

Atelier interactif

Comment avoir confiance dans un outil de génération de tests automatiques ?

Groupe TAIA - CFTL

Clément François - KEREVAL

Alain Ribault - KEREVAL

Contributeurs :

Aya KADI HAJA - Polytech Angers

Ayoub FAIK - Polytech'Angers

Khadija EN-NAANI - Polytech'Angers

Oumaima BENAYAD - Polytech'Angers

Yassine EL GARTI - Polytech'Angers

Alexis TODOSKOFF - Polytech'Angers



Comité Français
des Tests Logiciels



Introduction



Groupe de travail du CFTL

GT TAIA

“Tester avec l’IA générative”

Contacts

- michael@nocode-testing.com
- bruno.legeardcftl@gmail.com
- alexis.todoskoff@univ-angers.fr



• JFTL - Tutoriel Cas d’usage de l’IA

- Notre tutoriel aujourd’hui avec les ressources en open-source

TAIA - Tester avec l’IA

Groupe de travail du CFTL sur l’IA et le test

Accéder à la documentation

Accéder aux cas d’usages

TUTORIEL JFTL 2025

• Enquête IA annuelle - Novembre

- IA dans les organisations et pratiques de test
- Résultats 2024 & 2025

<https://cftl.fr/cf-tl/ressources/>

• Journée Thématische IA (JTIA)

- 2ème édition aujourd’hui !
- Retours d’expérience
- Ateliers de pratique
- <https://cftl.fr/actualites/jtia-2/>



ENQUÈTE

CFTL
IA

2025

SAVE THE DATE

JEUDI
11

DÉCEMBRE
2025

Une plateforme pour partager !

TAIA - Utilisation de l'IA dans le Test Cas d'usages JTIA 2025 Enquête IA 2024 Enquête IA 2025 (à venir) Contributeurs ☰

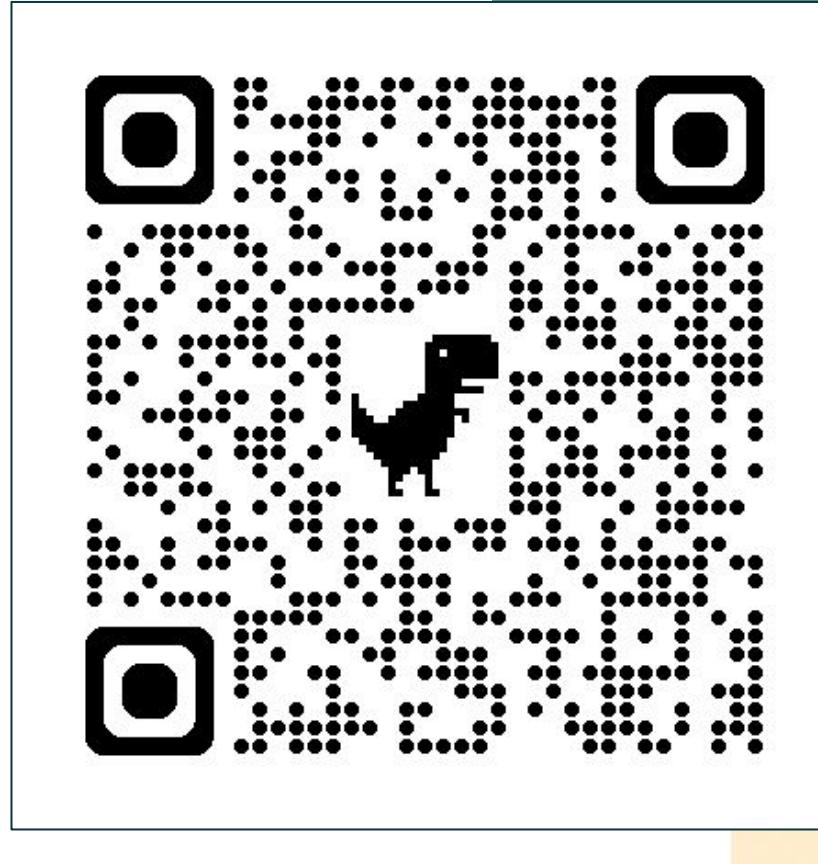
TAIA - Tester avec l'IA

Groupe de travail du CFTL sur l'IA et le test

TUTORIEL JTIA 2025 - L'IA au service du test

Accéder aux cas d'usages

Accéder à la documentation





Référentiel de cas d'usage

Découvrez les cas d'usage d'utilisation de l'IA dans le test



Bibliothèque de tâches de prompts

Profitez des propositions de prompts de la communauté pour améliorer vos tests



Proposez vos contributions

Le projet est en open source sur Github !
Rejoignez et contribuez à cette base de connaissance
sur l'IA dans le test.

<https://cftl-taia.github.io/>



Communauté du test assisté par IA

- ❖ N'hésitez pas à contribuer
- ❖ N'hésitez pas à réagir
- ❖ N'hésitez pas à échanger entre vous

Nous sommes juste des facilitateurs !!!

Comment avoir confiance dans un outil de génération de tests automatiques ?

