chatbot_culture.py",
        "template_personnalisable.py"
    ]

    succes = 0
    for interface in interfaces:
        try:
            with open(interface, 'r') as f:
                if "streamlit" in f.read():
                    print(f"✅ {interface} : Interface Streamlit valide")
                    succes += 1
        except FileNotFoundError:
            print(f"❌ {interface} : Fichier manquant")

    return succes == len(interfaces)

def lancer_test_interface(fichier, port=8501):
    """Lance une interface pour test"""
    print(f"🚀 Test {fichier} sur port {port}")

```

try:

Lancer Streamlit en arrière-plan

```
process = subprocess.Popen([
    "streamlit", "run", fichier, "--server.port", str(port)
], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
```

Attendre démarrage

```
time.sleep(3)
```

Tester connexion

```
response = requests.get(f"http://localhost:{port}")
```

```
if response.status_code == 200:
```

```
    print(f"✅ {fichier} : Interface accessible")
```

```
    success = True
```

```
else:
```

```
    print(f"❌ {fichier} : Interface non accessible")
```

```
    success = False
```

Arrêter processus

```
process.terminate()
```

```
return success
```

```
except Exception as e:
```

```
    print(f"❌ {fichier} : Erreur {e}")
```

```
    return False
```

Exécution tests

```
if __name__ == "__main__":
```

```
    print("\n1 Test installation...")
```

```
    test1 = test_streamlit_install()
```



```

print("\n 2 Test bases de données...")
test2 = test_bases_donnees()

print("\n 3 Test interfaces...")
test3 = test_interfaces()

if all([test1, test2, test3]):
    print("\n 🎉 TOUS LES TESTS RÉUSSIS !")
    print("Votre environnement IA déconnectée est prêt !")
else:
    print("\n ⚠️ Certains tests ont échoué")
    print("Vérifiez les erreurs ci-dessus")
    sys.exit(1)

```

EOF

Rendre exécutable et lancer

```
chmod +x test_complet.py
```

```
python3 test_complet.py
```

Test manuel des interfaces

Test individuel de chaque chatbot

```
echo " 🖋️ Lancement tests manuels..."
```

Test 1 : Éducation

```
echo " 📖 Test chatbot éducatif"
```

```
streamlit run chatbot_education.py --server.port 8501 &
```

```
sleep 3
```

```
echo " ➡️ Testez sur http://localhost:8501"
```

```
read -p "Appuyez sur Entrée après test..."
```

```
kill -f "streamlit run chatbot_education.py"
```

Test 2 : Agriculture

```
echo "🌾 Test chatbot agricole"

streamlit run chatbot_agricole.py --server.port 8502 &

sleep 3

echo "➡️ Testez sur http://localhost:8502"

read -p "Appuyez sur Entrée après test..."

pkill -f "streamlit run chatbot_agricole.py"
```

Test 3 : Santé

```
echo "🏥 Test assistant santé"

streamlit run chatbot_sante.py --server.port 8503 &

sleep 3

echo "➡️ Testez sur http://localhost:8503"

read -p "Appuyez sur Entrée après test..."

pkill -f "streamlit run chatbot_sante.py"
```

Test 4 : Culture

```
echo "🏛️ Test gardien culturel"

streamlit run chatbot_culture.py --server.port 8504 &

sleep 3

echo "➡️ Testez sur http://localhost:8504"

read -p "Appuyez sur Entrée après test..."

pkill -f "streamlit run chatbot_culture.py"
```

```
echo "✅ Tests manuels terminés"
```

🔧 DÉPANNAGE

Problèmes courants et solutions

Erreur : "streamlit command not found"

Solution 1 : Réinstaller streamlit

