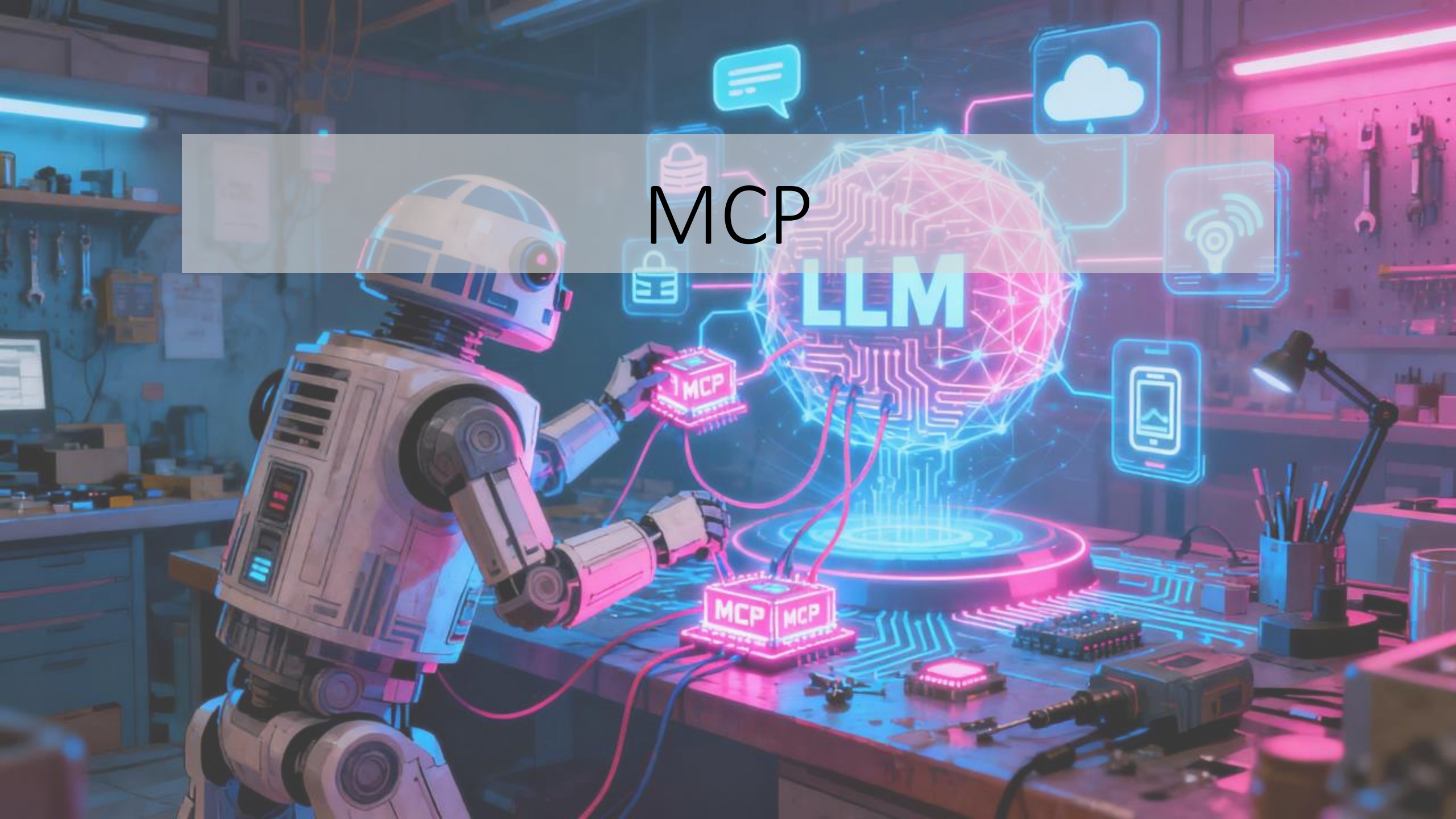
MCP



Objectifs

- IAG
- Théorie MCPs
  - Introduction
  - Architecture
  - Resources
  - Tools
  - Prompts
- Technologie autour des MCPs
  - JSONRPC
  - JSONSchema
- Implantation en C# « natif » (i.e. sans Framework)

