

Étude de cas : l'IA dans le développement logiciel

Table des matières

I. Développement logiciel.....	2
1. Préparation et analyse	2
2. Définir les exigences	2
3. Design.....	2
4. Développement	2
5. Expérimentation	2
6. Déploiement.....	3
7. Maintenance.....	3
II. Différents types d'IA.....	3
1. Introduction	3
2. 3 Principaux types d'IA	3
a. Les sous catégories des IA basées sur leurs fonctionnalités	4
b. Les sous catégories des IA génératives	4
c. L'IA générative en détails.....	5
III. L'impact de l'IA sur les diverses phases du développement logiciel	5
1. Préparation et analyse	5
2. Définir les exigences	5
3. Design.....	6
4. Développement	6
5. Expérimentation.....	6
6. Déploiement.....	6
7. Maintenance.....	6
IV. Développement logiciel IA.....	7
1. Préparation et analyse IA	7
2. Définir les exigences IA	14
3. Design IA.....	20
4. Développement IA	23
5. Expérimentation IA	25
6. Déploiement IA	28
7. Maintenance IA	33
V. Conclusion.....	34
VI. Bibliographie	35

Étude de cas : l'IA dans le développement logiciel

I. Développement logiciel

1. Préparation et analyse

Analyse de documents, interviews avec les parties prenantes, observation de logiciels existants, ateliers avec les utilisateurs pour prioriser les besoins de ceux-ci, prototypes pour montrer aux parties prenantes et aux utilisateurs.

Etablir les besoins des clients et utilisateurs. Evaluer la faisabilité du produit, les potentiels revenus, les coûts de productions...

Pour décider quoi faire il est possible d'utiliser des cadres de priorisation des fonctionnalités qui prennent en compte la Valeur du logiciel, le cout, le temps et d'autres facteurs

Une fois qu'il est décidé que le projet est en accord avec le business et les buts des parties prenantes, que le projet est faisable, adresse les besoins des clients. On peut passer à la prochaine phase

2. Définir les exigences

Phase critique pour convertir l'information rassemblée pendant la phase précédente en besoins clairs pour l'équipe de développeurs. Ce processus guide le développement de plusieurs documents : une spécification des exigences logicielles (SRS), un document sur les cas d'utilisation et un document sur la matrice de traçabilité des exigences.

Les ingénieurs logiciels peuvent préparer le terrain en dressant des listes de tâches, assigner les responsabilités aux équipes, anticiper les problèmes, et des estimations de coûts.

3. Design

La phase de design comporte le plan et la vision généralement sur un document de conception du logiciel qui contient le design, le langage de programmation, la plateforme à utiliser, et les applications de sécurité. Cette phase peut aussi avoir un prototype afin que le code puisse être visualisé.

Pendant le design, il faut prendre en compte les exigences fonctionnelles (tâches que le logiciel doit accomplir), les exigences non fonctionnelles (efficacité avec laquelle le logiciel doit fonctionner et le niveau de sécurité et enfin les restrictions techniques et budgétaires.

Il faut se focaliser sur la modularité et la testabilité du logiciel ainsi que son extensibilité pour le futur. Pendant cette phase, il peut aussi avoir les descriptions sur l'interface utilisateur.

4. Développement

Il est enfin temps d'écrire le code : transformer le design en un logiciel qui fonctionne. C'est en général la phase la plus longue. Cette phase transforme le design en application fonctionnelle. Pendant ce processus, il est important d'avoir une documentation continue, une collaboration et gestion de version. Il est important d'avoir un développement modulaire, et de toujours se servir du feedback et de faire des révisions. Cette phase est non seulement l'étape où le logiciel se prend forme, mais aussi celle qui requiert une attention méticuleuse pour assurer sa robustesse et sa flexibilité.

5. Expérimentation

Faire le test est une partie cruciale, il existe plusieurs types de test dans ce processus :

- Test de l'ensemble du système logiciel pour vérifier s'il répond à toutes les exigences.
- Tests unitaires : chaque petite partie du logiciel est testée

Étude de cas : l'IA dans le développement logiciel

- Test d'intégration : permettent de s'assurer que les différentes parties du logiciel peuvent fonctionner ensemble
- Tests fonctionnels : vérifier que le logiciel fait ce qu'il est censé faire, conformément aux exigences
- Tests de performance : tester la façon dont le logiciel gère différentes charges de travail, s'assurant qu'il fonctionne bien
- Tests d'acceptation : test final pour voir si le logiciel fait ce que les utilisateurs veulent et fonctionne correctement

6. Déploiement

Le déploiement est la phase où le logiciel est mis à disposition pour les utilisateurs, il existe plusieurs types de façons pour déployer le logiciel :

- Déploiement bleu vert avec une ancienne et une nouvelle version au cas où la nouvelle ne fonctionne pas
- Déploiement Canary : ce procédé envoie le trafic petit à petit et envoyer le reste une fois que l'on a confirmé le bon fonctionnement
- Déploiement shadow : une manière de tester des nouveautés sans que ce soit visible aux utilisateurs : c'est déployé dans un autre environnement pour éviter de perturber le service.

7. Maintenance

Une fois déployé, il faut toujours maintenir le logiciel avec des corrections de bug que les utilisateurs aient pu signaler, mises à jour et améliorations pour des fonctionnalités que les utilisateurs pourraient demander, suivi des performances, faire attention à la sécurité et apporter du soutien aux utilisateurs si besoin.

II. Différents types d'IA

1. Introduction

Les IA ont été utilisées depuis les débuts des modèles d'algorithmes d'apprentissage mais ont été maintenues par des analystes et avaient donc besoin d'interventions humaines pour toute tâche en dehors de leur entraînement initial.

Au début des années 2010, les réseaux de neurones artificiels sont devenus très performants et ont permis aux machines d'apprendre de façon similaire aux humains.

Ainsi, les modèles d'apprentissage en profondeur ont permis aux IA de pouvoir prendre des décisions sans interventions humaines. Ceci a permis aux IAs de pouvoir accomplir des tâches en automatisation, génération de contenu, maintenance en prévision et d'autres capacités pour les différentes industries.

2. 3 Principaux types d'IA

L'IA faible ou étroite qui accomplit des tâches spécifiques et ne vont pas plus loin : les assistants vocaux comme Siri et Alexa, les générateurs d'images et les traiteurs de langage naturels sont tous des formes d'IA étroite

Étude de cas : l'IA dans le développement logiciel

L'IA générale aussi connue sous le nom d'IA fort, qui est un concept théorique. Cette IA utilise ses précédentes compétences et pour pouvoir accomplir de nouvelles tâches sans avoir besoin d'intervention humaine et permet ainsi à l'IA d'apprendre et accomplir n'importe quelle tâche intellectuelle qu'un humain puisse faire.

“Super IA” ou superintelligence artificielle est purement théorique : on peut parler de conscience, cette IA peut faire des jugements et possède des capacités cognitives.

On peut aussi séparer les IA en se basant sur leurs fonctionnalités :

a. Les sous catégories des IA basées sur leurs fonctionnalités

i. *L'IA réactive :*

Ces IAs sont dépourvues de mémoires et font des tâches spécifiques, elles travaillent uniquement avec les données disponibles, elles sont issues de statistiques et produisent des résultats avec l'analyse de grandes quantités de données. Les robots industriels, les systèmes de gestion de trafic sont des exemples d'IA réactive.

ii. *L'IA à mémoire limitée :*

Celle-ci peut se souvenir d'éléments passés et surveiller des situations au fil du temps. Ces IAs peuvent donc utiliser des données du passé pour faire des décisions. Ces IA ont cependant les données pendant une durée déterminée et ne peuvent conserver les données d'expériences passées pour une utilisation à long terme. Les IA génératives comme ChatGPT, les assistants virtuels et les systèmes de conduite automatique sont des exemples d'IA à mémoire limitée.

iii. *La théorie de l'IA possédant une conscience*

Cette classe d'IA est dans la catégorie des IA générale et n'existe pas encore aujourd'hui : elle comprendrait les émotions et pensées d'autres entités. Celles-ci peuvent en théorie simuler des relations humaines. Cette IA est toujours en développement et utilisera d'autres manières que le texte pour analyser les saisies des humains.

iv. *L'IA consciente*

Une fois de plus, cette IA n'existe pas et peut-être jamais. Celle-ci est purement théorique : elle pourrait comprendre ses conditions et caractéristiques internes, ainsi que les émotions humaines et nos pensées, en plus d'avoir ses propres émotions et croyances.

b. Les sous catégories des IA génératives

i. *La génération de texte*

Celles-ci sont sûrement les plus utilisées, les chat bot IA sont régulièrement les sujets des actualités et médias. Ces modèles sont utilisés pour de nombreuses choses : écrire du code jusqu'à relire des textes.

ii. *La génération d'image*

Avec des prompts (textes apportés à l'IA), ces IA peuvent le transformer en une image. Elles sont de plus en plus utilisées car chaque mise à jour permet une création de dessins, maintenant de photos plus claires et plus personnalisables. Des questions peuvent se poser sur l'avenir de certains artistes et du manque de données d'entraînement qui poussent les IA à réutiliser ses propres créations et dégrader ses performances.

