CAHIER DE CONCEPTION

THEME: CONCEPTION ET REALISATION DE BOOT INSPECTOR, UN OUTIL D'ANALYSE DU DEMARRAGE D'UNE DISTRIBUTION LINUX: Cas de Manjaro.

Rédigé par:

- TCHINDA METEWOUE Florelle Ecladore
- OMGBA ESSENGUE Patrice

ENCADRANT: M. NGUIMBUS Emmanuel

Master 1 Sécurité des Systèmes d'Information 2025/2026

PLAN :

1. INTRODUCTION
2. CONTEXTE DU PROJET ET OBJECTIF
3. DIAGRAMME DE SÉQUENCE
4. DIAGRAMME DE CLASSE
5. DIAGRAMME DE COMPOSANTS
6. DIAGRAMME DE DÉPLOIEMENT
7. CONCLUSION

INTRODUCTION

Chaque projet informatique nécessite une phase d'analyse suivi d'une étape de conception. Pour la conception de notre application, nous avons utilisé une modélisation UML. Ce cahier de conception a pour objectif de décrire en détail la manière dont notre projet sera concu.

FR

CONTEXTE ET OBJECTIFS :

À une époque où la performance et la fiabilité des serveurs et des postes de travail Linux sont cruciales, un temps de démarrage lent peut entraîner une perte de productivité et des retards dans la disponibilité des services critiques. Le diagnostic des goulots d'étranglement du démarrage est un processus complexe qui oblige les administrateurs système et les développeurs à collecter manuellement des données à partir de divers outils (systemd-analyze, journalctl), puis à corréler et interpréter ces informations textuelles pour identifier les problèmes.

Boot Inspector a été créé pour répondre à ce défi. C'est une plateforme web intégrée qui simplifie radicalement l'optimisation du démarrage des systèmes Linux. En centralisant la collecte et l'analyse des données de démarrage et en y ajoutant un moteur de règles internes pour générer des recommandations pertinentes basées sur des seuils prédéfinis, l'application transforme une tâche ardue et technique en un processus rapide, visuel et accessible.

FR

OBJECTIFS

1. Compréhension et visibilité du démarrage
2. Diagnostic rapide et automatisé
3. Optimisation des performances
4. Amélioration de l’expérience utilisateur par
   une automatisation complète du processus de collecte de données et
   une interface intuitive

FR

DIAGRAMME DE SÉQUENCE

Les diagrammes de séquences permettent de représenter les interactions entre les objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois des messages entre objets.

Diagramme de sequence :