IAG

- Intelligence Artificielle Générative
- Principales classes populaires :
  - **Modèles de langage (LLM / SLM)** – génèrent du texte : GPT (OpenAI), Claude (Anthropic), LLaMa (Meta), Mistral, Gemini (Google), etc.
  - **Modèles de diffusion** – génèrent des images/vidéos : Stable Diffusion, DALL-E (OpenAI), MidJourney, Flux.1, etc.
  - **Modèles audios / musique** : MusicLM (Google), Suno, AI Music (Eleven Labs), etc.
  - Etc.

Utilisations – LLM/SLM

- Robot conversationnel : service à la clientèle
- Analyse de courriels et routage vers le bon destinataire (Ex. trouver le bon technicien pour répondre à une question)
- Extraction de données à partir d'un courriel (Ex. Modifier quantité d'une commande, etc.)
- Surveiller les réseaux sociaux avec analyse de sentiment et tendances
- Personnalisation de logiciel : permettre de créer des flux de travail à partir d'une description (Ex. à la réception d'une inscription, envoie un courriel à l'utilisateur et envoie une notification aux vendeurs d'offres promotionnelles)

Utilisations – Agent

- Bâtis sur les LLM
- Travaillent en asynchrone sur des tâches de longue durée
- Utilisent des sources de données externes
- Agissent sur des systèmes à la place d'un utilisateur
- Exemples :
  - Analyse les récits utilisateurs de l'itération courante du projet ABC et donne-les-moi les récits à améliorer et extrait ceux qui sont à risque
  - Analyse le récit utilisateur 123 et crée les tâches associées sur Azure DevOps
  - Donne-moi les avertissements pour l'application ABC d'aujourd'hui. Classe-les par niveau de risque

Utilisations – Agent

Analyse les récits utilisateurs de l'itération courante du projet et donne les moi les récits à améliorer et extrait ceux qui sont à risque.

ado

Pour cibler l'itération courante et en extraire les US, je récupère l'équipe du projet puis l'itération "current", et je listerai ses work items.

> `core_list_project_teams` a été exécuté – ado (serveur MCP) ✓

> `work_list_team_iterations` a été exécuté – ado (serveur MCP) ⊗

✓ Updated todo list

Reçu. Je cible l'itération courante (API ne renvoie pas de "current"; j'ai utilisé IterationPath=Iteration 1) et j'analyse les US pour proposer des améliorations et isoler les risques.

Récits de l'itération courante

US incluses dans Iteration 1:

- 7912 — Affichage des événements (3 pts, P1)
- 7929 — Création et connexion à un compte (2 pts)
- 7939 — Publication des événements (8 pts)
- 7940 — Filtration des événements par date (5 pts)

Récits à améliorer (concis et actionnables)

- 7912 Affichage des événements
  - Ajouter: cas vide/erreurs réseau, lien explicite vers la page détail, tri/pagination, contrat API (formats des champs, date/heure), règles d'exclusion (bannis/passés), accessibilité (alt images).
  - Nettoyer: déplacer "Header/Footer" en tickets techniques transverses.
- 7929 Création/connexion compte
  - Ajouter: cas d'échec/annulation Auth0, déconnexion, persistance profil (ID externe, email, rôle), guards de routes, configuration callbacks/env (dev/prod), récupération de mot de passe.
  - Revoir estimation (2 pts probablement faible) ou scinder (Auth0 vs persistance/guards).
- 7939 Publication des événements
  - Clarifier: liste des champs obligatoires et contraintes (longueurs, formats), upload image (taille/format/storage), fuseau horaire, validation serveur, gestion doublons, statut initial (Publié vs En attente), contrôle de rôle "organisateur".
  - Recommander split: A) Modèle/BD + API + validations, B) Formulaire/UX + validations client, C) Upload média.
- 7940 Filtration par date
  - Préciser: timezone/local vs UTC, journée locale vs plage de dates, reset du filtre, paramètre d'URL, cas bord (minuit/DST), contrat API cohérent avec 7912.

