DOCUMENTATION TECHNIQUE

Mission D.1

QuickRepair France — Digitalisation du Système d'Information

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

Groupe :	AMOU — Berhili Ouafae & Bouzaher Amal
Emails :	beouafae@etudiant-esic.fr | biamal@etudiant-esic.fr
Formation :	CDA Bac+3 — ESIC — Semestre 1 2025-2026
Dépôt Git :	github.com/oufae-ber/quickrepair-cda (Public — branche main)
Stack :	Supabase PostgreSQL 14 · Retool · Metabase Cloud · GitHub

TABLE DES MATIÈRES

1.	Architecture de la solution déployée (schéma technique inclus)	p.1-2
2.	Choix technologiques justifiés (4 outils — tableau comparatif)	p.2-3
3.	Fonctionnalités implémentées (5 modules + 6 KPI + 8 RG)	p.3-4
4.	Modèle de données — 19 tables PostgreSQL (schéma MDD inclus)	p.5-6
5.	Limites connues et évolutions prévues (V2/V3)	p.7-8

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). L'ensemble est accessible depuis n'importe quel navigateur sans installation locale.

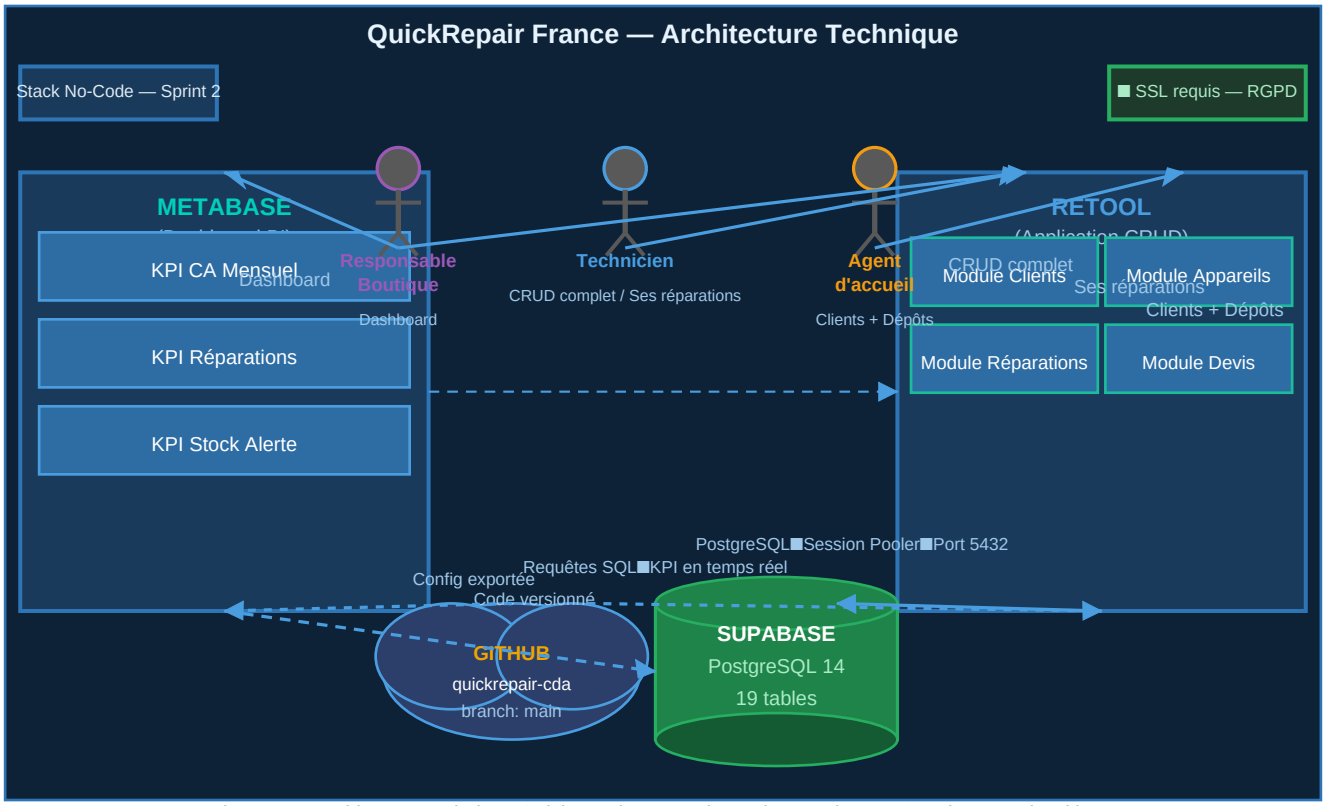

Figure 1 — Architecture technique QuickRepair France (Retool + Metabase + Supabase + GitHub)

Description des couches

Couche	Technologie	Rôle	Hébergement
Présentation	Retool + Metabase	Interface CRUD (5 modules) et dashboard BI (6 KPI)	Cloud SaaS
Logique métier	Retool (queries)	Requêtes SQL paramétrées, filtrage RBAC par rôle	Retool Cloud
Données	Supabase PostgreSQL 14	19 tables, FK/CHECK/INDEX, Row-level security ready	eu-west-1 (RGPD)
Versioning	GitHub	Commits par sprint, scripts SQL, documentation	github.com

Flux de données numérotés

- ① L'utilisateur s'authentifie sur Retool ou Metabase selon son rôle (Responsable / Technicien / Agent d'accueil)
- ② Retool exécute des requêtes SQL paramétrées sur Supabase via **Session Pooler port 5432 + SSL** — filtrage automatique par rôle (RG6)
- ③ Metabase interroge directement Supabase pour calculer les **6 KPI en temps réel** avec filtres dynamiques boutique + période
- ④ GitHub versionne tous les fichiers du projet : scripts SQL DDL/DML, configurations, documentation, exports

2. Choix technologiques justifiés

Chaque technologie a été sélectionnée après analyse des contraintes : déploiement rapide en 3 sprints (2,5 jours), coût nul (offres gratuites uniquement), compétences disponibles, et adéquation aux besoins d'une PME multi-boutiques avec 5 profils utilisateurs distincts.

