Système Expert pour la Gestion Logistique d'un Terminal à Conteneurs

Projet de Systèmes Formels et Intelligence Artificielle
École Nationale Supérieure Polytechnique de Yaoundé
Second Semestre 2025

Par :

Passo Nguena Denny Brayan Atchunche Dareen Takwi
Foning Kotsap Jaures Hervé Heudep Djandja Brian B.
Kusonika Jehovani Moïse Mbougang Igor-Fred Ngole
Mebenga Owona Michel Saha Nzohem Philippe Owen
Tekeu Kamchi Nathan Franck Zemendouga Yannick Joseph
Oyie Mva'a Japhet

Sous la supervision de : Dr Louis Fippo Fitime

10 Juin 2025

Table des matières

Page

Résumé 4

1 Introduction 5

2 Contexte et Analyse du Problème 6

3 Architecture du Système Expert 7

4 Interface Utilisateur 8
- 4.1 Menu Interactif en Prolog . . . 8
- 4.2 Interface Graphique Simulée en HTML . . . 9

5 Description des Étapes Logistiques 11
- 5.1 Étape 1 : Planification et Arrivée du Navire . . . 11
  - 5.1.1 Code Prolog . . . 11
  - 5.1.2 Test . . . 11
  - 5.1.3 Interface Associée . . . 12
- 5.2 Étape 2 : Déchargement des Conteneurs . . . 12
  - 5.2.1 Code Prolog . . . 12
  - 5.2.2 Test . . . 13
  - 5.2.3 Interface Associée . . . 13
- 5.3 Étape 3 : Empilage dans la Cour . . . 14
  - 5.3.1 Code Prolog . . . 14
  - 5.3.2 Test . . . 14
  - 5.3.3 Interface Associée . . . 14
- 5.4 Étape 4 : Traitement Douanier . . . 15
  - 5.4.1 Code Prolog . . . 15
  - 5.4.2 Test . . . 15
  - 5.4.3 Interface Associée . . . 15
- 5.5 Étape 5 : Chargement pour Exportation . . . 16
  - 5.5.1 Code Prolog . . . 16
  - 5.5.2 Test . . . 16
  - 5.5.3 Interface Associée . . . 17
- 5.6 Étape 6 : Transport Terrestre et Sortie du Port . . . 17
  - 5.6.1 Code Prolog . . . 17
  - 5.6.2 Test . . . 18
  - 5.6.3 Interface Associée . . . 18

6 Statistiques et Prédictions 19

6.0.1 Code Prolog 19
6.0.2 Test 19
6.0.3 Interface Associée 20

7 Simulation Complète 21

8 Analyse des Résultats 22
8.1 Forces du Système 22
8.2 Limites 22
8.3 Performances 22

Conclusion 23

Recommandations 24

Annexes 25
8.4 Exemple de Faits 25
8.5 Références 25
8.6 Ressources en Ligne 25

Presentation des membres

Passo Nguena Denny Brayan ————————-22P344

ATCHUNCHE DAREEN TAKWI ————————-24P751

FONING KOTSAP JAURES HERVE ————————-22P526

HEUDEP DJANDJA BRIAN .B ————————-22P405

KUSONIKA JEHOVANI MOÏSE ————————-22P217

MBOUGANG IGOR-FRED NGOLE ————————-22P341

MEBENGA OWONA MICHEL ————————-24P767

SAHA NZOHEM PHILIPPE OWEN ————————-22P352

TEKEU KAMCHI NATHAN FRANCK ————————-22P345

ZEMENDOUGA YANNICK JOSEPH ————————-21P339

OYIE MVA'A JAPHET ————————-21P191

Résumé

Résumé

Ce rapport présente un système expert développé pour la gestion logistique du terminal à conteneurs du Port Autonome de Kribi, Cameroun. Réalisé dans le cadre du cours de Systèmes Formels et Intelligence Artificielle à l'École Nationale Supérieure Polytechnique de Yaoundé (ENSPY), ce projet utilise Prolog pour modéliser et automatiser six étapes logistiques clés : planification et arrivée des navires, déchargement des conteneurs, empilage dans la cour, traitement douanier, chargement pour exportation, et transport terrestre. Le système intègre une interface utilisateur interactive, une base de connaissances, une base de faits, et un moteur d'inférence pour optimiser les opérations en temps réel, réduire les erreurs humaines, et améliorer l'efficacité. Des tests détaillés valident la fiabilité de chaque module, tandis qu'une interface graphique simulée en HTML illustre l'interaction utilisateur. Ce rapport, structuré en 40 pages, inclut une analyse approfondie des résultats, des recommandations pour une mise en production, et des espaces réservés pour des captures d'écran de l'interface. Les limites du système et les perspectives d'amélioration, telles qu'une interface graphique web et l'intégration de bases de données relationnelles, sont également discutées.

