SIMPLON.CO

Référentiels d'activités, de compétences et d'évaluation

Développeur en intelligence artificielle

REFERENTIEL D'ACTIVITES *Décrit les situations de travail et les activités exercées, les métiers ou emplois visés*	**REFERENTIEL DE COMPETENCES** *Identifie les compétences et les connaissances, y compris transversales, qui découlent du référentiel d'activités*	**REFERENTIEL D'EVALUATION** *Définit les critères et les modalités d'évaluation des acquis*	
		MODALITES D'EVALUATION	**CRITÈRES D'ÉVALUATION**
Bloc de compétences 1 : Réaliser la collecte, le stockage et la mise à disposition des données d'un projet en intelligence artificielle			
A1 : Programmer la collecte de données depuis plusieurs sources pour un projet en intelligence artificielle - Identification des contraintes techniques propres aux sources de données (consulter la documentation, les règles de confidentialité, etc.). - Rédaction des spécifications techniques pour l'extraction des données.	**C1. Automatiser l'extraction de données** depuis un service web, une page web (*scraping**), un fichier de données, une base de données et un système *big data** en programmant le *script** adapté afin de pérenniser la collecte des données nécessaires au projet.	**E1. Mise en situation (C1, C2, C3, C4, C5)** L'évaluation doit se faire dans un contexte de réalisation d'un service numérique réel ou fictif basé sur l'usage de données, à partir du cadrage pour la réalisation d'un service numérique (spécifications fonctionnelles et techniques par exemple). Le projet évalué a pour but d'optimiser, d'automatiser, de pérenniser et de mettre à disposition les flux de données et les données, utiles et nécessaires à la	- La présentation du projet et de son contexte est complète : acteurs, objectifs fonctionnels et techniques, environnements et contraintes techniques, budget, organisation du travail et planification. - Les spécifications techniques précisent : les technologies et outils, les services externes, les exigences de programmation (langages), l'accessibilité (disponibilité, accès). - Le périmètre des spécifications techniques est complet : il couvre

- Construction les requêtes HTTP pour la récupération des données depuis un service web (REST*). - Lecture d'un fichier de données dans un script (Python, R, etc.). - Téléchargement de l'HTML d'une ou plusieurs pages web visées par une action de *scraping*. - Connexion programmatique (Python, R, etc.) à un système de gestion de base de données et à un système big data (Hive, Apache Impala, etc.). - Programmation des filtrages/parsing des données utiles dans les résultats obtenus depuis : - une API, - depuis la lecture d'un ou de plusieurs fichiers de données, - et depuis l'HTML collecté d'un site web (*scraping*). - Exécution programmatique des requêtes d'extraction de type SQL*. - Exécution programmatique des requêtes d'extraction depuis un système big data.		réalisation du service numérique, par les équipes techniques (par exemple en analyse statistique, en business intelligence, en machine learning ou encore en intelligence artificielle). Livrable : rapport professionnel individuel Évaluation : - Correction du rapport professionnel - Soutenance orale individuelle	l'ensemble des moyens techniques à mettre en œuvre pour l'extraction et l'agrégation des données en un jeu de données brutes final. - Le script d'extraction des données est fonctionnel : toutes les données visées sont effectivement récupérées à l'issue de l'exécution du script. - Le script comprend un point de lancement, l'initialisation des dépendances et des connexions externes, les règles logiques de traitement, la gestion des erreurs et des exceptions, la fin du traitement et la sauvegarde des résultats. - Le script d'extraction des données est versionné* et accessible depuis un dépôt Git*. - L'extraction des données est faite depuis un mix entre au moins les sources suivantes : un service web (API REST), un fichier de données, un scraping, une base de données et un système big data.
- Ecriture des requêtes d'extraction de type SQL de récupération de données	**C2. Développer des requêtes de type SQL d'extraction des données** depuis un système		- Les requêtes de type SQL pour la collecte de données sont fonctionnelles : les

stockées en base de données et depuis un système big data (Hive, Spark, etc). - Documentation des requêtes d'extraction.	de gestion de base de données et un système big data en appliquant le langage de requête propre au système afin de préparer la collecte des données nécessaires au projet.		données visées sont effectivement extraites suites à l'exécution des requêtes. - La documentation des requêtes met en lumière choix de sélections, filtrages, conditions, jointures, etc., en fonction des objectifs de collecte. - La documentation explicite les optimisations appliquées aux requêtes .
- Rédaction des spécifications techniques pour l'agrégation des données. - Programmation des règles d'agrégation des données collectées depuis chaque source en un jeu de données brutes unique. - Programmation de l'identification des entrées corrompues dans le jeu de données (par exemple, données partielles et/ou manquantes). - Programmation de la suppression des entrées corrompues. - Programmation de l'identification des entrées au format non normalisé. - Programmation de la l'homogénéisation des formats de données (par exemple, format des dates, des unités, etc.).	**C3. Développer des règles d'agrégation de données** issues de différentes sources en programmant, sous forme de script, la suppression des entrées corrompues et en programmant l'homogénéisation des formats des données afin de préparer le stockage du jeu de données final.		- Le script d'agrégation des données est fonctionnel : les données sont effectivement agrégées, nettoyées et normalisées en un seul jeu de données à l'issue de l'exécution du script. - Le script d'agrégation des données est versionné et accessible depuis un dépôt Git. - La documentation du script d'agrégation est complète : dépendances, commandes, les enchaînements logiques de l'algorithme, les choix de nettoyage et d'homogénéisation des formats données.

