|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Informaticien/-ne CFC**  Travail pratique individuel 2025 (TPI) |  | |
|  | |

Modèle de rapport v1.4

Candidat No

**138922**

Nom du candidat : Clara Brodard

**« OutSidersGPT »**

### Sommaire

Résumé du rapport du TPI 2

1 Les grandes lignes du projet 3

1.1 Analyse de la situation initiale 3

1.2 Analyse de l’état désiré 3

1.3 Cahier des charges 4

1.4 Organisation du projet 4

2 Analyse 7

2.1 Objectifs du système 7

2.2 Variantes – Choix du modèle IA 10

2.3 Analyse de risque 11

2.4 Sécurité de l’information et protection des données 11

3 Conception 12

3.1 Exigences du système 12

3.2 Architecture du système 12

3.3 Plan d’intégration des systèmes 13

3.4 Concept d‘implémentation 13

3.5 Concept de tests 16

4 Réalisation 17

4.1 Spécifications détaillées 17

4.2 Design du système 28

5 Test 30

5.1 Procédure de test 30

5.2 Protocole de test 31

5.3 Signature du protocole de test 34

6 Conclusion 35

6.1 Améliorations possibles 35

6.2 Auto-évaluation 36

7 Bibliographie : liste des sources et références 37

8 Glossaire 38

9 Signatures 39

10 Annexes 40

# Résumé du rapport du TPI

**Situation de départ**

OutSiders Sàrl est une entreprise spécialisée en criminologie. Les ministères publics font appel à eux pour la rédaction de rapports sur l’« évaluation préliminaire des risques ». Ce sont des documents qui fournissent une analyse scientifique des risques qu’une personne a de récidivé un acte de violence, se reposant sur des documents comme le dossier pénal, la littérature scientifique ou les outils d’évaluation criminologiques.

Les ressources humaines étant limitées, ils souhaiteraient une solution afin de maintenir la qualité tout en optimisant les étapes de rédaction. Ils voudraient donc voire si une application utilisant l’intelligence artificielle pouvait être bénéfique dans leur travail. Il faudra cependant prendre en compte que le côté humain doit prédominer dans les analyses et que la sécurité des données est très importante.

Ce TPI consiste à réaliser une application web test pour cette entreprise qui répondrait aux exigences citées précédemment.

**Mise en œuvre**

La réalisation du projet est composée de 4 phases principales : l’analyse, la conception, la réalisation et les tests. La documentation est également une grande partie du projet.

L’analyse et la conception se sont faites durant les premiers jours du projet, avec la réalisation de plusieurs schémas, afin de me faire une idée du travail à fournir et comment structurer le projet. Je me suis ensuite lancé dans la réalisation, en ajoutant les unes après les autres les fonctionnalités désirées. Des tests ont été effectué tout au long de la réalisation, afin d’être sûr qu’il n’y avait pas de bug. L’application a donc une partie cliente en web et une partie serveur qui est en PHP. Une base de données a également été créée en SQL.

Enfin la documentation a été fait durant toute la durée du projet, et la réalisation d’un manuel d’installation en Markdown a été réaliser vers la fin du temps imparti.

**Résultats**

L’application remplit toutes les exigences du cahier des charges, mais pourrait être amélioré. Le manuel d’installation permet à n’importe qui d’installer l’infrastructure essentielle pour faire fonctionner l’application.

Un seul test ne s’est pas avéré positif, mais n’est pas gênant pour la fonctionnalité globale de l’application.

# Les grandes lignes du projet

Ce projet TPI a été proposé par M. Mario Ramalho aux apprentis informaticien de 4ème année à l’Ecole des Métiers de Fribourg. Le projet OutSidersGPT doit être réalisé en 80 heures réparties sur une période de 10 jours.

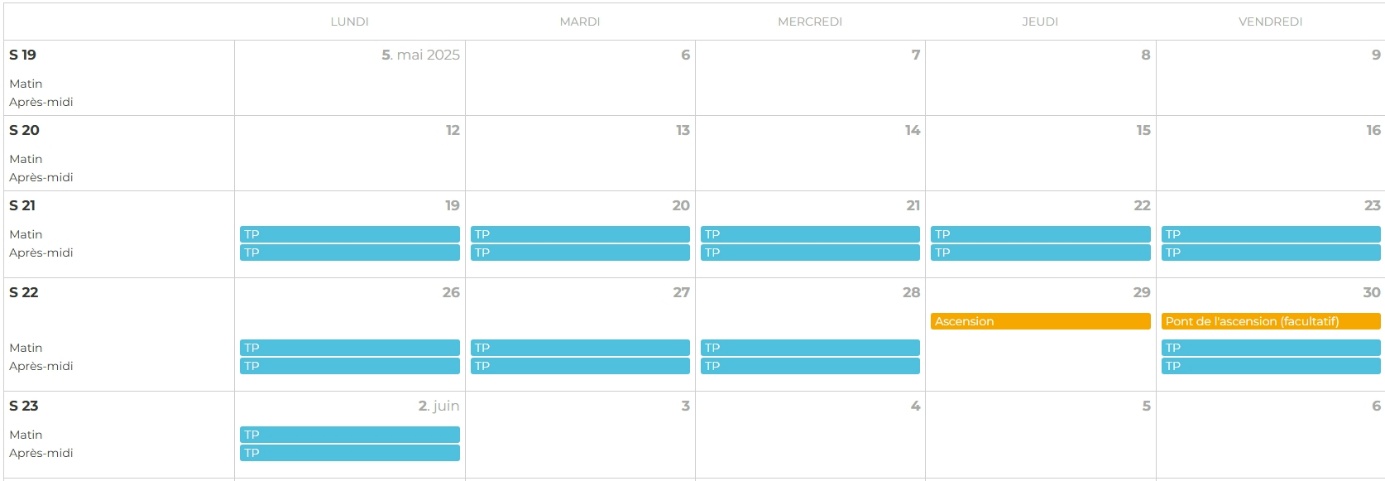


Figure 1 : Planning général du TPI sur PKORG

## Analyse de la situation initiale

L’entreprise OutSiders réalise des rapports pour les ministères publics. Ses rapports déterminent les risques qu’une personne ayant commis un délit ou un crime de récidiver. Ses documents font entre 5 et 7 pages et reposent sur différents éléments, comme le dossier pénal, les outils d’évaluation criminologiques et la littérature scientifique.

Les dossiers sont réalisés par les employés. Bien que standardisés, la rédaction de ceux-ci peut être longue. Ils cherchent donc une manière simple et sécurisée de rédiger ces rapports plus rapidement.

Ils souhaiteraient donc avoir une application utilisant l’intelligence artificielle.