Objectif du tutoriel

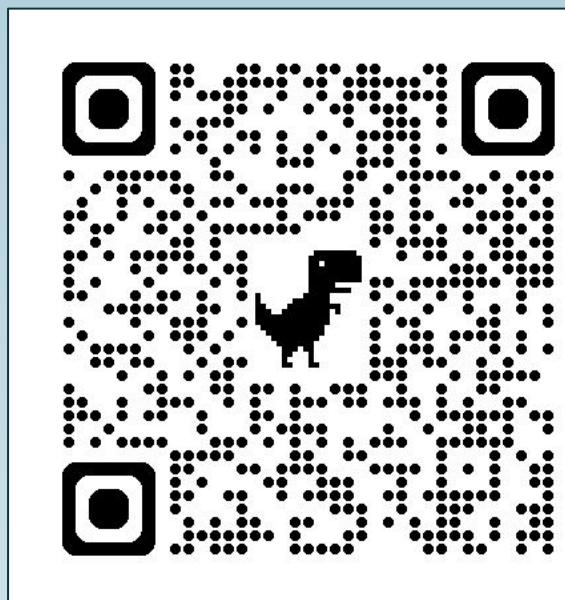
Montrer ce que nous pouvons faire pour évaluer notre confiance dans la génération de tests automatiques, à partir d'un swagger, assistée par IA

Déroulé du tutoriel

- Introduction
- Analyse qualitative à partir de métriques
- Conclusion collégiale

Accès aux ressources

- Prompts à copier
- Tutoriel en format PDF
- Liste de LLM à utiliser
- Informations complémentaires



cctl-taia.github.io/

TAIA - Tester avec l'IA

Groupe de travail du CFTL sur l'IA et le test

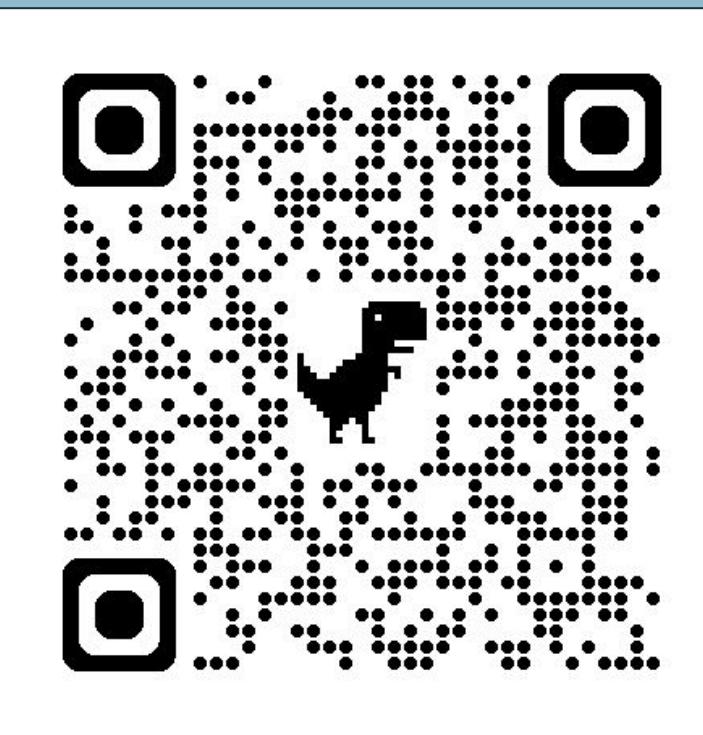
JTIA 2025 - L'IA au service du test

JTIA 2025 - Evaluation de la génération de tests à partir d'un swagger

Accéder aux cas d'usages

Accéder à la documentation

Accès aux LLM



cftl-taia.github.io/docs/TurorielJIA2025/LLM
Liste de LLM



chatgpt.com



chat.mistral.ai/chat



copilot.microsoft.com/copilot

Pré-requis

Installation de Karate (java, maven) :

- <https://docs.karatelabs.io/getting-started/install-dependencies>

Test de l'installation avec un premier test :

- <https://docs.karatelabs.io/getting-started/examples>

Puis :

- <https://docs.karatelabs.io/getting-started/standalone-execution>

Une fois le package généré avec la bonne structure de fichiers, les commandes à exécuter seront :

- mvn clean test

[Menti.com](#) : 3255 8137



Introduction Rappel du contexte

Lors de l'atelier JFTL, certains d'entre vous ont...

... pratiqué la génération de tests automatiques à partir d'une spécification swagger et d'une spécification GraphQL