- Versionnement* des scripts avec Git et un dépôt Git en ligne. - Documentation des scripts.			
<u>A2 : Développer la mise à disposition technique des données collectées pour un projet d'intelligence artificielle</u> - Rédaction des spécifications techniques pour le stockage des données. - Modélisation de la structure des données de la base de données selon la méthode Merise*. - Choix du système de gestion de base de données. - Création de la base de données dans le système de gestion de base de données. - Documentation de la procédure d'installation du système de gestion de base de données. - Rédaction ou mise à jour du registre des traitements de données personnelles en vue de la mise en conformité de la base de données avec le RGPD. - Rédaction des procédures de tri des	**C4. Créer une base de données** dans le respect du RGPD en élaborant les modèles conceptuels et physiques des données à partir des données préparées et en programmant leur import afin de stocker le jeu de données du projet.		- Les modélisations des données respectent la méthode et le formalisme Merise. - Le modèle physique des données est fonctionnel : il est intégré avec succès lors de la création de la base de données, sans erreur. - La base de données est choisie au regard de la modélisation des données et des contraintes du projet. - La reproduction des procédures d'installation décrites (base de données et API) a pour résultat un système conforme aux objets techniques attendus.. - Le script d'import fourni est fonctionnel : il permet l'insertion des données dans le système mis en place. - La documentation technique du script d'import est versionné à la racine du même dépôt Git que celui utilisé pour le script d'import. - Les documentations techniques des script

données personnelles stockées dans la base de données pour la mise en conformité avec le RGPD (par exemple : détection et suppression des données personnelles inutiles, trop anciennes, etc.). - Programmation du script d'import des données en base de données. - Documentation du script d'import.			couvrent les parties suivantes : - les dépendances nécessaires pour la réutilisation des scripts (langages, dépendances externes, etc) - les commandes pour l'exécution des scripts. - Le registre des traitements de données personnelles intègre l'ensemble des traitements de données personnelles impliqués dans la base de données. - Les procédures de tri des données personnelles pour la mise en conformité de la base de données avec le RGPD sont rédigées. - Les procédures de tri détaillent les traitements de conformité (automatisés ou non) à appliquer ainsi que leur fréquence d'exécution.
- Rédaction des spécifications techniques des moyens de mise à disposition et d'accès aux données du projet : API (REST) et accès direct à la base de données. - Configuration des accès au jeu de données depuis le serveur de l'API. - Développement de la réception et de la validation des requêtes client (web, mobile…)	**C5. Développer une API mettant à disposition le jeu de données** en utilisant l'architecture REST afin de permettre l'exploitation du jeu de données par les autres composants du projet.		- La documentation technique de l'API (REST) couvre tous les points de terminaisons. - La documentation technique couvre les règles d'authentification et/ou d'autorisation de l'API. - La documentation technique respecte les standards du modèle choisi (par exemple *OpenAPI**).

- Développement des requêtes à la base de données en fonction de la mise à disposition des données prévues par les spécifications. - Développement des réponses de l'API au client. - Développement des règles d'autorisation et d'accès aux points de terminaison de l'API REST. - Sécurisation de l'API : Top 10 OWASP API[1] par exemple. - Rédaction de la documentation technique de l'API REST. - Rédaction de la documentation technique d'accès à la base de données adaptée aux environnements techniques du projet.			- L'API REST est fonctionnelle pour l'accès aux données du projet : elle restreint par une autorisation (ou authentification) l'accès aux données, - L'API REST est fonctionnelle pour la mise à disposition : elle permet la récupération de l'ensemble des données nécessaires au projet, comme prévu selon les spécifications données.
Bloc de compétences 2 : Intégrer des modèles et des services d'intelligence artificielle			
A3. Accompagner le choix et l'intégration d'un service d'intelligence artificielle préexistant	**C6. Organiser et réaliser une veille technique et réglementaire** en animant le travail collectif de sélection des sources, de collecte, de traitement et de partage des informations afin	**E2. Cas pratique (C6, C7, C8)** L'évaluation doit se faire à partir de l'expression d'un besoin réel ou fictif de	- La thématique de veille choisie porte sur un outil et/ou une réglementation mobilisée dans la mise en situation.

[1] https://raw.githubusercontent.com/OWASP/API-Security/master/2019/en/dist/owasp-api-security-top-10.pdf

