HETIC — Web3

CAHIER DES CHARGES TECHNIQUE

Robot d'Assistance Autonome (RAA)

Prototype MVP pour ERP — Bibliothèque municipale / Centre administratif

Matériel	Yahboom Transbot (Jetson Nano + Lidar + RealSense + Bras Robotique)
Équipe	5 étudiants — 2 Back-end · 2 IoT/Robotique · 1 Front-end
Version	1.1 — Intégrant les retours de révision
Date	Février 2026
Auteur	CTO — Équipe RAA

1. Vision du Projet

1.1 Phrase de Vision

RAA (Robot d'Assistance Autonome) est un robot mobile autonome qui transporte des objets d'un point A à un point B au sein d'un bâtiment, pour le personnel d'une bibliothèque municipale ou d'un centre administratif, afin de réduire les déplacements répétitifs, d'améliorer la productivité opérationnelle et de démontrer la viabilité d'une solution robotique low-cost dans un environnement ERP réel.

1.2 Contexte et Motivation

Les environnements administratifs et bibliothécaires génèrent de nombreux déplacements internes répétitifs : transport de documents, de colis internes, de matériels d'un bureau à un autre. Ces tâches consomment du temps agent sans valeur ajoutée.

Le projet RAA vise à prototyper une réponse robotique accessible, construite sur du matériel grand public (Yahboom Transbot) et des logiciels open-source (ROS 2), dans le cadre d'un ERP d'entreprise. Ce MVP est développé en 5 semaines par une équipe de 5 étudiants HETIC.

1.3 Objectifs Stratégiques

- Démontrer la faisabilité technique d'un robot autonome sur du matériel accessible (<500€)
- Valider l'intégration full-stack : interface web ↔ serveur ↔ robot en temps réel
- Produire un MVP documenté, maintenable et extensible pour les promotions suivantes
- Préparer une démo jury convaincante, fiable et répétable

2. Objectifs et Périmètre

2.1 Objectifs Fonctionnels (IN SCOPE)

1. Navigation autonome point A → point B dans un environnement cartographié
2. Évitement d'obstacles statiques et dynamiques via Lidar
3. Pick-up et drop-off d'un objet standardisé via le bras robotique
4. Interface opérateur web pour créer et suivre les missions en temps réel
5. Mise à jour du statut de mission en temps réel (WebSocket)
6. Arrêt d'urgence physique et logiciel
7. Monitoring de la batterie avec seuil d'alerte
8. Interface accessible WCAG AA (navigation clavier, lecteur d'écran)

2.2 Non-Objectifs (OUT OF SCOPE)

> ⛔ Ces éléments sont explicitement hors périmètre pour ce MVP. Toute demande dans ce sens doit être refusée ou reportée à une version future.

- Navigation multi-étages ou ascenseur
- Reconnaissance visuelle d'objets (la saisie est positionnelle/préprogrammée)
- Gestion de plusieurs robots simultanément
- Authentification utilisateur (l'interface est mono-utilisateur pour le MVP)
- Déploiement en production ou en cloud
- Persistance longue durée / analytics des missions
- Application mobile native

2.3 Personas

Persona 1 — L'Opérateur (utilisateur principal)

Nom fictif	Marie, 42 ans, bibliothécaire référente
Contexte	Gère les demandes internes de transport de documents entre bureaux
Objectif	Lancer une mission rapidement sans formation technique
Frustrations	Perd du temps à se déplacer pour des transports mineurs
Besoin clé	Interface simple, statut en temps réel, notifications claires

Persona 2 — Le Jury / Décideur

Profil	Évaluateur technique ou responsable métier
Contexte	Évalue la solidité technique et la pertinence du projet
Objectif	Voir une démo fluide, comprendre l'architecture
Besoin clé	Fiabilité, documentation claire, scénario B de secours

3. Use Cases

3.1 Tableau Récapitulatif des Use Cases

UC #	Nom	Acteur principal	Priorité
UC 1	Créer une mission	Opérateur	🔴 Critique
UC 2	Suivre une mission en temps réel	Opérateur	🔴 Critique
UC 3	Déclencher l'arrêt d'urgence	Opérateur / Système	🔴 Critique
UC 4	Annuler une mission	Opérateur	🟠 Important
UC 5	Consulter l'historique des missions	Opérateur	🟡 Souhaitable
UC 6	Monitorer l'état du robot	Opérateur / Système	🟠 Important
UC 7	Naviguer vers destination	Robot (automatique)	🔴 Critique
UC 8	Pick-up / Drop-off objet	Robot (automatique)	🔴 Critique