## Analyse de l’état désiré

OutSiders voudrait donc une application, un peu comme ChatGPT, qui utiliserait l’IA comme aide pour réaliser ses rapports. On aimerait également que les données soient protégées et qu’aucune entreprise tierce puisse avoir accès aux données de ses rapports.

Ils voudraient l’application répartie en deux, une première partie une simple IA d’aide, qui pourrait répondre à des questions générales, et une seconde partie spécialisée dans la rédaction de certaines parties des rapports.

## Cahier des charges

Les exigences fonctionnelles sont les suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Exigence  fonctionnelle | Description | Priorité | Temps estimé |
| Analyse  Conception | Installation de l’application GPT4All Elaboration des diagrammes et schéma de conception | Haute | 1 jour |
| Implémentation Dialogue utilisateur | Configuration de GPT4All Élaboration du dialogue standard | Haute | 3 jours |
| Sauvegarde des dialogues en BD | Enregistrement des dialogues effectués par l’utilisateur dans la base de données MySQL | Moyenne | 1.5 jour |
| Implémentation  Enregistrement BD | Enregistrement des prompts dans la base de données MySQL | Moyenne | 1 jour |
| Implémentation GPT Personnalisé | Exécution des GPT personnalisé | Basse | 1 jour |
| Tests | Tests fonctionnels de l’application | Haute | 1 jour |
| Manuel  d’installation | Élaboration du manuel d’installation en markdown sur GitHub | Moyenne | 0.5 jour |
| Documentation | Rédaction de la documentation :  - Conception du diagramme d’infrastructure mis en place  - Réalisation des diagrammes de conception (activité, classe)  - Réalisation du diagramme de base de données  - Définition des maquettes de l’application MVP - Réalisation des diagrammes de cas d’utilisation | Haute | 1 jour |

## Organisation du projet

### Planification

La méthode choisie pour l’organisation et la planification en cascade/gantt. Cela signifie que chaque activité du projet est sous forme de phases séquentielles. Les phases principales sont :

* Analyse/Conception
* Implémentation
* Tests

Cela permet une vision claire des étapes à réaliser dans l’ordre et de remarquer rapidement les retards. La documentation est réalisée durant toute la durée du projet et les visites des experts ainsi que les discussions avec le supérieur professionnel sont notées dans la planification.

Les carrés de couleurs représentent dans la grille le temps de travail prévu, et les croix le temps de travail effectif. Pour chaque tâche, on peut savoir si elle est en cours ou si elle est terminée. Enfin, une ligne du temp permet de se rendre compte du temps qui est passé et le temps qu’il reste encore à disposition.

Une image contenant texte, capture d’écran, ligne, Parallèle

Le contenu généré par l’IA peut être incorrect.

Figure 2 : Planning du projet

### Journal de travail

Le journal de travail permet de garder une trace des activités réalisées, le temps effectif passé sur une tâche, et permet également de pouvoir documenté les éventuels problèmes rencontrés. Après chaque journée, une réflexion personnelle est rédigée. Elle permet de mettre son ressenti sur le travail effectué.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure 3 : Extrait du journal de travail

### Rôles des participants

Le TPI implique de multiples personnes, voici leurs noms et leurs rôles :

|  |  |
| --- | --- |
| Participant | Rôle |
| Clara Brodard | Candidate |
| Frédéric Hertling | Formateur en entreprise |
| Mario Ramalho | Supérieur professionnel |
| Marc Denervaud | Expert principal |
| David Jaquier | Expert secondaire |

### Méthodes de sauvegardes

J’ai décidé de sauvegarder la documentation et le code de la manière :

* Une copie en local sur l’ordinateur de l’école
* Une copie sur OneDrive
* Une copie en local sur mon ordinateur personnel

Je vérifie tous les soirs que toutes les versions sont identiques et à jour. L’avantage est que OneDrive permet un versioning des fichiers. Enfin, pour partager le code et avoir une manière de sauvegarder le code GitHub sera utiliser.

Les copies en locales sont renommées avec la date d’enregistrement de celles-ci.

### Travaux préparatoires

Les travaux préparatoires au TPI sont :

* Installation d’un IDE pour développer (Visual studio code)
* Installation de Wamp (PHP, MySQL)
* Installation de Google Chrome

# Analyse

L’analyse a été effectué le premier jour du TPI. Elle a permis de comprendre les enjeux de l’application et à bien débuter le projet.

## Objectifs du système

Les objectifs du projet sont de réaliser une application web locale qui permettra de communiquer avec des GPT. Les chapitres suivants détailent les éléments essentiels de l’application.

### Analyse de l’état désiré

L’application sera une application web héberger sur un serveur local. La maquette s’inspire fortement du visuel d’autres applications permettant des discussions avec des intelligences artificielles.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure 4 : Maquette de l'application web

Nous avons 3 principaux boutons, « Nouvelle discussion » pour créer une nouvelle discussion avec l’IA d’aide générale. La nouvelle discussion s’ajoutera à la liste à gauche.

« Envoyer » pour envoyer une requête à l’IA, la conversation s’affiche dans la grande fenêtre au milieu.

« Nouveau model » permet de créer un GPT personnalisé. Une fenêtre pop-up apparaîtra pour pouvoir entrer le système prompt que l’on souhaite pour notre GPT. Celui-ci apparaît dans la liste au-dessus du bouton. Ils permettront donc l’aide à la rédaction.

En cliquant sur une discussion, on a accès à l’historique et on peut continuer celle-ci. En cliquant sur un GPT, on peut l’utiliser, mais ceux-ci n’ont pas d’historique.

### Processus d’entreprise concernés

Le processus d’entreprise concerné est la rédaction du rapport d’évaluation préliminaire des risques.

Lorsqu'une demande est reçue, accompagnée d’un dossier pénal, l’équipe procède à une analyse manuelle approfondie croisant les informations contenues dans le dossier, des outils d’évaluation criminologique reconnus et la littérature scientifique spécialisée.

Cette analyse aboutit à la rédaction d’un rapport standardisé de 5 à 7 pages, présentant une synthèse claire et rigoureuse des risques identifiés à court terme.

L’ensemble du processus, entièrement réalisé par des spécialistes, est soumis à un délai de livraison de 48 heures maximum.

### Objectifs

On peut résumer les objectifs fonctionnels de l’application grâce au schéma Use-Case suivant :

Une image contenant texte, diagramme, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.