Récits à risque (avec raisons)

- Élevé

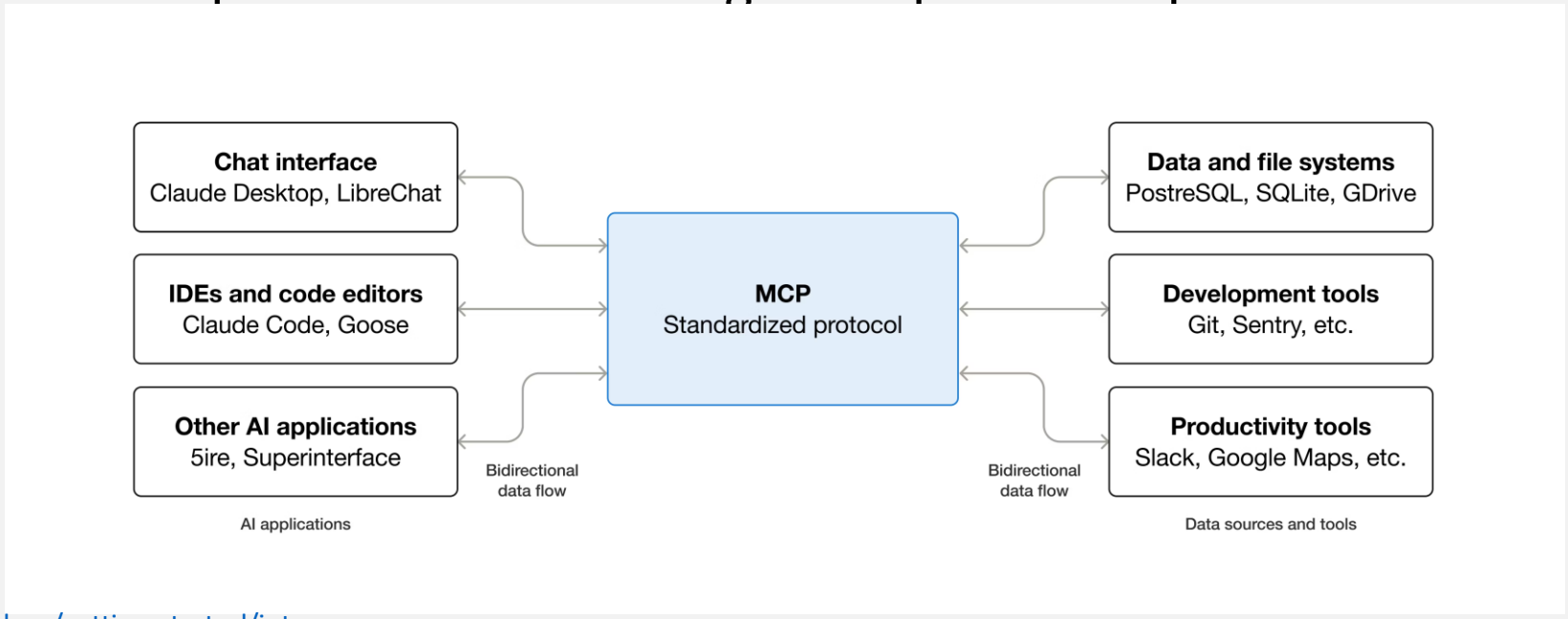
ADO : https://github.com/microsoft/azure-devops-mcp?tab=readme-ov-file

MCP - Introduction

- MCP : Model Context Protocol
- Utilise JSON-RPC et JSON-Schema
- But : permettre à des modèles d'intelligence artificielle générative (IAG) d'accéder à des services externes
- Exemple :
  - Système de fichiers / système d'exploitation
  - Calendrier Google
  - Courriels
  - Logiciels de modélisation comme blender, figma
  - Logiciels de gestion de projets comme Azure DevOps
  - Domotique
  - Données
  - Etc.