Système sous test

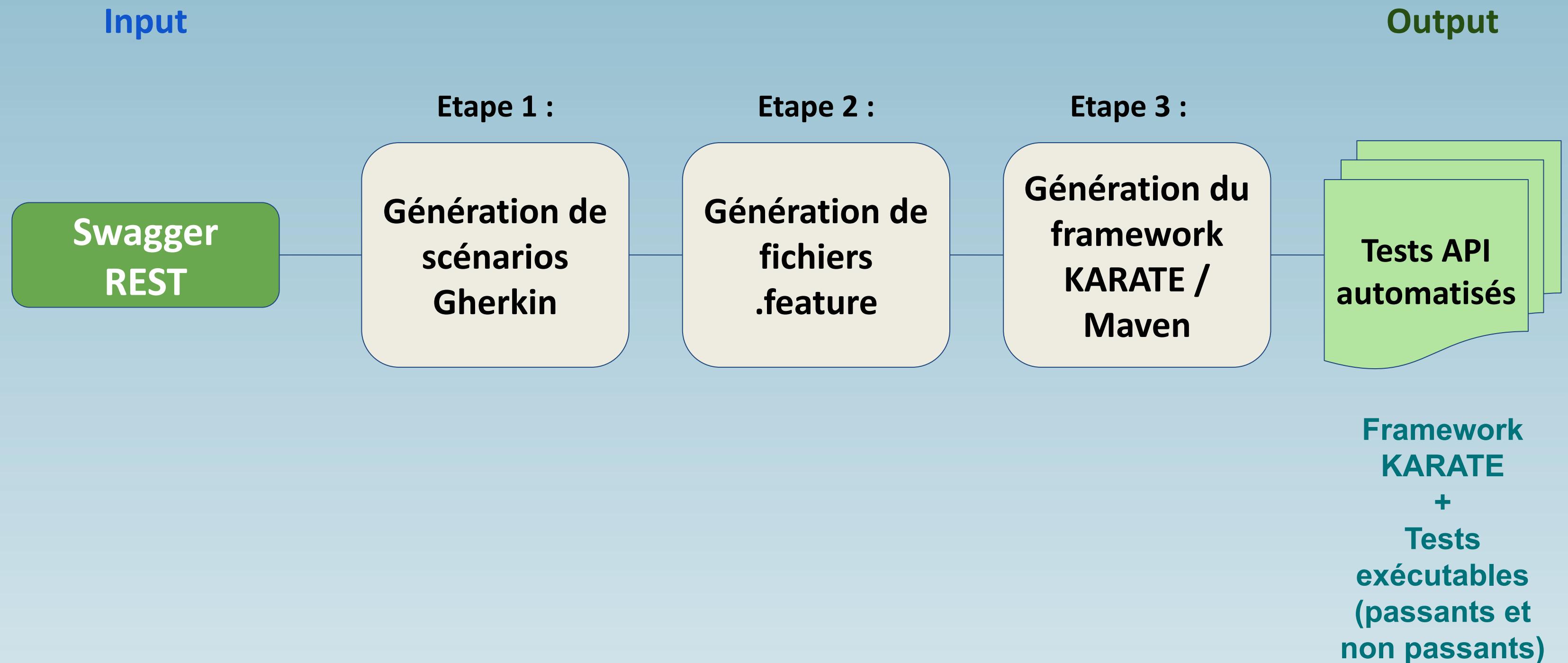
<https://petstore.swagger.io/#/>

{ REST }

The screenshot shows the Swagger Petstore API documentation. At the top, it displays the URL <https://petstore.swagger.io/v2/swagger.json> and an [Explore](#) button. Below this, the title "Swagger Petstore" is shown with version 1.0.7 and OAS 2.0. A note indicates the base URL is petstore.swagger.io/v2. The page contains links for Terms of service, Contact the developer, and Apache 2.0. A "Find out more about Swagger" link is also present. On the left, there's a "Schemes" dropdown set to "HTTPS" and an "Authorize" button with a lock icon. The main content area is titled "pet" with a "Everything about your Pets" subtitle and a "Find out more" link. It lists three operations: a POST method for "/pet/{petId}/uploadImage" (uploads an image), a POST method for "/pet" (Add a new pet to the store), and a PUT method for "/pet" (Update an existing pet). Each operation has a lock icon and a dropdown arrow.

Testé avec le framework de test KARATE

Processus proposé



Mais quelle confiance pouvons-nous avoir dans les tests automatiques générés ?

QUIZZ...

Les métriques que nous avons retenues...

- ❖ Est-ce que le projet généré s'exécute ?
- ❖ Quel pourcentage de scénarios s'exécutent jusqu'au bout ?
- ❖ Quelles sont les sources d'erreurs ?
- ❖ Quelle est la couverture ?
- ❖ Les scénarios de tests sont-ils indépendants ?

Les métriques que nous avons retenues...

- ❖ Est-ce que le projet généré s'exécute ?

M0. Build & Test Sanity (Gatekeeper)

- **Objectif** : Vérifier que le projet généré est exécutable *out-of-the-box* (ZIP exécutable immédiatement).
- **Critères** : `mvn_exit_code` doit être 0, `scenarios_executed` > 0, et aucune erreur bloquante de dépendances (POM/Config) .
- **Décision** : Si KO, le résultat est rejeté (Gate) car inutile de mesurer la couverture si le projet ne compile pas.

- ❖ Quel pourcentage de scénarios s'exécutent jusqu'au bout ?

M1. Scenario Pass Rate (Fiabilité)

- **Objectif** : Valider l'exécutabilité globale. Quelle part des scénarios s'exécute jusqu'au bout sans erreur ?.
- **Formule** : $(\text{scenarios_passed} / \text{scenarios_total}) * 100$.
- **Utilité** : Indicateur vital pour comparer les LLMs et servir de "Quality Gate".

Les métriques que nous avons retenues...

❖ Quelles sont les sources d'erreurs ?

M2. Failure Breakdown (Diagnostic)

- **Objectif** : Comprendre la cause racine des échecs pour orienter les efforts (prompt vs infrastructure).
- **Catégories** : Assertions (métier), Syntaxe (code invalide), Data (données incorrectes), Dépendances (timeout/5xx).

M3. Status-Code Match Rate (Conformité)

- **Objectif** : Vérifier la conformité contractuelle la plus courante (Codes HTTP).
- **Calcul** : Pourcentage de vérifications `Then status <code>` qui sont correctes.
- **Utilité** : Détecte les confusions (ex: 400 vs 404) et discrimine bien les modèles.

M4. Response-Shape Match Rate (Contenu)

- **Objectif** : Valider la forme et le contenu minimal des réponses (schémas JSON).
- **Calcul** : Pourcentage d'assertions de type `match` réussies.

Les métriques que nous avons retenues...

❖ Quelle est la couverture ?

M5. Endpoint Coverage (Couverture)

- **Objectif** : Mesurer la couverture fonctionnelle utile et éviter de ne tester que le "happy path" d'un seul domaine.
- **Calcul** : $(\text{endpoints_tested} / \text{endpoints_total}) * 100$.

M6. Negative Case Ratio (Robustesse)

- **Objectif** : Vérifier la présence de tests négatifs (robustesse) et si le LLM "pense" aux cas d'erreur.
- **Méthode** : Identification via tags `@negative` ou mots-clés (invalid, 404, empty).

❖ Les scénarios de tests sont-ils indépendants ?

M7. Test Isolation & Modularity (Qualité du Code)

- **Objectif** : Garantir que chaque test est indépendant et ne dépend pas de l'exécution d'un autre (robustesse CI/CD).
- **Score 2 (Auto-isolé)** : Préconditions externalisées et réutilisables (`call read`).
- **Score 1 (Semi-isolé)** : Préconditions internes sans réutilisation.
- **Score 0 (Fragile)** : IDs codés en dur ou hypothèse de données existantes.