iii. *La génération de son*

Ces IA peuvent identifier les motifs dans les différents genres et chez les artistes pour créer de la musique basée sur ces caractéristiques

Étude de cas : l'IA dans le développement logiciel

iv. *La génération de vidéo*

Cette forme d'IA générative combine toutes les formes précédemment citées, les données d'entraînement incluent l'audio, la vidéo et le texte pour créer des vidéos.

c. L'IA générative en détails

Les premières versions devaient soumettre leurs données via une interface de programmation d'applications (API) ou par un processus compliqué. Les utilisateurs devaient aussi savoir utiliser des langages de programmation pour pouvoir les utiliser. Maintenant, nous pouvons faire des requêtes en langage courant et modifier le résultat comme le style, le ton avec de simples commentaires. Les Modèles d'IA génératives combinent plusieurs algorithmes pour représenter et traiter du contenu. Par exemple, les IA qui traitent le langage naturel (NLP) et génèrent du texte, transforment des caractères comme les lettres, la ponctuation et les mots en phrases, parties de discours, entités et actions qui sont représentés en vecteurs grâce à plusieurs techniques d'encodage. Le même genre de techniques sont utilisés par les autres génération comme les images transformées en plusieurs éléments visuels en vecteurs. Il faut cependant faire attention car ces techniques peuvent encoder des biais et racisme contenu dans les données d'entraînement.

Une fois que les développeurs sont d'accord pour la manière de représentation, ils appliquent un réseau neuronal particulier pour générer un nouveau contenu en réponse à une requête. Des techniques telles que les auto encodeurs vibrationnels (VAE) : des réseaux neuronaux dotés d'un décodeur et d'un encodeur permettent de générer des visages humains réalistes. Le progrès récent dans le domaine des transformateurs comme le BERT de Google, le GPT d'OpenAI ont permis de créer des réseaux neuronaux capables d'encoder du langage, des images et du nouveau contenu.

Les IA génératives ont des limites :

- Nous n'avons pas toujours accès à ses sources
- Le contenu qui sonne réaliste rend plus difficile l'identification de fausse information
- Les résultats peuvent passer sous silence des partis pris, préjugés et haine
- Elles peuvent "halluciner" : inventer des fausses sources pour prouver ses informations

III. L'impact de l'IA sur les diverses phases du développement logiciel

1. Préparation et analyse

L'IA générative peut aider à la recherche de données pour retrouver des informations pour le projet, identifier les tendances du marchés, les demandes des utilisateurs et les risques. En étudiant les retours des utilisateurs et les réseaux sociaux les IA peuvent identifier le sentiment des consommateurs. En ce qui concerne la communication avec les parties prenantes, les IA n'ont pas de grands impacts sur cet aspect du développement logiciel : à part résumer les longs documents et automatiser la prise de note pendant les réunions, les humains restent essentiels pour cet aspect-là.

2. Définir les exigences

Cette phase est aussi très axée sur la rédaction de documents et de communication entre différentes équipes. Pendant cette phase, les IA génératives peuvent être utilisées principalement pour aider à la rédaction si possible, et lister certaines tâches à suivre.

Étude de cas : l'IA dans le développement logiciel

3. Design

« C'est un peu comme si nous avions un collègue de bureau avec qui faire du brainstorming » - Ramakrishnan vice-président de du département informatique et DSI de Freshworks. Pour cette phase les IA génératives peuvent apporter beaucoup de différentes idées pour les différentes fonctionnalités et exigences. Les prototypes peuvent parfois être fait par les IAs génératives pour visualiser le produit. Le design des interfaces peut aussi être fait en partie par des IA.

4. Développement

Etant la phase la plus longue, c'est aussi celle qui semble de plus en plus influencée par les IA. Les IA peuvent coder certaines parties du code, corriger des erreurs, faire des test... Le plus souvent, ce sont des assistants ou des outils qui vont accompagner les développeurs afin que le code s'écrive de manière plus fluide et rapide. D'après les retours des utilisateurs, les développeurs débutants ont de fortes améliorations en productivité. Les IA les plus utilisées sont développées sur du code comme le copilot GitHub. En général, l'utilisation est faite comme ce qui suit : les IA sont utilisées pour traduire du code source en python. Malgré les erreurs et suggestions pas toujours correctes, ceci crée une bonne structure pour commencer le code. Le codage est fait plus efficacement grâce à la complétion automatique qui permet de coder plus rapidement. Il y a une sorte de « créativité » avec les IA qui peuvent offrir des algorithmes ou autres solutions quand la syntaxe est nouvelle pour le programmeur ou autre. Pour le débogage, les IA sont aussi très utiles pour les utilisateurs qui peuvent simplement mettre leurs lignes et obtenir des solutions.

Malgré tous ces avantages, il y peut y avoir des problèmes avec l'utilisation des IA en particulier génératives pour le développement logiciel. Bien que les IA puissent proposer des différentes idées pour des projets, il est difficile de parler de créativité, ainsi il peut avoir une perte de créativité avec la surutilisation d'IA. Le manque de transparence sur l'obtention des résultats peut aussi présenter des soucis de responsabilité quand il y a des soucis. Il y a aussi des problèmes de sécurité : les données de l'entreprise qui sont données à l'IA exploitables par les entreprises derrière celles-ci.

5. Expérimentation

Afin de générer des tests, les IA génératives peuvent être utilisées pour générer les tests eux-mêmes automatiquement. De plus, des environnements de simulations peuvent être générés pour tester comment le logiciel fonctionne sous différentes conditions.

6. Déploiement

L'IA peut optimiser les pipelines d'intégration en automatisant le processus de déploiement ce qui réduit le risque d'erreur humaine : pour le déploiement en Canary, le trafic peut être automatiquement envoyé tout en vérifiant le bon fonctionnement. Pour les déploiements dans des environnements, les IA peuvent les configurer automatiquement. De plus, la surveillance peut se faire automatiquement par les IA tout en fournissant des alertes quand il y a des problèmes.

7. Maintenance

Afin d'assister à la maintenance, l'IA peut trier et hiérarchiser les demandes de maintenance, afin de traiter rapidement les problèmes critiques. Le logiciel peut aussi avoir un système de chat bot afin de répondre et résoudre les problèmes de moindre importance ou facilement solvables. Il est également possible que l'IA active des systèmes d'autoréparations. Un des outils les plus utiles pour la maintenance est l'analyse que les IA peuvent faire sur beaucoup de données et retours d'utilisateurs et ainsi suggérer des mises à jour.

Étude de cas : l'IA dans le développement logiciel

En conclusion, l'IA et surtout celle générative offre des avantages significatifs sur quasiment toutes les étapes du développement logiciel. En automatisant certaines tâches basiques, en améliorant l'analyse de donnée, l'IA permet aux développeurs de se concentrer sur les aspects stratégiques et créatifs.

IV. Développement logiciel IA

Application de Recettes de Cuisine :

Une application où les utilisateurs peuvent rechercher, enregistrer et partager des recettes, ainsi que planifier leurs repas hebdomadaires.

1. Préparation et analyse IA

Afin de commencer mon logiciel, j'ai décidé de faire un logiciel de gestion de recette : avec une possibilité de stockage des recettes sur un site. Je n'ai pas fait la toute première partie de la préparation et analyse avec une IA car je pense que dire comment jouer le rôle de partie prenante puis lui poser des questions est contreproductif car ce serait fortement influencé par comment je l'ai conditionné à jouer le rôle de partie prenante. J'ai donc commencé avec ce prompt qui mets le contexte sur ce que l'IA doit faire :

Bonjour, vous êtes un expert en analyse de marché. Je suis dirigeant d'une petite entreprise qui cherche à développer un logiciel de stockage/partage de recette, il faudrait que vous m'aidiez pour analyser le marché : établir les besoins des clients et utilisateur, la faisabilité du produit, les revenus, les coûts de production... Je souhaiterais utiliser des cadres de priorisation des fonctionnalités en fonction du coût, de la valeur du logiciel et autres facteurs.

Voici le résultat :

1. Analyse des besoins des clients et des utilisateurs

- **Segmentation de marché** : Identifier les différents segments de clients potentiels (ex. : amateurs de cuisine, chefs professionnels, blogueurs culinaires, etc.).
- **Études de marché** : Utiliser des sondages, des interviews et des groupes de discussion pour comprendre les besoins spécifiques et les attentes des utilisateurs.
- **Analyse de la concurrence** : Examiner les solutions existantes sur le marché, leurs fonctionnalités, leurs points forts et leurs faiblesses.

2. Faisabilité du produit

- **Analyse technique** : Évaluer les technologies nécessaires pour développer le logiciel (plateformes, langages de programmation, infrastructure cloud, etc.).
- **Ressources nécessaires** : Identifier les compétences et les ressources humaines requises (développeurs, designers, spécialistes UX/UI, etc.).
- **Risques potentiels** : Identifier les principaux risques techniques et de marché et proposer des stratégies pour les atténuer.

