# TP IA4GL

# Utilisation de l'IA pour l'Automatisation des Tests Fonctionnels

Phase 1 : Démarche d'Automatisation

Analyse du Cahier des Charges

Ryma Ben Salah

Master 2 Génie Logiciel

Université de Montpellier

Année universitaire 2025–2026

# Table des matières

# Chapitre 1

# Introduction

## 1.1 Contexte

Les tests fonctionnels jouent un rôle central dans le cycle de développement logiciel, car ils permettent de vérifier que le comportement d'un système est conforme aux exigences métier décrites dans un cahier des charges. Ils constituent un moyen privilégié pour valider les fonctionnalités du point de vue de l'utilisateur final.

Cependant, la conception de ces tests repose encore largement sur des activités manuelles, telles que l'analyse des spécifications, la rédaction des scénarios et l'identification des cas d'erreur, ce qui représente un coût important en temps et en effort.

## 1.2 Problématique

L'émergence des modèles de langage de grande taille (Large Language Models – LLMs) offre de nouvelles perspectives pour assister ces tâches répétitives. Ces modèles sont capables d'analyser des descriptions textuelles complexes et d'en extraire des informations structurées.

La problématique abordée dans cette phase est la suivante : comment utiliser l'intelligence artificielle pour assister l'analyse d'un cahier des charges fonctionnel afin d'identifier les éléments nécessaires à la génération de tests fonctionnels, tout en conservant un contrôle humain sur les résultats produits.

## 1.3 Objectifs de la Phase 1

La Phase 1 a pour objectifs :

— analyser le cahier des charges de l'application TodoManager ;
— identifier les fonctionnalités à tester ;
— extraire les règles métier et contraintes associées ;

— identifier les cas nominaux et les cas d'erreur ;
— proposer une démarche d'analyse assistée par l'IA.

## 1.4 Cas d'étude : Application TodoManager

TodoManager est une application de gestion de tâches permettant à un utilisateur de créer, consulter, modifier, finaliser et supprimer des tâches. Bien que simple, cette application présente un ensemble complet de fonctionnalités CRUD ainsi que des règles de validation intéressantes pour l'étude des tests fonctionnels.

# Chapitre 2

# Phase 1 : Analyse du Cahier des Charges

## 2.1 Vue d'ensemble de la Phase 1

La Phase 1 constitue le point d'entrée de la démarche d'automatisation des tests fonctionnels. Elle vise à transformer une description fonctionnelle textuelle en une base structurée exploitable pour la suite du processus, notamment la génération des scénarios de tests.

Cette phase repose sur une analyse conjointe entre l'étudiant et l'intelligence artificielle, l'IA jouant un rôle d'assistant à l'analyse.

## 2.2 Entrée de la Phase 1

L'entrée principale de cette phase est le cahier des charges fonctionnel de l'application TodoManager. Celui-ci décrit :

- les objectifs généraux de l'application
- les entités manipulées et leurs attributs
- les fonctionnalités offertes à l'utilisateur
- les règles métier et contraintes de validation
- les comportements attendus en cas d'erreur

## 2.3 Description générale de l'application

TodoManager est une application de gestion de tâches permettant à un utilisateur de créer et de gérer une liste de tâches personnelles. L'objectif principal de l'application est d'offrir des fonctionnalités simples et clairement définies pour manipuler des tâches, tout en respectant les exigences fonctionnelles spécifiées dans le cahier des charges.

Cette application constitue un cas d'étude adapté pour l'expérimentation de l'automatisation des tests fonctionnels, car elle présente un périmètre fonctionnel bien délimité, des règles métier explicites et plusieurs cas d'erreur à prendre en compte.

## 2.4 Entité *Tâche*

L'entité centrale de l'application est la *Tâche*. Une tâche représente une action à effectuer par l'utilisateur et constitue l'unité de base manipulée par le système.

Chaque tâche est identifiée de manière unique et possède un cycle de vie simple, allant de sa création jusqu'à sa finalisation ou sa suppression.