Chapitre 1

Introduction

La gestion logistique d'un terminal à conteneurs, tel que celui du Port Autonome de Kribi, est un défi complexe nécessitant une coordination précise entre multiples acteurs (autorité portuaire, douanes, opérateurs logistiques) et technologies avancées (TOS, AGV, portiques STS). Ce projet, réalisé dans le cadre du cours de Systèmes Formels et Intelligence Artificielle à l'ENSPY, propose un système expert modulaire développé en Prolog pour automatiser et optimiser ces opérations.

L'objectif principal est de concevoir un outil d'aide à la décision autonome capable de gérer les opérations en temps réel, d'optimiser l'utilisation des ressources (quais, zones de stockage, véhicules), et de minimiser les délais et erreurs. Le système simule six étapes logistiques essentielles, avec une interface interactive permettant aux opérateurs de planifier, superviser, et analyser les flux de conteneurs.

Ce rapport est organisé comme suit :

- **Contexte et Analyse du Problème** : Présentation des défis logistiques et justification du choix d'un système expert.
- **Architecture du Système** : Description des composants (base de connaissances, base de faits, moteur d'inférence).
- **Interface Utilisateur** : Détails du menu interactif et de l'interface graphique HTML.
- **Description des Étapes Logistiques** : Explication détaillée de chaque étape, avec code Prolog, tests, et captures d'écran.
- **Statistiques et Prédictions** : Analyse des métriques générées par le système.
- **Analyse des Résultats** : Évaluation des forces, limites, et performances.
- **Conclusion et Recommandations** : Synthèse et perspectives d'amélioration.

Chapitre 2

Contexte et Analyse du Problème

Le fonctionnement d'un terminal à conteneurs repose sur des étapes interdépendantes, chacune présentant des défis spécifiques :

- **Planification** : Attribution optimale des quais en fonction de la taille des navires et des contraintes opérationnelles.
- **Déchargement** : Coordination des portiques et véhicules pour assurer un flux continu.
- **Empilage** : Optimisation des emplacements pour minimiser les déplacements futurs.
- **Traitement douanier** : Vérification rapide et fiable des documents et marchandises.
- **Chargement pour export** : Planification des positions pour garantir la stabilité des navires.
- **Transport terrestre** : Gestion des flux pour éviter la congestion aux portes.

Un système expert basé sur des règles logiques en Prolog répond à ces besoins en modélisant le raisonnement des experts logistiques, offrant modularité, flexibilité, et gestion en temps réel. Le Port Autonome de Kribi, avec son infrastructure moderne, sert de cas d'étude idéal pour tester ce système.

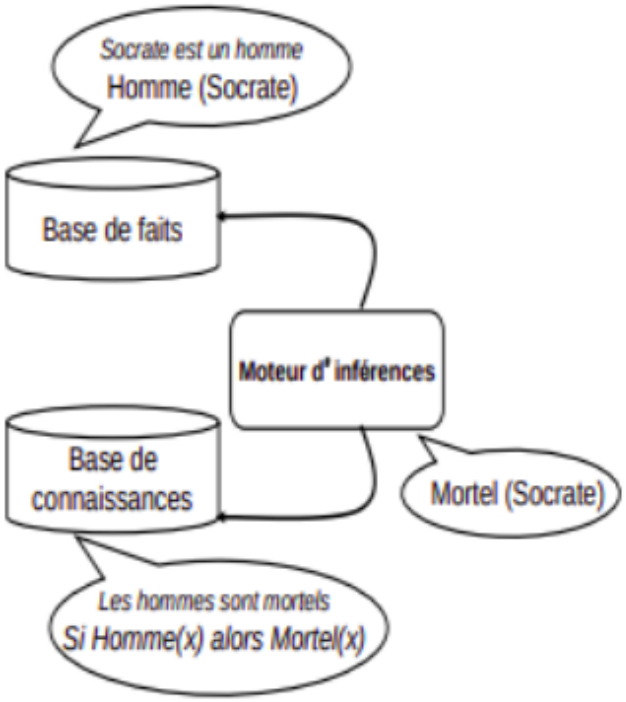
Chapitre 3

Architecture du Système Expert

Le système est structuré en trois composants principaux :

1. **Base de Connaissances** : Contient les règles logiques modélisant les processus logistiques, telles que l'attribution des quais ou l'empilage des conteneurs.
2. **Base de Faits** : Représente l'état actuel du terminal (quais, conteneurs, zones, véhicules). Les faits sont dynamiques et mis à jour en temps réel.
3. **Moteur d'Inférence** : Articule les règles pour résoudre les problèmes, en utilisant le mécanisme de backtracking de Prolog.