Figure 5 : Diagramme use-case

**Description des acteurs**

|  |  |
| --- | --- |
| Acteur | Description |
| User | Est l’utilisateur utilisant l’application |
| GPT4All | Est l’application fournissant l’API pour utiliser le modèle IA |
| BD | Est la base de données qui contient l’historique des conversations et les prompts systèmes des GPT personnalisés |

**Description des cas**

|  |  |
| --- | --- |
| Cas d’utilisation | Description |
| Nouveau dialogue | Création d’une nouvelle discussion avec le GPT général avec la possibilité d’envoyer des requêtes |
| Afficher discussion enregistrée | Affiche l’historique d’une discussion, avec la possibilité d’envoyer de nouvelle requête |
| Utiliser GPT personnalisé | Affichage pour faire une discussion avec un GPT personnalisé avec la possibilité d’envoyer des requêtes |
| Ajout GPT personnalisé | Création d’un nouveau GPT personnalisé, personnalisation du prompt système |
| Envoyer requête | Envoie d’une requête à l’API GPT4All et reçois la réponse pour l’afficher |
| Enregistrement de la discussion | Si la discussion est une discussion avec le GPT général, les requêtes et les réponses du GPT sont enregistrer dans la base de données |
| Enregistrement des paramètres | Lorsque l’on crée un nouveau GPT personnalisé, le prompt système est enregistré dans la base de données |

## Variantes – Choix du modèle IA

La variante principale pour le début du projet et de choisir quel modèle d’IA on souhaite utiliser. On veut une application qui puisse nous répondre rapidement et de manière cohérente avec l’infrastructure actuel. A savoir que le PC de travail a 64 Go de RAM, et un processeur Intel Core 17.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modèle | Taille | RAM | Paramètres | Quant. | Licence | API & Paramètres | Avantages | Désavantages |
| **Llama 3.2 1B Instruct** | 737 MB | 2 GB | 1B | q4\_0 | Inconnue  (Meta, usage flou) | ✅ Support API | - Ultra léger  - Réponses rapides  - Multilingue | - Peu puissant  - Licence peu claire  - Capacités limitées |
| **Mini Orca (Small)** | 1.98 GB | 4 GB | 3B | q4\_0 | Non commercial | ✅ Support API | - Très rapide  - Instruction tuned  - Taille/puissance équilibrées | - Pas utilisable commercialement  - Moins puissant que 7B modèles |
| **Deepseek R1**  **Distill Qwen 7B** | 4 GB | 8 GB | 7B | q4\_0 | Commercial libre | ✅ Paramètres API modifiables | - Très puissant- Commercial OK- Performances solides | - Lourd  - Démarrage lent sur petite machine |
| **Mistral Instruct 7B** | 3.83 GB | 8 GB | 7B | q4\_0 | Commercial libre | ✅ Paramétrage API total | - Rapide- Non censuré- Excellente qualité des réponses | - RAM élevée nécessaire  - Moins adapté à faible hardware |

Le meilleur choix serait entre le Llama et le Mini Orca. Ils sont les deux rapide, léger et spécialisé dans l’exécution d’instructions. Llama est spécialisé en langues donc il a un avantage pour la rédaction des rapports en français, et pourrait également être utilisé pour des tests futurs pour la rédaction de rapports en allemand. Cependant, Mini Orca ne prend pas en charge le format message, et cela rend ses paramétrages plus compliqués et le fait de retenir les historiques est également plus compliqué.

J’ai décidé de choisir le Llama 3.2 pour sa légèreté et sa rapidité. Elle prend très peu de ressource, ce qui est parfait pour l’application test que l’on développe et sa polyvalence est un avantage.

## Analyse de risque

Le projet a pour but d’être un premier test pour éventuellement aboutir à une application fonctionnelle pour l’entreprise OutSiders. Le risque est simplement que si ce test n’aboutit pas sur un résultat concluant, ils ne demanderont pas de mandat à l’EMF pour réaliser l’application finale.

## Sécurité de l’information et protection des données

OutSiders est soumis à la loi fédérale sur la protection des données dont voici quelques articles intéressants :

**Article 4 – Principes généraux** : Le traitement des données doit être licite, effectué de bonne foi et proportionné.

**Article 5 – Exactitude des données** : Les données doivent être exactes et, si nécessaire, mises à jour.

**Article 8 – Sécurité des données** : Le responsable du traitement doit assurer la sécurité des données personnelles par des mesures techniques et organisationnelles appropriées.

**Article 12 – Registre des activités de traitement** : Les entreprises doivent tenir un registre des activités de traitement des données personnelles.

**Article 22 – Analyse d'impact relative à la protection des données (DPIA)** : Une analyse d'impact est requise si le traitement présente un risque élevé pour les droits fondamentaux des personnes concernées.

**Article 24 – Notification des violations de la sécurité des données** : Les violations de la sécurité des données doivent être signalées sans délai au Préposé fédéral à la protection des données et à la transparence (PFPDT).

**Article 60 – Sanctions** : Les violations intentionnelles des obligations peuvent entraîner des amendes allant jusqu'à 250 000 francs suisses.

Ils doivent également respecter le code pénal suisse sur la violation du secret professionnel (art 321) et le code de procédure pénale suisse pour le traitement et la transmission des données pénales dans le cadre de procédures pénales.

L’infrastructure étant complètement locale va permettre que les données sensibles ne puissent pas quitter l’entreprise. L’IA est également hébergé directement sur le post en local, on ne risque donc pas de donner les informations sensibles à une entreprise tierce.

# Conception

## Exigences du système

Les critères principaux du projet sont :

* Une infrastructure locale
* Une sécurisation des données sensibles
* Une rapidité d’exécution

Les requêtes de l’application seront faites avec PHP et l’application tournera sur un serveur local Wamp, ce qui permettra à l’application de gérer également la base de données en SQL. Le frontend pourra être exécuté sur le navigateur Chrome ou Edge. L’interface web utilisera donc html, css et du javascript. Du json sera utilisé pour les appels à l’API de GPT4All, installé localement également.

## Architecture du système

La solution présentera la structure suivante :

L’application web sera hébergée sur un serveur local qui contiendra des fichiers PHP permettant d’accéder à une base de données également locale et l’application GPT4All via une API produite par celle-ci, et qui permettra d’accéder à un modèle d’IA et aux documents locaux

Une image contenant texte, diagramme, capture d’écran, Plan

Le contenu généré par l’IA peut être incorrect.

Figure 6 : Schéma de l'architecture de l'application

## Plan d’intégration des systèmes

Un manuel d’installation sera élaboré en Markdown qui sera disponible sur GitHub. Ce manuel permettra aux utilisateurs d’installer tous les éléments essentiels pour faire fonctionner l’application.

Il comprendra donc l’installation de Wamp, de comment intégrer l’application au serveur et comment lancer l’application.

## Concept d’implémentation

L’application contiendra un fichier HTML, un fichier JavaScript et de multiples fichier PHP. Nous allons développer ici quelques activités qui seront implémentées.