- Définition de la ou des thématiques de veille. - Planification des temps dédiés à la veille. - Choix d'un outil d'agrégation des flux d'informations et d'actualités. - Choix d'un outil de partage ou communication des synthèses des informations collectées. - Identification des sources et des flux d'informations utiles à la veille thématique visée (dont accessibilité, sécurité, gestion des données) - Qualification de la fiabilité des sources et des flux identifiés. - Configuration des outils d'agrégation selon les flux et sources et la ou les thématiques de veille. - Rédaction des synthèses des informations collectées. - Communications des synthèses aux parties prenantes du projet et aux éventuels intéressés (à intervalles	de formuler des recommandations pour le projet toujours en phase avec l'état de l'art.	fonctionnalités d'intelligence artificielle. Ce besoin peut résulter d'une commande client comme d'une sollicitation interne d'un collaborateur data scientist par exemple. Le cas pratique évalué a pour but l'installation et la configuration du service d'intelligence artificielle préconisé. <u>Livrable</u> : rapport professionnel individuel <u>Évaluation</u> : - Correction du rapport professionnel - Soutenance orale individuelle	- Les temps de veille sont planifiés régulièrement (à minima une récurrence d'une heure hebdomadaire). - Le choix des outils d'agrégation est cohérent avec les sources d'informations visées et le budget disponible (flux RSS, flux réseaux sociaux, agrégation newsletter, etc) - Les synthèses sont communiqués aux parties prenantes dans un format qui respecte les recommandations d'accessibilité (par exemples celles de l'association Valentin Haüy[2]ou de Atalan - AcceDe[3]). - Les informations partagées dans la synthèse répondent à la thématique de veille choisie. - Les sources et flux identifiés répondent aux critères de fiabilité : - L'auteur de la page est identifié - Des informations sur l'auteur sont disponibles et confirment ses compétences, sa notoriété et l'absence d'intérêts personnels - l'analyse du contenu est valable (date de publication récente, sources de l'information

[2] https://www.avh.asso.fr/fr/favoriser-laccessibilite/accessibilite-numerique/accessibilite-des-documents-et-des-courriels-0
[3] https://www.accede-web.com/wp-content/uploads/accessibility-guidelines-editors-template.pdf

réguliers ou à l'opportunité par exemple).			indiquées, niveau de langue correct), - la source (site) ou le document est structuré - les sources (sites) ou documents respectant les normes d'accessibilités sont privilégiés. - l'information peut être confirmée par d'autres sites de confiance
- Définition de la problématique technique et fonctionnelle d'intelligence artificielle à adresser à partir de l'expression de besoin. - Identification des contraintes de moyens, techniques et opérationnelles liées au contexte du projet. - *Benchmark* des outils et services d'intelligence artificielle accessibles et répondant au problème visé. - Rédaction des conclusions préconisant un ou plusieurs services d'intelligence artificielle.	**C7**. **Identifier des services d'intelligence artificielle préexistants** à partir de l'expression de besoin en fonctionnalités d'intelligence artificielle, en réalisant un *benchmark* de services existants et en analysant leurs caractéristiques pour formaliser une ou plusieurs recommandations de services adaptés au besoin.		- L'expression de besoin est reformulée et présente les objectifs et les contraintes du projet d'intégration d'une solution d'intelligence artificielle. - Le *benchmark* liste les services étudiés et les services non étudiés. - Les raisons pour écarter un service sont explicitées. - Le *benchmark* détaille le niveau d'adéquation du service étudié pour chaque ensemble fonctionnel souhaité par le commanditaire. - Le *benchmark* détaille le niveau de la démarche éco-responsable du service étudié, en fonction des informations disponibles. - Le *benchmark* détaille les principales contraintes techniques et les pré-requis pour chaque solution.

			- Les conclusions délimitent clairement les services répondant aux besoins, avec leurs avantages et leurs inconvénients, des services ne couvrant pas les besoins du commanditaire.
- Création de l'environnement d'exécution du service (compte SaaS*, VPS*, sur-site, etc). - Installation et configuration des éventuelles dépendances (sdk, outils, autre service SaaS, etc). - Création des accès à l'environnement d'exécution et de configuration du service (comptes, groupes d'utilisateurs, droits, etc). - Installation et configuration des outils de monitorage disponibles avec le service intégré. - Rédaction de la documentation.	**C8. Paramétrer un service d'intelligence artificielle** en suivant sa documentation technique et en respectant les spécifications du projet, afin de permettre l'intégration des connecteurs du service dans le système d'information.		- Le service installé est accessible, avec une éventuelle authentification. - Le service est configuré correctement, il répond aux besoins fonctionnels et aux contraintes techniques du projet. - Le monitorage disponible du service est opérationnel. - La documentation couvre la gestion des accès à la solution, les procédures d'installation et de test, les éventuelles dépendances et interconnexions avec d'autres solutions, les données impliquées dans l'utilisation de la solution. - La documentation est communiquée aux parties prenantes dans un format qui respecte les recommandations d'accessibilité (par exemples celles de l'association Valentin Haüy[4]ou de Atalan -

[4] https://www.avh.asso.fr/fr/favoriser-laccessibilite/accessibilite-numerique/accessibilite-des-documents-et-des-courriels-0