3. Prévision des revenus

- **Modèles de revenus** : Explorer différents modèles de monétisation (abonnements, freemium, publicités, achats intégrés, etc.).

Étude de cas : l'IA dans le développement logiciel

- **Estimation du marché** : Évaluer la taille du marché potentiel et les parts de marché réalisables.
- **Projections financières** : Élaborer des projections financières basées sur des hypothèses de croissance des utilisateurs, taux de conversion, etc.
- 4. **Coûts de production**
 - **Développement initial** : Estimer les coûts de développement du logiciel, y compris les salaires des développeurs, les coûts des outils et des licences, etc.
 - **Maintenance et support** : Prévoir les coûts récurrents pour la maintenance, les mises à jour et le support client.
 - **Marketing et distribution** : Calculer les coûts associés au lancement et à la promotion du produit.
- 5. **Cadres de priorisation des fonctionnalités**
 - **Analyse Coût-Bénéfice** : Utiliser des matrices coût-bénéfice pour évaluer et prioriser les fonctionnalités en fonction de leur coût de développement et de la valeur ajoutée pour l'utilisateur.
 - **Méthodes de priorisation** : Appliquer des méthodes telles que MoSCoW (Must have, Should have, Could have, Won't have) et Kano pour classifier les fonctionnalités.
 - **Feedback utilisateur** : Intégrer les retours des utilisateurs dans le processus de priorisation pour s'assurer que les fonctionnalités développées répondent aux besoins réels.

Pour commencer, il serait utile de recueillir des informations plus spécifiques sur votre entreprise et vos objectifs pour le logiciel de stockage/partage de recettes. Pourriez-vous fournir des détails supplémentaires sur votre vision pour ce produit, votre public cible, et tout autre renseignement pertinent ?

Ceci peut servir de base pour savoir quelles questions se poser quand on veut faire du développement logiciel. Je voulais voir à quel point le IA pouvaient faire les étapes seules :

N'étant pas un expert dans ce domaine, je souhaiterais que vous, en tant qu'expert faites certains choix pour moi, mon entreprise est petite, nous souhaitons créer un logiciel simple afin que tout le monde puisse l'utiliser, la cible serait plutôt les 20-50 ans à la recherche de quoi cuisiner. Dans le futur utiliser des algorithmes pour déterminer ce que nos clients préfèrent en termes de goûts pour recommander des recettes qu'ils préféreraient. Cependant pour l'instant c'est pour un logiciel simple.

Merci pour ces précisions. Voici un plan adapté à vos besoins pour développer un logiciel simple de stockage/partage de recettes.

1. Analyse des besoins des clients et des utilisateurs

Segmentation de marché :

- **Cible principale** : Adultes de 20 à 50 ans intéressés par la cuisine, incluant les amateurs de cuisine, les parents cherchant des idées de repas, et les jeunes adultes apprenant à cuisiner.

Étude de cas : l'IA dans le développement logiciel

Études de marché :

- **Sondages et interviews** : Créer des questionnaires simples pour recueillir des informations sur les habitudes culinaires, les préférences en matière de recettes, et les fonctionnalités souhaitées dans un logiciel de recettes.
- **Groupes de discussion** : Organiser des sessions avec des membres de votre cible pour discuter de leurs besoins et attentes.

Analyse de la concurrence :

- **Évaluation des applications existantes** : Analyser les applications populaires de gestion de recettes (ex. : Paprika, Yummly, BigOven) pour identifier les fonctionnalités clés et les lacunes à combler.

2. Faisabilité du produit

Analyse technique :

- **Technologies** : Utiliser des technologies courantes et faciles à maintenir, comme une application web basée sur React pour le front-end et Node.js pour le back-end. Une base de données comme MongoDB peut être utilisée pour stocker les recettes.
- **Infrastructure** : Hébergement cloud (par exemple, AWS ou Google Cloud) pour assurer la scalabilité et la fiabilité.

Ressources nécessaires :

- **Équipe** : Un petit groupe de développeurs full-stack, un designer UX/UI, et un chef de projet pour coordonner les efforts.

Risques potentiels :

- **Technologie** : Risque de bugs et de problèmes de performance initiale.
- **Adoption** : Risque que les utilisateurs ne trouvent pas l'application intuitive ou utile.

3. Prévision des revenus

Modèles de revenus :

- **Freemium** : Offrir une version gratuite avec des fonctionnalités de base, et une version premium avec des fonctionnalités avancées (ex. : organisation des recettes, suggestions personnalisées).
- **Publicités** : Intégrer des annonces pour les utilisateurs de la version gratuite.
- **Abonnements** : Offrir un abonnement mensuel ou annuel pour accéder aux fonctionnalités premium.

Estimation du marché :

Étude de cas : l'IA dans le développement logiciel

- **Taille du marché :** Le marché des applications culinaires est large, avec une forte adoption parmi les utilisateurs mobiles et les amateurs de cuisine.
- **Parts de marché :** Commencer par cibler une petite part de marché avec des campagnes de marketing ciblées et un bon référencement sur les app stores.

Projections financières :

- **Utilisateurs :** Estimer le nombre d'utilisateurs gratuits et payants après six mois et un an de lancement.
- **Revenus :** Calculer les revenus basés sur les abonnements et les publicités.

4. Coûts de production

Développement initial :

- **Salaires :** Calculer les coûts pour les développeurs, designers et autres membres de l'équipe.
- **Outils et licences :** Inclure les coûts des outils de développement, de conception, et des licences nécessaires.

Maintenance et support :

- **Coûts récurrents :** Prévoir les coûts pour la maintenance du serveur, les mises à jour de l'application, et le support utilisateur.

Marketing et distribution :

- **Lancement :** Prévoir un budget pour les campagnes de lancement, incluant les annonces sur les réseaux sociaux, les collaborations avec les influenceurs culinaires, etc.

5. Cadres de priorisation des fonctionnalités

Analyse Coût-Bénéfice :

- **Matériel nécessaire :** Créer une matrice pour évaluer les coûts de développement de chaque fonctionnalité par rapport à la valeur ajoutée pour les utilisateurs.

Méthodes de priorisation :

- **MoSCoW :** Classifier les fonctionnalités en Must have (doit avoir), Should have (devrait avoir), Could have (pourrait avoir), et Won't have (n'aura pas).
- **Kano :** Utiliser le modèle Kano pour évaluer l'impact des fonctionnalités sur la satisfaction des utilisateurs.

Feedback utilisateur :

Étude de cas : l'IA dans le développement logiciel

- **Boucles de rétroaction** : Mettre en place des mécanismes pour recueillir les retours des utilisateurs dès le lancement (ex. : enquêtes, évaluations dans l'application).

Prochaines étapes

1. **Réaliser une étude de marché** : Utiliser des sondages et des groupes de discussion pour affiner la compréhension des besoins des utilisateurs.
2. **Définir les fonctionnalités de base** : Prioriser les fonctionnalités essentielles pour la première version de l'application.
3. **Commencer le développement** : Former une petite équipe de développement pour commencer à travailler sur la version initiale.
4. **Planifier le lancement** : Préparer une stratégie de marketing pour le lancement de l'application.

Ici, l'IA reste assez vague avec plusieurs propositions, pour mon cas, je peux ignorer beaucoup d'éléments, cependant ils sont cruciaux pour le développement logiciel en entreprise. L'étude de marché, l'analyse de la concurrence, les ressources nécessaires, les modèles de revenue, la projection financière, les coûts, le marketing, les modèles de priorisation et vont être ignorés. Pour étudier le marché j'ai fais appel à l'IA :

Pouvez-vous faire cette partie ? : Analyse de la concurrence : ... texte que l'IA avait donné.

Pouvez-vous faire cette partie ? : Analyse de la concurrence : Pouvez-vous faire cette partie ? : Analyse de la concurrence :

Analyse de la concurrence

1. Paprika

Fonctionnalités clés : ...

Lacunes potentielles : ...

2. Yummly

Fonctionnalités clés :...

Lacunes potentielles :...

3. BigOven

Fonctionnalités clés :...

Lacunes potentielles :...

Synthèse des lacunes à combler

Personnalisation et recommandations :

Étude de cas : l'IA dans le développement logiciel

- Mettre en place un système de recommandations personnalisées basé sur les préférences des utilisateurs et l'historique des recettes consultées ou enregistrées.

Interface utilisateur intuitive :

- Développer une interface moderne et facile à utiliser, même pour les nouveaux utilisateurs.

Modèle freemium équilibré :

- Proposer une version gratuite sans trop de publicités intrusives et des fonctionnalités premium attractives sans être indispensables.

Fonctionnalités de base robustes :

- Assurer des fonctionnalités de base solides comme la génération de listes de courses, la planification des repas, et la capture facile de recettes depuis le web.

Intégration sociale :

- Ajouter des fonctionnalités sociales pour permettre le partage de recettes et d'idées entre utilisateurs, ainsi que des options de collaboration sur les listes de courses et les plans de repas.