3.2 Use Cases Détaillés

UC1 — Créer une Mission

Acteur principal	Opérateur (Marie)
Préconditions	Le robot est en état IDLE, la batterie > 20%, l'interface web est ouverte
Scénario nominal	1. L'opérateur ouvre l'interface web 2. Clique sur « Nouvelle mission » 3. Sélectionne le point d'origine sur la carte SVG 4. Sélectionne le point de destination 5. Décrit l'objet à transporter (champ texte) 6. Clique sur « Lancer la mission » 7. Le système valide et affiche la mission créée avec statut CREATED 8. Le robot reçoit l'assignation et démarre
Extensions (erreurs)	4a. Le point sélectionné est inaccessible → message d'erreur explicite 6a. Le robot est occupé → mission mise en file d'attente, message informatif 6b. Batterie insuffisante → mission refusée, alerte batterie
Postconditions	Une entrée en DB avec statut ASSIGNED, un événement WebSocket mission:created émis, le robot en route

UC3 — Déclencher l'Arrêt d'Urgence

Acteur principal	Opérateur ou Système (safety_monitor)
Préconditions	Une mission est en cours (robot en mouvement)
Scénario nominal — Logiciel	1. L'opérateur clique le bouton « Arrêt d'urgence » dans l'interface 2. Le front-end envoie POST /api/robot/emergency-stop 3. Le serveur relaye la commande au

	robot via WebSocket 4. Le robot stoppe immédiatement tous les moteurs 5. La mission passe en statut EMERGENCY_STOPPED 6. Un toast d'alerte apparaît dans l'interface
Scénario nominal — Physique	1. Une personne appuie sur le bouton rouge physique du Transbot 2. Les moteurs sont coupés électriquement 3. Le safety_monitor détecte l'arrêt et met à jour le statut en DB
Extensions	3a. Le robot est hors Wi-Fi → l'arrêt physique prend le relais (failsafe matériel)
Postconditions	Robot immobile, mission EMERGENCY_STOPPED, redémarrage manuel requis

UC6 — Monitorer l'État du Robot

Acteur principal	Opérateur (passif) / Système (actif)
Préconditions	Le robot est connecté au serveur
Scénario nominal	1. Le robot envoie un heartbeat WebSocket toutes les 5s 2. Le serveur met à jour le statut (batterie, position, état) 3. L'interface affiche en temps réel la position sur la carte et le niveau de batterie 4. Si batterie < 20% → alerte orange dans l'interface 5. Si pas de heartbeat pendant 15s → alerte rouge « Robot déconnecté »
Postconditions	L'opérateur a une vue temps réel de l'état du robot

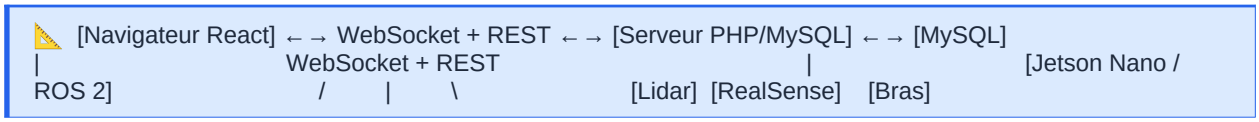
4. Architecture Technique

4.1 Vue d'Ensemble des Couches

L'architecture du RAA repose sur trois couches découplées communiquant par API REST et WebSocket. Ce découplage permet à chaque sous-équipe de travailler en parallèle et facilite le débogage lors de la démo.

Couche	Stack technique	Rôle
Front-end	React 18 + TypeScript + Tailwind	Interface opérateur accessible WCAG AA
Back-end	PHP + MySQL + Socket.io	API REST, gestion missions, temps réel
Robot	ROS 2 (Humble) + Python + Nav2	Navigation autonome, bras, capteurs

4.2 Schéma d'Architecture



4.3 Couche Front-end

Stack : React 18 avec TypeScript, Tailwind CSS, Socket.io-client, React Query, React Router v6.

