DOCUMENTATION TECHNIQUE

Mission D.1

QuickRepair France

Digitalisation du Système d'Information

Chaîne de réparation multi-boutiques — Île-de-France

Groupe AMOU

Ouafae Berhili & Amal Bouzaher

beouafae@etudiant-esic.fr | biamal@etudiant-esic.fr

CDA Bac+3 — ESIC — Semestre 1 2025-2026

■ Table des matières

1. Architecture de la solution déployée

La solution QuickRepair France repose sur une architecture trois-tiers cloud-native, entièrement hébergée sans infrastructure physique. Chaque couche est découplée et communique via des protocoles standardisés (PostgreSQL wire protocol, HTTPS/REST).

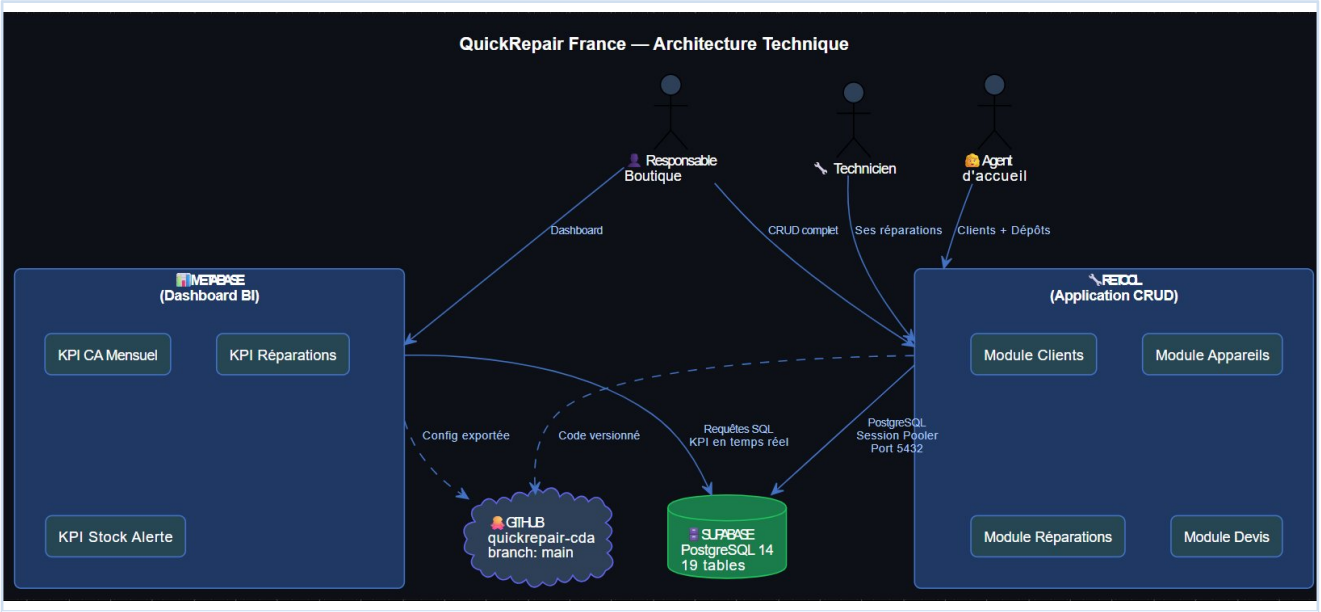

Figure — Architecture technique — QuickRepair France (Retool + Metabase + Supabase + GitHub)

Description des couches

Couche	Technologie	Rôle	Hébergement
Présentation	Retool + Metabase	Interface utilisateur CRUD et dashboard BI	Cloud (SaaS)
Logique métier	Retool (queries)	Requêtes SQL paramétrées, filtrage par rôle	Retool Cloud
Données	Supabase PostgreSQL 14	19 tables, contraintes FK, indexes	eu-west-1
Versioning	GitHub	Commits, branches, historique propre	github.com

Flux de données

- ① L'utilisateur s'authentifie sur Retool ou Metabase (rôle : Responsable / Technicien / Agent)
- ② Retool exécute des requêtes SQL paramétrées sur Supabase via Session Pooler (port 5432, SSL)
- ③ Metabase interroge directement Supabase pour calculer les 6 KPI en temps réel
- ④ Tous les fichiers (SQL, config, documentation) sont versionnés sur GitHub (branche main)

2. Choix technologiques justifiés

Chaque technologie a été sélectionnée après analyse des contraintes du projet : déploiement rapide, coût nul (offres gratuites), compétences disponibles, et adéquation aux besoins fonctionnels d'une PME multi-boutiques.

