ÉCOLE SUPÉRIEURE FRANÇAISE
keyce®
Informatique & Intelligence Artificielle

COLLEGE DE PARIS

GROUPE SIX DEVOPS CONCEPTION D'UNE APPLICATION WEB DE CONNEXION A ODOO ET PGADMIN

LISTE DE MEMBRE G6 B3 JOUR

ATANGANA NTOUH

JOHANN FREDERIC

KONDJO WANDJI

KOUAM OWEN

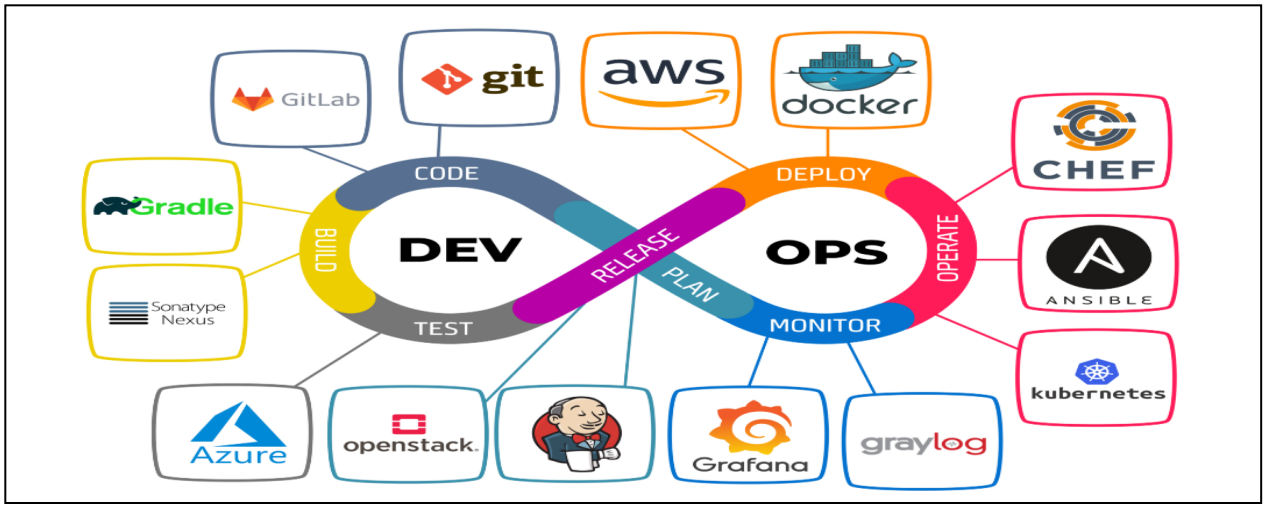
MBALLA ELI

WANDJA YOTCHOU

DIRIGE PAR MONSIEUR BOGNI (2024-2025)

Table des matières

I. INTRODUCTION



Le développement et l'exploitation (DevOps) sont des pratiques essentielles pour optimiser la collaboration entre les équipes de développement et d'exploitation. Dans le contexte des applications web, l'utilisation de frameworks comme Flask facilite la création d'applications légères et modulaires, tout en permettant une intégration rapide et efficace des processus de déploiement et de test. Flask, un micro-framework Python, est idéal pour construire des applications web en raison de sa simplicité et de sa flexibilité. En intégrant des outils de CI/CD (Intégration Continue / Déploiement Continu), les équipes peuvent automatiser les tests et les déploiements, garantissant ainsi une livraison plus rapide et fiable des fonctionnalités. Dans ce contexte, nous allons examiner deux applications Flask qui illustrent comment le DevOps peut être appliqué pour améliorer le développement d'applications web. La première application est conçue pour se connecter à un serveur Odoo, un système de gestion d'entreprise open source. L'application fournit une interface simple où les utilisateurs peuvent entrer leurs informations de connexion (nom de la base de données, email et mot de passe). En utilisant la bibliothèque `odoorpc`, l'application tente de se connecter au serveur Odoo et, si la connexion est réussie, redirige l'utilisateur vers l'interface web d'Odoo. En cas d'erreur, un message d'erreur est affiché à l'utilisateur. Ce type d'application est un excellent exemple de comment interfacer des systèmes externes à l'aide de Flask tout en maintenant une expérience utilisateur fluide. La deuxième application est axée sur la connexion à une base de données PostgreSQL. Elle permet aux utilisateurs de fournir les informations nécessaires pour établir une connexion (hôte, port, nom de base de données, utilisateur et mot de passe). En utilisant la bibliothèque `psycopg2`, l'application essaie de se connecter à la base de données. Si la connexion réussit, un message de succès est affiché ; sinon, l'utilisateur reçoit un message d'erreur. Cette application démontre comment Flask peut être utilisé pour interagir avec des bases de données et gérer les connexions de manière sécurisée.