Composants clés :

- Dashboard missions : liste des missions en cours/terminées avec statut en temps réel
- Formulaire de création de mission : sélection point A / point B sur carte SVG interactive
- Vue live robot : position sur la carte + flux caméra (optionnel MVP)
- Notifications toast : confirmation pick-up, livraison, erreurs

Accessibilité WCAG AA :

- Navigation clavier complète (focus visible, skip links, tabindex logique)
- Aria-live regions pour les mises à jour temps réel
- Contrastes valides (ratio 4.5:1 minimum), labels explicites
- Tests : axe-core en CI, tests manuels NVDA/VoiceOver

4.4 Couche Back-end — API REST

Méthode	Endpoint	Description
POST	/api/missions	Créer une nouvelle mission
GET	/api/missions	Lister toutes les missions
GET	/api/missions/:id	Détail d'une mission
PATCH	/api/missions/:id/status	Mise à jour statut (robot → serveur)
GET	/api/robot/status	État courant du robot (batterie, position)
POST	/api/robot/emergency-stop	Arrêt d'urgence

4.5 Couche Robot — Packages ROS 2

Package ROS 2	Fonction
nav2_bringup	Navigation autonome : planification de trajectoire + évitement d'obstacles
slam_toolbox	Cartographie initiale du lieu (SLAM) via Lidar
robot_localization	Fusion odometry + IMU pour localisation précise
micro_ros_agent	Bridge entre Jetson et microcontrôleurs du Transbot
mission_manager	Node custom : reçoit missions API, orchestre navigation + bras
arm_controller	Node custom : contrôle du bras (pick-up / drop-off)
safety_monitor	Node custom : watchdog, arrêt d'urgence, monitoring batterie

5. Diagrammes UML

📌 Les diagrammes UML ci-dessous sont représentés en notation textuelle (PlantUML). Les fichiers sources .puml sont disponibles dans le dossier docs/uml/ du monorepo. Ils doivent être rendus via PlantUML (https://plantuml.com) ou l'extension VS Code PlantUML.

5.1 Diagramme de Use Case

```
@startuml
left to right direction
actor "Opérateur" as Op
actor "Robot (auto)" as Robot
rectangle RAA {
  (UC1: Créer mission) as UC1
  (UC2: Suivre mission) as UC2
  (UC3: Arrêt d'urgence) as UC3
  (UC4: Annuler mission) as UC4
  (UC6: Monitorer robot) as UC6
  (UC7: Naviguer) as UC7
  (UC8: Pick-up/Drop-off) as UC8
}
Op --> UC1 Op --> UC2 Op --> UC3 Op --> UC4 Op --> UC6
Robot --> UC7 Robot --> UC8 Robot --> UC3
@enduml
```

5.2 Diagramme d'États — Machine à États Mission

```
@startuml
[*] --> CREATED : POST /api/missions
CREATED --> ASSIGNED : Robot disponible
ASSIGNED --> NAVIGATING_TO_PICKUP : Nav2 goal envoyé
NAVIGATING_TO_PICKUP --> PICKING_UP : Arrivée point A
PICKING_UP --> NAVIGATING_TO_DROP : Pick-up confirmé
NAVIGATING_TO_DROP --> DROPPING_OFF : Arrivée point B
DROPPING_OFF --> COMPLETED : Drop-off confirmé
COMPLETED --> [*]
CREATED --> CANCELLED ASSIGNED --> CANCELLED NAVIGATING_TO_PICKUP --> FAILED PICKING_UP
--> FAILED NAVIGATING_TO_DROP --> FAILED
NAVIGATING_TO_PICKUP --> EMERGENCY_STOPPED NAVIGATING_TO_DROP --> EMERGENCY_STOPPED
@enduml
```