Pour clôturer la première étape du développement logiciel, j'ai demandé une estimation du marché :

Très bien, pouvez-vous faire ceci : Estimation du marché :... (texte de l'IA)

J'ai trouvé que l'analyse était plus intéressante, la voici :

Estimation du marché

Taille du marché

Le marché des applications culinaires est vaste et en croissance continue, avec une forte adoption parmi les utilisateurs mobiles et les amateurs de cuisine. Voici quelques points clés :

1. Croissance du marché des applications mobiles :

- Selon diverses études, le marché des applications mobiles continue de croître à un rythme rapide. En 2023, il était estimé à plus de 935 milliards de dollars.
- Les applications de style de vie, y compris les applications culinaires, représentent une part significative de ce marché.

2. Popularité des applications culinaires :

- La pandémie de COVID-19 a entraîné une augmentation significative de l'utilisation des applications culinaires, car de plus en plus de personnes cuisinaient à domicile.

Étude de cas : l'IA dans le développement logiciel

- Les applications de recettes comme Yummly, Paprika et BigOven comptent des millions de téléchargements et d'utilisateurs actifs.

3. Tendances alimentaires et technologiques :

- Les tendances alimentaires comme le véganisme, les régimes cétogènes et la cuisine saine ont conduit à une demande accrue pour des recettes spécifiques.
- L'essor des technologies d'intelligence artificielle et de machine learning a permis le développement de fonctionnalités de personnalisation avancées dans les applications culinaires.

Parts de marché

Pour une petite entreprise entrant sur le marché des applications culinaires, il est important de cibler une part de marché réaliste avec des stratégies de marketing efficaces.

1. Segmentation du marché :

- **Public cible** : Adultes de 20 à 50 ans intéressés par la cuisine. Ce groupe comprend des amateurs de cuisine, des parents cherchant des idées de repas, et des jeunes adultes apprenant à cuisiner.
- **Géographie** : Concentrer les efforts de marketing sur des régions spécifiques où l'adoption des applications culinaires est élevée.

2. Campagnes de marketing ciblées :

- **Publicité sur les réseaux sociaux** : Utiliser des plateformes comme Instagram, Facebook et TikTok pour atteindre des audiences intéressées par la cuisine et les recettes.
- **Collaborations avec influenceurs** : Travailler avec des blogueurs culinaires et des influenceurs pour promouvoir l'application et atteindre une audience plus large.
- **Optimisation des app stores** : Utiliser l'ASO (App Store Optimization) pour améliorer la visibilité de l'application dans les app stores. Cela inclut l'utilisation de mots-clés pertinents, des descriptions attrayantes et des captures d'écran de haute qualité.

3. Estimation de la part de marché :

- **Objectif initial** : Viser à acquérir une part de marché de 0,1% à 0,5% dans la première année. Bien que cela puisse sembler modeste, cela représente des milliers d'utilisateurs actifs, ce qui est une base solide pour une petite entreprise.
- **Croissance progressive** : Augmenter progressivement la part de marché en fonction des retours des utilisateurs et des améliorations continues de l'application.

Étude de cas : l'IA dans le développement logiciel

4. Mesure et ajustement :

- **Analytiques et feedback** : Utiliser des outils analytiques pour suivre l'engagement des utilisateurs, les taux de rétention et les conversions. Recueillir des retours réguliers des utilisateurs pour améliorer l'application.
- **Adaptation des stratégies** : Adapter les stratégies de marketing et de développement en fonction des données et des retours pour maximiser l'acquisition et la rétention des utilisateurs.

2. Définir les exigences IA

Voici comment j'ai utilisé l'IA pour la deuxième phase :

Excellent, nous pouvons passer à la deuxième phase du développement logiciel : Définir les exigences : il faudrait pouvoir faire des documents : une spécification des exigences logicielles (SRS), un document sur les cas d'utilisation et un document sur la matrice de traçabilité des exigences.

Définir les exigences : Phase 2 du développement logiciel

Pour structurer cette phase, nous allons créer les documents suivants :

1. **Spécification des exigences logicielles (SRS)**
2. **Document sur les cas d'utilisation**
3. **Matrice de traçabilité des exigences**

1. Spécification des exigences logicielles (SRS)

Introduction

1.1 Objectif Ce document décrit les exigences fonctionnelles et non fonctionnelles pour le développement de l'application de gestion de recettes.

1.2 Portée L'application permettra aux utilisateurs de stocker, partager et rechercher des recettes. Elle sera ciblée principalement sur les adultes de 20 à 50 ans cherchant des idées de repas.

1.3 Définitions, acronymes, et abréviations

- **SRS** : Spécification des exigences logicielles
- **UI** : Interface utilisateur
- **API** : Interface de programmation d'application
- **DB** : Base de données

1.4 Références

- Documentation de React, Node.js, MongoDB
- Guides de bonnes pratiques UX/UI

Étude de cas : l'IA dans le développement logiciel

1.5 Aperçu Le document SRS contient les exigences fonctionnelles et non fonctionnelles, les interfaces externes, et les caractéristiques de performance.

Description générale

2.1 Perspective du produit L'application sera une application web accessible via des navigateurs mobiles et de bureau. Elle utilisera React pour le front-end, Node.js pour le back-end, et MongoDB pour la base de données.

2.2 Fonctions du produit

- Création et gestion de recettes
- Recherche de recettes par ingrédients, type de plat, etc.
- Planification de repas hebdomadaires
- Génération automatique de listes de courses
- Partage de recettes avec d'autres utilisateurs

2.3 Caractéristiques des utilisateurs Les utilisateurs seront des adultes de 20 à 50 ans, intéressés par la cuisine, incluant les amateurs de cuisine, les parents et les jeunes adultes.

2.4 Contraintes

- Doit fonctionner sur les navigateurs mobiles et de bureau courants
- Doit être intuitif et facile à utiliser

2.5 Hypothèses et dépendances

- Les utilisateurs auront accès à Internet
- L'application utilisera des services cloud pour le stockage et la synchronisation des données

Exigences spécifiques

3.1 Exigences fonctionnelles

RF-1 : Gestion des recettes

- **RF-1.1** : Les utilisateurs peuvent ajouter, modifier et supprimer des recettes.
- **RF-1.2** : Les recettes peuvent être catégorisées par type, ingrédient, temps de préparation, etc.

RF-2 : Recherche de recettes

- **RF-2.1** : Les utilisateurs peuvent rechercher des recettes par mots-clés.
- **RF-2.2** : Les utilisateurs peuvent filtrer les recettes par ingrédients, type de plat, etc.

Étude de cas : l'IA dans le développement logiciel

RF-3 : Planification des repas

- **RF-3.1** : Les utilisateurs peuvent planifier des repas sur un calendrier hebdomadaire.
- **RF-3.2** : Les utilisateurs peuvent ajouter des recettes aux repas planifiés.

RF-4 : Liste de courses

- **RF-4.1** : Les utilisateurs peuvent générer automatiquement des listes de courses à partir des recettes sélectionnées.
- **RF-4.2** : Les listes de courses peuvent être modifiées et partagées.

RF-5 : Partage de recettes

- **RF-5.1** : Les utilisateurs peuvent partager des recettes avec d'autres utilisateurs via des liens ou des réseaux sociaux.

3.2 Exigences non fonctionnelles

RNF-1 : Performance

- **RNF-1.1** : L'application doit charger en moins de 2 secondes sur une connexion standard.

RNF-2 : Sécurité

- **RNF-2.1** : Les données utilisateur doivent être sécurisées et chiffrées.
- **RNF-2.2** : L'application doit utiliser des protocoles de sécurité standard (HTTPS).

RNF-3 : Utilisabilité

- **RNF-3.1** : L'interface utilisateur doit être intuitive et facile à naviguer.
- **RNF-3.2** : L'application doit être accessible aux utilisateurs ayant des handicaps.

2. Document sur les cas d'utilisation

Cas d'utilisation 1 : Ajouter une recette

Acteurs : Utilisateur

Description : L'utilisateur ajoute une nouvelle recette à sa collection.

Scénario principal :

1. L'utilisateur clique sur "Ajouter une recette".
2. L'utilisateur entre les détails de la recette (nom, ingrédients, instructions).
3. L'utilisateur sélectionne les catégories appropriées.
4. L'utilisateur enregistre la recette.
5. L'application confirme que la recette a été ajoutée avec succès.

Étude de cas : l'IA dans le développement logiciel

Cas d'utilisation 2 : Rechercher des recettes

Acteurs : Utilisateur

Description : L'utilisateur recherche des recettes en utilisant des mots-clés.

Scénario principal :

1. L'utilisateur entre des mots-clés dans la barre de recherche.
2. L'utilisateur clique sur "Rechercher".
3. L'application affiche une liste de recettes correspondant aux mots-clés.
4. L'utilisateur peut cliquer sur une recette pour voir les détails.

Cas d'utilisation 3 : Planifier un repas

Acteurs : Utilisateur

Description : L'utilisateur planifie un repas pour une date spécifique.