## 2.5 Attributs et statuts

Une tâche est définie par les attributs suivants :

— **Identifiant** : identifiant unique généré par le système ;
— **Titre** : chaîne de caractères obligatoire permettant de décrire brièvement la tâche ;
— **Description** : champ optionnel fournissant des informations complémentaires ;
— **Date d'échéance** : date optionnelle associée à la tâche ;
— **Statut** : état courant de la tâche.

Les statuts possibles pour une tâche sont :

— `PENDING` : tâche créée mais non encore commencée ;
— `IN_PROGRESS` : tâche en cours de réalisation ;
— `DONE` : tâche finalisée.

Lors de la création d'une tâche, son statut initial est systématiquement `PENDING`.

## 2.6 Fonctionnalités de l'application

Le cahier des charges définit six fonctionnalités principales fournies par l'application TodoManager :

1. **Créer une tâche** : permettre à l'utilisateur de créer une nouvelle tâche en fournissant au minimum un titre
2. **Lister les tâches** : afficher l'ensemble des tâches existantes
3. **Consulter une tâche** : afficher les informations détaillées d'une tâche à partir de son identifiant
4. **Mettre à jour une tâche** : modifier les attributs d'une tâche existante
5. **Finaliser une tâche** : marquer une tâche comme terminée en changeant son statut
6. **Supprimer une tâche** : retirer définitivement une tâche du système

Ces fonctionnalités constituent la base des scénarios de tests fonctionnels définis et automatisés dans la suite du TP.

## 2.7 Rôle de l'intelligence artificielle dans l'analyse

Le modèle de langage est utilisé pour assister l'analyse sémantique du cahier des charges. Plus précisément, il permet :

— d'identifier et lister les fonctionnalités décrites
— de détecter les règles métier explicites (ex. : titre obligatoire)
— de distinguer les cas nominaux des cas d'erreur
— de proposer une structuration des informations extraites

Les résultats produits par l'IA sont systématiquement relus et validés manuellement afin de garantir leur conformité avec les exigences fonctionnelles.

## 2.8 Exemple de prompt utilisé

```
À partir du cahier des charges suivant, identifie les fonctionnalités,
les règles métier, les cas nominaux et les cas d'erreur à tester.
Présente les résultats de manière structurée et compréhensible.
```

## 2.9 Résultats de l'analyse

L'analyse du cahier des charges de TodoManager a permis d'identifier :

— six fonctionnalités principales (création, consultation, mise à jour finalisation et suppression des tâches)
— des règles métier clés, telles que l'obligation d'un titre lors de la création et le statut initial `PENDING`
— des cas d'erreur associés aux actions invalides ou aux ressources inexistantes

Ces éléments constituent la base nécessaire pour la génération des scénarios de tests fonctionnels dans les phases suivantes.

## 2.10 Limites et validation humaine

Bien que l'IA facilite et accélère l'analyse du cahier des charges, elle ne garantit pas à elle seule l'exactitude des résultats. Une validation humaine reste indispensable pour interpréter correctement les exigences, lever les ambiguïtés et compléter les règles métier implicites.

## 2.11 Conclusion de la Phase 1

Cette première phase démontre l'intérêt de l'intelligence artificielle comme outil d'assistance à l'analyse des cahiers des charges. Elle permet de structurer rapidement les informations nécessaires à la conception des tests fonctionnels, tout en conservant un contrôle humain essentiel.

Les résultats obtenus dans cette phase constituent une base solide pour la génération des scénarios de tests fonctionnels en langage Gherkin lors de la phase suivante.

# Chapitre 3

# Objectifs de la Phase 2

La Phase 2 a pour finalité de convertir les éléments fonctionnels et les règles métier identifiés lors de la phase précédente en scénarios de tests exploitables. Ces scénarios sont ensuite structurés à l'aide du langage Gherkin afin de décrire de manière formelle le comportement attendu du système.

Cette phase vise également à assurer une couverture des cas d'utilisation standards ainsi que des situations d'erreur, en s'appuyant sur l'intelligence artificielle comme outil d'assistance à la génération, tout en conservant une validation humaine des scénarios obtenus.