```
pip3 install --upgrade streamlit
```

Solution 2 : Vérifier PATH

echo \$PATH

export PATH=\$PATH:~/local/bin

Solution 3 : Utiliser python -m

python3 -m streamlit run votre_fichier.py

Erreur : "Port already in use"

Trouver processus utilisant le port

lsof -i :8501

Arrêter processus

pkill -f streamlit

Utiliser autre port

streamlit run chatbot_education.py --server.port 8502

Erreur : "Module not found"

Vérifier modules installés

pip3 list | grep streamlit

Réinstaller si nécessaire

pip3 uninstall streamlit

pip3 install streamlit

Vérifier version Python

python3 --version # Doit être 3.10+

Interface ne s'affiche pas correctement

Vider cache Streamlit

streamlit cache clear

Relancer avec debug

streamlit run chatbot_education.py --logger.level debug

```
# Vérifier navigateur (Chrome/Firefox recommandés)
```

Base de connaissances vide

```
# Vérifier contenu fichier
```

```
cat base_education.py | head -20
```

```
# Vérifier encodage
```

```
file base_education.py
```

```
# Recréer si nécessaire avec cat
```

```
cat > base_education.py << 'EOF'
```

```
# Contenu ici
```

```
EOF
```

Logs et diagnostic

```
# Créer script diagnostic
```

```
cat > diagnostic.py << 'EOF'
```

```
import sys
```

```
import streamlit
```

```
import os
```

```
print("🔍 Diagnostic IA Déconnectée")
```

```
print(f"Python version: {sys.version}")
```

```
print(f"Streamlit version: {streamlit.__version__}")
```

```
print(f"Répertoire actuel: {os.getcwd()}")
```

```
print(f"Fichiers présents:")
```

```
for f in os.listdir('.'):

```

```
    if f.endswith('.py'):

```

```
        print(f" - {f}")
```

```
EOF
```

```
python3 diagnostic.py
```

MAINTENANCE ET ÉVOLUTION

Mise à jour des bases de connaissances

```
# Script de sauvegarde

cat > sauvegarde.sh << 'EOF'

#!/bin/bash

DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p sauvegardes/$DATE

# Sauvegarder bases de données

cp base_*.py sauvegardes/$DATE/

cp chatbot_*.py sauvegardes/$DATE/

echo "  Sauvegarde créée dans sauvegardes/$DATE"

EOF

chmod +x sauvegarde.sh

./sauvegarde.sh
```

Enrichissement communautaire

```
# Template ajout questions fréquentes

cat > enrichir_base.py << 'EOF'

# Script pour enrichir les bases de connaissances

import json

from datetime import datetime

def ajouter_question_reponse(domaine, categorie, question, reponse):

    """Ajouter une nouvelle Q&R à la base"""

    # Charger base existante

    fichier_base = f"base_{domaine}.py"

    # Format d'ajout
```

```
nouvelle_entree = {
    "question": question,
    "reponse": reponse,
    "date_ajout": datetime.now().strftime("%Y-%m-%d"),
    "source": "communautaire"
}
```

Log des ajouts

```
with open("ajouts_communautaires.json", "a") as f:
    json.dump(nouvelle_entree, f)
    f.write("\n")
```

```
print(f"✅ Ajouté: {question} → {reponse}")
```

```
print(f"📄 Domaine: {domaine}, Catégorie: {categorie}")
```

Exemple d'utilisation

```
if __name__ == "__main__":
    print("🔄 Enrichissement base de connaissances")
```

Exemple éducation

```
ajouter_question_reponse(
    domaine="education",
    categorie="mathematiques",
    question="Comment calculer un pourcentage",
    reponse="Pourcentage = (Partie ÷ Total) × 100"
)
```

Exemple agriculture

```
ajouter_question_reponse(
    domaine="agricole",
    categorie="riz",
```

```

    question="Quand planter le riz à Antananarivo",
    reponse="Plantation optimale: novembre-décembre avec les premières pluies"
)
EOF

```

Script de mise à jour automatique

```

cat > mise_a_jour.py << 'EOF'

#!/usr/bin/env python3

import os

import shutil

from datetime import datetime

def mettre_a_jour_bases():
    """Met à jour les bases de connaissances"""

    print("🔄 Mise à jour des bases de connaissances...")