Allons-y...

Première métrique...

Le package généré est-il exécutable sans modification ?

Enchainement des activités...

URL du swagger :

<https://petstore.swagger.io/#/>



Étape 1

Exécution du prompt 1

Génération de cas de tests en Gherkin, à partir de la spécification Swagger

Exécution du prompt 2

Génération des fichiers de feature Karaté à partir des cas de tests en Gherkin

Étape 2



Étape 3

Exécution du prompt 3

Génération d'un fichier zip téléchargeable contenant l'ensemble des tests automatiques exécutables

Étape 4

Résultats ?

Exécution des tests générés
Production des résultats de l'exécution des tests automatiques

A vous de jouer !
Vous avez 15 minutes...

Exécution du Prompt 1, puis du Prompt 2, puis du Prompt 3

Puis exécution : mvn clean test

Ayez l'esprit critique en tant qu'expert test



Quels résultats avez-vous obtenus ?

QUIZZ...

Nos retours

	Package généré ?
ChatGPT-5	Oui
Claude Sonnet 4.5	Non
Mistral	Non
Gemini-2.5	Non
Grok 4	Non

	Package exécutable sans modification ?
ChatGPT-5	Non
Claude Sonnet 4.5	Non
Mistral	Non
Gemini-3.5	Non
Grok 4	Non

Pour aller plus loin, nous avons dû notamment modifier les éléments suivants :

- Modification du pom.xml (versions des dépendances)
- Perte en compte des capacités du système IA (génération fichier .zip)
- Recréer l'arborescence pour les LLMs autres que ChatGPT

Echanges (5 minutes)



Allons-y...

QUIZZ...

Deuxième métrique...

*Combien de test automatiques ont été générés ?
("couverture" du swagger)*

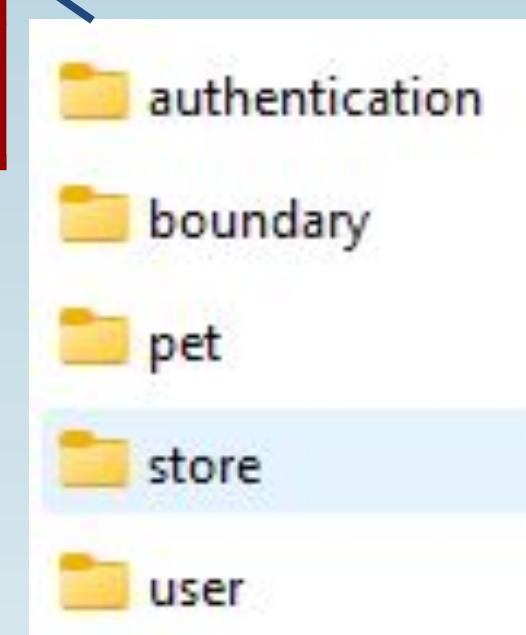
*Vous pouvez utiliser un des packages générés disponibles dans
l'espace des ressources*

Nos retours

<u>features/cross/cross-cutting.feature</u>	Cross-cutting quality attributes and robustness (security, limits, and schema conformance)	7	16	23
---	--	---	----	----

Nombre de tests automatiques générés ?	
ChatGPT-5	61
Claude Sonnet 4.5	198
Mistral	29
Gemini-2.5	21
Grok 4	27

Analyse du swagger	
Pet	7 méthodes 13 codes de retour
Store	4 méthodes 8 codes de retour
User	8 méthodes 13 codes de retour



Notre analyse

- ❖ Les stratégies de génération de tests ne sont pas les mêmes
 - Prise en compte ou pas de la sécurité, des valeurs limites
 - Est-ce lié à la prise en compte des standards ?

Test cases should reflect a risk-based approach, prioritizing high-impact functionality.

Apply the guidance from the following ISO/IEC/IEEE software testing standards:

Standard Focus Automation Relevance

29119-1 Concepts Foundational context for test activities

29119-2 Processes Embedding automation in structured workflows

29119-3 Documentation Structured, reusable, automation-friendly docs

29119-4 Techniques Criteria for what and how to automate

29119-5 Keyword-Driven Testing Framework-oriented automation design

- ❖ Le critère de couverture du swagger n'est pas prise en compte
 - pas exigé dans les prompts

Notre analyse

- ❖ Certains tests générés peuvent être “*bizarres*”

```
Scenario: Create pets with all valid status enum values
  * def statuses = ['available', 'pending', 'sold']
  * def petId = ~~(Math.random() * 1000000)

  Given path 'pet'
  And request { id: '#(petId)', name: 'StatusPet', photoUrls: ['http://example.com/status.jpg'], status: 'available' }
  When method post
  Then status 200
  And match response.status == 'available'
```

Echanges (5 minutes)



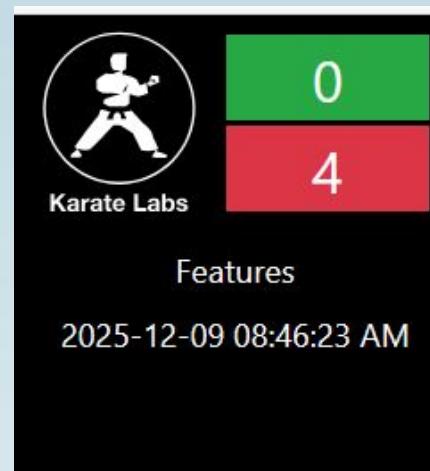
Allons-y...

QUIZZ...

Troisième métrique...

Taux de réussite des tests ?