■■ Supabase (PostgreSQL 14)

Choix retenu	Supabase — PostgreSQL managé cloud
Justification	BDD relationnelle puissante, hébergement cloud gratuit, API REST auto-générée, dashboard visuel, SSL natif
Alternative envisagée	PlanetScale (MySQL) — écarté car syntaxe différente et pas de FK natives
Alternative envisagée	Firebase Firestore — écarté car NoSQL inadapté au modèle relationnel complexe (19 tables)
Avantage clé	Compatible avec Retool et Metabase nativement, région EU disponible (RGPD)

■ Retool (Application CRUD)

Choix retenu	Retool — Low-code pour applications internes
Justification	Connexion PostgreSQL native, composants UI prêts à l'emploi, gestion des rôles intégrée, déploiement instantané
Alternative envisagée	AppSmith — écarté car hébergement self-hosted complexe dans le délai imparti
Alternative envisagée	Développement React custom — écarté car délai trop long pour 3 sprints
Avantage clé	Queries SQL paramétrées permettant le filtrage par rôle (technicien voit ses réparations uniquement)

■ Metabase (Dashboard BI)

Choix retenu	Metabase Cloud — BI self-service
Justification	Interface no-code pour créer des KPI, connexion PostgreSQL directe, visualisations variées (camembert, barres, tableaux)
Alternative envisagée	Power BI — écarté car licence payante et intégration Supabase complexe
Alternative envisagée	Grafana — écarté car orienté monitoring, pas BI métier
Avantage clé	Filtres dynamiques par boutique et par période sans développement supplémentaire

■ GitHub (Versioning)

Choix retenu	GitHub — Versioning Git
Justification	Standard industriel, gratuit, historique des commits, collaboration multi-membres, intégration CI/CD possible

Alternative envisagée	GitLab — viable mais GitHub plus universel pour un portfolio CDA
Avantage clé	Historique propre démontrant la progression du projet (commits datés par sprint)

3. Fonctionnalités implémentées

3.1 Application Retool — 5 modules CRUD

Module	Fonctionnalités	Rôles concernés	Tables utilisées
Module Clients	Créer, lister, rechercher, modifier un client Recherche par nom / email / téléphone	agent, responsable	client
Module Appareils	Ajouter un appareil lié à un client Catégories : Smartphone, PC, Tablette, Console, Montre	agent, responsable	appareil, client
Module Réparations	Créer une réparation, assigner technicien Changer statut, voir timeline historique Filtrer par boutique, technicien, statut	tous les rôles	reparation, historique_statut, statut
Module Devis	Créer un devis avec lignes OPERATION/PIECE Calcul automatique du montant total	agent, responsable	devis, devis_ligne
Gestion Rôles	Filtrage automatique selon le rôle connecté Technicien : ses réparations uniquement Responsable : toutes boutiques	admin	employe, role

3.2 Dashboard Metabase — 6 KPI métiers

KPI	Description	Visualisation	Requête SQL
KPI 1	CA mensuel par boutique	Graphique barres	SUM(montant) GROUP BY boutique, mois
KPI 2	Réparations en cours	Compteur numérique	COUNT WHERE statut = "En cours"
KPI 3	Répartition par statut	Camembert	COUNT GROUP BY statut
KPI 4	Top 5 types réparation	Barres horizontales	COUNT GROUP BY type_reparation LIMIT 5
KPI 5	Délai moyen réparation	Tableau par boutique	AVG(date_fin - date_depot)
KPI 6	Stock sous seuil alerte	Liste critique	WHERE quantite < seuil_alerte

3.3 Règles de gestion implémentées

ID	Règle de gestion
RG1	Un client peut avoir plusieurs appareils — relation 1:N (client → appareil)
RG2	Chaque réparation est assignée à un technicien principal obligatoire
RG3	Les pièces utilisées sont tracées dans reparation_piece avec quantité et prix appliqué

RG4	Chaque changement de statut crée une entrée dans historique_statut (traçabilité complète)
RG5	Un devis contient des lignes de type OPERATION ou PIECE
RG6	Un technicien ne voit que ses réparations (filtrage par technicien_principal_id)
RG7	Le stock déclenche une alerte si quantite < seuil_alerte
RG8	Un paiement est unique par réparation (contrainte UNIQUE sur id_reparation)

4. Modèle de données — 19 tables PostgreSQL

Le modèle de données a été conçu pour couvrir l'ensemble du cycle de vie d'une réparation : de l'accueil du client jusqu'au paiement, en passant par le diagnostic, la commande de pièces et la traçabilité des statuts. Le schéma comporte 19 tables organisées en 3 groupes fonctionnels.