    # Créer backup
    backup_dir = f"backup_{datetime.now().strftime('%Y%m%d_%H%M%S')}"
    os.makedirs(backup_dir, exist_ok=True)

    # Sauvegarder fichiers actuels
    fichiers_a_sauver = [
        "base_education.py",
        "base_agricole.py",
        "base_sante.py",
        "base_culture.py"
    ]

    for fichier in fichiers_a_sauver:
        if os.path.exists(fichier):
            shutil.copy2(fichier, backup_dir)

```

```
print(f"✅ Sauvegardé: {fichier}")
```

```
print(f"📁 Backup créé dans: {backup_dir}")
```

```
# Ici, ajouter logique de mise à jour
```

```
# Ex: télécharger nouvelles versions, merger contenu, etc.
```

```
if __name__ == "__main__":
```

```
    mettre_a_jour_bases()
```

```
EOF
```

```
chmod +x mise_a_jour.py
```

Collecte feedback utilisateurs

```
cat > collecte_feedback.py << 'EOF'
```

```
import streamlit as st
```

```
import json
```

```
from datetime import datetime
```

```
def interface_feedback():
```

```
    """Interface de collecte de feedback"""
```

```
st.title("📝 Améliorer votre IA locale")
```

```
st.write("Aidez-nous à enrichir les réponses !")
```

```
# Formulaire feedback
```

```
with st.form("feedback_form"):
```

```
    domaine = st.selectbox(
```

```
        "Domaine concerné:",
```

```
        ["Éducation", "Agriculture", "Santé", "Culture", "Autre"]
```

```
    )
```



```

question_posee = st.text_area(
    "Question que vous avez posée:",
    placeholder="Ex: Comment calculer une surface ?"
)

reponse_obtenue = st.text_area(
    "Réponse obtenue de l'IA:",
    placeholder="Copiez la réponse reçue..."
)

evaluation = st.radio(
    "Cette réponse était-elle utile ?",
    ["😊 Très utile", "🙂 Utile", "😐 Moyenne", "😞 Peu utile", "😡 Inutile"]
)

suggestion = st.text_area(
    "Suggestion d'amélioration:",
    placeholder="Comment améliorer cette réponse ?"
)

submitted = st.form_submit_button("📤 Envoyer feedback")

if submitted:
    # Sauvegarder feedback
    feedback_data = {
        "timestamp": datetime.now().isoformat(),
        "domaine": domaine,
        "question": question_posee,
        "reponse": reponse_obtenue,
        "evaluation": evaluation,
        "suggestion": suggestion
    }

```

```
}
```

```
# Ajouter au fichier feedback
```

```
with open("feedback_utilisateurs.json", "a") as f:
```

```
    json.dump(feedback_data, f, ensure_ascii=False)
```

```
    f.write("\n")
```

```
st.success("✅ Merci pour votre feedback !")
```

```
st.info("Vos suggestions aideront à améliorer l'IA pour toute la communauté")
```

```
if __name__ == "__main__":
```

```
    interface_feedback()
```

```
EOF
```

Monitoring utilisation

```
cat > monitoring.py << 'EOF'
```

```
import json
```

```
import pandas as pd
```

```
from datetime import datetime, timedelta
```

```
import streamlit as st
```

```
def analyser_utilisation():
```

```
    """Analyse l'utilisation des chatbots"""
```

```
st.title("📊 Statistiques d'utilisation IA locale")
```

```
# Simuler données d'usage (à remplacer par vrais logs)
```

```
stats_exemple = {
```

```
    "education": {"questions": 45, "satisfaction": 4.2},
```

```
    "agricole": {"questions": 38, "satisfaction": 4.5},
```

```
    "sante": {"questions": 22, "satisfaction": 4.0},
```

```
    "culture": {"questions": 15, "satisfaction": 4.8}
```

```
}
```

```
col1, col2, col3 = st.columns(3)
```

```
with col1:
```

```
    st.metric(
        "Questions totales",
        sum(s["questions"] for s in stats_exemple.values())
    )
```