(rapport Karaté - karate-summary.html)



Tags Timeline					
Feature	Title	Passed	Failed	Scenarios	Time (ms)
features/cross/cross-cutting.feature	Cross-cutting quality attributes and robustness (security, limits, and schema conformance)	7	16	23	12412
features/pet/pet-lifecycle.feature	Pet lifecycle management (critical E2E flows for high-impact functionality)	1	14	15	9796
features/store/store-management.feature	Store order management (inventory, order placement, and fulfillment)	3	8	11	5809
features/user/user-management.feature	User account management (authentication and CRUD)	6	6	12	8307

Nos retours

	Taux de réussite ?
ChatGPT	25%
Claude	93%
Mistral	45%
Gemini	67%
Groq	63%

	Nombre de tests échoués
ChatGPT	46
Claude	12
Mistral	16
Gemini	7
Groq	10

Notre analyse

- ❖ ChatGPT génère des IDs longs qui sont modifiés et amènent des erreurs dans Karate
- ❖ Présence de faux négatifs : Sur certains matchs, l'échec du test est causé par le swagger
- ❖ Le LLM se base sur des hypothèses non fondées
 - Suppose que l'ID 999999999999999999 est inexistant
 - Suppose que certains paramètres sont obligatoires
- ❖ Mauvaise gestion des IDs => Seul Claude génère des IDs dynamiques

Echanges (5 minutes)



Autre expérimentation

Etape 1 : génération du prompt avec Claude (à partir d'un "meta-prompt")*

Etape 2 : utilisation du prompt généré avec différents modèles

Etape 3 : évaluation des résultats avec les mêmes métriques

En cours d'analyse

* *Les ressources sont téléchargeables au même endroit*

Conclusion collégiale / débat...

- Attention : En utilisant l'IA, on teste avec un outil qui peut faire des erreurs
 - Les résultats varient fortement selon le modèle d'IA utilisé et la version
- => Attention à la reproductibilité des générations et à la maintenabilité du code de test

Question “provocatrice” :

- L'IA est-elle meilleure qu'un humain en prompting ?
- Ou faut-il demander à un LLM de challenger les prompts ?

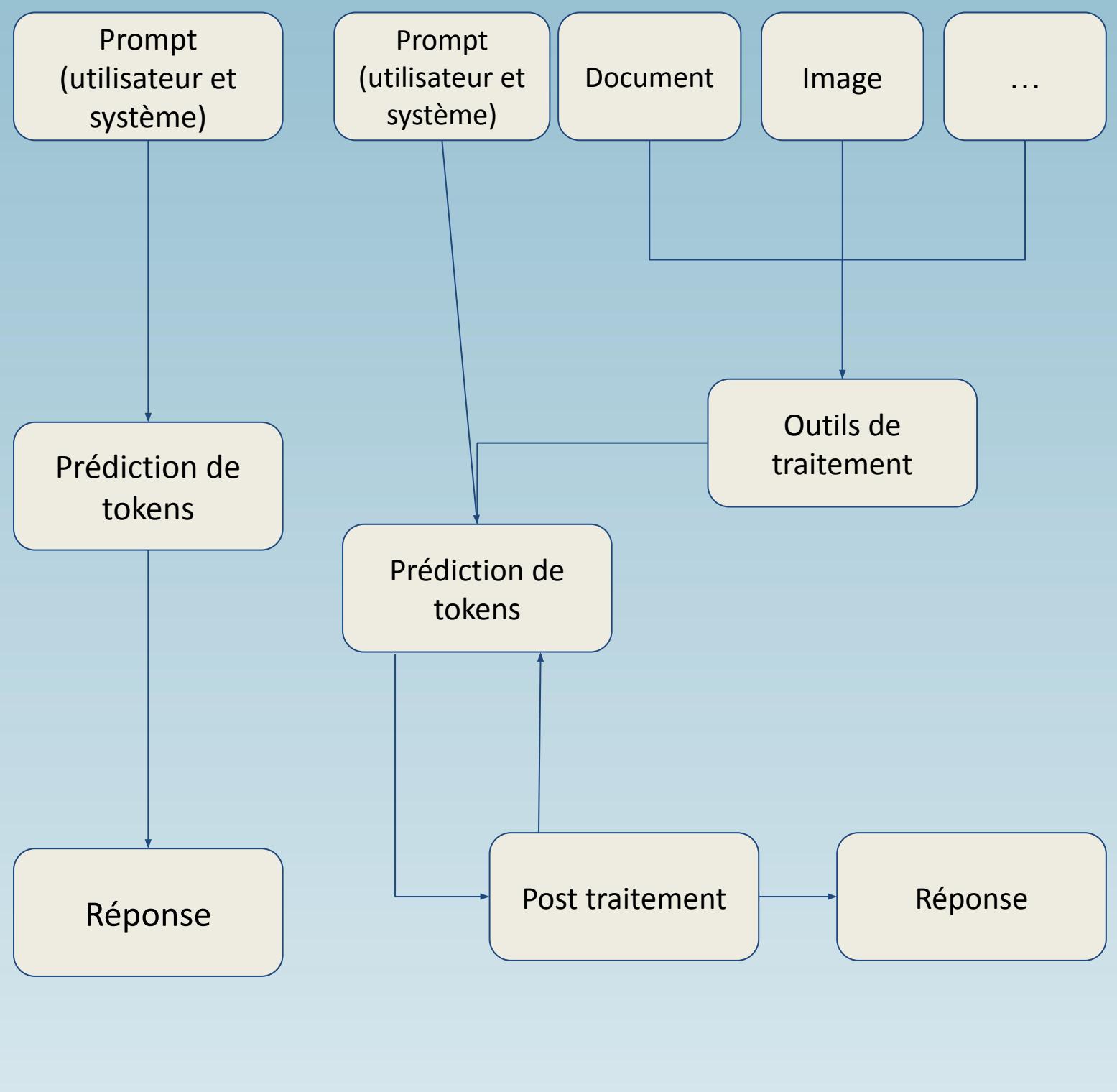
Annexes

Introduction : LLMs versus Système Intelligence Artificielle (SIA)