Scénario principal :

1. L'utilisateur ouvre le calendrier de planification des repas.
2. L'utilisateur sélectionne une date.
3. L'utilisateur ajoute des recettes au repas pour cette date.
4. L'application enregistre le plan de repas.

3. Matrice de traçabilité des exigences

ID Exigence	Description Exigence	Cas d'utilisation associé
RF-1.1	Ajouter une recette	CU-1
RF-2.1	Rechercher des recettes	CU-2
RF-3.1	Planifier un repas	CU-3
RF-4.1	Générer une liste de courses	CU-4
RF-5.1	Partager une recette	CU-5

Ces documents pouvant être très longs, je n'ai pas demandé de les rédiger en entier, cependant il est tout à fait possible de rédiger une grande partie en utilisant une IA générative. En effet, avec le prompt suivant, on observe la capacité des IA générative pouvant être utilisées pour ce genre de tests.

Très bien, étant aussi expert pour la rédaction documents SRS, pouvez-vous me faire simplement : 1.0. Introduction vide mais qui contient : 1.1. Objet de l'étude 1.2. Portée du projet

1.0 Introduction

Étude de cas : l'IA dans le développement logiciel

1.1 Objet de l'étude

L'objet de cette étude est de définir les exigences logicielles pour le développement d'une application de gestion de recettes. Ce document servira de référence pour l'équipe de développement, les parties prenantes et les utilisateurs afin de garantir que les fonctionnalités et les caractéristiques de l'application répondent aux besoins identifiés.

1.2 Portée du projet

La portée du projet inclut la création d'une application web permettant aux utilisateurs de stocker, partager et rechercher des recettes. L'application sera principalement destinée aux adultes âgés de 20 à 50 ans qui cherchent des idées de repas. Les principales fonctionnalités incluront la gestion des recettes, la planification des repas, la génération de listes de courses et le partage de recettes. Le projet vise à fournir une interface utilisateur intuitive et une expérience utilisateur agréable, tout en intégrant des pratiques de sécurité et de performance élevées.

Pour finaliser la deuxième phase, voici mon prompt qui change le rôle de l'IA :

Vous êtes maintenant un excellent ingénieur logiciel, pouvez-vous faire une liste de tâche, assigner les responsabilités aux équipes, anticiper les problèmes et estimer les coûts concrets

Liste des tâches, responsabilités des équipes, anticipation des problèmes et estimation des coûts

Liste des tâches et responsabilités

Phase 1 : Planification et définition des exigences

1. Définir les exigences (1 semaine)
 - ...
2. Conception de l'architecture (2 semaines)
 - ...

Phase 2 : Développement

3. Développement du backend (4 semaines)
 - ...
4. Développement du frontend (4 semaines)
 - ...
5. Intégration et tests (3 semaines)
 - ...

Phase 3 : Déploiement et maintenance

1. Déploiement initial (2 semaines)
 - ...

Étude de cas : l'IA dans le développement logiciel

2. Support et maintenance (continu)

- ...

Anticipation des problèmes

1. Problèmes de performance

- ...

2. Sécurité des données

- ...

3. Évolution des exigences

- ...

4. Défauts de l'interface utilisateur

- ...

Estimation des coûts concrets

1. Ressources humaines

- **Chef de projet** : 1 personne, 10 000 € / mois
- **Analyste business** : 1 personne, 8 000 € / mois
- **Architecte logiciel** : 1 personne, 12 000 € / mois
- **Développeurs (frontend et backend)** : 4 personnes, 8 000 € / mois chacune
- **Designer UI/UX** : 1 personne, 7 000 € / mois
- **Testeurs QA** : 2 personnes, 6 000 € / mois chacune
- **Ingénieur DevOps** : 1 personne, 9 000 € / mois
- **Ingénieur sécurité** : 1 personne, 10 000 € / mois

Total mensuel des ressources humaines : $10\,000 + 8\,000 + 12\,000 + (4 \times 8\,000) + 7\,000 + (2 \times 6\,000) + 9\,000 + 10\,000 = 90\,000$ €

Durée estimée du projet : 12 semaines (3 mois)

Coût total des ressources humaines : $90\,000 \text{ €} \times 3 \text{ mois} = 270\,000 \text{ €}$

2. Ressources matérielles et logicielles

- **Outils de développement et licences** : 10 000 € (pour la durée du projet)
- **Serveurs et hébergement cloud** : 5 000 € / mois
- **Tests utilisateurs et autres frais divers** : 5 000 €

Total coûts matériels et logiciels : $10\,000 + (5\,000 \times 3) + 5\,000 = 30\,000$ €

Coût total estimé du projet : $270\,000 \text{ € (ressources humaines)} + 30\,000 \text{ € (matériels et logiciels)} = 300\,000 \text{ €}$

Étude de cas : l'IA dans le développement logiciel

Les détails des étapes sont supprimés car je les ai estimés assez basiques, de plus, l'IA décide des logiciels, ce qui peut être un peu contreproductif si l'on se limite à seuls ceux proposés.

3. Design IA

Pour la phase de design, j'avais déjà demandé une base pour voir si le projet était possible donc mon prompt impose certains langages :

Passons maintenant à la phase de design, celle ci doit comporter le plan et la vision général , le document de conception doit contenir le design, le langage de programmation, j'utiliserai python et en particulier Flask pour le backend, avec html et css pour le frontend ainsi que du react en java.

N'étant pas tout à fait satisfait avec le résultat, qui était trop simpliste, j'ai refait un prompt :

Pour continuer la phase il faut prendre en compte les exigences fonctionnelles... (III.3)

Afin d'avoir une réponse moins large et vague, j'ai ajouté mon paragraphe sur la description de la phase design, voici la réponse :

Phase de design : Plan et vision générale (Complété)

1.0 Introduction

La phase de design du projet de développement de l'application de gestion de recettes vise à fournir une vue d'ensemble claire de la structure et de l'architecture de l'application. Cette phase inclut la définition du design global, du choix des technologies, et de la structure de l'application. Nous allons également nous concentrer sur les exigences fonctionnelles, les exigences non fonctionnelles, et les restrictions techniques et budgétaires.

2.0 Vision générale

L'application de gestion de recettes est conçue pour être intuitive, performante et sécurisée, offrant aux utilisateurs une expérience fluide pour la gestion de leurs recettes. La vision est de créer une application web accessible sur les navigateurs mobiles et de bureau, qui permette de stocker, partager et rechercher des recettes, ainsi que de planifier des repas et générer des listes de courses.

3.0 Plan de conception

3.1 Architecture de l'application

L'architecture de l'application sera basée sur une structure MVC (Modèle-Vue-Contrôleur), séparant les préoccupations et facilitant la maintenance et l'évolution du code.

- **Frontend (Vue) :**
 - **Technologies :** HTML, CSS, React (JavaScript)
 - **Description :** Le frontend est responsable de l'interface utilisateur et de l'interaction avec l'utilisateur. React sera utilisé pour créer des composants UI réactifs et dynamiques.
- **Backend (Contrôleur) :**
 - **Technologies :** Python, Flask

Étude de cas : l'IA dans le développement logiciel

- **Description :** Le backend gère la logique de l'application, les règles métier, et la communication avec la base de données. Flask sera utilisé pour créer une API RESTful pour la communication entre le frontend et le backend.
- **Base de données (Modèle) :**
 - **Technologies :** MongoDB
 - **Description :** La base de données stocke les informations des recettes, des utilisateurs, des commentaires, etc. MongoDB est choisi pour sa flexibilité et sa performance.

3.2 Exigences fonctionnelles

- **Gestion des recettes**
 - Ajouter, modifier et supprimer des recettes
 - Catégoriser les recettes par type, ingrédient, temps de préparation, etc.
- **Recherche de recettes**
 - Rechercher des recettes par mots-clés
 - Filtrer les recettes par ingrédients, type de plat, etc.
- **Planification des repas**
 - Planifier des repas sur un calendrier hebdomadaire
 - Ajouter des recettes aux repas planifiés
- **Liste de courses**
 - Générer automatiquement des listes de courses à partir des recettes sélectionnées
 - Modifier et partager les listes de courses
- **Partage de recettes**
 - Partager des recettes avec d'autres utilisateurs via des liens ou des réseaux sociaux

3.3 Exigences non fonctionnelles

- **Performance**
 - L'application doit charger en moins de 2 secondes sur une connexion standard.
- **Sécurité**
 - Les données utilisateur doivent être sécurisées et chiffrées.
 - L'application doit utiliser des protocoles de sécurité standard (HTTPS).
- **Utilisabilité**
 - L'interface utilisateur doit être intuitive et facile à naviguer.

Étude de cas : l'IA dans le développement logiciel

- L'application doit être accessible aux utilisateurs ayant des handicaps.
- **Modularité**
 - Le code doit être organisé en modules réutilisables pour faciliter la maintenance et l'extensibilité.
- **Testabilité**
 - Le code doit être écrit de manière à faciliter les tests unitaires et d'intégration.
- **Extensibilité**
 - L'architecture doit permettre l'ajout de nouvelles fonctionnalités sans modifications majeures.