```
with col2:
```

```
    avg_satisfaction = sum(s["satisfaction"] for s in stats_exemple.values()) / len(stats_exemple)
    st.metric("Satisfaction moyenne", f"{avg_satisfaction:.1f}/5")
```

```
with col3:
```

```
    st.metric("Domaines actifs", len(stats_exemple))
```

```
# Graphiques
```

```
st.subheader("📊 Usage par domaine")
```

```
df_stats = pd.DataFrame([
    {"Domaine": k.title(), "Questions": v["questions"], "Satisfaction": v["satisfaction"]}
    for k, v in stats_exemple.items()
])
```

```
st.bar_chart(df_stats.set_index("Domaine")["Questions"])
```

```
# Recommandations
```

```
st.subheader("💡 Recommandations")
```

```
domaine_populaire = max(stats_exemple.keys(), key=lambda x:
stats_exemple[x]["questions"])
```

```
domaine_satisfaisant = max(stats_exemple.keys(), key=lambda x:
stats_exemple[x]["satisfaction"])
```

```
st.info(f"🔥 Domaine le plus utilisé: **{domaine_populaire.title()}**")
```

```
st.info(f"★ Domaine le plus apprécié: **{domaine_satisfaisant.title()}**")
```

```
if st.button("📊 Générer rapport complet"):
```

```
    st.download_button(
```

```
        label="📁 Télécharger rapport",
```

```
        data=df_stats.to_csv(index=False),
```

```
        file_name=f"rapport_ia_locale_{datetime.now().strftime('%Y%m%d')}.csv",
```

```
        mime="text/csv"
```

```
    )
```

```
if __name__ == "__main__":
```

```
    analyser_utilisation()
```

```
EOF
```

📚 RESSOURCES COMPLÉMENTAIRES

Documentation technique

```
# Créer documentation complète
```

```
cat > README_technique.md << 'EOF'
```

```
# IA Déconnectée - Documentation Technique
```

🎯 Vue d'ensemble

Cette solution implémente des chatbots intelligents avec bases de connaissances françaises préprogrammées, optimisés pour fonctionner sans connexion Internet.

Architecture

projet_ia_deconnectee/

```
├── bases_connaissances/      # Données structurées par domaine
|   ├── base_education.py    # Connaissances pédagogiques
|   ├── base_agricole.py     # Conseils agricoles locaux
|   ├── base_sante.py        # Information santé préventive
|   └── base_culture.py      # Patrimoine culturel local
├── interfaces/              # Applications utilisateur finales
|   ├── chatbot_education.py # Assistant pédagogique
|   ├── chatbot_agricole.py  # Conseiller agricole virtuel
|   ├── chatbot_sante.py     # Assistant santé communautaire
|   └── chatbot_culture.py   # Gardien patrimoine culturel
├── templates/               # Modèles réutilisables
|   └── template_personnalisable.py # Base pour nouveaux cas d'usage
├── tests/                   # Validation et contrôle qualité
|   └── test_complet.py      # Tests automatisés système
└── utils/                   # Outils de maintenance
    ├── sauvegarde.sh        # Script de backup automatique
    └── monitoring.py        # Surveillance système
```

Technologies utilisées

- Python 3.10+ : Langage principal
- Streamlit : Interface web
- Reconnaissance mots-clés : Logique de réponse
- JSON : Stockage configurations

Avantages techniques

- Ultra-léger : Quelques KB vs 800MB pour TinyLLaMA
- Instantané : Pas de temps de chargement
- Fiable : Réponses cohérentes garanties
- Personnalisable : Bases de connaissances modifiables
- Multiplateforme : Linux, Windows, macOS

Cas d'usage implémentés

1. Éducation : Assistant pédagogique multilingue
2. Agriculture : Conseiller technique Madagascar
3. Santé : Information préventive sécurisée
4. Culture : Gardien patrimoine malgache

Déploiement