Analyser les données de démarrage

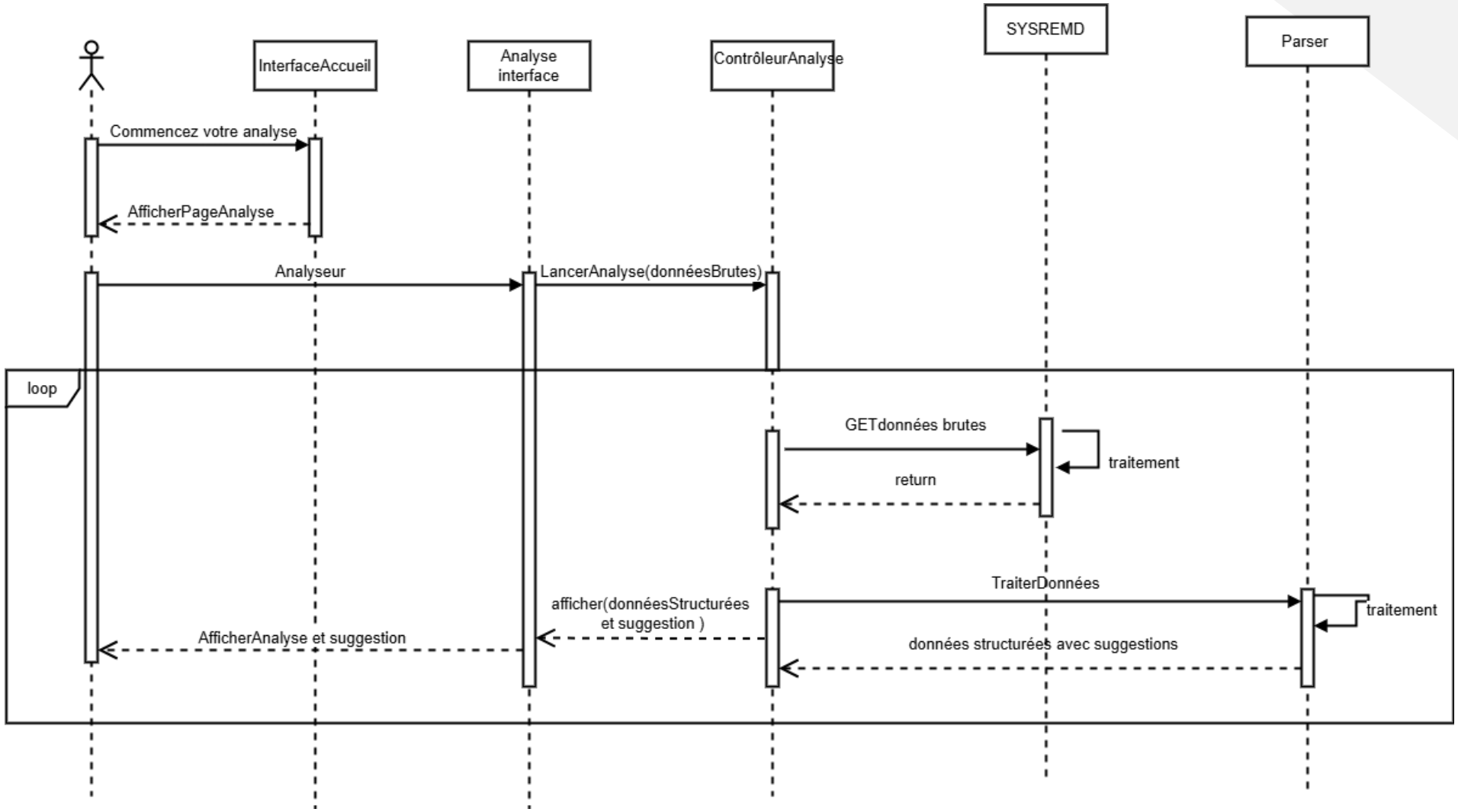


Diagramme de sequence :