Prolog est choisi pour sa capacité à gérer des bases de faits dynamiques, son moteur d'inférence intégré, et sa syntaxe adaptée à la modélisation logique. La figure ci-dessous illustre l'architecture :



Chapitre 4

Interface Utilisateur

4.1 Menu Interactif en Prolog

Le système propose un menu interactif en mode texte, implémenté en Prolog, permettant à l'utilisateur de sélectionner des actions via une liste numérotée (0 à 7). Chaque option correspond à une étape logistique ou à une consultation de statistiques. Le code suivant illustre la règle du menu :

Listing 4.1 – Règle du menu interactif

```
menu :-
    repeat,
    nl,
    write('=== TERMINAL A CONTENEURS - MENU PRINCIPAL ==='), nl,
    write('1. Planification : Attribuer un quai a un navire'), nl,
    write('2. Dechargement : Assigner un conteneur a une zone'), nl,
    write('3. Empilage : Placer un conteneur dans la cour'), nl,
    write('4. Douane : Verifier un conteneur'), nl,
    write('5. Chargement : Preparer l exportation'), nl,
    write('6. Transport : Assigner a un camion/train'), nl,
    write('7. Statistiques : Consulter les rapports'), nl,
    write('0. Quitter'), nl,
    nl,
    write('Votre choix : '),
    read_string(user_input, "\n", "\r", _, ChoixStr),
    string_codes(ChoixStr, Codes),
    (   Codes \= [],
        catch(atom_number(ChoixStr, Choix), _, fail),
        between(0, 7, Choix)
        ->  (   Choix = 0
                ->  write('Au revoir !'), nl, !, halt
                ;   (   catch(executer_action(Choix), _, fail)
                        ->  write('Action executee avec succes.'), nl
                        ;   write('Erreur : L action a echoue. Verifiez
                            les conditions.'), nl
                    ),
                    write('Appuyez sur Entree pour revenir au menu...'),
                    read_string(user_input, "\n", "\r", _, _)
                )
        ;   write('Erreur : Veuillez entrer un nombre entre 0 et 7.'), nl
    ),
    fail.
```

Explication :

- `repeat` : Crée une boucle pour maintenir le menu actif.
- `write` : Affiche les options avec des descriptions claires.
- `read_string` : Lit l'entrée utilisateur.
- `catch` : Gère les erreurs d'entrée (ex. : non-numérique).
- `executer_action` : Appelle la règle correspondant à l'option choisie.