			AcceDe[5]).
A4. Réaliser l'intégration d'un modèle ou d'un service d'intelligence artificielle - Analyse des spécifications fonctionnelles et techniques fournies par le commanditaire interne ou externe. - Conception de l'architecture de l'API : les points de terminaison ; les règles d'accès, etc. - Choix des outils et langages de programmation pour le développement de l'API. - Installation et configuration de l'environnement de développement. - Développement de la vérification et de l'éventuelle transformation des paramètres envoyés par le client (web, mobile…) au format attendu par le modèle. - Développement de l'exécution du modèle à partir de la requête client. - Développement de la réponse de l'API au client avec le résultat de l'exécution du modèle.	**C9. Développer une API exposant un modèle d'intelligence artificielle** en utilisant l'architecture REST pour permettre l'interaction entre le modèle et les autres composants du projet.	**E3. Mise en situation (C9, C10, C11, C12, C13)** L'évaluation doit se faire dans un contexte réel ou fictif de réalisation d'un service d'intelligence artificielle à partir d'un modèle fourni. Le projet évalué a pour but la mise en service (*packaging**, monitorage, test…) du modèle fourni, et son intégration dans une application existante. Livrable : rapport professionnel individuel Évaluation : - Correction du rapport professionnel - Soutenance orale individuelle intégrant une démonstration du projet.	- L'API restreint l'accès au modèle d'intelligence artificielle avec un moyen d'authentification. - L'API permet l'accès aux fonctions du modèle, comme attendu selon les spécifications. - Les recommandations de sécurisation d'une API du top 10 OWASP sont intégrées quand nécessaires. - Les sources sont versionnées et accessibles depuis un dépôt Git distant. - Les tests couvrent tous les points de terminaison dans le respect des spécifications. - Les tests s'exécutent sans bug. - Les résultats des tests sont correctement interprétés. - La documentation couvre l'architecture et tous les points de terminaisons de l'API. - La documentation couvre les règles d'authentification et/ou d'autorisation d'accès à l'API.

[5] https://www.accede-web.com/wp-content/uploads/accessibility-guidelines-editors-template.pdf

- Développement des règles d'autorisation et d'accès aux points de terminaison de l'API. - Sécurisation de l'API : Top 10 OWASP API[6] par exemple. - Développement des tests d'intégration validant le bon fonctionnement des points de terminaison de l'API développée. - Versionnement des sources avec Git, et depuis un dépôt distant. - Rédaction et/ou génération de la documentation de l'API.			- La documentation et l'API respectent les standards d'un modèle choisi (par exemple *Open API**). - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).
- Installation de l'environnement de développement de l'application. - Programmation des éventuelles étapes d'authentification ou d'autorisation avec l'API. - Programmation de la communication avec les points de terminaison de l'API du modèle ou du service d'intelligence artificielle.	**C10. Intégrer l'API d'un modèle ou d'un service d'intelligence artificielle** dans une application, en respectant les spécifications du projet et les normes d'accessibilité en vigueur, à l'aide de la documentation technique de l'API, afin de créer les fonctionnalités d'intelligence artificielle de l'application.		- L'application de départ est installée et fonctionnelle en environnement de développement. - La communication avec l'API depuis l'application fonctionne. - Les éventuelles étapes d'authentification et de renouvellement de l'authentification (expiration des jetons par exemple) sont intégrées correctement en suivant la documentation de l'API.

[6] https://raw.githubusercontent.com/OWASP/API-Security/master/2019/en/dist/owasp-api-security-top-10.pdf

- Intégration des éventuelles adaptations d'interface du à l'intégration de l'intelligence artificielle. - Test et validation, en fonction des enjeux, du niveau d'accessibilité des interfaces modifiées. - Développement des tests d'intégration sur le périmètre de l'API exploité. - Versionnement des sources avec Git et sur le dépôt de l'application.			- Tous les points de terminaison de l'API concernés par le projet sont intégrés à l'application selon les spécifications fonctionnelles et techniques. - Les adaptations d'interfaces nécessaires et en accord avec les spécifications sont intégrées à l'application. - Les tests d'intégration couvrent tous les points de terminaison exploités. - Les tests s'exécutent en totalité : il n'y a pas de bug dans les programmes des tests en eux-mêmes. - Les résultats des tests sont correctement interprétés. - Les sources sont versionnées et accessibles depuis le dépôt Git de l'application.
<u>A5. Faciliter le déploiement d'un modèle d'intelligence artificielle avec une approche MLOps</u> - Liste des métriques du modèle à monitorer et des éventuels déclencheurs pour le réentraînement, à l'aide des équipes de data analyse et de data science : stabilité des données,	**C11. Monitorer un modèle d'intelligence artificielle** à partir des métriques courantes et spécifiques au projet, en intégrant les outils de collecte, d'alerte et de restitution des données du monitorage pour permettre l'amélioration du modèle de façon itérative.		- Les métriques faisant l'objet du monitorage du modèle sont expliquées sans erreur d'interprétation. - Le ou les outils pour l'intégration du monitorage du modèle sont adaptés au contexte et aux contraintes techniques du projet.

performance du modèle d'intelligence artificielle, bonne santé du système, etc. - Choix d'une solution ou d'un outil pour le monitorage du modèle d'intelligence artificielle et pour la consolidation des indicateurs. - Intégration de la solution ou de l'outil de monitorage du modèle : - intégration des collecteurs. - intégration des éventuels déclencheurs (pour l'entraînement continu). - Sélection d'un outil de restitution des métriques de surveillance du modèle (Grafana, Dash, Kibana…). - Intégration de l'outil de restitution et intégration des éventuelles alertes (notifications, e-mails, etc). - Test et validation du bon fonctionnement de toute la chaîne de monitorage. - Versionnement des sources avec Git et sur un dépôt distant.			- Au moins un vecteur de restitution des métriques évaluées, en temps réel, est proposé (*dashboard*, feuille de calcul, etc). - Les enjeux d'accessibilité, pour toutes les parties prenantes du projet, sont pris en compte lors de la sélection de l'outil de restitution. - La chaîne de monitorage est d'abord testée dans un bac à sable ou environnement de test dédié. - La chaîne de monitorage est en état de marche. Les métriques visées sont effectivement évaluées et restituées. - Les sources sont versionnées et accessibles depuis un dépôt Git distant. - La documentation technique de la chaîne de monitorage couvre la procédure d'installation de la chaîne, de configurations, et d'utilisation du monitorage à destination des équipes techniques. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin

