MCP : Entrainement CH - RH

Créé par	rim belabadia
Heure de création	@9 avril 2025 15:25
Étiquettes	

Architecture MCP pour un Chatbot Multi-Sources

1. Introduction au Modèle MCP (Modèle de Connexion Polyvalente)

1.1 Définition et Rôle

Le **MCP** est une architecture middleware qui agit comme un **médiateur sémantique** entre un chatbot et plusieurs sources de données hétérogènes (Oracle, SAP, SQL Server).

Fonctions principales :

✅ **Abstraction** : Cache la complexité des sources (schémas, langages de requête, protocoles).

✅ **Unification** : Fournit une **API unique** au chatbot, peu importe la source sous-jacente.

✅ **Optimisation** : Gère le routage intelligent, l'agrégation et le cache des requêtes.

1.2 Schéma Conceptuel du MCP

```
flowchart TD
    A[Chatbot] -->|Requête Naturelle| B[MCP]
    B -->|Traduction| C1[(Oracle)]
    B -->|Traduction| C2[(SAP)]
    B -->|Traduction| C3[(SQL Server)]
    C1 -->|Données Brutes| B
```

```
    C2 →|Données Brutes| B
    C3 →|Données Brutes| B
    B →|Réponse Unifiée| A
```

Remarque :

- Le chatbot **ne connaît pas** la structure des bases de données.
- Le MCP **traduit** la requête en langage natif (SQL, RFC, BAPI, etc.).
- Il **normalise** les réponses dans un format standard (JSON).

---

2. Intégration du MCP avec les 3 Sources de Données

2.1 Architecture Technique

Composant	Rôle	Technologies (Next.js)
Connecteur Oracle	Exécute des requêtes SQL/PLSQL	`oracledb` (Node.js)
Connecteur SAP	Appelle des BAPI/RFC	`node-rfc`
Connecteur SQL Server	Requête T-SQL	`tedious` (Node.js)
Couche MCP	Traduction + Agrégation	API Routes (Next.js)

2.2 Comment Construire le MCP ?

Étape 1 : Définir le Métamodèle

- Créer une **représentation unifiée** des concepts.

Étape 2 : Implémenter les Connecteurs

- Implementer des connecteurs pour chaque source de données.

Étape 3 : API MCP (Next.js)

```
// pages/api/mcp/query.ts
export default async function handler(req: NextApiRequest, res: NextApiRe
sponse) {
  const { intent } = req.body;
  const data = await MCPService.fetch(intent); // Routage automatique
```

```
  res.json(data);
}
```

---

3. Entraînement du Chatbot avec le MCP

3.1 Stratégie d'Apprentissage

Le chatbot doit être entraîné sur **deux niveaux** :

1. **Compréhension des Intentions** (NLP)

   1.1. Cas d'Usage RH Prioritaires

   1. **Gestion des congés**
      - Intentions : `checkLeaveBalance`, `submitLeaveRequest`
      - Sources : SAP HR (données) + Oracle HCM (workflow)
   2. **Gestion des compétences**
      - Intentions : `listTeamSkills`, `findInternalExperts`
      - Sources : Oracle HCM (compétences) + SQL Server (projets)
   3. **FAQ Employés**
      - Intentions : `getHRPolicy`, `requestDocument`
      - Sources : SQL Server (base documentaire)

2. **Mapping des Intentions vers le MCP**
   - Chaque intention appelle une **requête MCP prédéfinie**.
   - Exemple de dataset d'entraînement :

```
{
  "intent": "checkLeaveBalance",
  "questions": [
    "Combien me reste-t-il de jours de congé ?",
    "Quel est mon solde de RTT ?"
  ],
  "mcp_query": {
    "operation": "getLeaveBalance",
```

```
    "sources": ["SAP", "Oracle"],
    "params": {"employeeId": "$user.id"}
  }
}
```

3. **Flux d'Exécution**

```
sequenceDiagram
    participant User
    participant Chatbot
    participant MCP
    participant Databases

    User->>Chatbot: "Combien me reste-t-il de jours de congé ?"
    Chatbot->>MCP: { intent: "getLateCustomers" }
    MCP->>Databases: Requête Oracle + SAP + SQL
    Databases-->>MCP: Résultats bruts
    MCP-->>Chatbot: Données normalisées
    Chatbot-->>User: "Il vous reste 15 jours."
```