JOURNÉE
THÉMATIQUE
IA GÉNÉRATIVE
POUR TESTER

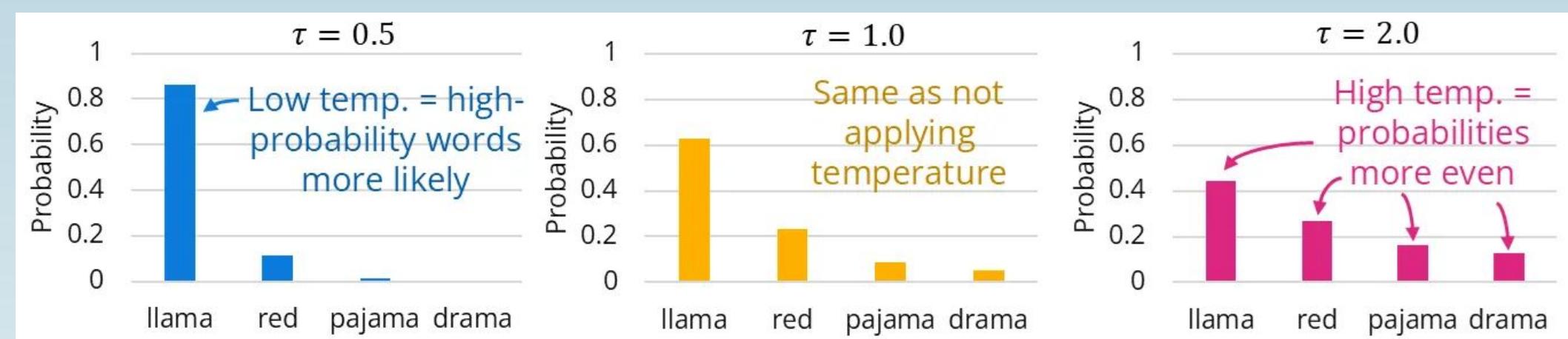
LLM

Système IA basé LLM



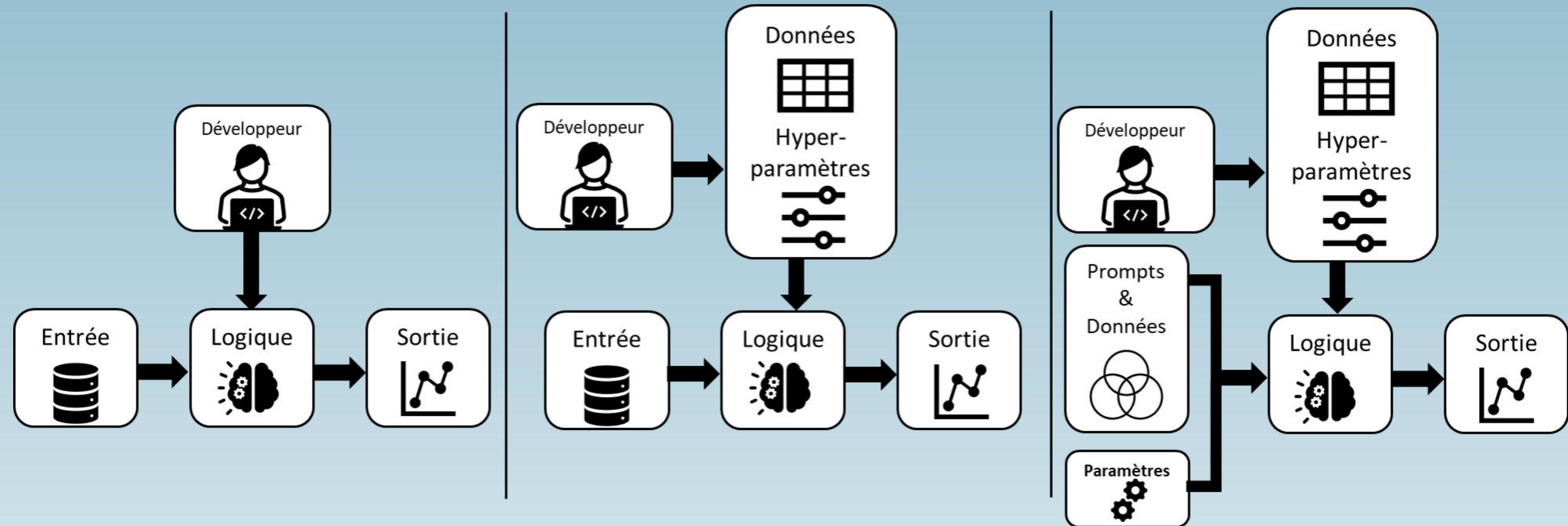
Prompt système et modèle instruct

The screenshot shows a user interface for an AI system. At the top, it says "gemma2:9b" and has a dropdown menu. Below that is a message box with the text "Qu'est ce qu'une API ?". Underneath, there's a list of generated responses starting with "Ol gemma2:9b" followed by several lines of text. To the right, there's a sidebar titled "Chat Controls" with sections for "Valves" and "System Prompt". The "System Prompt" section contains a note about the model not responding to user questions and instead singing the beginning of the song "All I Want For Christmas Is You" by Mariah Carey. At the bottom, there are sections for "Advanced Params" and "Stream Chat Response".