Consulter l'historique des analyses

FR

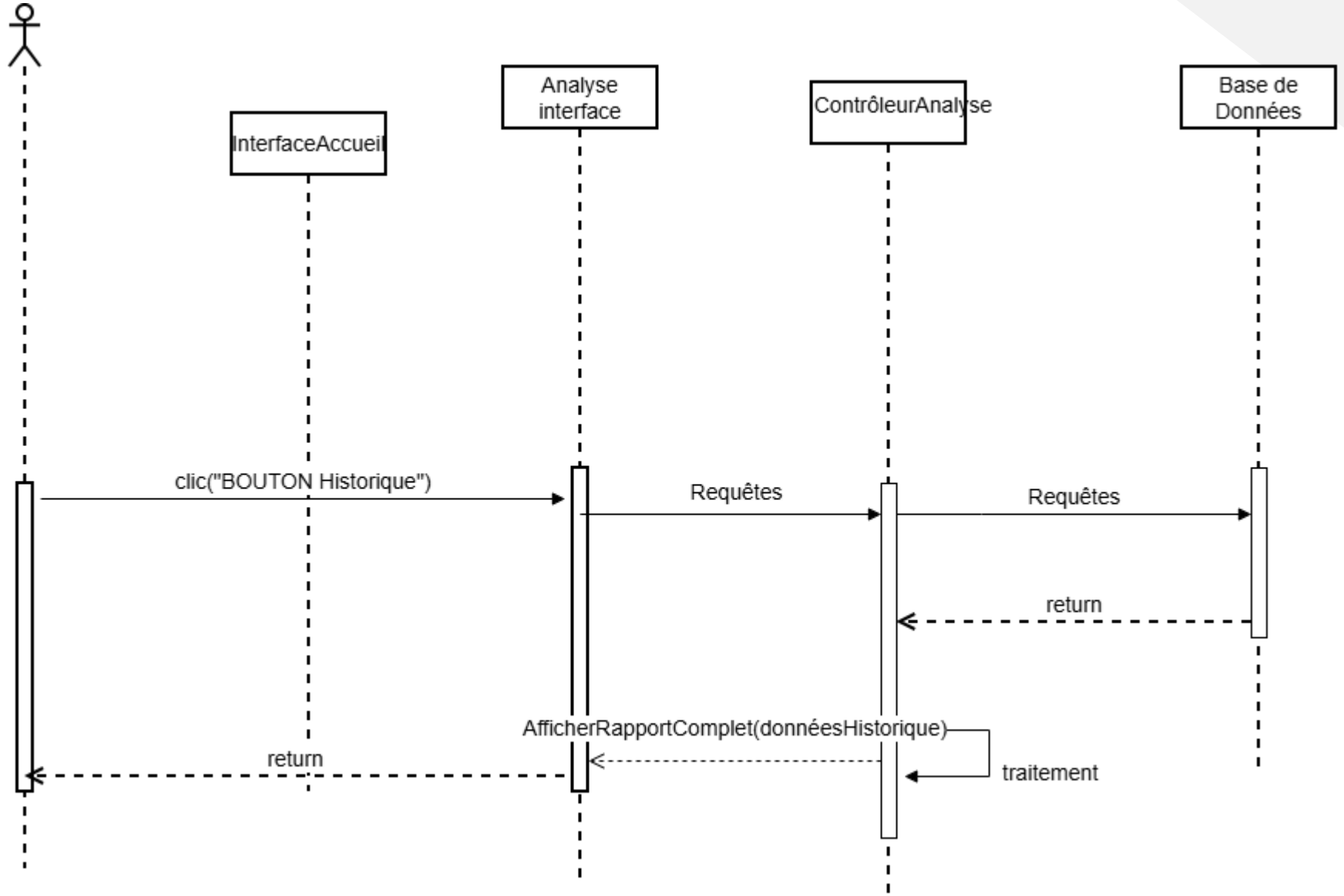
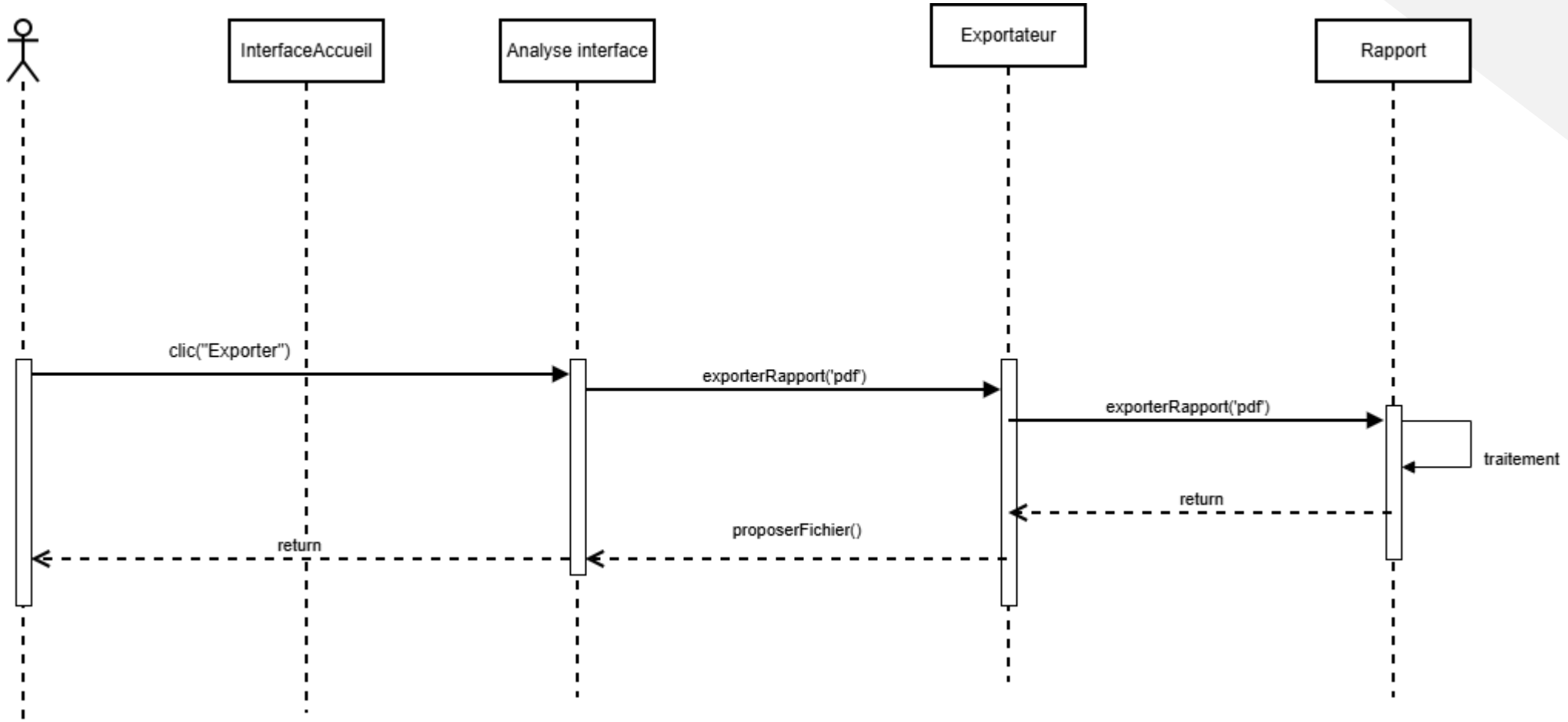