Groupe 1 — Tables de référence (7 tables)

Table	Nb enreg.	Colonnes clés	Remarques
role	5 rôles RBAC	code, libelle	ADMIN, RESP_BOUTIQUE, TECH, ACCUEIL, LOGISTIQUE
boutique	5 boutiques IDF	id_boutique (PK), nom, adresse, ville, code_postal	B01→B05
employe	12 employés	id_employe, id_boutique (FK), id_role (FK), nom, email_pro, actif	Lié boutique + rôle
client	20 clients	id_client, nom, prenom, telephone, email, ville	Données personnelles
appareil	30 appareils	id_appareil, id_client (FK), marque, modele, categorie	5 catégories possibles
type_reparation	20 types	code_type (PK), libelle, prix_moyen, duree_moy_h	Catalogue tarifs
statut	11 statuts	id_statut, libelle, ordre	Cycle de vie réparation

Groupe 2 — Logistique et pièces (6 tables)

Table	Nb enreg.	Colonnes clés	Remarques
piece	20 pièces	id_piece, reference, libelle, categorie	Références internes
fournisseur	5 fournisseurs	id_fournisseur, nom, specialite, delai_moy_jours	Délai livraison
fournisseur_piece	N:N	(id_fournisseur, id_piece) PK, prix_unitaire	Prix par fournisseur
stock	Par boutique	(id_boutique, id_piece) PK, quantite, seuil_alerte	Alerte si qte < seuil
commande_fournisseur	Commandes	id_commande, id_fournisseur, id_boutique, statut_commande	3 statuts possibles
commande_ligne	Lignes cmd	(id_commande, id_piece) PK, quantite, prix_achat	Détail commandes

Groupe 3 — Atelier et réparations (6 tables)

Table	Nb enreg.	Colonnes clés	Remarques

reparation	20 répar.	id_reparation, numero_suivi (UNIQUE), technicien_principal_id, devis_accepte	Objet central du SI
historique_statut	Traçabilité	id_hist, id_reparation (FK), id_statut (FK), date_changement	RG4 — audit trail
devis	0..1/répar.	id_devis, id_reparation (UNIQUE FK), montant_total, statut_devis	4 statuts devis
devis_ligne	Lignes devis	id_devis_ligne, type_ligne (OPERATION/PIECE), qte, prix_unitaire	Détail facturation
reparation_piece	N:N	(id_reparation, id_piece) PK, quantite, prix_applique	Pièces consommées
paiement	0..1/répar.	id_paiement, id_reparation (UNIQUE FK), montant, mode	CB/especes/virement

Indexes créés pour les performances

Index	Colonnes	Utilité
idx_client_nom	client(nom, prenom)	Recherche rapide client par nom
idx_reparation_numero_suivi	reparation(numero_suivi)	Lookup par numéro de suivi (unique)
idx_histo_rep_date	historique_statut(id_reparation, date_changement)	Timeline statuts
idx_stock_seuil	stock(id_boutique, seuil_alerte, quantite)	KPI alertes stock
idx_employe_boutique_role	employe(id_boutique, id_role)	Filtrage par rôle

5. Limites connues et évolutions prévues

5.1 Limites connues du prototype actuel

ID	Domaine	Description de la limite	Impact
L1	Authentification	Retool gère l'auth mais sans SSO — pas d'authentification unifiée avec Supabase Auth	Moyen
L2	Notifications	Pas de notifications automatiques au client (SMS/email) lors du changement de statut	Élevé
L3	Offline	L'application nécessite une connexion internet — pas de mode hors-ligne	Faible
L4	Mobile	Retool est optimisé desktop — interface non responsive sur mobile	Moyen
L5	Performance	Session Pooler Supabase limité à 500 connexions simultanées (offre gratuite)	Faible
L6	Rapports PDF	Pas de génération automatique de devis PDF depuis l'application	Élevé
L7	Sécurité	Mots de passe de test en clair dans la documentation — à changer en production	Critique

5.2 Évolutions techniques prévues (V2)

ID	Horizon	Évolution	Solution technique	Version
E1	Court terme	Notifications SMS/email	Twilio API ou SendGrid intégré dans Retool	Sprint 4
E2	Court terme	Génération PDF devis	Bibliothèque PDF dans Retool (jsPDF)	Sprint 4
E3	Moyen terme	Application mobile	React Native + API Supabase REST	V2
E4	Moyen terme	Authentification unifiée	Supabase Auth + Row Level Security (RLS)	V2
E5	Moyen terme	CI/CD automatisé	GitHub Actions → déploiement automatique	V2
E6	Long terme	IA prédictive	Prédiction délais réparation par ML (Python)	V3
E7	Long terme	Module facturation	Intégration API comptable (Pennylane/Sage)	V3

5.3 Bilan du prototype

Le prototype livré couvre l'ensemble des fonctionnalités prioritaires définies dans le cahier des charges. Les 11 user stories ont été implémentées sur les 3 sprints planifiés. Le système permet à QuickRepair France de gérer ses réparations de manière centralisée, avec une traçabilité complète et des indicateurs décisionnels en temps réel.

Critère	Réalisé	Taux	Statut
User Stories	11/11	100%	■ Toutes implémentées
Tables BDD	19/19	100%	■ Toutes créées et alimentées
KPI Metabase	6/6	100%	■ Tous fonctionnels
Modules Retool	5/5	100%	■ Tous opérationnels
Tests fonctionnels	10/10	100%	■ 9 OK + 1 KO corrigé
Commits Git	8+	100%	■ 2 auteurs — messages propres