La première activité est l’affichage d’une discussion enregistrée. On récupère la discussion pour savoir si elle existe, et si elle existe, on va l’afficher dans l’application. Les actions dans la DB ici sont figuratives pour signifier que des données de la base de données sont récupérées puis testées dans l’application.

Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

Figure 7 : Diagramme d'activité enregistrement discussion

La deuxième action est l’envoi d’une requête. On vérifie si elle est vide, si c’est le cas, on affichera une erreur. Sinon, elle va être envoyée à l’API de GPT4All. On vérifie ensuite si cette discussion doit avoir un historique, si c’est le cas, on ajoute la requête et la réponse de la requête dans la base de données. Dans les deux cas, on va afficher dans l’application la requête de l’utilisateur et la réponse.

Une image contenant texte, diagramme, capture d’écran, Parallèle

Le contenu généré par l’IA peut être incorrect.

Figure 8 : Diagramme d'activité envoi d'une requête

La structure de la base de données est assez simple. Chaque discussion est nommée, et ces discussions peuvent avoir de multiples messages.

Pour chaque message, on sait de quelle discussion il appartient, s’il a été généré par un GPT ou par l’utilisateur (boolean), son ordre dans la conversation et enfin le contenu du message.

La table T\_GPT contient les prompts personnalisés qui influencera le comportement de l’IA et le nom du GPT

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure 9 : Schéma base de données

Pour avoir une idée de la structure globale du projet, on peut étudier le diagramme de classe suivant :

Une image contenant texte, capture d’écran, Rectangle, diagramme

Le contenu généré par l’IA peut être incorrect.

Figure 10 : Diagramme de classe

On a donc un fichier html qui fera office d’interface utilisateur. Il est relié à la logique de l’application, app.js. Il pourrait être divisé par worker, à voir avec l’organisation et le temps à disposition. Dans tous les cas, la logique fera appel à des fichiers php pour faire les requêtes à la base de données à l’API de GPT4All.

## Concept de tests

Les tests seront réalisés en blackbox. Cela signifie que l’on va vérifier le fonctionnement de l’application sans prendre en compte le code source ou la structure interne. On teste uniquement ce que le logiciel est censé faire, en se basant sur les spécifications fonctionnelles, les entrées possibles et les résultats attendus. Les avantages sont la facilité de réalisation, même pour une personne qui ne connait pas le code et de tester avec le point de vue de l’utilisateur final. Le désavantage principal est que l’on ne teste pas toutes les branches de code, ce qui peut nous faire manquer des bugs liés à la structure du programme.

Les entêtes du tableau des tests sera le suivant :

|  |  |
| --- | --- |
| Entête | Description |
| Nr. | Numéro du test |
| Objet testé | Le cas d’utilisation testé, ce qui est testé |
| Description du test | Un complément d’information sur le test, comment le réaliser |
| Attente | Le résultat attendu si le test fonctionne |
| Résultat | Le résultat obtenu |
| Etat | L’état du test OK/KO en fonction si le test est réussi ou non |
| Visa | Date et initiale de la personne qui a effectué le test. |

# Réalisation

## Spécifications détaillées

Nous allons voir le code de l’application de plus près, notamment le code en lien à l’ajout d’une nouvelle discussion à la discussion avec le modèle de GPT4All.

### Code accès api

**Dans gpt4all.php**

On démarre une session pour pouvoir gérer les variables de session, on définit le type du contenu de la réponse qui sera du json et on inclut le fichier de connexion à la base de données

<?php

session\_start();

header("Content-Type: application/json");

require\_once '../includes/db.php';

On essaye ensuite de récupérer une instance PDO pour la connexion à la base de données, en cas d’erreur, on retourne un message d’erreur et on stop toute exécution.

try {

  $pdo = getPDO();

} catch (PDOException $e) {

  echo json\_encode(['error' => 'Connexion DB échouée : ' . $e->getMessage()]);

  exit;

}

On a ici une fonction qui vérifie si un ID de discussion existe dans la base de données, et on retourne true si elle existe, sinon false. Ensuite on va vérifier la réponse et vérifier si on a bien un message reçu.

function isDiscussionIdValide($pdo, $id) {

  $stmt = $pdo->prepare("SELECT 1 FROM t\_discussion WHERE PK\_Discussion = ?");

  $stmt->execute([$id]);

  return $stmt->fetch() !== false;

}

// Récupère le corps JSON

$data = json\_decode(file\_get\_contents("php://input"), true);

// Vérifie l'existence des messages

if (!isset($data["messages"]) || !is\_array($data["messages"])) {

    echo json\_encode(["error" => "Aucun message fourni."]);

    exit;

}

Si aucune discussion n’est active ou si l’ID stocké n’est pas valide, on va créer une nouvelle discussion dans la base de données basé sur la date et l’heure courante. Puis on va stocker le nouvel ID dans la variable de session.

// Si aucune discussion active → en créer une

if (!isset($\_SESSION["discussion\_id"]) || !isDiscussionIdValide($pdo, $\_SESSION["discussion\_id"])) {

  $stmt = $pdo->prepare("INSERT INTO t\_discussion (Nom) VALUES (:nom)");

  $stmt->execute(["nom" => "Discussion " . date("Y-m-d H:i:s")]);

  $\_SESSION["discussion\_id"] = $pdo->lastInsertId();

}

On va ensuite récupérer l’ID de la discussion active et le tableau des messages, on va ensuite ordré les messages.

$discussionId = $\_SESSION["discussion\_id"];

$messages = $data["messages"];

$stmt = $pdo->prepare("SELECT COALESCE(MAX(Ordre), 0) FROM t\_message WHERE FK\_Discussion = ?");

$stmt->execute([$discussionId]);

$lastOrder = (int)$stmt->fetchColumn();

On récupère le dernier message, si le dernier message est un message utilisateur, on va l’ajouter à la base de données.

$dernierMsg = end($messages);

if ($dernierMsg['role'] === 'user') {

    $userOrder = $lastOrder + 1;

    $stmt = $pdo->prepare("

      INSERT INTO t\_message (GPT, Ordre, Message, FK\_Discussion)

      VALUES (0, :ordre, :message, :discussion)

    ");

    $stmt->execute([

      'ordre'      => $userOrder,

      'message'    => $dernierMsg['content'],

      'discussion' => $discussionId

    ]);

} else {

    // Si par hasard ce n'est pas un user, on arrête

    echo $response;

    exit;

}

On va ensuite préparer la conversation avec l’API de GPT4All en configurant le cURL. On va utiliser du POST.