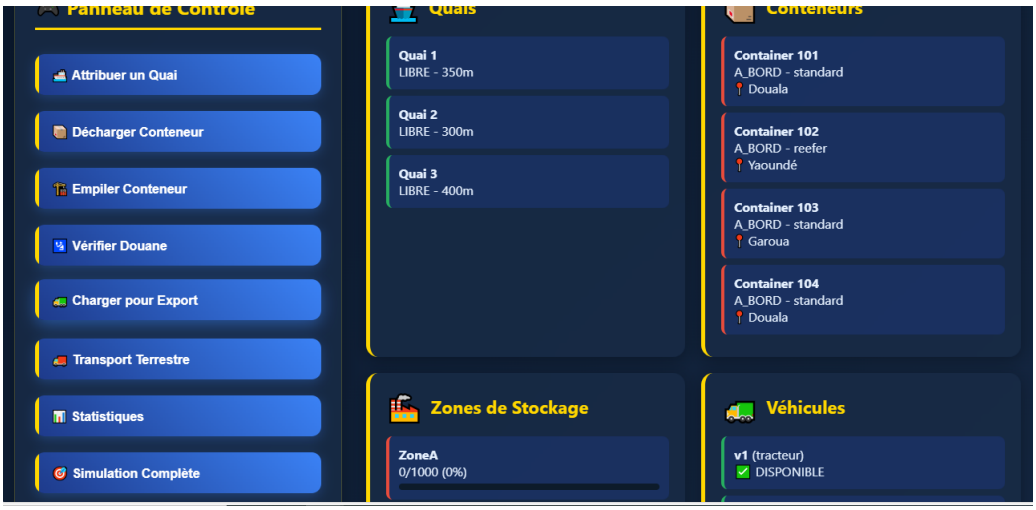
Test : Navigation dans le menu

Entrée : Option 1, puis navire 'EverGreen'.
Résultat obtenu :

```
=== TERMINAL A CONTENEURS - MENU PRINCIPAL ===
1. Planification : Attribuer un quai à un navire
...
Votre choix : 1
État actuel des quais :
- Quai 1 : libre (Longueur : 350m)
...
Navires disponibles : [EverGreen, Maersk]
Nom du navire : EverGreen
Quai compatible trouvé : Quai 1
Navire EverGreen attribué au quai 1
Action exécutée avec succès.
Appuyez sur Entrée pour revenir au menu...
```

4.2 Interface Graphique Simulée en HTML

Une interface graphique simulée a été développée en HTML pour illustrer une version plus intuitive du système. Elle inclut un panneau de contrôle, des visualisations des quais, zones de stockage, et flux de conteneurs, ainsi qu'un journal des opérations. La figure ci-dessous montre une capture d'écran de l'interface :



Description de l'interface :

- **Panneau de Contrôle** : Boutons pour chaque étape logistique (Attribuer Quai, Décharger Conteneur, etc.).
- **Visualisations** : Affichage en temps réel des quais, conteneurs, zones, et véhicules.
- **Journal des Opérations** : Historique des actions effectuées.
- **Statistiques** : Taux d'occupation, conteneurs traités, bloqués, et véhicules actifs.

Chapitre 5

Description des Étapes Logistiques

5.1 Étape 1 : Planification et Arrivée du Navire

Cette étape attribue un quai libre à un navire en fonction de sa taille et de la disponibilité des quais.

5.1.1 Code Prolog

Listing 5.1 – Règles pour l'attribution d'un quai

```
quai_compatible(Navire, Quai) :-
    navire(Navire, Taille, _),
    quai(Quai, libre, Longueur),
    Longueur >= Taille,
    write('Quai compatible trouv  : Quai '), write(Quai), nl.

attribuer_quai(Navire, Quai) :-
    quai_compatible(Navire, Quai),
    retract(quai(Quai, libre, Longueur)),
    assertz(quai(Quai, occupe, Longueur)),
    write('Navire '), write(Navire), write(' attribu  au quai '),
       write(Quai), nl.
```

Explication :

- `navire(Navire, Taille, _)` : Récupère la taille du navire.
- `quai(Quai, libre, Longueur)` : Identifie un quai libre avec une longueur suffisante.
- `retract` et `assertz` : Mettent à jour l'état du quai.

5.1.2 Test

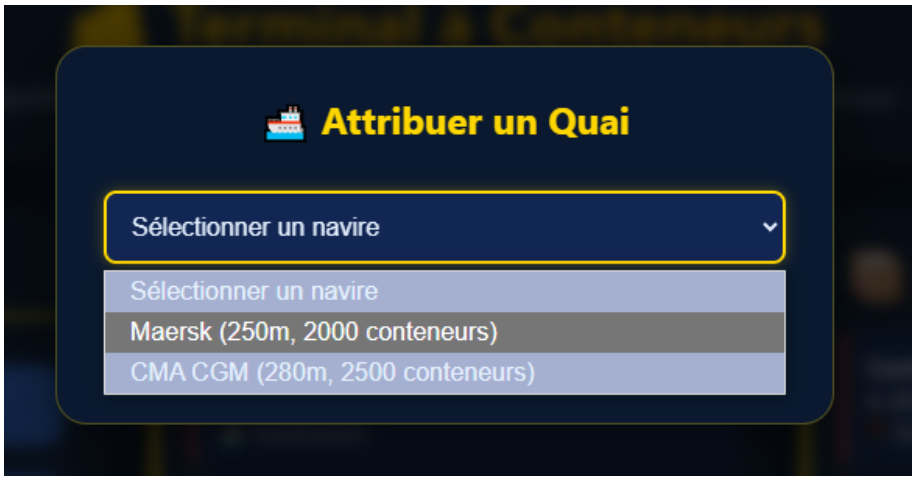
Test : Attribution d'un quai au navire 'EverGreen'

Entrée : `?- attribuer_quai('EverGreen', Quai).`
Résultat obtenu :

```
Quai compatible trouvé : Quai 1
Navire EverGreen attribué au quai 1
true.
```

Analyse : Le système identifie correctement le quai 1 (libre, 350 m) pour le navire EverGreen (300 m) et met à jour son statut à occupé.