- Rédaction de la documentation technique des procédures d'installation, de maintenance du monitorage du modèle. - Rédaction de la documentation utilisateur, pour l'utilisation du monitorage.			Haüy ou de Microsoft).
- Définition, avec les équipes responsables du développement du modèle, du périmètre des tests pour chaque composante du modèle : format des données, complétude et labellisation des données, phase d'entraînement, phase de validation du modèle, etc. - Choix des outils de test (unittest par exemple.) - Installation et configuration de l'environnement d'exécution des tests. - Intégration des tests : assertions, mocks*, fixtures*…	**C12. Programmer les tests automatisés d'un modèle d'intelligence artificielle** en définissant les règles de validation des jeux de données, des étapes de préparation des données, d'entraînement, d'évaluation et de validation du modèle pour permettre son intégration en continu et garantir un niveau de qualité élevé.		- L'ensemble des cas à tester sont listés et définis : la partie du modèle visée par le test, le périmètre du test et la stratégie de test. - Les outils de test (framework, bibliothèque, etc.) choisis sont cohérent avec l'environnement technique du projet. - Les tests sont intégrés et respectent la couverture souhaitée établie. - Les tests s'exécutent sans problème technique en environnement de test. - Les sources sont versionnées et accessibles depuis un dépôt Git distant (DVC, Gitlab…). - La documentation couvre la procédure d'installation de l'environnement de test, les dépendances installées, la procédure d'exécution des tests et de calcul de la couverture.

- Versionnement des sources et si possible des données, avec sur un dépôt Git distant (DVC, Gitlab…). - Rédaction de documentation technique pour la configuration de l'environnement et pour l'exécution des tests.			- La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).
- Définition des étapes, tâches et déclencheurs de la chaîne de livraison continue. Par exemple : réentrainement, exécution des tests, livraison en environnement de pré-production. - Paramétrage d'une chaîne de livraison continue : variables d'environnement, versions des environnements et dépendances, déclencheurs, etc. - Intégration de l'exécution des tests du modèle et des données. - Intégration des étapes d'entraînement et d'évaluation du modèle d'intelligence artificielle. - Intégration de la génération des éventuels rapports des résultats obtenus	**C13. Créer une chaîne de livraison continue d'un modèle d'intelligence artificielle** en installant les outils et en appliquant les configuration souhaitées, dans le respect du cadre imposé par le projet et dans une approche MLOps*, pour automatiser les étapes de validation, de test, de *packaging** et de déploiement du modèle.		- La documentation pour l'utilisation de la chaîne couvre toutes les étapes, les tâches et tous les déclencheurs disponibles. - Les déclencheurs sont intégrés comme préalablement définis. - Le ou les fichiers de configuration de la chaîne sont correctement reconnus et exécutés par le système selon les déclencheurs configurés. - L'étape de test des données est intégrée à la chaîne et s'exécute sans erreur. - La ou les étapes de test, d'entraînement et de validation du modèle sont intégrées à la chaîne et s'exécutent sans erreur. - Les sources de la chaîne sont versionnées et accessibles depuis le dépôt Git distant du projet.

lors de l'évaluation du modèle (*accuracy**, *confusion matrix**…). - intégration de l'étape de livraison (sous forme de *pull request** par exemple), avec le ou les rapports d'évaluation du modèle attachés à la livraison. - Les sources de la chaîne (fichiers de configuration par exemple) sont versionnées et accessibles depuis le dépôt Git du projet (DVC, Gitlab…). - Documentation des procédures d'installation et de test de la chaîne. - Documentation utilisateur, pour le déclenchement de la chaîne.			- La documentation de la chaîne de livraison continue couvre la procédure d'installation, de configuration et de test de la chaîne. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).
Bloc de compétences 3 : Réaliser une application intégrant un service d'intelligence artificielle			
A6. Concevoir une application intégrant un service d'intelligence artificielle - Modélisation des données de l'application (par exemple, entités-relations, MCD/MPD, etc). - Modélisation des parcours utilisateurs (schéma fonctionnel, story board, etc).	**C14. Analyser le besoin d'application d'un commanditaire intégrant un service d'intelligence artificielle,** en rédigeant les spécifications fonctionnelles et en le modélisant, dans le respect des standards d'utilisabilité et d'accessibilité, afin d'établir avec précision les objectifs de développement correspondant au besoin et à la faisabilité technique.	**E4. Mise en situation (C14, C15, C16, C17, C18, C19)** L'évaluation doit se faire dans un contexte réel ou fictif de développement d'une application intégrant un service d'intelligence artificielle. Le projet évalué a pour but d'analyser un besoin en développement d'application	- La modélisation des données respecte un formalisme : Merise, entités-relations, etc. - La modélisation des parcours utilisateurs respecte un formalisme : schéma fonctionnel, wireframes, etc. - Chaque spécification fonctionnelle couvre le contexte, les scénarios d'utilisation et les critères de validation.