// URL locale du service GPT4All pour générer une réponse

$gptUrl = "http://localhost:4891/v1/chat/completions";

// Initialise une session cURL pour envoyer la requête à GPT4All

$ch = curl\_init($gptUrl);

// Configure cURL pour retourner la réponse sous forme de chaîne (et pas l'afficher directement)

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

// Indique qu'on va envoyer une requête POST

curl\_setopt($ch, CURLOPT\_POST, true);

// Définit l'entête HTTP Content-Type à application/json

curl\_setopt($ch, CURLOPT\_HTTPHEADER, [

    "Content-Type: application/json"]);

// Envoie les données JSON (messages reçus) au service GPT4All

curl\_setopt($ch, CURLOPT\_POSTFIELDS, json\_encode($data));

// Exécute la requête cURL et stocke la réponse dans $response

$response = curl\_exec($ch);

curl\_close($ch); // Ferme la session cURL

Pou des raisons de débogage, on va créer la réponse dans un fichier. Cela va être utile si on reçoit un message d’erreur de la part de l’API ou que l’API n’est pas accessible.

// Analyse de la réponse

file\_put\_contents('debug\_gpt\_response.txt', $response); // écrit la réponse dans un fichier pour debug

// Décode la réponse JSON reçue en tableau PHP

$responseData = json\_decode($response, true);

// Récupère le contenu du message généré par GPT (s'il existe)

$reply = $responseData["choices"][0]["message"]["content"] ?? null;

Si la réponse du GPT n’est pas vide, on va l’enregistrer dans la base de données.

// Insérer la réponse de GPT

if (!empty($reply)) {

  $gptOrder = $userOrder + 1;

  $stmt = $pdo->prepare("

    INSERT INTO t\_message (GPT, Ordre, Message, FK\_Discussion)

    VALUES (1, :ordre, :message, :discussion)

  ");

  $stmt->execute([

    'ordre'      => $gptOrder,

    'message'    => $reply,

    'discussion' => $discussionId

  ]);}

// Retourner la réponse telle quelle

echo $response;

### Paramètre gpt4all et installation GPT

Dans GPT4all, il va falloir installer le modèle. Il faut aller dans le menu Models puis appuyer sur le bouton +add Model.

Une image contenant texte, Police, capture d’écran, conception

Le contenu généré par l’IA peut être incorrect.

Figure 11 : Menu models de GPT4All

Une image contenant texte, Police, logo, symbole

Le contenu généré par l’IA peut être incorrect.

Figure 12 : Ajouter un modèle

Dans la liste des GPT, il faut installer Llama 3.2 1B Instruct.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure 13 : Modèle Llama 3.2

Il va donc s’ajouter dans les modèles installés.

Dans les settings, application, il faut ouvrir l’API et vérifier que le port est bien 4891 :



Figure 14 : Paramètres API

Enfin, on va lier nos documents à notre GPT. On va dans LocalsDocs puis on appuie sur + Add Collection.

Une image contenant texte, Police, logo, symbole

Le contenu généré par l’IA peut être incorrect.

Figure 15 : Ajouter une collection de documents

On appelle la collection OuSidertsDoc et on désigne le dossier contenant les fichiers de références de OutSiders.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Figure 16 : Création de la collection

Enfin, à chaque fois que l’on lancera GPT4All, il va falloir aller dans les Chats, aller sur la discussion avec le server et désigner les localDocs :

Une image contenant texte, capture d’écran, logiciel, Page web

Le contenu généré par l’IA peut être incorrect.

Figure 17 : Lier une collection à un modèle

### App.js

**Ajouter une nouvelle discussion**

La fonction suivante permet de démarrer une nouvelle discussion en envoyant une requête HTTP à l'API côté serveur. On utilise fetch pour appeler le fichier new\_discussion.php, lequel crée une nouvelle entrée dans la base de données pour une nouvelle discussion. On attend la réponse et on la traite sous forme de JSON.

Si la réponse indique un succès (data.success == true), on réinitialise l’historique de conversation côté client. On crée un message système par défaut qui définit le comportement de l’assistant.

function startNewConversation() {

  fetch("http://localhost/OutsidersGPT/api/new\_discussion.php", { method: "GET" })

    .then(response => response.json())

    .then(data => {

      if (data.success) {

        // Réinitialise l'historique côté client

        conversationHistory = [

          {

            role: "system",

            content: "Tu es un assistant utile. Réponds clairement et poliment."

          }

        ];

On efface les messages affichés à l’écran et on vide le champ de saisie de l'utilisateur. On ajoute dynamiquement un nouveau lien vers cette discussion dans la barre latérale, avec le nom renvoyé par le serveur (exemple "Discussion 2025-06-02 14:30:45").

        document.getElementById("chat").innerHTML = "";

        document.getElementById("prompt").value = "";

        // Ajoute dynamiquement le lien dans la barre latérale

        const discussionsContainer = document.querySelector(".discussions");

        const newLink = document.createElement("a");

        newLink.href = "#";

        newLink.textContent = data.nom;

        discussionsContainer.appendChild(newLink);

        loadDiscussions();

      } else {

        console.error(data.message);

      }

    })

    .catch((err) => {

      console.error("Erreur lors de la création de la discussion :", err);

    });

}

**Charger discussions**

La fonction suivante permet de récupérer depuis la base de données la liste des discussions existantes et de les afficher dynamiquement dans l’interface utilisateur.

function loadDiscussions() {

  fetch("http://localhost/OutsidersGPT/api/get\_discussions.php")

On attend la réponse du serveur, qu’on transforme ensuite en JSON pour la traiter en JavaScript. Si la réponse renvoyée contient une erreur, on affiche un message d’erreur et on arrête l’exécution.

    .then(res => res.json())

    .then(data => {

      if (!data.success) {

        console.error("Erreur récupération discussions :", data.message);

        return;

      }

On sélectionne l’élément HTML qui contiendra la liste des discussions, puis on vide son contenu pour éviter les doublons.

      const container = document.querySelector(".discussions");

      container.innerHTML = ""; // vide l'existant

Pour chaque discussion reçue, on crée dynamiquement un lien (<a>) avec comme texte le nom de la discussion, et on stocke son identifiant dans un attribut data-id. On ajoute ensuite un écouteur d’événement sur chaque lien pour qu’au clic, on charge les messages associés à cette discussion via la fonction loadMessagesForDiscussion(). Finalement, on ajoute chaque lien à l’élément HTML conteneur. En cas d’erreur de requête (par exemple si le serveur ne répond pas), on affiche un message dans la console.

      data.discussions.forEach(disc => {

        const a = document.createElement("a");

        a.href = "#";

        a.textContent = disc.Nom;

        a.dataset.id = disc.PK\_Discussion;

        a.addEventListener("click", () => {

          loadMessagesForDiscussion(disc.PK\_Discussion);

        });

        container.appendChild(a);

      });

    })

    .catch(err => {

      console.error("Erreur chargement discussions :", err);

    });

}

**Afficher l’historique d’une discussion**

Cette fonction permet de charger l’historique des messages associés à une discussion précise à partir de son identifiant. On envoie une requête GET à la base de données en précisant l’ID de la discussion à charger. Si la réponse contient une erreur (par exemple, ID de discussion inexistant), un message est affiché dans la console et l’exécution s’arrête.

function loadMessagesForDiscussion(id) {

  fetch(`http://localhost/OutsidersGPT/api/load\_messages.php?id=${id}`)

    .then(res => res.json())

    .then(data => {

      if (!data.success) {

        console.error("Erreur récupération messages :", data.message);

        return;

      }

Avant d’afficher les nouveaux messages, on vide l’affichage existant et on réinitialise la variable conversationHistory avec un message système par défaut.