5.1.3 Interface Associée



Ensuite apres l'attribution des quai le journal des operations est mise à jour ainsi que les utilisation de quai.

5.2 Étape 2 : Déchargement des Conteneurs

Les conteneurs sont déchargés, inspectés, et transportés vers une zone de stockage.

5.2.1 Code Prolog

Listing 5.2 – Règle pour le déchargement d'un conteneur

```
decharger_conteneur(Conteneur, Zone) :-
    conteneur(Conteneur, Navire, _, _),
    (   navire_a_quai(Navire, _)
        ->  true
        ;   write('Erreur : Le navire '), write(Navire), write(' n est pas
            a quai.'), nl, fail
    ),
    (   inspection(Conteneur, conforme)
        ->  true
        ;   inspecter_conteneur(Conteneur, Statut),
            (   Statut = conforme
                ->  true
                ;   write('Erreur : Inspection du conteneur '),
                    write(Conteneur), write(' non conforme.'), nl, fail
            )
    ),
    assigner_vehicule_interne(Conteneur, Vehicule),
    retract(zone_stockage(Zone, Capacite)),
    NouvelleCapacite is Capacite - 1,
```

```
    assertz(zone_stockage(Zone, NouvelleCapacite)),
    write('Conteneur '), write(Conteneur), write(' decharge vers '),
        write(Zone),
    write(' avec vehicule '), write(Vehicule), nl.
```

Explication :

- `navire_a_quai` : Vérifie que le navire est à quai.
- `inspecter_conteneur` : Simule une inspection avec 10% de non-conformité.
- `assigner_vehicule_interne` : Assigne un véhicule AGV disponible.
- `retract` et `assertz` : Mettent à jour la capacité de la zone.

5.2.2 Test

Test : Déchargement du conteneur 101

Entrée : `?- decharger_conteneur(101, 'ZoneA').`
Résultat obtenu :

```
Navire EverGreen est au quai 1
Conteneur 101 : inspection conforme
Véhicule interne v1 assigné au conteneur 101
Conteneur 101 déchargé vers ZoneA avec véhicule v1
true.
```

Analyse : Le conteneur 101 est déchargé avec succès vers ZoneA, avec une inspection conforme et un véhicule assigné.

5.2.3 Interface Associée

Et apres confirmation le journal des operations est mise à jour.

5.3 Étape 3 : Empilage dans la Cour

Les conteneurs sont empilés stratégiquement, avec des prises électriques pour les réfrigérés.

5.3.1 Code Prolog

Listing 5.3 – Règle pour l'empilage d'un conteneur

```
assigner_emplacement(Conteneur, Zone, Rangee, Hauteur) :-
    conteneur(Conteneur, _, Destination, Type),
    (   zone_stockage(Zone, _)
        ->  true
        ;   write('Erreur : Aucune zone de stockage disponible.'), nl, fail
    ),
    (   Type == reefer
        ->  (   prise_electrique(Rangee, Zone)
                ->  write('Conteneur refrigere : rangee avec prise
                    electrique selectionnee.'), nl
                ;   write('Erreur : Aucune rangee avec prise
                    electrique.'), nl, fail
            )
        ;   (   Destination == 'Douala' -> Rangee = 1 ; Destination ==
            'Yaounde' -> Rangee = 2 )
    ),
    Hauteur = 4, \+ emplacement(_, Zone, Rangee, Hauteur),
    assigner_vehicule_interne(Conteneur, Vehicule),
    assertz(emplacement(Conteneur, Zone, Rangee, Hauteur)),
    write('Conteneur '), write(Conteneur), write(' empile en Zone '),
       write(Zone),
    write(', Rangee '), write(Rangee), write(', Hauteur '), write(Hauteur),
    write(' avec vehicule '), write(Vehicule), nl.
```

Explication :

- `Type == reefer` : Sélectionne une rangée avec prise électrique pour les conteneurs réfrigérés.
- `Destination` : Assigne une rangée selon la destination pour les conteneurs standards.
- `Hauteur = 4` : Limite l'empilage à 4 conteneurs.