5.3 Diagramme de Séquence — Mission Complète

```
@startuml
actor Opérateur participant Front participant Serveur database MySQL participant Robot
Opérateur -> Front: Sélectionne A, B, objet Front -> Serveur: POST /api/missions Serveur
-> MySQL: INSERT mission (CREATED) Serveur -> Front: WS mission:created Serveur -> Robot:
WS mission:assign Robot -> Serveur: WS robot:heartbeat Robot -> Robot: Nav2 goal point A
Robot -> Serveur: WS mission:status NAVIGATING_TO_PICKUP Serveur -> Front: WS
robot:position (500ms) Robot -> Serveur: WS PICKING_UP Robot -> Robot: Bras saisit objet
Robot -> Serveur: WS NAVIGATING_TO_DROP Robot -> Serveur: WS DROPPING_OFF Robot ->
Serveur: WS COMPLETED Serveur -> MySQL: UPDATE status=COMPLETED Serveur -> Front: WS
mission:completed Front -> Opérateur: Toast notification
@enduml
```

6. Stack Technique — Choix et Justifications

💡 Cette section documente les décisions techniques (ADR — Architecture Decision Records). Pour chaque choix, les alternatives considérées et les raisons du rejet sont explicitées.

6.1 Front-end

Technologie	Choix retenu	Alternatives écartées	Justification
Framework UI	React 18 + TypeScript	Vue 3, Svelte, Angular	Maîtrise de l'équipe, écosystème riche, typage fort pour la fiabilité
CSS	Tailwind CSS	CSS Modules, styled-components	Développement rapide, cohérence visuelle, pas de config complexe
WebSocket client	Socket.io-client	ws natif, SockJS	Reconnexion automatique intégrée, compatibilité avec Socket.io serveur
Data fetching	React Query	SWR, Redux Toolkit Query	Cache intelligent, invalidation simple, gestion loading/error native

6.2 Back-end

Technologie	Choix retenu	Alternatives écartées	Justification
Langage	PHP (via Express-like)	Node.js pur, Python/FastAPI, Go	Compétence existante de l'équipe back, hébergement simple
Base de données	MySQL	PostgreSQL, SQLite, MongoDB	Relations simples, pas de JSON complexe, connu de l'équipe. PostgreSQL serait préférable en prod.
WebSocket serveur	Socket.io	ws natif, µWebSockets	Rooms, broadcast, reconnexion côté client, API simple
Tests API	Jest + Supertest	Mocha, Vitest, Postman	Standard de l'écosystème Node/PHP, intégration CI facile

6.3 Robot (IoT)

Technologie	Choix retenu	Alternatives écartées	Justification
Middleware robot	ROS 2 Humble	ROS 1 Noetic, custom firmware	Support long terme (LTS), Nav2 intégré, communauté active, compatible Jetson
Navigation	Nav2 (navigation2)	Move Base (ROS 1), custom PID	Stack de navigation complète avec planification, costmaps, recovery behaviors
Cartographie	slam_toolbox	gmapping, cartographer	Compatible ROS 2, SLAM en ligne et hors-ligne, carte rechargeable
Langage robot	Python 3	C++ ROS 2	Développement plus rapide pour les nodes custom, maîtrise équipe

Technologie	Choix retenu	Alternatives écartées	Justification
			IoT

7. Risques Techniques et Mitigations

Crit.	Risque	Impact	Mitigation
⚠	Wi-Fi instable en démo	Robot déconnecté, mission bloquée	Hotspot dédié · Auto-reconnect · Mode dégradé autonome
⚠	SLAM imprécis	Robot désorieinté, collision	Cartographier le lieu exact · Balises visuelles · Vitesse réduite
⚠	Bras ne saisit pas	Mission incomplète	Objet standardisé · Séquence calibrée · Plan B : plateau fixe
⚠	Latence WebSocket	Position décalée sur carte	Interpolation front · Réduire fréquence · Tester réseau salle
⚠	Batterie insuffisante	Robot s'éteint en mission	Monitoring 20% · Charge complète · Batterie externe
⚠	Sécurité physique	Blessure personne / mobilier	0.3 m/s max · Arrêt urgence · Périmètre délimité · Safety monitor
⚠	Intégration tardive	Pièces incompatibles	API Contract First · Mock serveur · Mock robot · CI dès Sprint 1

7.1 Stratégie Anti-Risque Démo

9. Toujours avoir un scénario de démo B (simplifié) prêt si le scénario A échoue
10. Filmer une vidéo de la démo qui fonctionne la veille en backup
11. Faire 10+ répétitions complètes avant le jour J
12. Apporter votre propre routeur Wi-Fi (pas le Wi-Fi de la salle)
13. Prévoir 15 min de setup avant le passage devant le jury