3.4 Restrictions techniques et budgétaires

- **Techniques**
 - Doit fonctionner sur les navigateurs mobiles et de bureau courants
 - Utiliser les technologies choisies (Flask, React, MongoDB)
- **Budgétaires**
 - Le budget total estimé est de 300 000 € pour la durée du projet

3.5 Description de l'interface utilisateur

- **Pages principales :**
 - **Page d'accueil :** Présentation générale, recherche de recettes
 - **Page de recherche :** Barre de recherche, résultats filtrés
 - **Page de recette :** Détails de la recette, ingrédients, instructions
 - **Page de planification des repas :** Calendrier hebdomadaire, ajout de recettes
 - **Page de liste de courses :** Liste générée à partir des recettes sélectionnées
 - **Page de profil utilisateur :** Informations de l'utilisateur, recettes favorites
- **Composants UI :**
 - **Barre de navigation :** Liens vers les différentes pages, barre de recherche
 - **Cartes de recette :** Affichage des recettes sous forme de cartes avec image, titre, et description courte
 - **Formulaires :** Pour l'ajout et la modification de recettes, les commentaires, et la planification des repas

Conclusion

Cette phase de design définit la structure, les exigences et les technologies de l'application de gestion de recettes. En se concentrant sur la modularité, la testabilité et l'extensibilité, nous

Étude de cas : l'IA dans le développement logiciel

assurons que l'application pourra évoluer et s'améliorer au fil du temps. La prochaine étape consistera à implémenter ce design en suivant les exigences définies précédemment.

Ce prompt semble un peu ambitieux, bien qu'il donne des idées pour les dépendances fonctionnelles et non fonctionnelles, si l'on lui demande de faire certaine, il n'y aurait pas vraiment de garantie que l'IA puisse les implémenter, surtout celles non fonctionnelles.

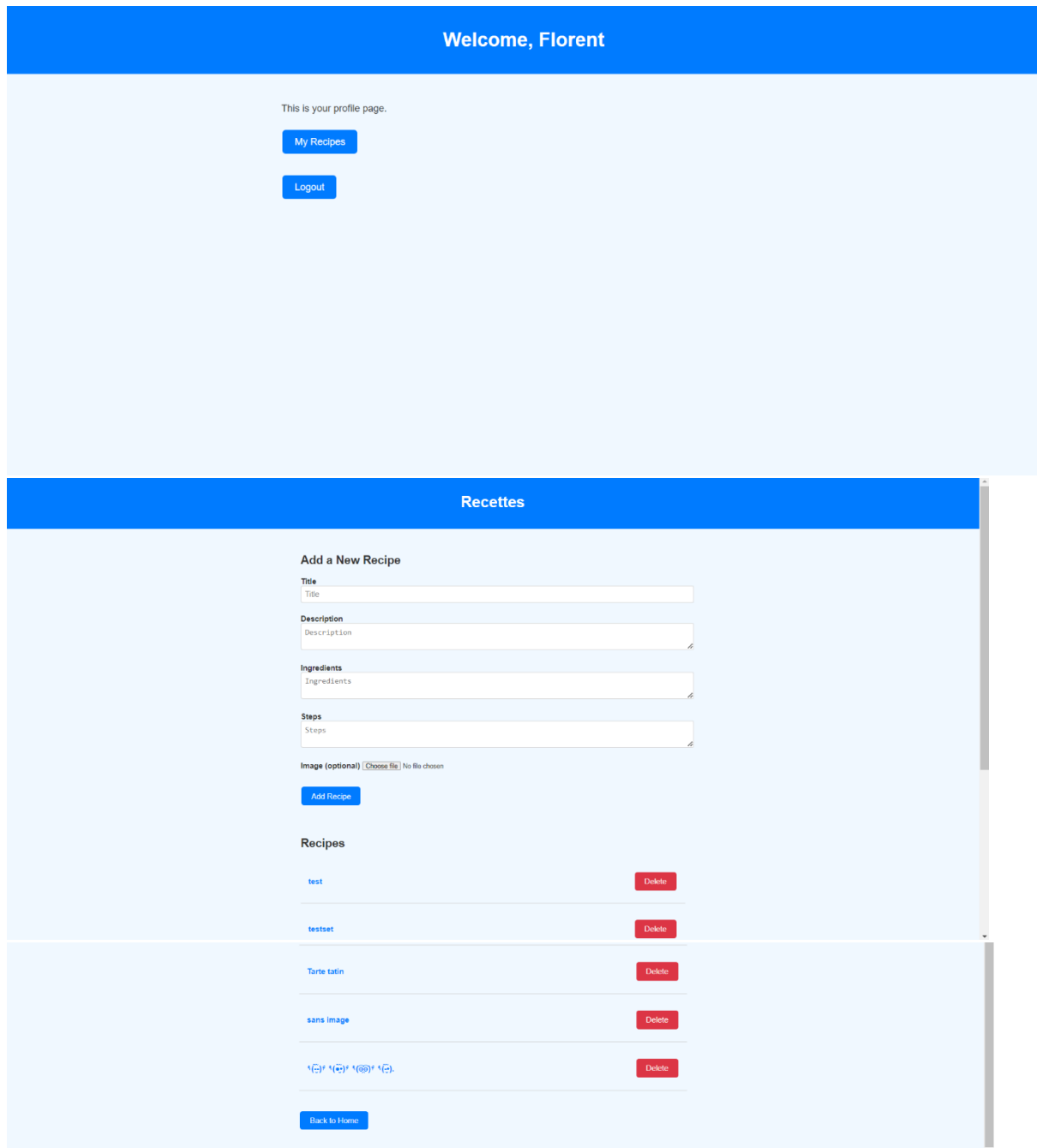
4. Développement IA

Pour le code, tout est sur la page github : <https://github.com/florent88/Recette>

Sinon, voici des screenshots :

The image displays two screenshots of a web application interface. The top screenshot is the 'Login' page, featuring a blue header with the title 'Login'. Below the header, there are input fields for 'Email' (containing 'florentlee@gmail.com') and 'Password' (masked with dots). A blue 'Login' button is positioned below the password field. At the bottom, there is a link that reads 'Don't have an account? Register'. The bottom screenshot is the 'Register' page, also with a blue header titled 'Register'. It contains input fields for 'Username' (containing 'florentlee@gmail.com'), 'Email', and 'Password' (masked with dots). A blue 'Register' button is located below the password field. At the bottom, there is a link that reads 'Already have an account? Log in'.


Étude de cas : l'IA dans le développement logiciel



Étude de cas : l'IA dans le développement logiciel

Tarte tatin

Tarte tatin



Tarte tatin rapide et facile

Ingredients

- 6 pommes
- 150g de sucre
- 150g de beurre
- 1 pâte feuilletée
- Rhum

Steps

1. Mettre le sucre dans une poêle pouvant aller au four à feu doux-moyen
2. Une fois que le sucre est caramélisé, arrêtez la cuisson avec un peu d'eau et de rhum
3. Rajoutez le beurre à feu doux et laissez jusqu'à ce que de grosses bulles se forment
4. Enlevez du feu et rajoutez les pommes épluchées, épépinées et coupées en 2, bien les serrer entre elles.
5. Rajoutez la pâte feuilletée par dessus, y faire des petits trous à l'aide d'une fourchette
6. Enfouez au four à 200°C pendant 20-25 minutes.

Edit Recipe

Title
Tarte tatin

Description
Tarte tatin rapide et facile

Ingredients
6 pommes
150g de sucre
150g de beurre

Steps
1. Mettre le sucre dans une poêle pouvant aller au four à feu doux-moyen
2. Une fois que le sucre est caramélisé, arrêtez la cuisson avec un peu d'eau et de rhum

Update Image No file chosen

Bien que l'IA a pu faire une grande partie du code, pour avoir un site qui fonctionne, j'ai dû faire quelques modifications au code. Un des problèmes que j'ai rencontrés souvent est le fait que Chat GPT ne se rendait pas compte qu'il faisait des importations circulaires, ce qui empêchait le code de pouvoir être lancé. Etant la première fois que j'utilisais flask et son shell, j'ai dû utiliser quelques solutions en ligne pour une utilisation qui quelquefois, malgré les instructions, ne fonctionnait pas. De plus, le tout premier code que Chat GPT m'a donné ne fonctionnait pas, c'est en donnant le code de base de visual studio que l'IA a pu se servir pour faire une base sans erreurs. Certains problèmes n'ayant pas pu être réglés sont : la page d'accueil est directement le login car parfois mais pas tout le temps, le code se lançait directement dans l'accueil pour voir les recettes sans demander un login. Un autre problème est le fait que les utilisateurs peuvent modifier les recettes de n'importe qui et ajouter ce qu'ils veulent. Les IA avaient du mal avec la génération de base de données et flask qui, par-dessus, générait des erreurs. Je n'ai donc pas modifié trop le code pour que ce soit toujours fait principalement par une IA générative. Un autre petit souci est le fait que j'ai dû demander du css pour avoir moins de redondance car l'IA faisait directement sur html pour chaque fichier le style spécifique.