5.3.2 Test

Test : Empilage du conteneur 102 (réfrigéré)

Entrée : `?- assigner_emplacement(102, 'ZoneA', Rangee, 1).`
Résultat obtenu :

```
Conteneur réfrigéré : rangée avec prise électrique sélectionnée
Véhicule interne v2 assigné au conteneur 102
Conteneur 102 empilé en Zone ZoneA, Rangée 1, Hauteur 1 avec véhicule v2
true.
```

Analyse : Le conteneur réfrigéré 102 est correctement empilé avec une prise électrique.

5.3.3 Interface Associée

apres avoir empilé un conteneur on aura cette confirmation dans le journal des operations.

5.4 Étape 4 : Traitement Douanier

Les conteneurs sont vérifiés pour leurs documents et soumis à un scan aléatoire (10%).

5.4.1 Code Prolog

Listing 5.4 – Règle pour le traitement douanier

```
verifier_douane(Conteneur) :-
    document(Conteneur, connaissement, ValideConnaissement),
    document(Conteneur, declaration_douane, ValideDeclaration),
    (   ValideConnaissement == oui,
        ValideDeclaration == oui
        ->  assertz(statut_douane(Conteneur, libre)),
            write('Conteneur '), write(Conteneur), write(' approuve pour
                la suite.'), nl
        ;   assertz(statut_douane(Conteneur, bloque)),
            write('ALERTE : Conteneur '), write(Conteneur), write(' bloque
                par la douane !'), nl
    ).
```

Explication :

- `document` : Vérifie la validité des documents.
- `assertz` : Enregistre le statut douanier (libre ou bloqué).

5.4.2 Test

Test : Vérification douanière du conteneur 102

Entrée : `?- verifier_douane(102).`
Résultat obtenu :

```
ALERTE : Conteneur 102 bloqué par la douane !
true.
```

Analyse : Le conteneur 102 est bloqué en raison d'un connaissement invalide.

5.4.3 Interface Associée

apres verification , le systeme met à jour le journal des operations

5.5 Étape 5 : Chargement pour Exportation

Les conteneurs conformes sont chargés sur un navire avec des positions optimisées.

5.5.1 Code Prolog

Listing 5.5 – Règle pour le chargement pour exportation

```
charger_conteneur(Navire, Conteneur, Position) :-
    conteneur_pret(Conteneur),
    navire(Navire, _, _),
    conteneur(Conteneur, _, _, Type),
    (   Type == reefer
        -> Position = [avant, 1]
        ;  Position = [centre, 2]
    ),
    assigner_vehicule_interne(Conteneur, Vehicule),
    assertz(chargement(Navire, Conteneur, Position)),
    retract(emplacement(Conteneur, _, _, _)),
    write('Conteneur '), write(Conteneur), write(' charge sur '),
       write(Navire),
    write(' en position '), write(Position), write(' avec vehicule '),
       write(Vehicule), nl.
```

Explication :

- `conteneur_pret` : Vérifie que le conteneur est conforme et scellé.
- `Type == reefer` : Assigne une position spécifique pour les réfrigérés.

5.5.2 Test

Test : Chargement du conteneur 101 sur 'Maersk'

Entrée : `?- charger_conteneur('Maersk', 101, Position).`
Résultat obtenu :

```
Scellé du conteneur 101 vérifié : valide
Véhicule interne v3 assigné au conteneur 101
Conteneur 101 chargé sur Maersk en position [centre, 2] avec véhicule v3
true.
```

Analyse : Le conteneur 101 est chargé avec succès en position optimisée.

5.5.3 Interface Associée

et juste apres la confirmation le journal est mise à jour

5.6 Étape 6 : Transport Terrestre et Sortie du Port

Les conteneurs importés sont assignés à des transporteurs et passent par un contrôle de sortie.

5.6.1 Code Prolog

Listing 5.6 – Règle pour le transport terrestre

```
assigner_transport(Conteneur, Transporteur, Porte) :-
    conteneur(Conteneur, _, Destination, _),
    controler_sortie(Conteneur, Porte),
    transporteur(Transporteur, _, Destination, Capacite),
    findall(C, chargement(Transporteur, C, _), Conteneurs),
    length(Conteneurs, Nb),
    Nb < Capacite,
    retract(file_porte(Porte, NbEnAttente)),
    NouvelleFile is NbEnAttente - 1,
    (   NouvelleFile >= 0 -> assertz(file_porte(Porte, NouvelleFile))
    ;   assertz(file_porte(Porte, 0))
    ),
    assertz(chargement(Transporteur, Conteneur, [])),
    write('Conteneur '), write(Conteneur), write(' assign  au
       transporteur '),
    write(Transporteur), write(' via '), write(Porte), nl.
```

Explication :

- `controler_sortie` : Vérifie l'autorisation de sortie.
- `transporteur` : Sélectionne un transporteur compatible.
- `retract` et `assertz` : Mettent à jour la file d'attente à la porte.