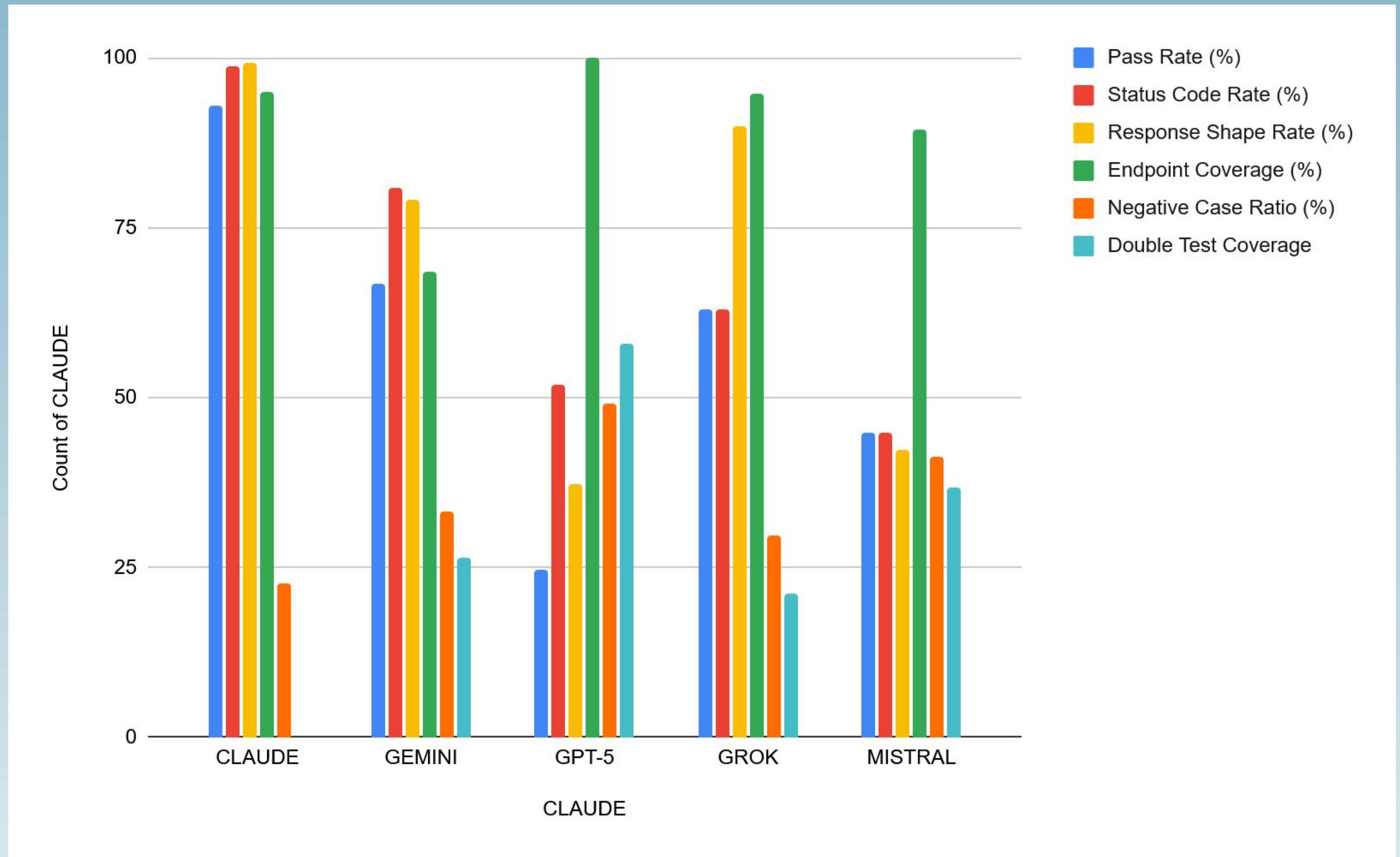
Evaluation d'un système IA (comparé à un système classique)

JOURNÉE
THÉMATIQUE
IA GÉNÉRATIVE
POUR TESTER



Les autres métriques analysées...