MCP - Introduction

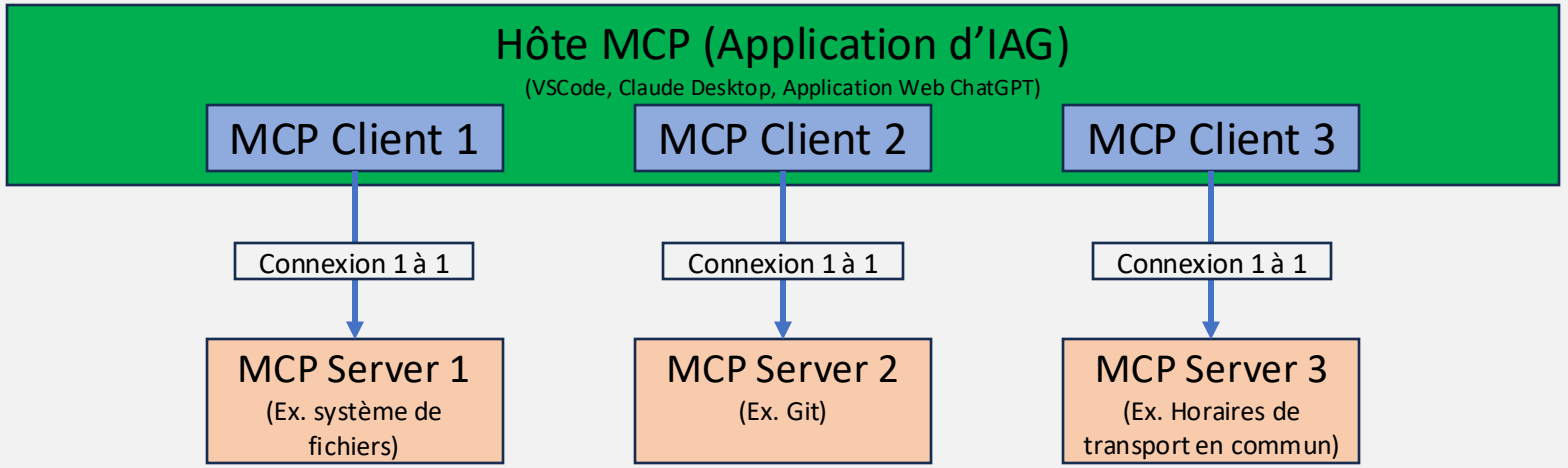
- Avantages :
  - Développeurs : réduire le temps d'intégration entre les IAG et les programmes et les services de tiers ou d'entreprise
  - Application et Agents : permettre d'accéder à des données et leur permet d'exécuter des actions sur « le monde »
  - Utilisateurs : permettre à l'IAG d'agir à sa place lorsque nécessaire



https://modelcontextprotocol.io/docs/getting-started/intro

MCP – Architecture

- Hôte MCP (MCP Host) : application qui coordonne et gère un ou plusieurs clients MCP
- Client MCP (MCP Client) : composant qui maintient une connexion avec un serveur MCP et qui permet à l'hôte de l'utiliser
- Serveur MCP (MCP Serveur) : un programme qui expose un contexte à des clients MCP ← **Notre but**



https://modelcontextprotocol.io/docs/learn/architecture

MCP – Blocs de base

- Les MCPs s'appuient sur trois blocs :
  - **Ressources (Resources)** : données statiques non modifiables utilisables dans le contexte. Ex. : documentation, dossier client, contenu de fichiers, etc.
  - **Outils (Tools)** : fonctionnalités que l'IAG peut appeler en fonction des demandes de l'utilisateur. Ex. : écriture de données, appel d'API, déclenchements de flux de travail, écriture de fichiers, etc.
  - **Instructions (Prompts) :** gabarits d'instructions qui indiquent au modèle d'utiliser des outils et ressources spécifiques

https://modelcontextprotocol.io/docs/learn/server-concepts

MCP – Ressources

- Données non modifiables par l'IAG
- Comme pour REST, les ressources sont accessibles à partir d'une URI
- Deux types de ressources :
  - Ressources : URIs fixes qui désignent des données spécifiques. Ex. « file:///monsuperfichier.md »
  - Gabarit de ressources (Resource Templates) : URIs dynamiques avec des paramètres. Ex. « file:///evenements/{pays}/{ville} »
- Opérations :
  - « resources/list » : renvoie une collection de descriptions de ressources
  - « resources/templates/list » : renvoie une collection de descriptions de gabarits de ressources
  - « resources/read » : renvoie le contenu d'une ressource à partir de son URI
  - « resources/subscribe » : s'abonne aux modifications d'une ressource (si la ressource peut changer : exemple les scores de matchs de ccer)