      // Réinitialise l'affichage

      document.getElementById("chat").innerHTML = "";

      conversationHistory = [

        {

          role: "system",

          content: "Tu es un assistant utile. Réponds clairement et poliment."

        }

      ];

On parcourt ensuite chaque message retourné par l’API. En fonction de la colonne GPT (1 pour assistant, 0 pour utilisateur), on détermine le rôle du message.

Chaque message est affiché à l’écran à l’aide de la fonction appendMessage(), puis ajouté à l’historique de conversation en mémoire (conversationHistory), afin de pouvoir être envoyé de nouveau à l'API si besoin.

Si une erreur survient durant la requête (exemple : API inaccessible), un message d’erreur est affiché dans la console.

      data.messages.forEach(msg => {

        const role = msg.GPT == 1 ? "assistant" : "user";

        appendMessage(role, msg.Message);

        conversationHistory.push({

          role,

          content: msg.Message

        });

      });

    })

    .catch(err => {

      console.error("Erreur chargement messages :", err);

    });

}

### Db.php

Ce fichier contient une fonction utilitaire permettant d’établir une connexion sécurisée à la base de données MySQL via PDO.

On définit les paramètres de connexion à la base de données :

* host : l’adresse du serveur de base de données (ici, en local)
* dbname : le nom de la base
* username et password : les identifiants de connexion à MySQL

On crée une instance de PDO avec les paramètres fournis, en spécifiant également l'encodage UTF-8 pour la gestion correcte des caractères spéciaux.

On active le mode d’exception pour que toute erreur de requête lève une exception PDOException, ce qui permet une meilleure gestion des erreurs.

Enfin, on retourne l’objet $pdo, prêt à être utilisé pour exécuter des requêtes SQL.

<?php

function getPDO() {

    $host = 'localhost';

    $dbname = 'mydb';

    $username = 'root';

    $password = '';

    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);

    $pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

    return $pdo;

}

### Get\_discussion.php

Ce fichier permet de récupérer, depuis la base de données, la liste de toutes les discussions enregistrées. La réponse est formatée en JSON pour être utilisée facilement côté client (dans l’interface JavaScript).

On essaye d’ouvrir une connexion à la base de données et d’exécuter une requête pour récupérer toutes les discussions, triées par ordre décroissant (les plus récentes en premier).

Le résultat de la requête est transformé en tableau associatif contenant tous les enregistrements retournés par la base.

La réponse est ensuite envoyée au format JSON, avec une clé success à true et la liste des discussions.

Si une erreur survient (par exemple, problème de connexion ou erreur SQL), on capture l’exception PDOException et on renvoie un message d’erreur dans la réponse JSON.

<?php

header('Content-Type: application/json');

require\_once '../includes/db.php';

try {

    $pdo = getPDO();

    $stmt = $pdo->query("SELECT PK\_Discussion, Nom FROM t\_discussion ORDER BY PK\_Discussion DESC");

    $discussions = $stmt->fetchAll(PDO::FETCH\_ASSOC);

    echo json\_encode(['success' => true, 'discussions' => $discussions]);

} catch (PDOException $e) {

    echo json\_encode(['success' => false, 'message' => $e->getMessage()]);

}

?>

### New\_discussion.php

Ce script est appelé lorsque l'utilisateur démarre une nouvelle conversation. Il crée une entrée dans la base de données et mémorise l’ID de cette discussion dans la session.

On tente d’ouvrir une connexion PDO à la base de données. En cas d’erreur, une exception sera attrapée.

On crée un nom par défaut pour la nouvelle discussion, basé sur la date et l’heure actuelles.

On prépare et exécute une requête SQL pour insérer une nouvelle ligne dans la table t\_discussion.

Une fois la discussion insérée, on récupère son identifiant (clé primaire) et on le stocke en session pour pouvoir s’y référer dans les prochaines requêtes.

On renvoie une réponse JSON indiquant que la discussion a bien été créée, avec son nom.

En cas d’erreur (connexion à la base ou exécution de la requête), on retourne un message d’échec.

<?php

session\_start();

header('Content-Type: application/json');

require\_once 'db.php';

try {

    $pdo = getPDO();

    $nom = 'Discussion ' . date('Y-m-d H:i:s');

    $stmt = $pdo->prepare("INSERT INTO t\_discussion (Nom) VALUES (:nom)");

    $stmt->execute(['nom' => $nom]);

    // Stocke l'ID de la discussion créée

    $\_SESSION['discussion\_id'] = $pdo->lastInsertId();

    echo json\_encode(['success' => true, 'message' => 'Discussion créée', 'nom' => $nom]);

} catch (PDOException $e) {

    echo json\_encode(['success' => false, 'message' => 'Erreur : ' . $e->getMessage()]);

}

?>

## Design du système

Le diagramme de l’application web est la suivante :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure 18 : Diagramme de classe final

App.js aurait pu être divisé en plusieurs worker. Mais le manque de temps et au vu de la taille assez petite du projet, toute la logique est dans ce fichier. Chaque fonction fait appel à un fichier php différent, voici le résumé :

|  |  |  |
| --- | --- | --- |
| Fonction JavaScript | Fichier PHP appelé | Rôle de la fonction |
| sendPrompt() | /api/gpt4all.php | Envoie une requête à GPT pour générer une réponse. |
| startNewConversation() | /api/new\_discussion.php | Crée une nouvelle discussion côté serveur. |
| loadDiscussions() | /api/get\_discussions.php | Récupère la liste des discussions existantes. |
| loadMessagesForDiscussion(id) | /api/load\_messages.php?id=${id} | Charge les messages d’une discussion spécifique. |
| (dans saveGptBtn event listener) | /api/save\_gpt.php | Sauvegarde un nouveau GPT personnalisé (nom + prompt système). |
| loadCustomGpts() | /api/get\_gpts.php | Charge la liste des GPT personnalisés. |
| loadGptAsSystemPrompt(id, name) | /api/get\_gpt\_prompt.php?id=${id} | Récupère un prompt système d’un GPT personnalisé, selon son ID. |

Nous allons dans les chapitres suivant voir les problèmes rencontrés durant le projet.