DIAGRAMME DE SEQUENCE :
Exporter un rapport d'analyse

FR



FR

DIAGRAMME DE CLASSE

Le diagramme de classe est le point central dans le développement orienté objet, il représente la structure du système sous forme de classes et les relations entre elles. Les classes constituent la base pour la génération de code et la génération des schémas des bases de données.

DIAGRAMME DE CLASSE

FR

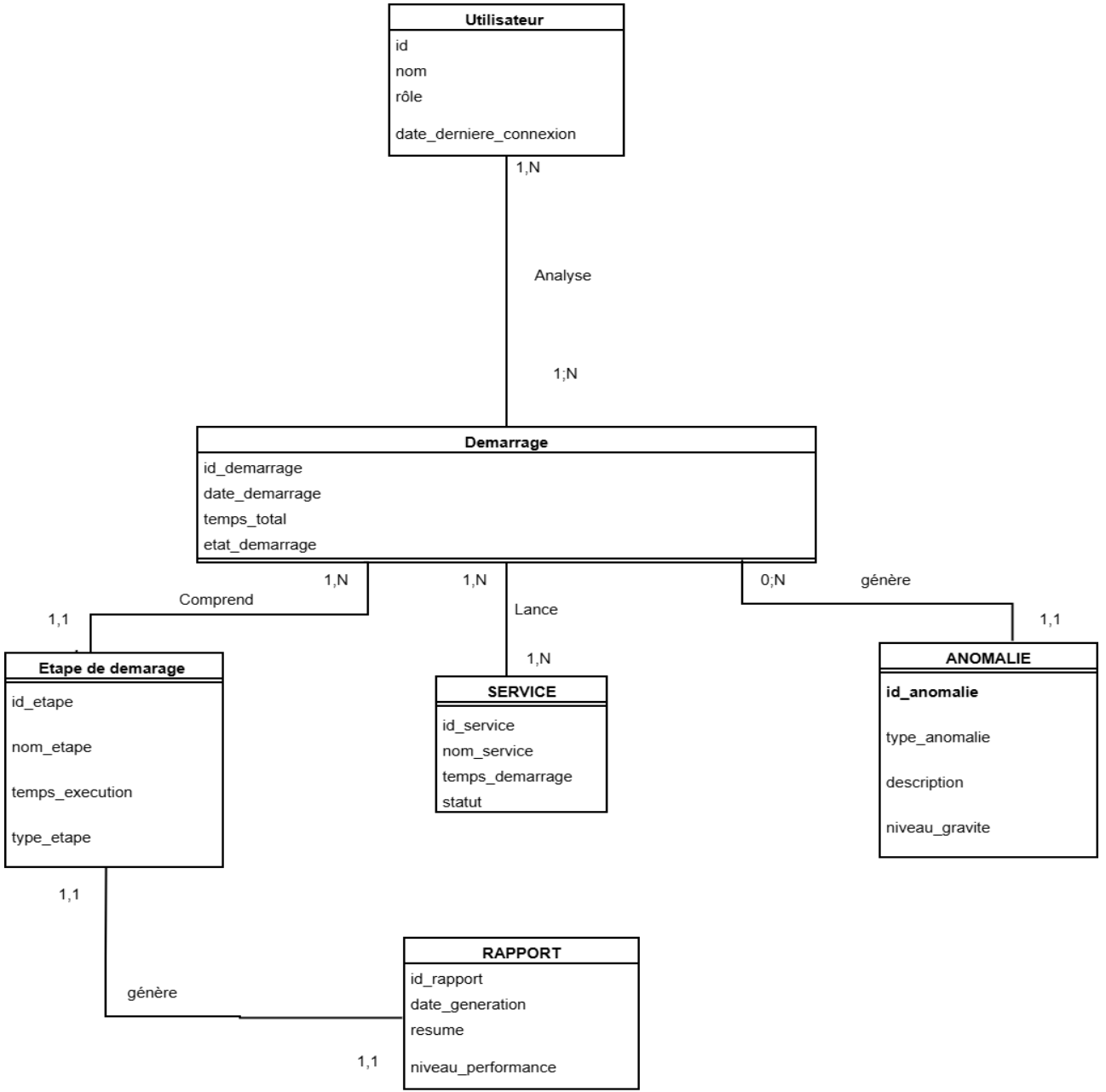
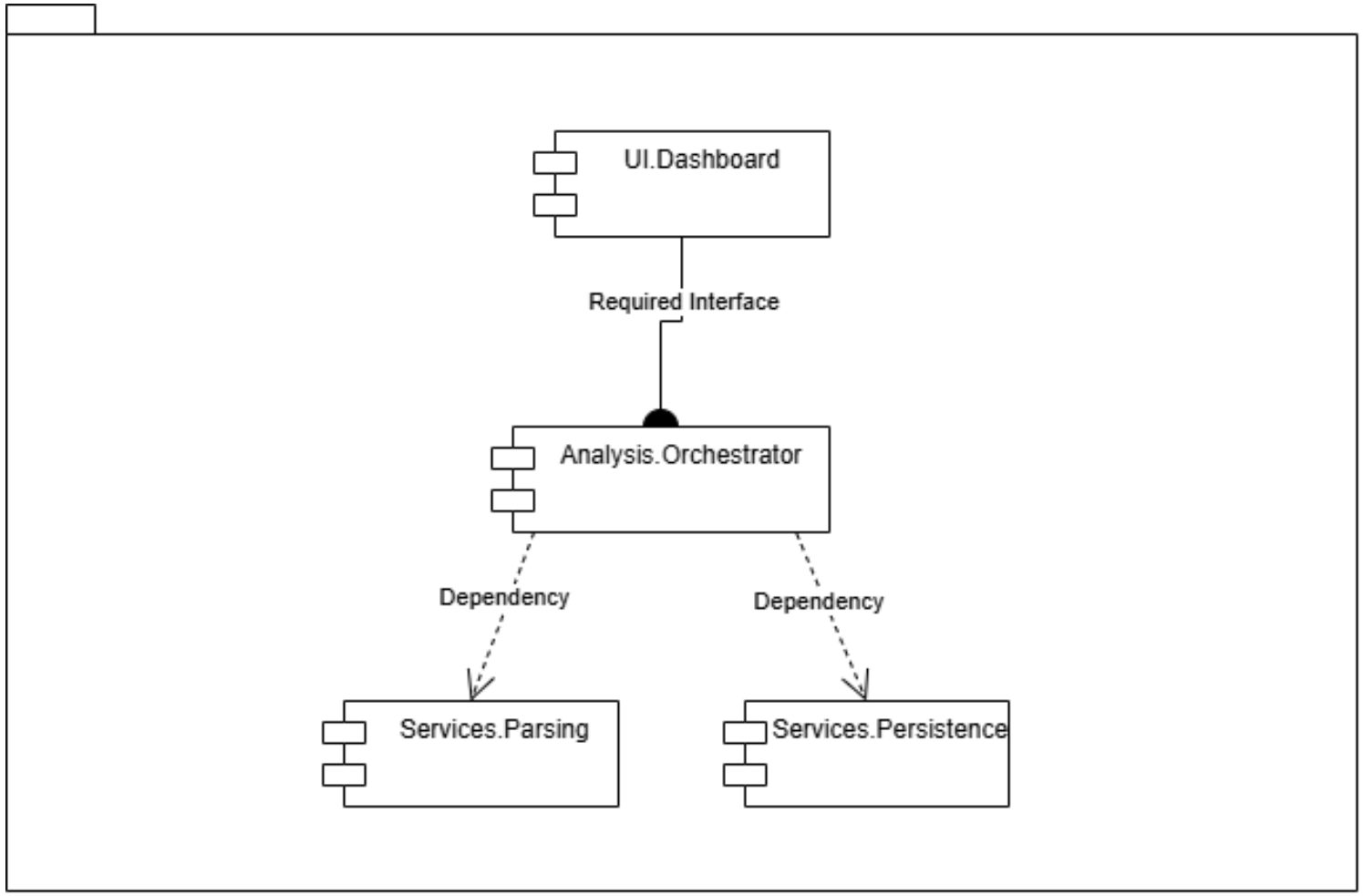