MCP – Ressources – Lister

```
+---------------------- MCP Request ----------------------+
{
  "jsonrpc": "2.0",
  "id": 4,
  "method": "resources/list",
  "params": {
    "_meta": {
      "progressToken": 4
    }
  }
}+-------------------------------------------------------+
```

```
+---------------------- MCP Response ---------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "resources": [
      {
        "uri": "file:///couleurs.md",
        "name": "couleurs.md",
        "title": "Couleurs",
        "description": "Un fichier de documentation sur les couleurs.",
        "mimeType": "text/markdown"
      }
    ]
  },
  "id": 4
}+-------------------------------------------------------+
```

MCP – Ressources – Lire

```
+--------------------- MCP Request ---------------------+
{
  "jsonrpc": "2.0",
  "id": 5,
  "method": "resources/read",
  "params": {
    "_meta": {
      "progressToken": 5
    },
    "uri": "file:///couleurs.md"
  }
}+-------------------------------------------------------+
```

```
+--------------------- MCP Response --------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "contents": [
      {
        "uri": "file:///couleurs.md",
        "name": "couleurs.md",
        "title": "Couleurs",
        "mimeType": "text/markdown",
        "text": "# Couleurs\n\n- Rouge\n- Vert\n- Bleu\n"
      }
    ]
  },
  "id": 5
}+-------------------------------------------------------+
```

MCP – Outils

- Les outils peuvent être invoqués par les IAG
- Les outils sont semblables à des appels de procédures distantes (RPC). Contrairement aux ressources qui ne font que lire, les outils permettent d'agir sur les systèmes externes (Modifier un fichier, ajouter un évènement dans un calendrier, envoyer un courriel, etc.)
- Opérations :
  - « tools/list » : renvoie la description d'une collection d'outils (Entrées / sorties)
  - « tools/call » : invoque un outil avec les paramètres d'entrée

MCP – Outils – Lister

```
+---------------------- MCP Request ----------------------+
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/list",
  "params": {
    "_meta": {
      "progressToken": 1
    }
  }
}+-------------------------------------------------------+
```

```
+---------------------- MCP Response ---------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "tools": [
      {
        "name": "use_my_super_operation",
        "title": "Calcul entre deux nombres",
        "description": "Calcul super complexe entre deux nombres",
        "inputSchema": {
          "type": "object",
          "properties": {
            "valeur1": {
              "type": "integer",
              "description": "Premier nombre"
            },
            "valeur2": {
              "type": "integer",
              "description": "Deuxi\u00E8me nombre"
```

```
              "type": "integer",
              "description": "Deuxi\u00E8me nombre"
            }
          }
        },
        "outputSchema": {
          "type": "object",
          "properties": {
            "result": {
              "type": "integer",
              "description": "Le r\u00E9sultat du calcul"
            }
          }
        }
      }
    ]
  },
  "id": 2
}+-------------------------------------------------------+
```

MCP – Outils – Invoquer

```
+--------------------- MCP Request ---------------------+
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "tools/call",
  "params": {
    "_meta": {
      "progressToken": 3
    },
    "name": "use_my_super_operation",
    "arguments": {
      "valeur1": 19,
      "valeur2": 23
    }
  }
}+-------------------------------------------------------+
```

```
+--------------------- MCP Response --------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "content": [
      {
        "type": "text",
        "text": "{\u0022result\u0022:42}"
      }
    ],
    "structuredContent": {
      "result": 42
    }
  },
  "id": 3
}+-------------------------------------------------------+
```