5. Expérimentation IA

Je n'ai pas inclus les tests de l'IA car ceux sur le code et son fonctionnement étaient trop basiques, et ceux sur la sécurité et la flexibilité ne fonctionnaient pas. Je pense que le problème était le fait que le code était devenu trop large pour que l'IA se souvienne de chaque ligne dans différents fichiers. Ainsi, les tests modifiaient d'autres lignes de code qui eux-mêmes étaient

Étude de cas : l'IA dans le développement logiciel

responsables pour le bon fonctionnement du logiciel. Je n'ai donc pas inclus ceux-ci dans mon projet. En utilisant Claude, voici le code que j'ai eu pour les tests :

```
import pytest
from Recette import app, db
from Recette.models import User, Recipe
from werkzeug.security import generate_password_hash

@pytest.fixture
def client():
    app.config['TESTING'] = True
    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///memory:'
    client = app.test_client()

    with app.app_context():
        db.create_all()

    yield client

    with app.app_context():
        db.drop_all()

@pytest.fixture
def sample_user():
    user = User(username='testuser', email='test@example.com',
password=generate_password_hash('password'))
    db.session.add(user)
    db.session.commit()
    return user

@pytest.fixture
def sample_recipe(sample_user):
    recipe = Recipe(
        title='Test Recipe',
        description='A test recipe',
        ingredients='Ingredient 1, Ingredient 2',
        steps='Step 1, Step 2',
        user_id=sample_user.id
    )
    db.session.add(recipe)
    db.session.commit()
    return recipe

def test_home_page(client):
    response = client.get('/')
    assert response.status_code == 200
    assert b'Home Page' in response.data
```

Étude de cas : l'IA dans le développement logiciel

```
def test_register(client):
    response = client.post('/register', data={
        'username': 'newuser',
        'email': 'newuser@example.com',
        'password': 'newpassword'
    }, follow_redirects=True)
    assert response.status_code == 200
    assert b'Registration successful!' in response.data

def test_login_logout(client, sample_user):
    # Test login
    response = client.post('/login', data={
        'email': 'test@example.com',
        'password': 'password'
    }, follow_redirects=True)
    assert response.status_code == 200
    assert b'Profile' in response.data

    # Test logout
    response = client.get('/logout', follow_redirects=True)
    assert response.status_code == 200
    assert b'Login' in response.data

def test_create_recipe(client, sample_user):
    client.post('/login', data={
        'email': 'test@example.com',
        'password': 'password'
    })
    response = client.post('/recipes', data={
        'title': 'New Recipe',
        'description': 'A new test recipe',
        'ingredients': 'New Ingredient 1, New Ingredient 2',
        'steps': 'New Step 1, New Step 2'
    }, follow_redirects=True)
    assert response.status_code == 200
    assert b'New Recipe' in response.data

def test_delete_recipe(client, sample_user, sample_recipe):
    client.post('/login', data={
        'email': 'test@example.com',
        'password': 'password'
    })
    response = client.post(f'/recipes/delete/{sample_recipe.id}', follow_redirects=True)
    assert response.status_code == 200
    assert b'Test Recipe' not in response.data

def test_view_recipe(client, sample_recipe):
    response = client.get(f'/recipes/{sample_recipe.id}')
```

Étude de cas : l'IA dans le développement logiciel

```
assert response.status_code == 200
assert b'Test Recipe' in response.data
```

```
def test_unauthorized_access(client):
    response = client.get('/profile', follow_redirects=True)
    assert response.status_code == 200
    assert b'Please log in to access this page' in response.data
```

Je ne l'ai pas inclus dans mon github car tous les tests concluaient à des erreurs, malheureusement, le mix des erreurs dans les tests et des différentes IA a fait qu'ils n'étaient pas très efficaces et que le code généré par Chat GPT pas très robuste.

6. Déploiement IA

Pour le déploiement, ne sachant pas entièrement comment fonctionne ce processus, j'ai demandé à l'IA comment s'y prendre :

Passons au déploiement de mon application, comment feriez-vous pour automatiser le processus de déploiement en Canary par exemple

Pour automatiser le déploiement de votre application en utilisant la méthode de déploiement Canary, vous pouvez suivre ces étapes générales :

Automatisation du Déploiement Canary

1. Configuration de l'Environnement de Développement

Assurez-vous que votre environnement de développement est configuré avec les outils nécessaires :

- **Système de contrôle de version** : Utilisez Git pour gérer le code source.
- **Outils de build** : Par exemple, utilisez Docker pour la conteneurisation et des outils comme Jenkins ou GitLab CI/CD pour l'automatisation des builds.

2. Stratégie de Déploiement Canary

La stratégie Canary consiste à déployer une nouvelle version de manière progressive en dirigeant un pourcentage du trafic vers la nouvelle version tout en laissant le reste du trafic aller vers l'ancienne version. Voici comment vous pouvez l'automatiser :

- **Script de déploiement** : Écrivez des scripts (par exemple, des scripts shell ou des scripts de déploiement Ansible) qui automatisent le processus de déploiement Canary.
- **Configuration de l'infrastructure** : Utilisez des outils de gestion d'infrastructure comme Terraform pour provisionner et gérer l'infrastructure nécessaire (par exemple, les serveurs, les conteneurs Docker, etc.).

3. Pipeline CI/CD

Utilisez un pipeline CI/CD pour automatiser le déploiement Canary :

- **Intégration continue (CI)** : Configurez des étapes CI qui compilent le code, exécutent les tests unitaires et génèrent des artefacts prêts pour le déploiement.

Étude de cas : l'IA dans le développement logiciel

- **Déploiement continu (CD)** : Configurez des étapes CD qui automatisent le déploiement Canary en utilisant les artefacts générés par CI.

4. Configuration de Kubernetes (optionnel)

Si vous utilisez Kubernetes pour l'orchestration des conteneurs :

- **Déploiement Kubernetes** : Utilisez des objets comme les Deployments et les Services pour gérer les versions et le routage du trafic.
- **Rolling Updates** : Configurez les stratégies de mises à jour progressives pour diriger le trafic de manière contrôlée vers la nouvelle version.

5. Monitoring et Rollback Automatique

- **Monitoring** : Intégrez des outils de monitoring comme Prometheus pour surveiller la performance de la nouvelle version.
- **Rollback Automatique** : Configurez des scripts ou utilisez des outils comme Istio pour automatiser le rollback vers la version précédente en cas de détection de problèmes majeurs.

Exemple de Script de Déploiement Canary (sous forme simplifiée)

Voici un exemple simplifié de script de déploiement Canary utilisant des outils comme Kubernetes :

```
# Déploiement de la nouvelle version
```

```
kubectl apply -f deployment-new.yaml
```

```
# Configuration du service pour le routage
```

```
kubectl apply -f service-canary.yaml
```

```
# Validation du déploiement Canary
```

```
# Exemples de vérification de santé, de tests, etc.
```

```
# Si les vérifications réussissent, progressivement rediriger le trafic
```

```
kubectl apply -f deployment-new-canary.yaml
```

```
# Monitorer les performances de la nouvelle version
```

```
# Exemples de vérifications de performance avec Prometheus
```

```
# Si OK, augmenter progressivement le pourcentage de trafic vers la nouvelle version
```

```
# Exemple d'ajustement des règles de trafic sur Istio
```

```
# Si des problèmes sont détectés, rollback automatique vers la version stable
```

```
# Exemple de script de rollback avec kubectl
```

Pour tester les capacités de l'IA, j'ai demandé de faire la première étape, cependant, pour la lisibilité du mémoire et à cause de la limite de mes connaissances en déploiement logiciel, il n'y aura que sa réponse.

Étude de cas : l'IA dans le développement logiciel

Pouvez vous faire la première étape?

Bien sûr ! La première étape consiste à configurer votre environnement de développement avec les outils nécessaires pour automatiser le processus de déploiement en Canary. Voici comment vous pouvez procéder :

Étape 1 : Configuration de l'Environnement de Développement

1.1 Système de Contrôle de Version

Utilisez Git pour gérer le code source de votre application. Assurez-vous que Git est installé sur votre machine de développement et que vous avez un référentiel Git configuré pour votre projet.

- Installez Git si ce n'est pas déjà fait :
 - Sur Linux (Ubuntu) :

```
sudo apt update
```

```
sudo apt install git
```

- Sur macOS, vous pouvez utiliser Homebrew :

```
brew install git
```

- Sur Windows, vous pouvez télécharger Git à partir du site officiel et suivre les instructions d'installation.

- Configurez votre identité Git :

```
git config --global user.name "Votre Nom"
```

```
git config --global user.email "votre@email.com"
```

- Initialisez un nouveau référentiel Git pour votre projet :

```
cd chemin/vers/votre/projet
```

```
git init
```

1.2 Outils de Build et d'Automatisation

Pour automatiser le processus de build et de déploiement, vous pouvez utiliser Docker pour la conteneurisation de votre application et un outil CI/CD tel que Jenkins ou GitLab CI/CD pour l'automatisation des builds.