DIAGRAMME DE COMPOSANTS

FR

Le diagramme de composants est le point central dans la conception de l'architecture logicielle. Il représente la structure du système sous forme de blocs modulaires et remplaçables (les composants) ainsi que les relations et les dépendances entre eux. Les composants constituent la base pour l'organisation du code en modules indépendants, la gestion des interfaces et la planification du déploiement physique du système.

DIAGRAMME DE COMPOSANTS



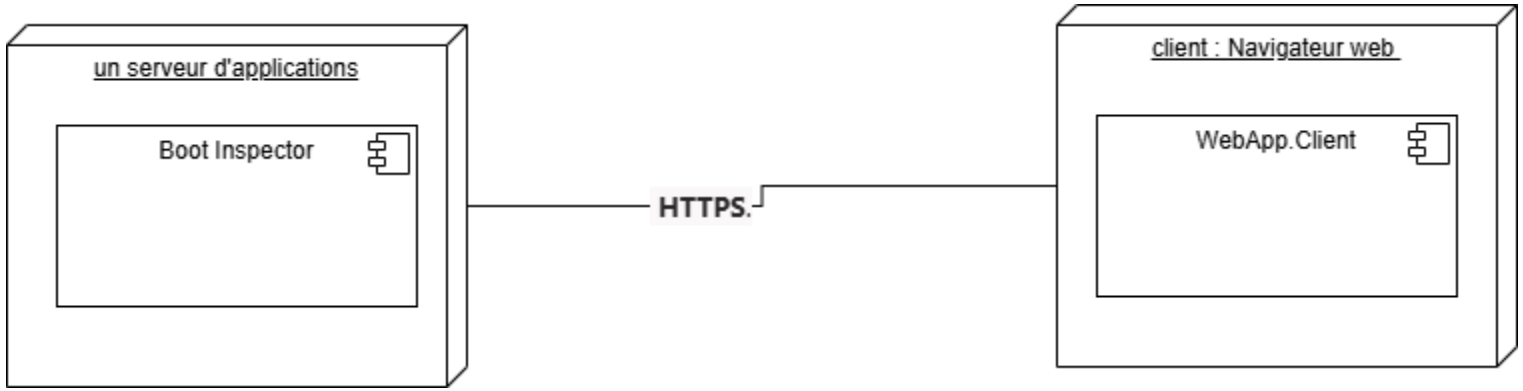
FR

DIAGRAMME DE DEPLOIEMENT

Le diagramme de déploiement est le point central dans la modélisation de l'**architecture physique** du système. Il représente la topologie matérielle et logicielle sur laquelle les artefacts logiciels sont exécutés, sous forme de nœuds (serveurs, appareils clients) et des chemins de communication entre eux. Le diagramme de déploiement constitue la base pour la planification de l'infrastructure, la distribution des composants logiciels et la compréhension de la configuration d'exécution du système final.

FR

DIAGRAMME DE DEPLOIEMENT



CONCLUSION

Ce cahier de conception a présenté l'architecture et le fonctionnement du projet *Boot Inspector* à travers une modélisation UML complète. Les différents diagrammes ont permis de décrire clairement les interactions, la structure du système et son déploiement. Cette conception fournit un cadre fiable pour la phase de réalisation et assure la cohérence, la performance et l'évolutivité de l'application.