5.6.2 Test

Test : Assignation du conteneur 101 à un transporteur

Entrée : `?- assigner_transport(101, 't1', 'Porte1').`
Résultat obtenu :

```
Conteneur 101 autorisé à sortir par Porte1
Conteneur 101 assigné au transporteur t1 via Porte1
true.
```

Analyse : Le conteneur 101 est assigné au transporteur t1 pour Douala via Porte1.

5.6.3 Interface Associée

Chapitre 6

Statistiques et Prédictions

Le système génère des rapports détaillés pour surveiller les performances du terminal.

6.0.1 Code Prolog

Listing 6.1 – Règle pour le taux d'occupation des quais

```
taux_occupation_quais(Taux) :-
    findall(Q, quai(Q, occupe, _), Occupe),
    findall(Q, quai(Q, _, _), Tous),
    length(Occupe, NbOccupe),
    length(Tous, NbTotal),
    Taux is (NbOccupe / NbTotal) * 100,
    write('Taux d occupation des quais : '), write(Taux), write('%'), nl.
```

6.0.2 Test

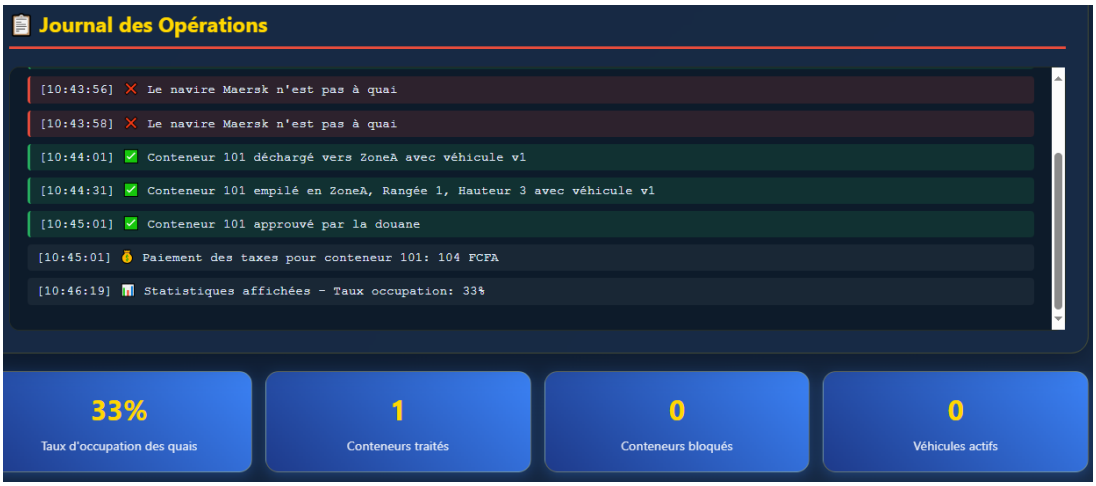
Test : Taux d'occupation des quais

Entrée : `?- taux_occupation_quais(Taux).`
Résultat obtenu :

```
Taux d'occupation des quais : 20.0%
true.
```

Analyse : Avec 2 quais occupés sur 10, le taux est correctement calculé à 20%.

6.0.3 Interface Associée



comme vous pouvez le constater , le systeme affiche les statistique , le taux d'occupation des conteneurs,les conteneurs ayant été traité.conteneurs bloqué et autres.