# Chapitre 4

# Principe des scénarios Gherkin

L'approche Behavior Driven Development (BDD) vise à améliorer la collaboration entre les différents acteurs d'un projet logiciel en décrivant le comportement attendu du système de manière claire et compréhensible. Dans ce contexte, le langage Gherkin est utilisé comme un langage de description des scénarios de tests fonctionnels, permettant de formaliser les exigences sous une forme structurée et lisible.

Un scénario Gherkin décrit une situation concrète d'utilisation du système et se concentre sur le comportement observé plutôt que sur les détails d'implémentation. Il constitue un lien direct entre les exigences fonctionnelles exprimées dans le cahier des charges et les tests automatisés.

## 4.1 Structure d'un scénario Gherkin

Un scénario Gherkin est structuré autour de trois éléments principaux, chacun ayant un rôle précis dans la description du comportement attendu :

- **Given** : cette clause permet de définir le contexte initial du scénario. Elle décrit l'état du système avant l'exécution de l'action testée. Le contexte peut inclure, par exemple, la disponibilité du système ou l'existence préalable de certaines données
- **When** : cette clause décrit l'action effectuée par l'utilisateur ou par le système. Elle représente l'événement déclencheur du scénario, tel qu'une création, une modification ou une suppression de données
- **Then** : cette clause exprime le résultat attendu après l'exécution de l'action. Elle permet de vérifier que le système se comporte conformément aux exigences fonctionnelles

Cette structure favorise une séparation claire entre le contexte, l'action et le résultat attendu, ce qui contribue à la lisibilité et à la précision des scénarios.

## 4.2 Lisibilité et collaboration

L'un des principaux avantages du langage Gherkin réside dans sa lisibilité. Les scénarios sont rédigés en langage quasi naturel, ce qui les rend compréhensibles par des acteurs techniques (développeurs, testeurs) comme par des acteurs non techniques (analystes métier, chefs de projet).

Cette lisibilité facilite la collaboration et la validation des exigences, car les scénarios peuvent être relus et discutés par l'ensemble des parties prenantes avant même leur implémentation technique.

## 4.3 Lien avec l'automatisation des tests

Au-delà de leur rôle descriptif, les scénarios Gherkin servent également de support à l'automatisation des tests. Chaque étape définie par les mots-clés `Given`, `When` et `Then` peut être associée à une implémentation technique, appelée *step definition*, permettant d'exécuter automatiquement le scénario.

Ainsi, les scénarios Gherkin constituent un point de convergence entre la spécification fonctionnelle et l'exécution automatique des tests, garantissant une meilleure traçabilité entre les exigences et leur validation.

# Chapitre 5

# Démarche de génération des scénarios

## 5.1 Entrées de la Phase 2

Les entrées de cette phase sont les résultats de la Phase 1, à savoir :

— la liste des fonctionnalités de l'application TodoManager

— les règles métier associées

— les cas nominaux et les cas d'erreur identifiés

## 5.2 Pipeline d'automatisation

Le pipeline d'automatisation adopté dans ce TP est volontairement simple et réaliste, afin de rester cohérent avec le travail effectivement réalisé. Il se compose des étapes suivantes :

1. analyse du cahier des charges fonctionnel .
2. génération assistée par IA de scénarios de tests fonctionnels en langage Gherkin .
3. validation et ajustement manuels des scénarios générés .
4. implémentation minimale de l'application servant de support à l'exécution des tests
5. génération et écriture assistée des scripts de tests automatisés (Cucumber et Selenium) .
6. exécution automatique des tests et analyse des résultats.

L'utilisation de LLM intervient principalement lors des phases de génération des scénarios et d'assistance à la production des scripts de tests.

## 5.3 Entrées, sorties et rôle de l'IA

Les principales entrées de la démarche sont le cahier des charges fonctionnel et la description des fonctionnalités à tester. Les sorties attendues sont des scénarios de tests

fonctionnels formalisés, des scripts de tests automatisés et des résultats d'exécution.

L'IA joue un rôle d'assistant intelligent : elle propose des scénarios et des scripts, qui sont ensuite validés, ajustés et intégrés par l'étudiant. Cette validation humaine est indispensable pour garantir la conformité aux exigences.