MCP – Instructions

- Les instructions (Prompt) sont des gabarits réutilisables pour créer des instructions paramétrables :
  - Prennent des paramètres dynamiques directement de l'utilisateur (ie non appelé par le LLM)
  - Peuvent utiliser des données de la conversation
  - Permettent d'enchainer plusieurs interactions et guider l'IAG dans un flux de travail complexe (ie. Plusieurs étapes)
- Ex. :
  - « Crée le fichier nommé {{filename}} dans le dossier {{path}} avec le contenu suivant :\n\n{{content}} »
  - « README.md », « /tmp », « Bonjour tout le monde ! » => « Crée le fichier README.md dans le dossier /tmp avec le contenu suivant :\n\nBonjour tout le monde ! »
- Opérations :
  - « prompts/list » : renvoie une collection de description d'instructions
  - « prompts/get » : renvoie le détail d'une instruction où les paramètres sont appliqués

MCP – Instructions – Lister

```
+---------------------- MCP Request ----------------------+
{
  "jsonrpc": "2.0",
  "id": 9,
  "method": "prompts/list",
  "params": {
    "_meta": {
      "progressToken": 9
    }
  }
}+-------------------------------------------------------+
```


```
+--------------------- MCP Response --------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "prompts": [
      {
        "name": "creer_fichier_avec_contenu",
        "title": "Cr\u00E9er un fichier avec du contenu",
        "description": "Demande au LLM de cr\u00E9er un fichier avec un contenu sp\u00E
9cifique.",
        "arguments": [
          {
            "name": "fileName",
            "required": true
          },
          {
            "name": "path",
            "required": true
          },
          {
            "name": "content",
            "required": true
          }
        ]
      }
    ]
  },
  "id": 9
}+-------------------------------------------------------+
```

MCP – Instructions – Obtenir

```
+---------------------- MCP Request ----------------------+
{
  "jsonrpc": "2.0",
  "id": 11,
  "method": "prompts/get",
  "params": {
    "_meta": {
      "progressToken": 11
    },
    "name": "creer_fichier_avec_contenu",
    "arguments": {
      "fileName": "README.md",
      "path": "/tmp",
      "content": "Bonjour tout le monde !"
    }
  }
}+-------------------------------------------------------+
```

```
+---------------------- MCP Response ---------------------+
{
  "jsonrpc": "2.0",
  "result": {
    "description": "Demande au LLM de cr\u00E9er un fichier avec un contenu sp\u00E9cif
ique.",
    "messages": [
      {
        "role": "user",
        "content": {
          "type": "text",
          "text": "Cr\u00E9e un fichier nomm\u00E9 \u0027README.md\u0027 dans le r\u00E
9pertoire \u0027/tmp\u0027 avec le contenu suivant :\n\nBonjour tout le monde !"
        }
      }
    ]
  },
  "id": 11
}+-------------------------------------------------------+
```

JSON-RPC

- Normalisation d’appels de procédures distantes avec la syntaxe JSON

Syntax:

```
--> data sent to Server
<-- data sent to Client
```

rpc call with positional parameters:

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
<-- {"jsonrpc": "2.0", "result": 19, "id": 1}

--> {"jsonrpc": "2.0", "method": "subtract", "params": [23, 42], "id": 2}
<-- {"jsonrpc": "2.0", "result": -19, "id": 2}
```

rpc call with named parameters:

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23, "minuend": 42}, "id": 3}
<-- {"jsonrpc": "2.0", "result": 19, "id": 3}

--> {"jsonrpc": "2.0", "method": "subtract", "params": {"minuend": 42, "subtrahend": 23}, "id": 4}
<-- {"jsonrpc": "2.0", "result": 19, "id": 4}
```

https://www.jsonrpc.org/specification#request_object

JSON-Schema

- Notation définition de schéma de données JSON
- Permet la documentation et la validation de documents JSON

```json
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    }
  },
  "patternProperties": {},
  "additionalProperties": false
}
```

```json
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    },
    "phones": {
      "type": "array"
    }
  }
}
```

https://tour.json-schema.org/content/03-Objects/01-Pattern-Properties
https://tour.json-schema.org/content/04-Arrays/01-Specifying-Length-of-an-Array

Références

- Concept : https://modelcontextprotocol.io/docs/getting-started/intro
- Architecture : https://modelcontextprotocol.io/docs/learn/architecture
- Schémas : https://modelcontextprotocol.io/specification/2025-06-18/schema
- JSONRPC : https://www.jsonrpc.org
- JSONSchema : https://json-schema.org/learn