<table>
<tr>
<td>- Rédaction des spécifications fonctionnelles sous forme de user story.

- Définition des objectifs techniques d’accessibilité des interfaces.</td>
<td></td>
<td rowspan="2">d’intelligence artificielle, de concevoir, développer, tester et enfin livrer l’application.

<u>Livrable</u> : rapport professionnel individuel

<u>Évaluation</u> :
- Correction du rapport professionnel
- Soutenance orale individuelle intégrant une démonstration du projet.

E5. Cas pratique (C20, C21)

L’évaluation doit se faire à partir d’une application existante présentant au moins une erreur technique, en contexte réel ou fictif.

Le cas pratique évalué a pour but la mise en place du monitorage applicatif et de la résolution d’un incident technique dans l’application.

<u>Livrable</u> : documentation technique du monitorage. Documentation de la résolution de l’incident technique.

<u>Évaluation</u> :
- Correction de la documentation
- Soutenance orale individuelle présentant le monitorage de l’application et la solution implémentée en réponse à l’incident technique traité.</td>
<td>- Les objectifs d’accessibilités sont directement intégrés aux critères d’acceptation des user stories.

- Les objectifs d’accessibilité sont formulés en s’appuyant sur un des standards d'accessibilité : WCAG, RG2AA, etc.</td>
</tr>
<tr>
<td>- Conception de l’architecture de l’application (n-tiers*, serverless*, micro-service*, model-vue-controller*, …).

- Choix des langages de programmation et des outils pour l’environnement de développement (Python, Java, Node.js, React, Dash, Github...).

- Identification des flux de données et des zones de stockage : base de données des données de l’application, fichiers de données issus de la journalisation, etc.

- Choix des éventuels services externes complémentaires (par exemple : BaaS*, paiements, cartographie, etc.)

- Rédaction des spécifications techniques.</td>
<td>C15. Concevoir le cadre technique d’une application intégrant un service d’intelligence artificielle, à partir de l'analyse du besoin, en spécifiant l’architecture technique et applicative et en préconisant les outils et méthodes de développement, pour permettre le développement du projet.</td>
<td>- Les spécifications techniques rédigées couvrent l’architecture de l’application, ses dépendances et son environnement d’exécution (langage de programmation, framework, outils, etc).

- Les éventuels services (PaaS, SaaS, etc) et prestataires ayant une démarche éco-responsable sont favorisés lors des choix techniques.

- Les flux de données impliqués dans l’application sont représentés par un diagramme de flux de données.

- La preuve de concept est accessible et fonctionnelle en environnement de pré-production.

- La conclusion à l’issue de la preuve de concept donne un avis précis permettant une prise de décision sur la poursuite du projet.</td>
</tr>
</table>

A7. Développer les interfaces et les fonctionnalités d'une application d'intelligence artificielle - Mise en place des outils et des supports de pilotage de la production technique, selon la méthode agile retenue (SCRUM, Shape up, etc.) - Facilitation des rituels agiles et du respect du cadre des rituels. - Communication de l'avancement des productions techniques. - Communication des imprévus et des changements.	**C16. Coordonner la réalisation technique d'une application d'intelligence artificielle** en s'intégrant dans une conduite agile de projet et un contexte MLOps et en facilitant les temps de collaboration dans le but d'atteindre les objectifs de production et de qualité.		- Les cycles, les étapes de chaque cycle, les rôles, les rituels et les outils de la méthode agile appliquée sont respectés dans sa mise en place et tout au long du projet. - Les outils de pilotage (tableau kanban, burndown chart, backlog, etc.) sont disponibles dans les conditions prévues par la méthode appliquée. - Les objectifs et les modalités des rituels sont partagés à toutes les parties prenantes et rappeler si besoin. - Les éléments de pilotage sont rendus accessibles à toutes les parties du projet et ce tout au long du projet, en accord avec les recommandations de la méthode de gestion de projet appliquée.
- Installation de l'environnement de développement du projet. - Intégration de la mise en page visée selon les maquettes graphiques. - Intégration des contenus des interfaces (textes, images, mise en page, …). - Développement des fonctionnalités *front-end* (par exemple des animations, validations, interactions avec des APIs et services tiers.	**C17. Développer les composants techniques et les interfaces d'une application** en utilisant les outils et langages de programmation adaptés et en respectant les spécifications fonctionnelles et techniques, les standards et normes d'accessibilité, de sécurité et de gestion des données en vigueur dans le but de répondre aux besoins fonctionnels identifiés.		- L'environnement de développement installé respecte les spécifications techniques du projet. - Les interfaces sont intégrées et respectent les maquettes. - Les comportements des composants d'interface (validation formulaire, animations, etc.) et la navigation respectent les spécifications fonctionnelles. - Les composants métier sont développés et fonctionnent comme prévu par les spécifications techniques et fonctionnelles.