II. CONFIGURATION DE L'APPLICATION

Bibliothèques utilisées dans chaque code

Application 1 : Connexion à Odoo

- -Flask : Framework web utilisé pour créer l'application.
- odoorpc : Bibliothèque utilisée pour interagir avec le serveur Odoo.

Application 2 : Connexion à PostgreSQL

- Flask: Framework web utilisé pour créer l'application.
- psycopg2: Bibliothèque utilisée pour se connecter à la base de données PostgreSQL.

Clés secrètes de chaque code

Application 1 :

- **Clé secrète :** `"5f2c7f3b8e2e4b9a3d6b8e1f75c12aaf098cf2f3df1c27ea"`

Application 2 :

- **Clé secrète :** `"5f2c7f3b8e2e4b9a3d6b8e1f75c12aaf098cf2f3df1c27eb"

Ces clés secrètes sont utilisées pour sécuriser les sessions dans les applications Flask.

III. ROUTE DE CONNEXION A LA BASE DE DONNEES

Dans la première application, la route définie est `"/"`, qui gère à la fois les méthodes GET et POST. Lorsque l'utilisateur accède à cette route via une requête GET, l'application affiche un formulaire de connexion. Si l'utilisateur soumet le formulaire (méthode POST), les informations de connexion telles que le nom de la base de données, l'email et le mot de passe sont récupérées. L'application tente ensuite d'établir une connexion avec le serveur Odoo à l'aide des informations fournies. En cas de succès, l'utilisateur est redirigé vers l'interface web d'Odoo ; sinon, un message d'erreur est affiché.

```python
# Configuration Odoo
ODOO_HOST = "localhost"  # Adresse de votre serveur Odoo
ODOO_PORT = 8069         # Port Odoo par défaut


@app.route("/", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        # Récupérer les données du formulaire
        db_name = request.form.get("db_name")  # Nom de la base de données
        email = request.form.get("email")      # Email utilisateur
        password = request.form.get("password")  # Mot de passe utilisateur
```

Figure 1 route connexion au serveur Odoo

Dans la deuxième application, la route est également `"/"`, servant un objectif similaire. Lorsqu'une requête GET est effectuée, le formulaire de connexion à la base de données PostgreSQL est affiché. Si le formulaire est soumis via une requête POST, les détails de connexion (hôte, port, nom de la base de données, utilisateur et mot de passe) sont extraits. L'application utilise ces informations pour tenter de se connecter à la base de données. Si la connexion est réussie, un message de succès est retourné ; sinon, un message d'erreur est affiché, informant l'utilisateur du problème rencontré.

```python
@app.route("/", methods=["GET", "POST"])
def connect_db():
    if request.method == "POST":
        # Récupérer les informations du formulaire
        host = request.form.get("host")
        port = request.form.get("port")
        dbname = request.form.get("dbname")
        user = request.form.get("user")
        password = request.form.get("password")
```

Figure 2 route de connexion PostgreSQL

Dans les deux cas, la route `"/"` joue un rôle central dans la gestion des connexions aux systèmes respectifs, offrant une interface utilisateur simple et intuitive.

IV. CONNEXION A LA BASE DE DONNEES

Dans la première application, qui se connecte à un **serveur Odoo**, les informations nécessaires pour établir la connexion sont récupérées à partir d'un formulaire soumis par l'utilisateur. Les données requises incluent le **nom de la base de données** (db_name), **l'email de l'utilisateur**, et **le mot de passe**. Lorsque l'utilisateur soumet le formulaire, ces informations sont envoyées à l'application, qui utilise la bibliothèque **odoorpc** pour tenter de se connecter au serveur Odoo. Si la connexion est réussie, l'utilisateur est redirigé vers l'interface web d'Odoo.

```
if request.method == "POST":
    # Récupérer les données du formulaire
    db_name = request.form.get("db_name")  # Nom de la base de données
    email = request.form.get("email")      # Email utilisateur
    password = request.form.get("password")  # Mot de passe utilisateur

    try:
        # Connexion au serveur Odoo
        odoo = odoorpc.ODOO(ODOO_HOST, port=ODOO_PORT)
        odoo.login(db_name, email, password)

        # Si connexion réussie, redirigez vers l'interface web d'Odoo
        return redirect(f"http://{ODOO_HOST}:{ODOO_PORT}/web?db={db_name}")
```

Figure 3 Connexion a Odoo

Dans la deuxième application, qui se connecte à une base de données PostgreSQL, le processus est similaire. L'utilisateur entre les informations suivantes dans un formulaire : **hôte** (host), **port**, **nom de la base de données** (dbname), **utilisateur**(user), et **mot de passe**. Ces informations sont cruciales pour établir une connexion sécurisée à la base de données. Lors de la soumission du formulaire, l'application utilise la bibliothèque **psycopg2** pour tenter de se connecter à la base de données PostgreSQL avec les détails fournis. Si la connexion est réussie, un message de succès est affiché ; en cas d'erreur, l'utilisateur reçoit un message explicatif.

```
try:
    # Tentative de connexion à la base de données PostgreSQL
    connection = psycopg2.connect(
        host=host,
        port=port,
        dbname=dbname,
        user=user,
        password=password
    )
    connection.close()
```

Figure 4 Connexion à PGAMIN

Dans les deux applications, les informations saisies par l'utilisateur sont essentielles pour établir une connexion sécurisée et fonctionnelle aux systèmes respectifs, permettant ainsi d'interagir avec les données de manière efficace.