### Problème rencontré

Deux problèmes ont été rencontré durant le projet. Heureusement, ils ont été résolus assez rapidement et n’ont pas causés de retard dans la planification.

#### Problème 1 : N’arrive pas à créer de nouvelle discussion

Il y a eu un problème pour enregistrer une nouvelle discussion dans la base de données.

Quand je cliquais sur le bouton nouvelle discussion, l’erreur suivante s’affichait :

SQLSTATE[23000]: Integrity constraint violation: 1062 Duplicata du champ '0' pour la clef 't\_discussion.PRIMARY'

La cause était que la table t\_discussion n’avait pas une clé primaire qui s’auto-incrémentait.

La solution a été de modifier la table pour qu’elle devienne auto-incrémentée

ALTER TABLE t\_discussion

MODIFY COLUMN PK\_Discussion INT NOT NULL AUTO\_INCREMENT;

Et de mettre à jour new\_discussion.php avec cette ligne :

$\_SESSION['discussion\_id'] = $pdo->lastInsertId();

#### Problème 2 : Enregistrement à double de certains messages dans la base de données

Le deuxième problème rencontré était que à chaque message envoyé dans une discussion, tous les messages précédents étaient réinsérés dans la base de données.

La cause était que dans gpt4all.php, on insérait tous les messages du tableau messages à chaque requête.

foreach ($messages as $msg) {

if ($msg["role"] === "user") {

// INSERT ...

}

}

Mais ce tableau contient l’historique complet de la discussion, donc les anciens messages étaient insérés à nouveau.

La solution était de remplacé la boucle par une insertion du dernier message utilisateur.

$dernierMessage = end($messages);

if ($dernierMessage["role"] === "user") {

// INSERT une seule fois ce message}

Puis afin d’avoir un ordre correct dans la base de données, il a fallu récupérer le numéro du dernier message sauvegardé :

$stmt = $pdo->prepare("SELECT MAX(Ordre) FROM t\_message WHERE FK\_Discussion = ?");

$stmt->execute([$discussionId]);

$order = ($stmt->fetchColumn() ?? 0) + 1;

# Test

## Procédure de test

A chaque fois qu’une fonctionnalité principale était implantée, le code a été testé. L’interface a été testé sur Edge Version 137.0.3296.52 (64 bits) et sur Windows 11 Entreprise 24H2. La version de Wamp était 3.3.0 64 bits et la version de GPT4All était la v3.10.0. Le résultat des tests est dans le chapitre ci-dessous.

## Protocole de test

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Etat | Visa |
| 1.0 | Nouvelle discussion générale | Création d’une nouvelle conversation en cliquant sur le bouton « nouvelle discussion » | Elle s’ajoute dans la liste avec le nom « discussion xxx» et il n’y a pas d’historique, s’il y avait une autre discussion affichée elle s’efface. |  | OK | 22.05.25  Clara  Brodard |
| 1.1 | Envoyer une requête dans la nouvelle discussion | Envoyer une question à l’IA. | L’IA répond de manière pertinente et rapidement |  | OK | 22.05.2025  Clara  Brodard |
| 1.2 | Envoyer une requête « vide » | Envoyer une requête vide à l’IA | Un message d’erreur s’affiche | Aucune requête n’est envoyée mais il n’y a pas de message d’erreur | KO | 23.05.2025  Clara  Brodard |
| 2.0 | Afficher une ancienne discussion | On clique dans la liste des discussions sur une d’entre elle | L’historique de la discussion s’affiche |  | OK | 23.05.2025  Clara  Brodard |
| 2.1 | Envoyer une requête dans une ancienne discussion | On envoie une question à l’IA | L’IA répond de manière pertinente et rapidement, et connait le contexte de la discussion |  | OK | 23.05.2025  Clara Brodard |
| 2.2 | Vérifier que la discussion est bien enregistrée | Quitter l’application et retourner, puis rouvrir la discussion | L’historique de la discussion est à jour | Après avoir quitté puis revenu sur la discussion, l’historique de la discussion est toujours là et à jour | OK | 23.05.2025 |
| 3.0 | Utiliser un GPT personnalisé | On clique sur un des GPT personnalisé | Il n’y a pas d’historique, s’il y avait une autre discussion affichée elle s’efface. |  | OK | 27.05.2025 |
| 3.1 | Envoyer une requête vide à un GPT personnalisé | On envoie une requête en lien avec le paragraphe que le GPT personnalisé doit produire | Le GPT produit le paragraphe selon sa personnalisation |  | OK | 27.05.2025 |
| 3.2 | Ajout d’un GPT personnalisé | On clique sur le GPT personnalisé | Une fenêtre s’affiche dans laquelle on peut ajouter un prompt pour le GPT personnalisé. En cliquant sur OK, le nouveau GPT personnalisé est créé et ajouté à la liste | L’ajout se fait mais il faut recharger la page pour voir le GPT personnalisé être ajouté à la liste. | OK | 27.05.2025 |
| 4.0 | Faire une longue discussion | On envoie plusieurs messages jusqu’à ce que la fenêtre soit trop petite de tout afficher | La fenêtre de se déforme pas et on a une scrollbar pour naviguer dans la discussion |  | OK | 30.05.2025 |
| 4.1 | Ajouter beaucoup de discussion | Ajouter des discussions jusqu’à ce que la zone soit trop petite pour tout afficher | La fenêtre de se déforme pas et on a une scrollbar pour naviguer | La liste s’agrandit à l’infini, et il n’y a pas de scrollbar | KO | 30.05.2025 |
| 4.2 | Ajouter beaucoup de GPT personnalisés | Ajouter des discussions jusqu’à ce que la zone soit trop petite pour tout afficher | La fenêtre de se déforme pas et on a une scrollbar pour naviguer | La liste s’agrandit à l’infini, et il n’y a pas de scrollbar | KO | 30.05.2025 |

## Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
| 30.05.2025 | Clara Brodard |  |

# Conclusion

Tous les objectifs du cahier des charges sont atteints. Le seul bémol par rapport aux objectifs est que bien qu’avec un prompt personnalisé, l’IA choisi, Llama, a du mal à suivre les instructions. Je pense que tester avec une IA plus puissante pourrait régler ce problème.

Bien que des améliorations sont possibles, l’application est fonctionnelle. Pour une bonne exécution du rapport, ChatGPT a été utiliser pour corriger les fautes d’orthographe et pour formuler certaines phrases.

## Améliorations possibles

Plusieurs modifications peuvent être apporté à l’application :

**Suppression d’une conversation**

En effet, le cahier des charges ne demandait pas de pouvoir supprimer une conversation. Cela est présent dans toutes les applications de GPTs. Cela éviterait également de surcharger la base de données.