```
```bash
```

```
Installation
```

```
pip3 install streamlit
```

```
Lancement
```

```
streamlit run chatbot_education.py
```

## Performance

- **Temps réponse** : < 100ms
- **Mémoire** : < 50MB RAM
- **Stockage** : < 1MB par chatbot
- **Compatibilité** : Raspberry Pi 4+

## Évolutivité

- Ajout facile nouveaux domaines
- Enrichissement communautaire
- Intégration APIs externes (optionnel)
- Multi-langue simple

EOF

Guide de déploiement en production

```
```bash
```

```
cat > deployment_production.md << 'EOF'
```

```
# Déploiement Production - IA Déconnectée
```

Environnement de production

Prérequis serveur

- OS : Ubuntu Server 22.04 LTS
- RAM : 2GB minimum (4GB recommandé)
- CPU : 2 vCPU minimum
- Stockage : 10GB SSD
- Réseau : Accès local uniquement

Installation automatisée

```
```bash
```

```
#!/bin/bash
```

```
Script d'installation production
```

```
Mise à jour système
```

```
sudo apt update && sudo apt upgrade -y
```

```
Installation Python et dépendances
```

```
sudo apt install python3 python3-pip nginx -y
```

```
Installation Streamlit
```

```
pip3 install streamlit
```

```
Configuration nginx (reverse proxy)
```

```
sudo tee /etc/nginx/sites-available/ia-locale << 'NGINX_CONF'
```

```
server {
```

```
 listen 80;
```

```
 server_name ia-locale.local;
```

```
 location / {
```

```
 proxy_pass http://127.0.0.1:8501;
```

```
 proxy_http_version 1.1;
```

```
 proxy_set_header Upgrade $http_upgrade;
```

```

 proxy_set_header Connection 'upgrade';
 proxy_set_header Host $host;
 proxy_cache_bypass $http_upgrade;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;
}
}

```

NGINX\_CONF

# Activer site

```
sudo ln -s /etc/nginx/sites-available/ia-locale /etc/nginx/sites-enabled/
```

```
sudo systemctl restart nginx
```

# Service systemd pour auto-démarrage

```
sudo tee /etc/systemd/system/ia-locale.service << 'SERVICE_CONF'
```

[Unit]

Description=IA Locale Streamlit App

After=network.target

[Service]

Type=simple

User=ubuntu

WorkingDirectory=/home/ubuntu/ia-locale

Environment=PATH=/home/ubuntu/.local/bin

ExecStart=/home/ubuntu/.local/bin/streamlit run chatbot\_education.py --server.port 8501 --server.address 127.0.0.1

Restart=always

[Install]

WantedBy=multi-user.target

SERVICE\_CONF



```
Activer service
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable ia-locale
```

```
sudo systemctl start ia-locale
```

```
echo "✅ Installation production terminée"
```

```
echo "🌐 Accès: http://ia-locale.local"
```