V. GESTION DES ERREURS ET DES MESSAGES

La gestion des erreurs et des messages d'information est cruciale pour offrir une expérience utilisateur fluide et informative dans les deux applications Flask.

Dans la première application, qui se connecte à Odoo, la gestion des erreurs se fait principalement lors de la tentative de connexion au serveur. Si une erreur survient, par exemple si les informations de connexion sont incorrectes, l'application capture l'exception OperationalError et affiche un message d'erreur à l'utilisateur via la fonction flash. Ce message indique clairement la nature du problème, par exemple : "Erreur de connexion : [détails de l'erreur]". Cela permet à l'utilisateur de comprendre ce qui s'est mal passé et d'apporter les corrections nécessaires avant de réessayer.

```
        # Si connexion réussie, redirigez vers l'interface web d'Odoo
        return redirect(f"http://{ODOO_HOST}:{ODOO_PORT}/web?db={db_name}")

    except Exception as e:
        # En cas d'erreur (connexion échouée)
        flash(f"Erreur de connexion : {str(e)}")
        return redirect("/")
```

Figure 5 Gestion des message et erreurs Odoo

De même, dans la deuxième application, dédiée à la connexion à PostgreSQL, la gestion des erreurs suit un schéma similaire. Lorsqu'une tentative de connexion échoue, l'application intercepte l'exception et utilise également la fonction flash pour afficher un message d'erreur. Par exemple, si les informations fournies sont incorrectes ou si le serveur est inaccessible, un message comme "Erreur de connexion : [détails de l'erreur]" est présenté à l'utilisateur. En cas de succès, un message de confirmation est également affiché, indiquant que la connexion à la base de données a été réussie.

```
    flash("Connexion réussie à la base de données PostgreSQL.", "success")
except OperationalError as e:
    flash(f"Erreur de connexion : {e}", "danger")

return redirect("/")
```

Figure 6 Gestion des messages et erreurs PGADMIN

VI. REDIRECTION DES TEMPLATES

la première application :après une tentative de connexion au serveur Odoo, l'utilisateur est redirigé vers un template spécifique en fonction du résultat de la connexion. Si la connexion est réussie, l'application redirige l'utilisateur vers le template de l'interface utilisateur d'Odoo, généralement désigné par un chemin d'URL qui charge l'application Odoo. En revanche, si la connexion échoue, l'utilisateur est renvoyé au même template avec un message d'erreur affiché, offrant ainsi une opportunité de corriger les informations saisies.

```
# Afficher le formulaire de connexion
return render_template("login.html")
```

Figure 7 Template html pour Odoo

Dans la deuxième application, la logique de redirection est similaire. Après que l'utilisateur a soumis les informations de connexion à la base de données PostgreSQL, si la connexion est réussie, l'application redirige vers un template qui confirme le succès de la connexion, souvent avec des détails supplémentaires sur l'accès à la base de données. Si la connexion échoue, l'utilisateur reste sur le même template et un message d'erreur est affiché, lui permettant de corriger les informations avant de réessayer.

```
    return redirect("/")

return render_template("login2.html")
```

Figure 8 Template html pour PGADMIN

VII. LANCEMENT DE L'APPLICATION

Pour lancer une application Flask, il faut d'abord s'assurer que l'environnement de développement est configuré correctement. Voici les étapes à suivre :

1**. Installer Flask :** Si ce n'est pas déjà fait, installez Flask en utilisant pip. Exécutez la commande suivante dans le terminal : pip install Flask.

2. **Créer un fichier Python :** Écrivez le code de votre application Flask dans un fichier Python, par exemple app.py.

3. **Définir la variable d'environnement :** Avant de lancer l'application, définissez la variable d'environnement FLASK_APP. Dans le terminal, exécutez la commande : export FLASK_APP=app.py pour les systèmes Unix ou set FLASK_APP=app.py pour Windows.

4. **Lancer le serveur Flask :** Exécutez la commande flask run dans le terminal. Cela démarre le serveur de développement Flask.

5. **Ouvrir l'application dans le navigateur :** Une fois le serveur lancé, ouvrez un navigateur web et entrez l'URL http://127.0.0.1:5000. Cela vous permettra d'accéder à l'application Flask que vous avez créée.