8. Conventions Équipe

8.1 Conventions Git

Branches

Type	Pattern	Exemple
Main	main	Toujours déployable, protégée
Front-End	feature/front-<ticket>-<slug>	feature/front-RAA-12-dashboard-missions
Back-End	feature/back-<ticket>-<slug>	feature/back-RAA-21-crud-missions
IoT	feature/iot-<ticket>-<slug>	feature/iot-RAA-35-nav2-navigation

Convention de Commit (Conventional Commits)

```
<type>(<scope>): <description courte>

feat(front): add mission creation form
fix(robot): fix websocket reconnection after wifi loss
docs(api): update openapi.yaml with robot endpoints
test(back): add jest tests for mission state machine
```

Types autorisés : feat, fix, docs, test, refactor, chore, style, ci
Scopes autorisés : front, back, robot, api, db, ci, docs

8.2 Processus de Review (Pull Request)

- Toute PR doit cibler develop (jamais main directement)
- Minimum 1 reviewer obligatoire (idéalement 2 si impact cross-équipe)
- La CI doit être verte (lint + tests) avant de merger
- Le reviewer approuve dans les 24h ou commente pour bloquer
- Pas de self-merge, même pour les hotfixes urgents (sauf accord exprès du CTO)
- Les merges vers main sont faits par le CTO uniquement, en fin de sprint

8.3 Conventions de Nommage

Contexte	Convention	Exemple
Variables JS/TS	camelCase	missionStatus, robotPosition
Composants React	PascalCase	MissionCard, RobotStatusBadge
Fichiers composants	PascalCase.tsx	MissionDashboard.tsx
Hooks custom	use + camelCase	useMissionStatus, useRobotPosition
Endpoints API	kebab-case	/api/robot/emergency-stop
Tables MySQL	snake_case	mission_logs, map_points
Nodes ROS 2	snake_case	mission_manager, arm_controller

Contexte	Convention	Exemple
Topics ROS 2	/namespace/topic	/robot/position, /robot/battery
Events WebSocket	namespace:action	mission:updated, robot:obstacle-detected

8.4 Structure Monorepo

```
raa-project/
├─ apps/web/          → Front-end React
├─ apps/server/       → Back-end Express/PHP
├─ apps/robot/        → Packages ROS 2 (Python)
├─ packages/shared/ → Types TypeScript partagés
├─ docs/              → openapi.yaml, ADRs, schémas
│   └─ uml/           → Fichiers .puml PlantUML
├─ scripts/           → Scripts de déploiement, setup
└─ .github/workflows/ → CI/CD
```

9. Roadmap et Questions Ouvertes

9.1 Plan de Développement en 3 Sprints

Sprint 0 — Fondations (Semaine 1)

Objectif : aligner l'équipe, définir les interfaces, configurer les environnements.

Qui	Tâches	Livrable
Tous	Rédiger le contrat API (OpenAPI 3.0)	openapi.yaml
Tous	Configurer Git (monorepo), CI, environnements dev	Repo + CI vert
IoT 1+2	Installer ROS 2 sur Jetson, tester moteurs + Lidar + bras	Robot qui bouge
IoT 1	SLAM : cartographier la pièce de démo	Carte .pgm/.yaml
Back 1+2	Setup Express + MySQL + Socket.io (scaffolding)	Serveur qui répond
Front	Setup React + Tailwind + routing + stubs API	Shell app navigable

Sprint 1 — Intégration Verticale (Semaines 2–3)

Objectif : une mission complète fonctionne de bout en bout (clic → navigation → confirmation).

Qui	Tâches	Livrable
Front	Dashboard missions + formulaire création + carte SVG	UI fonctionnelle
Front	Intégration WebSocket (statut temps réel)	Live updates
Back 1	CRUD missions complet + machine à états	API stable
Back 2	Bridge WebSocket (relay robot ↔ front) + emergency stop	Temps réel OK
IoT 1	Nav2 : navigation point A → B + évitement obstacles	Robot navigue
IoT 2	Client API Python + pick/place basique du bras	Robot connecté

Sprint 2 — Robustesse + Démo (Semaines 4–5)

Objectif : fiabiliser, gérer les cas d'erreur, polir l'expérience pour la démo jury.