### **Monitoring production**

```
Script surveillance
```

```
cat > monitor_prod.sh << 'EOF'
```

```
#!/bin/bash
```

```
Vérification service
```

```
if systemctl is-active --quiet ia-locale; then
```

```
 echo "✅ Service IA-locale actif"
```

```
else
```

```
 echo "❌ Service IA-locale arrêté"
```

```
 sudo systemctl restart ia-locale
```

```
fi
```

```
Vérification nginx
```

```
if systemctl is-active --quiet nginx; then
```

```
 echo "✅ Nginx actif"
```

```
else
```

```
 echo "❌ Nginx arrêté"
```

```
 sudo systemctl restart nginx
```

```
fi
```

```
Vérification espace disque
```

```
DISK_USAGE=$(df -h | awk 'NR==2 {print $5}' | sed 's/%//')
```

```

if [$DISK_USAGE -gt 80]; then

 echo " ⚠ Espace disque faible: ${DISK_USAGE}%"

fi

Vérification mémoire
MEMORY_USAGE=$(free | grep Mem | awk '{printf("%.0f", $3/$2 * 100)}')
if [$MEMORY_USAGE -gt 80]; then

 echo " ⚠ Mémoire élevée: ${MEMORY_USAGE}%"

fi

echo " 🇫🇷 Monitoring terminé - $(date)"
EOF

Cron job surveillance (toutes les 5 minutes)
echo "*/5 * * * * /path/to/monitor_prod.sh >> /var/log/ia-locale-monitor.log 2>&1" | crontab -
EOF

Liste de vérification finale
```bash
cat > checklist_finale.md << 'EOF'

```

✅ Checklist Finale - IA Déconnectée

Avant mise en production

Technique

- [] Python 3.10+ installé et fonctionnel
- [] Streamlit installé et testé
- [] Toutes les bases de connaissances créées
- [] Interfaces testées individuellement
- [] Script de test automatisé passé
- [] Sauvegarde configurée

Contenu

- [] Bases de connaissances validées par experts locaux
- [] Réponses adaptées au contexte culturel
- [] Langues locales intégrées si nécessaire
- [] Avertissements de sécurité ajoutés (santé)
- [] Contact centres ressources mentionnés

Utilisateurs

- [] Formation animateurs/formateurs réalisée
- [] Documentation utilisateur disponible
- [] Procédure feedback mise en place
- [] Support technique identifié
- [] Plan de maintenance défini

Production

- [] Environnement production configuré
- [] Monitoring mis en place
- [] Sauvegardes automatisées
- [] Accès réseau local sécurisé
- [] Plan de récupération défini

Après déploiement

Suivi régulier (hebdomadaire)

- [] Vérifier logs d'utilisation
- [] Collecter feedback utilisateurs
- [] Identifier questions non couvertes
- [] Mettre à jour bases si nécessaire

Évolution (mensuelle)

- [] Analyser statistiques d'usage
- [] Enrichir bases les plus utilisées
- [] Former nouveaux utilisateurs
- [] Planifier nouvelles fonctionnalités

Maintenance (trimestrielle)

- [] Audit sécurité complet
- [] Mise à jour système/dépendances
- [] Optimisation performances
- [] Révision documentation

Objectif final : Une IA locale autonome, utile et maintenue par la communauté !

EOF

SUPPORT ET CONTACT

Ressources d'aide

-  **Email support** : contact@ekm-conseils.eu
-  **Site web** : <https://ekmconseils.eu>
-  **Chaîne YouTube** : @emormin
-  **Documentation** : Blog EKM Conseils

Communauté

-  **Questions** : Commentaires YouTube vidéos IA Déconnectée
-  **Partage** : Partagez vos adaptations et améliorations
-  **Collaboration** : Contactez-nous pour projets communs

CONCLUSION

Cette procédure technique vous guide pour implémenter concrètement les 4 cas d'usage présentés dans la **Vidéo 3 : Cas d'usage inspirants**.

Ce que vous avez maintenant :

- **4 chatbots fonctionnels** prêts à l'emploi
- **Template personnalisable** pour vos propres besoins
- **Procédures de test** et validation
- **Scripts de maintenance** et évolution
- **Documentation complète** pour déploiement

Prochaines étapes :

1. **Testez** chaque chatbot dans votre environnement
2. **Adaptez** les bases de connaissances à votre contexte
3. **Déployez** en production avec monitoring
4. **Collectez** feedback et enrichissez régulièrement
5. **Partagez** votre expérience avec la communauté

 **Mission accomplie** : Votre IA déconnectée est opérationnelle et utile pour votre communauté !

 Document généré pour accompagner la Vidéo 3 - IA Déconnectée

 EKM Conseils - Innovation Souveraine & Résilience Numérique

 Version 1.0 - Juillet 2025