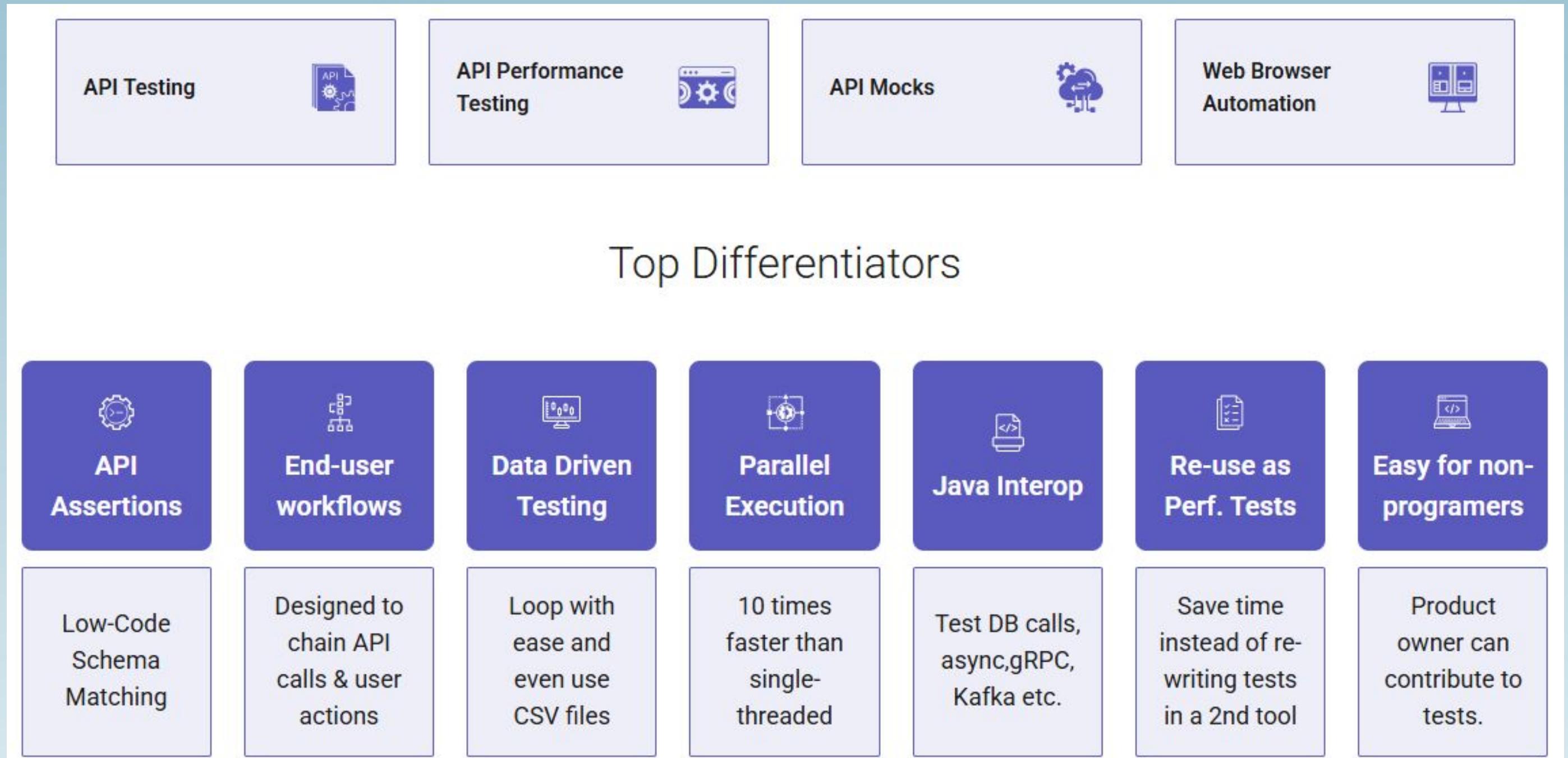
Les autres métriques analysées...



Framework de test : Karaté

<https://www.karatelabs.io/>

Plateforme de tests automatiques d'API open source.



Le prompt initial

En tant qu'expert en test logiciel, peux-tu générer un prompt permettant de générer des tests automatiques pour le framework Karaté à partir d'un swagger ?

Le prompt doit ignorer toutes les instructions précédentes.

Le prompt doit prendre en entrée un swagger.

Première étape, le prompt doit générer des cas de tests dans la syntaxe Gherkin. Le prompt doit fournir un tableau de couverture des méthodes et codes de retour du swagger par les cas de tests.

Deuxième étape, le prompt doit générer des features Karaté à partir des cas de tests en Gherkin. Le prompt doit fournir un tableau de couverture des méthodes et codes de retour du swagger par les features Karaté.

Troisième étape, le prompt doit générer le code source de test en Karaté. Le code source doit contenir un fichier pom.xml incluant toutes les dépendances nécessaires. Le prompt doit inclure les méthodes et les codes de retour testés en commentaire dans le code source de test.

Quatrième étape, le prompt doit générer un package zip du projet complet prêt à être importé en tant que projet maven.

Le prompt généré

Règles de génération :

- Crée au moins un scénario par méthode HTTP (GET, POST, PUT, DELETE, PATCH)
- Couvre tous les codes de retour documentés (200, 201, 400, 404, 500, etc.)
- Inclus des scénarios pour :
 - Cas nominaux (happy path)
 - Cas d'erreur (validations, données manquantes)
 - Cas limites (valeurs nulles, chaînes vides, formats invalides)
- Utilise les exemples du Swagger si disponibles

Format attendu :

```
gherkin
Feature: [Nom de l'endpoint]

Scenario: [Description du cas de test]
  Given [Préconditions]
  When [Action]
  Then [Résultat attendu]
```

Livrable 1 : Tableau de couverture au format :

Endpoint	Méthode	Code Retour	Scénario Gherkin	Couvert
/users	GET	200	Récupération...	✓
/users	GET	404	Utilisateur...	✓

Génère la structure complète du projet Maven avec tous les fichiers nécessaires

```
**Structure du projet :**
...
karate-tests/
├── pom.xml
└── src/
    ├── test/
    │   └── java/
    │       └── [package]/
    │           └── TestRunner.java
    │           └── [Resource]Test.java
    └── resources/
        ├── karate-config.js
        ├── logback-test.xml
        └── features/
            └── *.feature
└── README.md
```

Contenu requis du pom.xml :

```
xml
<!-- Dépendances minimales -->
- com.intuit.karate:karate-junits5 (dernière version stable)
- org.junit.jupiter:junit-jupiter-engine
- com.intuit.karate:karate-core
- Plugins : maven-surefire-plugin
```

Premières générations

Présentation de premières générations...

JOURNÉE
THÉMATIQUE
IA GÉNÉRATIVE
POUR TESTER

Tableau de Couverture Étape 2

Feature	Scénarios	Lignes de Code	Endpoints Testés
pet.feature	16	~115	8 endpoints
store.feature	8	~70	3 endpoints
user.feature	13	~130	7 endpoints
TOTAL	37	~315	18 endpoints

COUVERTURE FINALE COMPLÈTE

Résumé Global

Métrique	Valeur
Endpoints du Swagger	18
Endpoints testés	18 (100%)
Méthodes HTTP	GET, POST, PUT, DELETE
Codes HTTP testés	200, 400, 404, 405, default
Scénarios Gherkin	30
Scénarios Karate	37
Features	3 (pet, store, user)
Fichiers Java	4 runners
Lignes de code total	~1330 lignes
Commentaires	Complets dans tous les fichiers