Qui	Tâches	Livrable
Front	Audit accessibilité + gestion erreurs + animations statut	WCAG AA validé
Front	Bouton arrêt d'urgence + notifications livraison	UX complète
Back 1	Gestion erreurs + retry + logs structurés	API robuste

Qui	Tâches	Livrable
Back 2	Timeout missions + heartbeat robot + reconnexion auto	Résilience
IoT 1	Tuning Nav2 (vitesse, marges sécurité) + tests répétés	Navigation fiable
IoT 2	Scénario démo complet pick → move → place + safety monitor	Démo prête
Tous	Répétitions démo (minimum 10 runs complets)	Zéro surprise

9.2 Questions Ouvertes

⚠️ Ces questions doivent être résolues avant la fin du Sprint 0. Chaque question a un owner désigné.

#	Question	Owner	Échéance
Q 1	Quel est le lieu de démo exact ? Dimensions, obstacles fixes, luminosité ?	IoT 1	Fin Sprint 0
Q 2	Quel objet standardisé utilisera-t-on pour le pick-up ? (forme, poids, matériau)	IoT 2	Fin Sprint 0
Q 3	Le back-end est-il PHP ou Node.js/Express ? Les deux sont mentionnés dans le doc.	Back 1+2	Jour 1 Sprint 0
Q 4	Quel réseau Wi-Fi sera disponible ? Peut-on apporter notre propre routeur ?	Tous	Fin Sprint 1
Q 5	Les points A et B de la démo sont-ils fixes ou variables ? Combien de waypoints prédéfinis ?	Front + IoT 1	Sprint 0
Q 6	Faut-il une authentification pour l'interface (même basique) ou mono-user suffit ?	Front + Back 1	Sprint 0
Q 7	La caméra RealSense est-elle utilisée en MVP ou uniquement le Lidar pour la nav ?	IoT 1+2	Sprint 0

10. Recommandations CTO et Checklist Démo

10.1 Principes Non Négociables

- Sécurité d'abord : le robot ne doit jamais pouvoir blesser quelqu'un. Vitesse plafonnée, arrêt d'urgence physique + logiciel, périmètre délimité.
- API Contract First : aucune ligne de code avant l'accord sur openapi.yaml. C'est ce qui permet le travail en parallèle.
- Intégration continue : chaque jour, les pièces doivent se connecter. Pas d'intégration big bang la dernière semaine.
- Démo > Features : mieux vaut une démo qui marche à 100% avec 3 features qu'une démo qui plante avec 10 features.

10.2 Outils Recommandés

Catégorie	Outil	Pourquoi
Gestion projet	GitHub Projects (Kanban)	Gratuit, intégré au code, suffisant pour 5 personnes
CI/CD	GitHub Actions	Lint + tests auto sur chaque PR
API Doc	Swagger UI + openapi.yaml	Documentation vivante, testable
Monitoring robot	RViz2 + rqt_graph	Visualiser la nav, les topics ROS en temps réel
Tests API	Jest + Supertest	Tests unitaires + intégration back-end
Tests accessibilité	axe-core + pa11y	Détection automatique des violations WCAG
Diagrammes UML	PlantUML (VS Code ext.)	Diagrammes versionnable en texte dans le repo

10.3 Checklist Démo Jury

Statut	Item à valider
☐	Le robot démarre et se connecte au serveur en < 30 secondes
☐	L'interface web est navigable entièrement au clavier
☐	Une mission complète s'exécute de A à B sans intervention
☐	Le statut se met à jour en temps réel sur l'interface
☐	L'arrêt d'urgence fonctionne (physique + bouton web)
☐	Le robot évite un obstacle placé sur son chemin
☐	La notification de livraison apparaît à l'écran (et est annoncée aria-live)
☐	Le scénario B (dégradé) est prêt et a été testé
☐	La vidéo de backup est filmée (démo qui marche)
☐	Le routeur Wi-Fi dédié est chargé et testé sur place
☐	10+ runs complets effectués sans erreur
☐	Les questions ouvertes (Q1–Q7) sont toutes résolues

HETIC Web3 — Équipe RAA — Février 2026 — Document v1.1