**Affichage des conversations**

A ce jour, le modèle renvoi des réponses avec un certains formatage qui n’est pas lu par l’application, ce qui peut rendre certains messages difficiles à lire. Il serait possible d’interpréter ce formatage.

Affichage dans GPT4All :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Figure 19 : Exemple formatage GPT4All

Affichage dans l’application OutSidersGPT :

Une image contenant texte, Police, nombre, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure 20 : Exemple formatage dans l'application web

**Adaptation des listes**

La liste des discussions et la liste des GPTs personnalisés ne s’adaptent pas très bien à la fenêtre. Lorsque l’espace dédié est plein, il faudrait, comme pour la conversation, avoir une barre déroulante pour naviguer à travers ses deux listes facilement. Cependant, en l’état actuel, cela n’empêche pas d’utiliser l’application.

**Possibilité de changer de modèle**

Une dernière fonctionnalité que l’on pourrait ajouter serait de pouvoir changer de modèle quand on le souhaite. Cela pourrait être intéressant pour demander des tâches plus complexes.

**Message d’attente**

Lorsque l’on envoi une requête, GPT4All peut prendre plus ou moins de temps pour répondre. On pourrait ajouter un message « réponse en chargement » ou un message du même genre afin de faire comprendre à l’utilisateur que la réponse est en train d’être généré par l’IA.

Une autre solution serait de récupérer la réponse par stream et afficher petit à petit celle-ci sur l’interface.

## Auto-évaluation

J’ai bien aimé le projet. Je trouve toujours intéressant de travailler avec l’IA, qui est en train de prendre beaucoup de terrain sur de nombreux domaines. J’aurais aimé ajouter quelques améliorations mais les objectifs ont été atteints.

Je pense que la bonne planification du projet m’a permis de réaliser tout à temps, et même d’avoir plus de temps pour la documentation. Malheureusement, ce surplus de temps m’a également fait me détendre et j’étais facilement distrait. J’aurais dû me concentrer jusqu’au bout.

# Bibliographie : liste des sources et références

Nom description lien tableau

|  |  |  |
| --- | --- | --- |
| Nom | Description | Lien |
| GPT4All doc | La documentation de l’application GPT4All | [GPT4All](https://docs.gpt4all.io/index.html) |
| GPT4All app | Lien d’installation de l’application de GPT4All | [GPT4All – The Leading Private AI Chatbot for Local Language Models](https://www.nomic.ai/gpt4all) |
| Sécurité des données | Loi fédérale sur la sécurité des données | [RS 235.1 - Loi fédérale du 25 septembre 2020 sur... | Fedlex](https://www.fedlex.admin.ch/eli/cc/2022/491/fr) |
| GitHub | Lien pour le projet OutSidersGPT |  |
| Wamp | Lien de téléchargement du server Wamp | [Download wampserver3.3.0\_x86.exe (WampServer)](https://sourceforge.net/projects/wampserver/files/WampServer%203/WampServer%203.0.0/wampserver3.3.0_x86.exe/download) |

# Glossaire

|  |  |
| --- | --- |
| Mots | Description |
| API | Interface de programmation qui permet à deux applications de communiquer entre elles. |
| BD | Abréviation de Base de Données, utilisée pour stocker des informations de manière structurée. |
| ChatGPT | Modèle de langage développé par OpenAI, capable de générer du texte et de répondre à des questions. |
| crud | Acronyme pour Create, Read, Update, Delete, opérations de base sur une base de données. |
| css | Feuilles de style en cascade utilisées pour styliser les pages HTML (couleurs, polices, mises en page…). |
| gantt | un outil de planification visuelle utilisé pour représenter l’avancement d’un projet dans le temps. Chaque tâche y est affichée sous forme de barre horizontale, indiquant sa durée, sa date de début et de fin, ainsi que ses dépendances éventuelles avec d'autres tâches. |
| github | Plateforme de partage et de gestion de code source utilisant Git. |
| GPT | Generative Pretrained Transformer, type de modèle d'IA conçu pour comprendre et générer du langage. |
| html | Langage de balisage utilisé pour structurer le contenu des pages web. |
| IA | Intelligence Artificielle, discipline qui vise à créer des systèmes capables de simuler l’intelligence humaine. |
| javascript | Langage de programmation utilisé principalement pour rendre les pages web interactives. |
| json | Format léger d’échange de données, facile à lire et à écrire pour les humains et les machines. |
| markdown | Langage de balisage léger permettant de formater du texte simplement (titres, listes, liens, etc.). |
| onedrive | Service de stockage en ligne proposé par Microsoft pour sauvegarder et partager des fichiers. |
| PDO | PDO et une interface PHP qui permet à une base de données et d’y exécuter des requêtes SQL de manière sécurisé et flexible. |
| php | Langage de script côté serveur utilisé pour créer des pages web dynamiques. |
| Pkorg | Nouvelle plateforme fédérale pour l’organisation des TPI. |
| pop-up | Fenêtre qui s’ouvre automatiquement dans une page web, souvent pour afficher une information ou une alerte. |
| prompt | Entrée ou consigne textuelle envoyée à un système comme ChatGPT pour générer une réponse. |
| sql | Structured Query Language, langage de requête utilisé pour interagir avec des bases de données relationnelles. |
| stream | Un stream est une méthode de transfert de données où les données sont envoyées petit à petit (en continu) au lieu d’attendre que tout soit prêt avant de les transmettre. |
| TPI | Travail Pratique Individuel, généralement un projet de fin d’étude ou d’évaluation en informatique. |
| wamp | |  | | --- | | Acronyme de Windows, Apache, MySQL, PHP — environnement de développement web local sous Windows. | |

# Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le 02.06.2025 dans le cadre de la procédure de qualification de mon CFC d’informaticienne, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
| 02.06.2025 | Clara Brodard |  |

# Annexes

* Lien GitHub et les fichiers du zip
* Code source de l’application : [emf-info-tpi/24-25-OutSidersGPT](https://github.com/emf-info-tpi/24-25-OutSidersGPT)
* Dans le fichier .zip se trouvent
  + Journal de travail (Brodard\_Clara\_Journal\_2025.xlsx
  + Planning (Brodard\_Clara\_Planning\_2025.xlsx)
  + Schéma de la base de données (db.mwb)
  + SQL de la base de données (DB.sql)
  + La maquette de l’application (maquette.png)
  + Les schémas (SCHEMA.qea)
  + Le PV de la première visite des experts (PV\_Visite\_experts\_1.docx)
  + Dans le dossier Exemples\_Doc, se trouvent les documents fournis par OutSiders Sàrl
  + Code source de l’application (dans le dossier OutSidersGPT)