Ces étapes vous permettront de lancer et d'accéder à votre application Flask facilement.

Figure 9 Test connexion ODOO

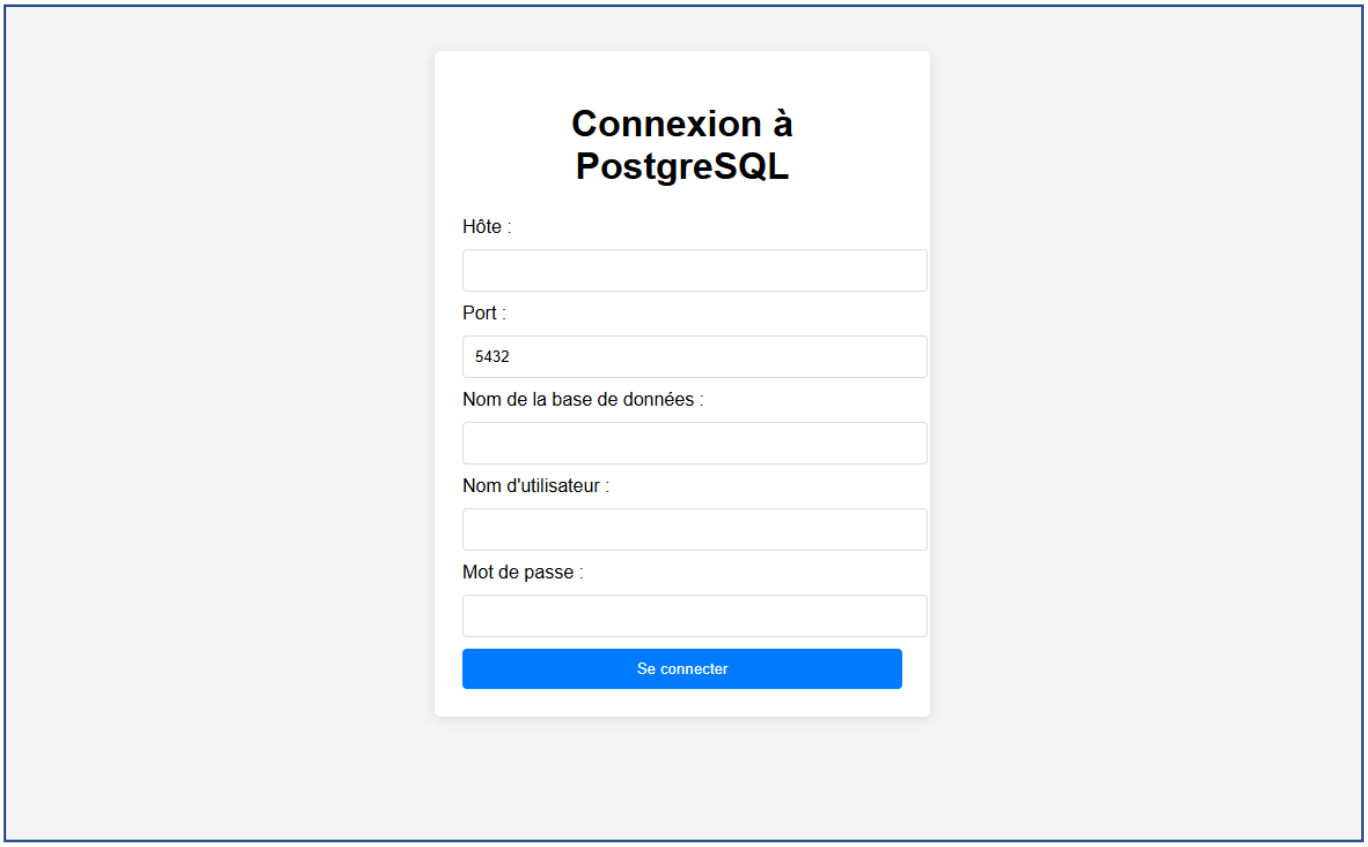


Figure 10 Test connexion PGADMIN

VIII. CONCLUSION

Pour résumer, nous avons exploré deux applications Flask distinctes, chacune dédiée à la connexion à des systèmes spécifiques : l'une à Odoo et l'autre à une base de données PostgreSQL. Nous avons examiné les routes utilisées pour gérer les connexions, les informations requises pour établir ces connexions, ainsi que la gestion des erreurs et des messages informatifs destinés aux utilisateurs. De plus, nous avons discuté des redirections vers des templates appropriés en fonction des résultats des tentatives de connexion. Enfin, les étapes pour lancer une application Flask et l'ouvrir dans un navigateur ont été détaillées, offrant ainsi une vue d'ensemble complète sur le développement et l'interaction avec des applications web basées sur Flask.