- Développement des composants métiers (calculs spécifiques, fonctionnalités, email, …). - Gestion des droits et des accès à l'application (authentification, permissions, groupes, …). - Intégration des composants d'accès aux données. - Sécurisation de l'application (top 10 OWASP). - Intégration de tests automatisés (par exemple unitaires, fonctionnels, …). - Versionnement des sources avec Git et sur un dépôt distant. - Les enjeux d'accessibilité sont pris en compte lors du développement de l'application, et notamment des interfaces. - Rédaction de la documentation technique.			- La gestion des droits d'accès à l'application ou à certains espaces de l'application est développée et respecte les spécifications fonctionnelles. - Les flux de données sont intégrés dans le respect des spécifications techniques et fonctionnelles. - Les développements sont réalisés dans le respect des bonnes pratiques d'éco-conception d'une application (Les recommandations d'éco-index ou Green IT par exemple) - Les préconisations du top 10 d'OWASP sont implémentées dans l'application quand nécessaire. - Des tests d'intégration ou unitaires couvrent au moins les composants métier et la gestion des accès. - Les sources sont versionnées et accessibles depuis un dépôt Git distant. - La documentation technique couvre l'installation de l'environnement de développement, l'architecture applicative, les dépendances, l'exécution des tests. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).

<u>A8. Développer les fonctions de tests et de contrôle d'une application d'intelligence artificielle</u> - Choix de l'outil d'intégration continue. - Définition des étapes, tâches et déclencheurs de la chaîne d'intégration continue. - Production des fichiers de configuration de la chaîne. - Configuration de l'automatisation : déclencheurs, variables d'environnement, dépendances, commandes à exécuter, etc. - Test et validation du bon fonctionnement de la chaîne. - Versionnement des sources pour la configuration de la chaîne. - Documentation des procédures d'installation et de test de la chaîne. - Documentation utilisateur, pour le déclenchement de la chaîne.	**C18. Automatiser les phases de tests du code source** lors du versionnement des sources à l'aide d'un outil d'intégration continue* de manière à garantir la qualité technique des réalisations.		- La documentation pour l'utilisation de la chaîne couvre les outils, toutes les étapes, les tâches et tous les déclencheurs de la chaîne. - Un outil de configuration et d'exécution d'une chaîne d'intégration continue est sélectionné de façon cohérente avec l'environnement technique du projet. - La chaîne intègre toutes les étapes nécessaires et préalables à l'exécution des tests de l'application (build, configurations…). - La chaîne exécute les tests de l'application disponibles lors de son déclenchement. - Les configuration sont versionnées avec les sources du projet d'application, sur un dépôt Git distant. - La documentation de la chaîne d'intégration continue couvre la procédure d'installation, de configuration et de test de la chaîne. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).

- Définition des étapes, tâches et déclencheurs de la chaîne de livraison continue. - Paramétrage d'une chaîne de livraison continue : variables d'environnement, versions des environnements et dépendances, déclencheurs, etc. - Configuration des étapes automatisées (*build*, *pull request*, etc.) de la chaîne de livraison continue à partir de la chaîne d'intégration continue. - Versionnement des configurations de la chaîne de livraison continue depuis le Git du projet. - Test la chaîne de livraison continue. - Documentation de la procédure de livraison continue : outils, configurations, exécution, débug.	**C19. Créer un processus de livraison continue** d'une application en s'appuyant sur une chaîne d'intégration continue et en paramétrant les outils d'automatisation et les environnements de test afin de permettre une restitution optimale de l'application.		- La documentation pour l'utilisation de la chaîne couvre toutes les étapes de la chaîne, les tâches et tous les déclencheurs disponibles. - Le ou les fichiers de configuration de la chaîne sont correctement reconnus et exécutés par le système. - La ou les étapes de *packaging* (compilation, minification, *build* de containers, etc.) de l'application sont intégrées à la chaîne et s'exécutent sans erreur. - L'étape de livraison (*pull request* par exemple) est intégrée et exécutée une fois la ou les étapes de packaging validées. - Les sources de la chaîne sont versionnées et accessibles depuis le dépôt Git distant du projet d'application. - La documentation de la chaîne de livraison continue couvre la procédure d'installation, de configuration et de test de la chaîne. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin

			Haüy ou de Microsoft).
A9. Assurer le maintien en condition opérationnelle d'une application d'intelligence artificielle - Définition des métriques pour le monitorage de l'application. - Définition des seuils ou des valeurs devant générer une alerte. - Choix d'une solution ou d'un outil pour la consolidation et le suivi des indicateurs de monitorage. - Configuration de l'outil ou de la solution de monitorage. - Intégration de la journalisation nécessaire aux objectifs de monitorage, dans l'application. - Intégration d'alertes (e-mail, push…) en fonction des indicateurs. - Documentation du monitorage et des procédures d'installation et de configuration de l'outillage utilisé dans	**C20. Surveiller une application d'intelligence artificielle,** en mobilisant des techniques de monitorage et de journalisation, dans le respect des normes de gestion des données personnelles en vigueur, afin d'alimenter la *feedback loop** dans une approche MLOps, et de permettre la détection automatique d'incidents.		- La documentation liste les métriques et les seuils et valeurs d'alerte pour chaque métrique à risque. - La documentation explicite les arguments en faveur des choix techniques pour l'outillage du monitorage de l'application. - Les outils (collecteurs, journalisation, agrégateurs, filtres, dashboard, etc.) sont installés et opérationnels à minima en environnement local. - Les règles de journalisation sont intégrées aux sources de l'application, en fonction des métriques à surveiller. - Les alertes sont configurées et en état de marche, en fonction des seuils préalablement définis. - La documentation couvre la procédure d'installation et de configuration des dépendances pour l'outillage du monitorage de l'application. - La documentation est communiquée dans un format qui respecte les recommandations d'accessibilité (par exemple celles de l'association Valentin Haüy ou de Microsoft).