## 5.4 Schéma global de la démarche

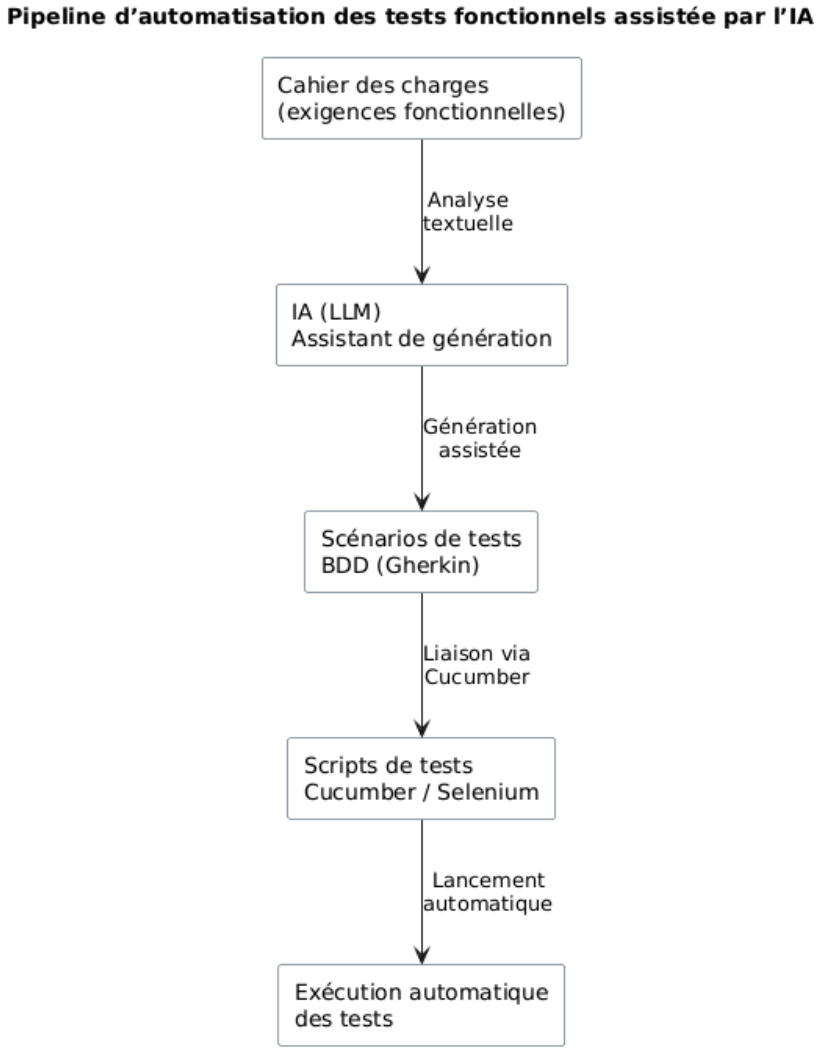


FIGURE 5.1 – Pipeline d'automatisation des tests fonctionnels assisté par IA

Ce schéma illustre la chaîne complète allant du cahier des charges jusqu'à l'exécution automatique des tests fonctionnels. L'intelligence artificielle intervient à deux niveaux principaux : la génération des scénarios de tests et l'assistance à la production des scripts d'exécution.

## 5.5 Vue d'ensemble

La démarche proposée dans ce TP consiste à automatiser certaines tâches liées aux tests fonctionnels à l'aide de l'intelligence artificielle. Le processus global peut être résumé comme suit :

Cahier des charges → IA → Scénarios Gherkin → IA → Scripts de tests → Exécution automatique

## 5.6 Rôle de l'intelligence artificielle

L'intelligence artificielle est utilisée comme un assistant de rédaction afin de proposer automatiquement des scénarios de tests fonctionnels en langage Gherkin. À partir des descriptions textuelles issues du cahier des charges, l'IA suggère des scénarios structurés suivant l'approche BDD.

L'IA ne génère pas des scénarios définitifs, mais fournit une base de travail qui est ensuite exploitée et validée par l'étudiant.