Docker

Docker permet de créer des conteneurs légers et portables pour les applications, ce qui facilite la gestion des dépendances et assure une portabilité accrue entre les environnements de développement, de test et de production.

- **Installation de Docker :**
 - Sur Linux, vous pouvez suivre les instructions sur le site officiel de Docker.
 - Sur macOS et Windows, Docker Desktop offre une installation simple.

Étude de cas : l'IA dans le développement logiciel

- Une fois Docker installé, vous pouvez créer un fichier Dockerfile à la racine de votre projet pour définir l'environnement d'exécution de votre application.

Exemple de Dockerfile pour une application Flask avec React en frontend :

Dockerfile

Utiliser une image Python officielle en tant qu'image de base

FROM python:3.8-slim

Définir le répertoire de travail dans le conteneur

WORKDIR /app

Copier les fichiers nécessaires dans le conteneur

COPY requirements.txt .

COPY app.py .

COPY static static/

COPY templates templates/

Installer les dépendances Python spécifiées

RUN pip install --no-cache-dir -r requirements.txt

Exposer le port 5000 pour Flask

EXPOSE 5000

Commande par défaut à exécuter lorsque le conteneur démarre

CMD ["python", "./app.py"]

CI/CD (Continuous Integration / Continuous Deployment)

Pour automatiser le processus de build, de test et de déploiement, vous pouvez utiliser Jenkins, GitLab CI/CD, Travis CI ou d'autres outils similaires. Voici comment configurer un pipeline CI/CD simplifié avec Jenkins :

- Installez Jenkins sur votre serveur ou utilisez Jenkins en tant que service cloud.
- Configurez un nouveau projet Jenkins et associez-le à votre référentiel Git.
- Définissez un fichier Jenkinsfile à la racine de votre projet pour décrire les étapes du pipeline :

Exemple de Jenkinsfile :

groovy

Copy code

pipeline {

agent any

Étude de cas : l'IA dans le développement logiciel

```
stages {
  stage('Build') {
    steps {
      // Étape de build, par exemple compilation, tests unitaires
      sh 'docker build -t monapp:latest .'
    }
  }
  stage('Test') {
    steps {
      // Étape de test, par exemple tests d'intégration
      sh 'docker run monapp:latest pytest'
    }
  }
  stage('Deploy Canary') {
    steps {
      // Étape de déploiement Canary
      sh 'kubectl apply -f deployment-canary.yaml'
    }
  }
}
```

Exemple de configuration de Jenkins pour un pipeline CI/CD :

1. Installez Jenkins et configurez-le pour qu'il puisse accéder à votre référentiel Git.
2. Créez un nouveau projet Jenkins de type "Pipeline".
3. Définissez le pipeline en utilisant le Jenkinsfile précédemment décrit.

En suivant ces étapes, vous configurez votre environnement de développement avec les outils nécessaires pour automatiser le processus de déploiement en Canary pour votre application de gestion de recettes. Cette configuration vous permettra de compiler, tester et déployer automatiquement les nouvelles versions de votre application de manière contrôlée et progressive.

Au vu des résultats, je pense qu'il serait difficile de faire toute la configuration du déploiement avec une IA, je pense que cette étape requiert plus d'intervention humaine par rapport aux autres.

Étude de cas : l'IA dans le développement logiciel

7. Maintenance IA

La maintenance peut être faite en partie en par l'IA, il faudrait idéalement un système de gestion des demandes et d'autoréparation :

Pour finaliser ce projet, il faudrait que l'IA assiste pour la maintenance du logiciel, vous pourriez coder un système de gestion et hiérarchie des demandes de maintenance. s'il y a des bugs il faudrait peut être des systèmes d'autoréparation

Ceci à rendu de simples étapes pour faire ceci, j'ai donc reformulé ma requête :

[...] Pour coder le système des demandes de modifications, faites comme si vous aviez un fichier texte avec celles-ci

Le code rendu n'était toujours pas satisfaisant, j'ai donc répondu avec :

Pourriez vous coder un système de gestion qui hiérarchise les demandes, par exemple si beaucoup de demandes ont le même mot (des mots uniques comme freeze ou autre), mettre la première (celle la plus haut dans le fichier) puis la mettre en avant par exemple avec votre script qui affiche le texte

```
import os

from collections import defaultdict

def lire_demandes_de_modification(nom_fichier):
    """ Fonction pour lire les demandes de modifications à partir d'un fichier texte """
    if os.path.exists(nom_fichier):
        with open(nom_fichier, 'r') as fichier:
            lignes = fichier.readlines()
            demandes = []
            demande_actuelle = {}
            for ligne in lignes:
                if ligne.strip() == "":
                    if demande_actuelle:
                        demandes.append(demande_actuelle)
                        demande_actuelle = {}
                    else:
                        cle, valeur = ligne.split(':', 1)
                        demande_actuelle[cle.strip()] = valeur.strip()
            # Ajouter la dernière demande
            if demande_actuelle:
                demandes.append(demande_actuelle)
            return demandes
    else:
        raise FileNotFoundError(f"Le fichier {nom_fichier} n'existe pas.")

def afficher_demandes(demandes):
    """ Fonction pour afficher les demandes de modifications """
    for index, demande in enumerate(demandes, start=1):
        print(f"Demande {index}:")
        print(f"Titre : {demande['Titre']}")
```

Étude de cas : l'IA dans le développement logiciel

```
print(f"Description : {demande['Description']}")
print()

def hierarchiser_demandes(demandes):
    """ Fonction pour hiérarchiser les demandes en fonction de la fréquence des mots clés """
    mots_cles_freq = defaultdict(int)

    # Compter la fréquence des mots clés dans les descriptions
    for demande in demandes:
        mots = demande['Description'].lower().split()
        for mot in mots:
            # Ignorer les mots courts (ex. : et, de, la, le, etc.)
            if len(mot) > 3:
                mots_cles_freq[mot] += 1

    # Trier les demandes en fonction de la fréquence des mots clés
    demandes_triees = sorted(demandes, key=lambda demande: -max(mots_cles_freq[mot] for
mot in demande['Description'].lower().split()))

    return demandes_triees

# Exemple d'utilisation
nom_fichier_demandes = 'demandes_modifications.txt'
demandes = lire_demandes_de_modification(nom_fichier_demandes)
demandes_triees = hierarchiser_demandes(demandes)
afficher_demandes(demandes_triees)
```

Voici uniquement le code, assez simpliste, un développeur qui veut gérer les demandes le mieux possibles utilisera sûrement un modèle avec une régression linéaire. De plus, quand je demandais un mécanisme d'autoréparation, celui-ci était très basique et ne pourrait pas fonctionner pour un logiciel.

V. Conclusion

D'après les études et papiers scientifiques et avec ce projet, pour l'instant, les IA ne sont pas assez développées pour faire du développement logiciel en entier. Avant que les IA ne soient peut-être plus développées, ce qui n'est pas sûr à cause des limitations des données utilisables, il semble être difficile de faire des logiciels complexes et même assez simple entièrement avec des IA. Les IA semblent avoir un souci de robustesse dans le code : il peut facilement avoir des erreurs ou des failles, quand on leur demande de les régler, nous obtenons souvent les mêmes codes ou très peu de modifications ce qui revient à du code qui est parfois non fiable.

Cependant, il est important de savoir les utiliser et s'en servir comme outil. En effet, les IA, en particulier génératives, peuvent simplifier de nombreuses étapes du développement logiciel. Utilisées comme outils pour coder des parties simples mais fastidieuses très rapidement, une autre utilisation est pour la rédaction de certains documents qui peuvent aussi être très longs, il est possible de faciliter ce processus.

Étude de cas : l'IA dans le développement logiciel

VI. Bibliographie

<https://theproductmanager.com/topics/software-development-life-cycle/>

<https://www.intelivita.com/en-fr/blog/software-development-process/>

<https://www.ibm.com/think/topics/artificial-intelligence-types>

<https://www.adobe.com/products/firefly/discover/generative-ai-vs-other-ai.html>

<https://www.techtarget.com/searchenterpriseai/definition/generative-AI>

<https://www.cio.com/article/1310540/the-early-returns-on-gen-ai-for-software-development.html#:~:text=Generative%20AI%20is%20already%20having,optimization%20and%20refactoring%2C%20among%20others.>

<https://kpmg.com/us/en/articles/2023/generative-artificial-intelligence.html>

<https://github.blog/2023-04-14-how-generative-ai-is-changing-the-way-developers-work/>

<https://www.cprime.com/resources/blog/15-impacts-of-generative-ai-on-software-development/>

<https://www.linkedin.com/pulse/impact-generative-ai-software-development-finsenseafrica1-flkirf/>

IAs utilisées :

- Chat GPT 4o (principalement)
- Chat GPT 3.5
- Claude 3.5
- Gemini