l'environnement de l'application.			
- Analyse d'un message d'erreur, par exemple en console ou depuis un fichier de journalisation. - Recherche d'une solution à l'aide de ressources externes : documentation, plateforme en ligne, etc. - Test et validation d'une solution. - Versionnement de la solution depuis le dépôt Git du projet d'application. - Documentation de l'erreur et de la solution implémentée.	**C21. Résoudre les incidents techniques** en apportant les modifications nécessaires au code de l'application et en documentant les solutions pour en garantir le fonctionnement opérationnel.		- La ou les causes du problème sont identifiées correctement. - Le problème est reproduit en environnement de développement. - La procédure de débogage du code est documentée depuis l'outil de de suivi. - La solution documentée explicite chaque étape de la résolution et de son implémentation. - La solution est versionnée dans le dépôt Git du projet d'application (par exemple avec une *merge request*).

Glossaire :

- **Scraping** : c'est une technique d'extraction du contenu de sites Web, via un script ou un programme, dans le but de le transformer pour permettre son utilisation dans un autre contexte comme l'enrichissement de bases de données, par exemple.
- **Big data** : des jeux de données plus variées, arrivant dans des volumes croissants et avec des flux dont la vitesse est élevée. C'est ce que l'on appelle les trois « V » : variété, volume, vitesse.
- **Script** : un programme dédié à une unique tâche, souvent dans le cadre d'automatisation.
- **REST** (API) : c'est une architecture d'API utilisée pour la création de services web (HTTP).
- **SQL** : c'est un langage informatique servant à exploiter des bases de données relationnelles. Il est également à la base de multiples adaptations, formant une famille de dérivées propre à des systèmes de stockage dont les besoins en requêtage sortent du cadre initialement prévu par SQL.

- **Dépôt Git** : C'est un entrepôt virtuel d'un projet en développement informatique. C'est l'endroit où sont stockées les versions d'un logiciel, de ses sources et dépendances (images, données, …)
- **Merise** : La méthode Merise est une méthode d'analyse, de conception et de réalisation de systèmes d'informations. Aujourd'hui principalement utilisée dans sa déclinaison pour la conception de structure de données relationnelles.
- **SaaS** : Le SaaS, ou Software (Logiciel) en tant que Service, est un modèle de distribution de logiciel uniquement accessible via un navigateur web, en ligne. Les SaaS sont hébergées par l'éditeur, et les accès au logiciel sont souvent conditionnés par un système d'abonnement et de création de compte utilisateur.
- **VPS** : pour virtual private server ou serveur virtuel privé.
- **Packaging** (d'un modèle d'intelligence artificielle) : C'est l'action de transformation d'un modèle en un format d'exécution générique et auto-suffisant (embarquant les dépendances et données nécessaires à l'exécution du modèle). Cette opération est menée à l'aide d'outils dédiés comme ONNX ou encore Docker.
- **Mocks, fixtures** : Ce sont essentiellement des initialiseurs personnalisés pour simplifier la construction des dépendances nécessaires aux tests.
- **MLOps** : Le MLOps est une pratique qui établit des règles de collaboration entre les concepteurs/développeurs et les opérateurs des infrastructures informatiques.
- **N-tiers** : En génie logiciel, l'architecture à plusieurs niveaux (souvent appelée architecture n-tiers) ou architecture multicouche est une architecture client-serveur dans laquelle les fonctions de présentation, de traitement des applications et de gestion des données sont physiquement séparées.
- **Serverless** : Il s'agit d'un modèle d'exécution de l'informatique dans le Cloud dans lequel le fournisseur de services Cloud fait fonctionner le serveur et gère dynamiquement l'allocation des ressources de la machine.
- **Model-vue-contrôleur** : Le modèle-vue-contrôleur (généralement connu sous le nom de MVC) est un modèle de conception de logiciel couramment utilisé qui divise la logique de programme correspondante en trois éléments interconnectés.
- **Micro-services** : L'architecture de micro-services organise une application comme une collection de services indépendants.
- **Back-end** : Il s'agit de la couche serveur d'une application.
- **Front-end** : Il s'agit des couches "visibles" d'une application, dont les interfaces.
- **Déployer/déploiement** : action ou processus de mise en ligne / en production d'un programme, d'une application.
- **OpenAPI** : OpenAPI est une norme de description des interfaces de programmation. OpenAPI encadre le développement et la documentation des API conformes à l'architecture REST.
- **BaaS** : il s'agit d'un modèle de service cloud dans lequel les développeurs externalisent tous les aspects génériques d'un back-end d'une application web ou mobile (authentification, comptes utilisateurs, notifications…) afin de concentrer l'effort sur la valeur ajoutée du produit.

- **Feedback loop** : désigne le processus par lequel les résultats prédits d'un modèle d'intelligence artificielle sont réutilisés pour former de nouvelles versions du modèle.