## 5.7 Exemple de prompt utilisé

```
À partir des fonctionnalités et règles métier suivantes,
génère des scénarios de tests fonctionnels en Gherkin
(Given / When / Then), incluant les cas d'erreur.
```

# Chapitre 6

# Validation humaine des scénarios

La génération automatique des scénarios de tests fonctionnels à l'aide de l'intelligence artificielle constitue un gain de temps significatif. Toutefois, les scénarios produits par un modèle de langage ne peuvent pas être considérés comme directement exploitables sans vérification. Une phase de validation humaine est donc indispensable afin de garantir la qualité et la pertinence des scénarios générés.

## 6.1 Nécessité de la validation humaine

Les modèles de langage sont capables de produire des scénarios syntaxiquement corrects et globalement cohérents. Cependant, ils ne disposent pas d'une compréhension complète du contexte métier ni des contraintes spécifiques de l'application. En conséquence, certaines imprécisions, ambiguïtés ou interprétations erronées peuvent apparaître dans les scénarios générés automatiquement.

La validation humaine permet de s'assurer que chaque scénario reflète fidèlement le comportement attendu du système et respecte strictement les exigences fonctionnelles définies dans le cahier des charges.

## 6.2 Types de corrections apportées

Lors de la validation des scénarios, plusieurs types d'ajustements peuvent être nécessaires :

- **Correction des ambiguïtés** : reformulation de certaines étapes lorsque le contexte ou l'action décrite n'est pas suffisamment précise
- **Adaptation au périmètre réel** : suppression ou modification de scénarios couvrant des fonctionnalités non implémentées ou hors du périmètre du TP
- **Alignement avec les règles métier** : vérification que les résultats attendus correspondent exactement aux règles définies (statut initial, conditions d'erreur,

etc.)

— **Uniformisation du vocabulaire** : harmonisation des termes utilisés afin de conserver une cohérence entre les scénarios, le cahier des charges et l'implémentation

## 6.3 Processus de validation

Le processus de validation humaine s'effectue de manière itérative. Les scénarios proposés par l'IA sont d'abord analysés individuellement, puis comparés au cahier des charges. Chaque scénario est ensuite validé, modifié ou, le cas échéant, rejeté.

Cette démarche permet d'exploiter efficacement les capacités de génération rapide de l'IA tout en conservant un contrôle total sur la qualité finale des scénarios.

## 6.4 Importance pour la qualité des tests

La validation humaine joue un rôle clé dans la qualité des tests fonctionnels produits. Elle garantit que les scénarios ne se contentent pas d'être syntaxiquement corrects, mais qu'ils sont également pertinents, complets et conformes aux exigences fonctionnelles.

Cette étape constitue un point de contrôle essentiel dans la démarche d'automatisation, en assurant un équilibre entre automatisation et expertise humaine.

# Chapitre 7

# Exemples de scénarios générés

Ce chapitre présente des exemples représentatifs de scénarios de tests fonctionnels générés à partir du cahier des charges de l'application TodoManager avec l'assistance de l'intelligence artificielle. Les scénarios sélectionnés illustrent à la fois un cas nominal et un cas d'erreur, afin de démontrer la couverture des comportements attendus du système.

## 7.1 Cas nominal : Création d'une tâche avec un titre valide

Le scénario suivant illustre un cas nominal correspondant à une utilisation normale de l'application. Il permet de vérifier que le système se comporte correctement lorsqu'un utilisateur crée une tâche en respectant les contraintes définies dans le cahier des charges.

**Objectif du scénario** L'objectif de ce scénario est de valider la fonctionnalité de création d'une tâche lorsque les données fournies sont valides. Il vérifie que la tâche est correctement enregistrée par le système et que son statut initial est correctement initialisé.

**Description du scénario**

```
Scenario: Création d'une tâche avec un titre valide
Given le système TodoManager est disponible
When l'utilisateur crée une tâche avec le titre "Acheter du lait"
Then la tâche est créée avec le statut "PENDING"
```

**Interprétation fonctionnelle** La clause `Given` définit le contexte initial du scénario, dans lequel le système est opérationnel et prêt à traiter des requêtes. La clause `When` décrit l'action effectuée par l'utilisateur, à savoir la création d'une tâche en fournissant un titre valide. Enfin, la clause `Then` exprime le résultat attendu : la tâche doit être créée avec le statut `PENDING`, conformément aux règles métier définies dans le cahier des charges.