4. **Workflow Complet du Chatbot RH avec MCP**

3.2 Workflow Global du Chatbot RH

```
flowchart TD
    A[Question Employé] --> B{NLU}
    B -->|Intent| C[MCP Routing]
    C --> D[SAP HR]
    C --> E[Oracle HCM]
    C --> F[SQL Server]
    D & E & F --> G[Normalisation]
    G --> H[Génération Réponse]
    H --> I[Réponse Naturelle]
```

Étapes Clés :

1. **Capture de la Question** (Interface Next.js)
2. **Compréhension du Langage** (NLP)

3. **Routage Intelligent** (MCP)
4. **Interrogation des Sources** (Connecteurs)
5. **Agrégation/Normalisation**
6. **Génération de la Réponse** (LLM ou Templates)

3.2.1 Entraînement du Chatbot RH

3.2.1.1 Outils proposés

Composant	Technologie/Approche	Cas d'Usage RH
NLP (NLU)	Rasa / Dialogflow / LLM (GPT-3.5)	Détection d'intentions ("congé", "formation")
Métadonnées MCP	JSON Schema	Mapping SAP → Oracle → SQL Server
Entraînement	Jeu de données annoté	Ex: 500 questions RH étiquetées

3.2.1.2 Jeu de Données d'Entraînement

Exemple :

```
{
  "text": "Comment poser un RTT ?",
  "intent": "leave_request_process",
  "entities": {"leave_type": "RTT"},
  "mcp_operation": "getLeavePolicy",
  "params": {"policy_type": "RTT", "source": "SAP"}
}
```

3.2.1.3 Pipeline d'Entraînement

```
flowchart LR
    A[Raw Data RH] --> B(Annotation)
    B --> C[NLP Model Training]
    C --> D[Intent Classification]
    D --> E[MCP Mapping]
    E --> F[Validation Métier]
```

---

4. Utilisation de Next.js pour le MCP

4.1 Pourquoi Next.js ?

- **API Routes** → Expose le MCP comme une API REST.
- **Middleware** → Logging, Cache, Sécurité centralisée.
- **Vercel Edge Functions** → Réduit la latence (global).
- **ISR (Incremental Static Regeneration)** → Met en cache les requêtes fréquentes.

4.2 Architecture Frontend :

```
flowchart TB
    subgraph Next.js App
        A[Chat UI] -->|API Call| B[App Router]
        B --> C[/api/rh/leave\\]
        B --> D[/api/rh/skills\\]
        C --> E[MCP Service]
        D --> E
        E --> F[SAP HR]
        E --> G[Oracle HCM]
    end
```

5. Conclusion

5.1 Bénéfices Clés

🚀 **Chatbot plus intelligent** (accès à toutes les données sans complexité).

⚡ **Performance optimisée** (cache, requêtes parallèles).

🔌 **Évolutivité** (ajout facile de nouvelles sources).

5.2 Prochaines Étapes

- **Améliorer le MCP** avec de l’apprentissage automatique (meilleur routage).
- **Dashboard d’analytique** (Next.js + Vercel Analytics).
- **Extension à d’autres sources** (MongoDB, APIs externes).

Annexe : Exemple de Flux Complet

```
sequenceDiagram
    participant Employé
    participant Chatbot
    participant MCP
    participant Systèmes

    Employé->>Chatbot: "Puis-je poser 5 jours en août ?"
    Chatbot->>MCP: {intent: "checkLeaveBalance", employeeId: "123"}
    MCP->>SAP HR: BAPI_EMPLOYEE_GETDATA(123)
    MCP->>Oracle HCM: GET /leave-requests?employee=123
    SAP HR-->>MCP: Solde: 12 jours
    Oracle HCM-->>MCP: Congés approuvés: 3 jours en août
    MCP-->>Chatbot: {balance: 12, approvedAugust: 3}
    Chatbot-->>Employé: "Oui ! Il vous reste 12 jours (dont 3 déjà posés en août)."
```