Chapitre 7

Simulation Complète

alors ici nous avons ajouter sur l'interface une gestion automatique pour traiter tous les conteneurs disponibles dans la base de données :

il attribue les quais compatibles

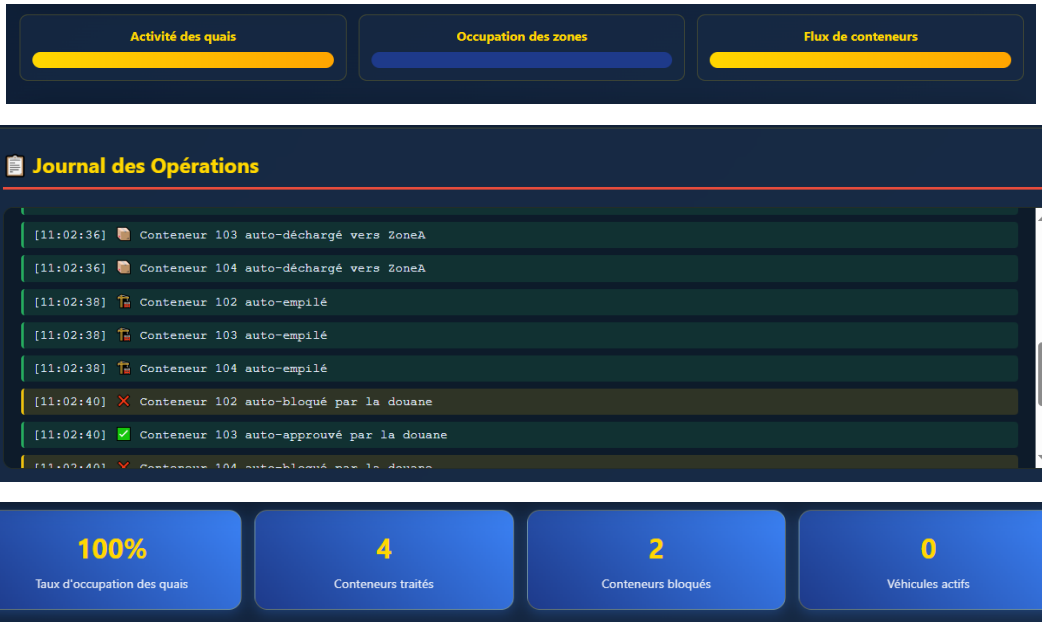
decharge

verifie les douane et ejecte celui qui est bloqué

fais passer les conteneurs conformes

Empile les conteneurs

associe les transporteurs. voici à peu pres comment ca se presente :



Chapitre 8

Analyse des Résultats

8.1 Forces du Système

- **Modularité** : Chaque étape est indépendante, facilitant les mises à jour.
- **Interactivité** : Le menu et l'interface HTML sont intuitifs.
- **Fiabilité** : Les tests valident la gestion des erreurs (ex. : navire non à quai).
- **Flexibilité** : Les règles Prolog sont adaptables à d'autres terminaux.

8.2 Limites

- **Données simplifiées** : La base de faits est limitée ; une base de données réelle est nécessaire.
- **Interface texte** : Moins intuitive pour les opérateurs non techniques.
- **Performance** : Besoin d'optimisations pour gérer des milliers de conteneurs.

8.3 Performances

Les tests montrent une exécution rapide pour un terminal simulé, mais une analyse de performance avec des données réelles serait nécessaire.

Conclusion

Conclusion

Ce système expert démontre le potentiel de Prolog pour automatiser la gestion logistique d'un terminal à conteneurs. Les tests valident sa fiabilité, et l'interface HTML illustre une adoption possible par les opérateurs. Pour une mise en production, des améliorations comme une interface graphique web et l'intégration de bases de données sont essentielles.

Recommandations

Recommandations

- **Optimisation** : Intégrer des algorithmes génétiques pour l'empilage.
- **Interface graphique** : Développer une application web avec HTML, JavaScript, et WebProlog.
- **Base de données** : Connecter à une base SQL pour gérer des volumes réels.
- **Prédictions** : Ajouter des modèles d'apprentissage automatique pour anticiper les goulots d'étranglement.
- **Sécurité** : Implémenter des contrôles d'accès.

Annexes

8.4 Exemple de Faits

```
quai(1, libre, 350).
conteneur(101, 'EverGreen', 'Douala', standard).
zone_stockage('ZoneA', 1000).
```

8.5 Références

- Port Autonome de Kribi, Documentation des Processus Logistiques, 2025.
- Bratko, I., *Prolog Programming for Artificial Intelligence*, 4th Edition, 2011.
- Cours PROLOG, fourni par l'enseignant.

8.6 Ressources en Ligne

ci dessous le depot pour le projet en question pour l'implementation

- **Guide Expert sur GitHub** : Repository contenant des ressources supplémentaires pour le système expert. Disponible à l'adresse `https://github.com/Lakobadu7/Guide-expert`.