Ce scénario permet ainsi de valider le comportement nominal de la fonctionnalité de création.

## 7.2 Cas d'erreur : Création d'une tâche sans titre

Le scénario suivant illustre un cas d'erreur permettant de tester la robustesse du système face à une action invalide de l'utilisateur.

**Objectif du scénario** L'objectif de ce scénario est de vérifier que le système rejette correctement une demande de création de tâche lorsque l'une des contraintes principales n'est pas respectée, en l'occurrence l'absence de titre.

**Description du scénario**

```
Scenario: Création d'une tâche sans titre
Given le système TodoManager est disponible
When l'utilisateur crée une tâche sans fournir de titre
Then une erreur est retournée
```

**Interprétation fonctionnelle** Dans ce scénario, le contexte initial est identique au cas nominal. L'action décrite par la clause `When` correspond à une tentative de création de tâche ne respectant pas les règles métier. La clause `Then` spécifie que le système doit retourner une erreur, ce qui permet de vérifier que les contraintes fonctionnelles sont correctement appliquées.

Ce scénario contribue à garantir la robustesse du système en s'assurant que les comportements invalides sont correctement détectés et gérés.

# Chapitre 8

# Limites de la génération assistée

L'utilisation de l'intelligence artificielle pour assister la génération des scénarios de tests fonctionnels présente de nombreux avantages, notamment en termes de rapidité et de réduction de l'effort de rédaction. Toutefois, cette approche comporte également plusieurs limites qu'il est nécessaire d'identifier et de prendre en compte afin de garantir la qualité des tests produits.

## 8.1 Limites liées à la compréhension du contexte

Les modèles de langage sont capables d'analyser des descriptions textuelles et de produire des scénarios globalement cohérents. Cependant, leur compréhension du contexte métier reste limitée à ce qui est explicitement exprimé dans le texte fourni. Les règles métier implicites ou les contraintes non formalisées peuvent ainsi être partiellement ou incorrectement interprétées.

Cette limitation peut conduire à la génération de scénarios incomplets ou ne reflétant pas exactement le comportement attendu du système.

## 8.2 Précision et ambiguïtés des scénarios générés

Bien que les scénarios générés par l'IA soient généralement syntaxiquement corrects, ils peuvent parfois manquer de précision. Certaines étapes peuvent être formulées de manière trop générique, laissant place à des interprétations ambiguës lors de l'implémentation des tests automatisés.

Dans ce contexte, une reformulation ou un enrichissement manuel des scénarios est souvent nécessaire afin de lever ces ambiguïtés et d'assurer une interprétation unique du comportement testé.

## 8.3 Dépendance à la qualité des entrées

La qualité des scénarios générés dépend fortement de la qualité des informations fournies en entrée. Un cahier des charges incomplet, ambigu ou mal structuré peut conduire à des scénarios incorrects ou partiels.

Cette dépendance souligne l'importance d'une analyse préalable rigoureuse du cahier des charges, ainsi que d'une interaction maîtrisée avec l'outil d'intelligence artificielle.

## 8.4 Rôle indispensable de la validation humaine

Face à ces limites, la validation humaine demeure une étape essentielle du processus. Elle permet de vérifier la conformité des scénarios avec les exigences fonctionnelles, d'identifier les incohérences et d'adapter les scénarios au périmètre réel de l'application.

La combinaison de la génération assistée par l'IA et de l'expertise humaine permet ainsi de tirer parti des avantages de l'automatisation tout en garantissant la qualité et la pertinence des tests fonctionnels.

# Chapitre 9

# Conclusion de la Phase 2

Cette phase a permis de démontrer l'intérêt de l'intelligence artificielle comme assistant à la génération des scénarios de tests fonctionnels. En combinant l'approche BDD, le langage Gherkin et la validation humaine, il est possible de produire rapidement des scénarios structurés et exploitables pour l'automatisation des tests.

Les scénarios obtenus constituent l'entrée principale de la phase suivante, dédiée à l'implémentation et à l'exécution automatisée des tests.