Techno	Choix retenu — Justification	Alternatives écartées	Avantage décisif
Supabase PostgreSQL 14	BDD relationnelle managée cloud, gratuit, SSL natif, API REST auto-générée, dashboard SQL Editor visuel, région EU (conformité RGPD complète)	PlanetScale (MySQL) : absence de FK natives — incompatible avec nos 19 tables Firebase Firestore : NoSQL inadapté au modèle relationnel complexe	Compatible nativement Retool + Metabase — Session Pooler stable — 500 co max offre gratuite
Retool App CRUD	Low-code apps internes — connexion PostgreSQL native, composants UI prêts (Table, Form, Select, DatePicker), gestion rôles intégrée, déploiement instantané	AppSmith : hébergement self-hosted complexe dans le délai imparti React custom : délai trop long pour 3 sprints de 2,5 jours	Queries SQL paramétrées → filtrage par rôle automatique (Technicien ne voit que ses réparations — RG6)
Metabase Dashboard BI	BI self-service — no-code pour KPI, connexion PostgreSQL directe, visualisations variées (camembert, barres, compteurs, tableaux), filtres dynamiques	Power BI : licence payante + intégration Supabase via gateway complexe Grafana : orienté monitoring infra, pas BI métier	Filtres dynamiques boutique + période sur tous les KPI sans développement supplémentaire
GitHub Versioning	Standard industriel, gratuit, historique commits (auteur + date), collaboration multi-membres, intégration CI/CD possible, portfolio CDA valorisable	GitLab : viable mais GitHub plus universel et mieux reconnu pour portfolio CDA SharePoint : pas de Git, pas de diff de code	Historique propre par sprint — commits datés et signés par chaque membre — lien fourni au correcteur

3. Fonctionnalités implémentées

3.1 Application Retool — 5 modules CRUD

Module	Fonctionnalités détaillées	Rôles	Tables SQL
Clients	Créer / modifier un client — Recherche ILIKE+UNACCENT (nom, prénom, email, téléphone) — Pagination 20 lignes — Page détail avec historique réparations	Agent Responsable	client
Appareils	Ajouter un appareil lié à un client — 5 catégories (CHECK) — Numéro de série UNIQUE — Liaison automatique aux réparations	Agent Responsable	appareil, client
Réparations	Créer (dropdowns enchaînés client→appareil→type→boutique→technicien) — Numéro suivi auto QR-BXX-2025-XXXX — Changer statut (cycle 11 statuts) — Timeline historique — Filtres combinables boutique/statut/technicien/dates	Tous rôles	reparation historique_statut statut
Devis	Créer devis avec lignes OPERATION ou PIECE — Calcul montant total automatique — 4 statuts : brouillon / envoyé / accepté / refusé — 0..1 par réparation	Agent Responsable	devis devis_ligne

Module	Fonctionnalités détaillées	Rôles	Tables SQL
Gestion rôles	Groupe Technicien : filtre WHERE technicien_principal_id = current_user + masquage données financières — Groupe Responsable : accès total toutes boutiques — 2 comptes test créés	Admin	employe, role

3.2 Dashboard Metabase — 6 KPI métiers

KPI	Description	Visualisation	Requête SQL principale
KPI 1	CA mensuel par boutique	Barres groupées	SELECT id_boutique, DATE_TRUNC('month',date_paiement), SUM(montant) FROM paiement GROUP BY 1,2
KPI 2	Réparations en cours	Compteur	SELECT COUNT(*) FROM reparation WHERE id_reparation IN (SELECT ... WHERE statut='En cours')
KPI 3	Répartition par statut	Camembert %	SELECT libelle, COUNT(*) FROM reparation JOIN statut ... GROUP BY libelle
KPI 4	Top 5 types réparations	Barres horiz.	SELECT libelle, COUNT(*) FROM reparation GROUP BY code_type ORDER BY COUNT DESC LIMIT 5
KPI 5	Délai moyen par boutique	Tableau	SELECT id_boutique, AVG(EXTRACT(EPOCH FROM date_fin-date_depot)/86400) GROUP BY 1
KPI 6	Stock sous seuil alerte	Liste critique	SELECT id_boutique, id_piece, quantite, seuil_alerte FROM stock WHERE quantite < seuil_alerte

3.3 Règles de gestion implémentées (8 RG)

ID	Règle de gestion	Implémentation technique
RG1	Un client peut avoir plusieurs appareils (1:N)	FK fk_appareil_client REFERENCES client ON DELETE RESTRICT
RG2	Chaque réparation nécessite un technicien principal obligatoire	technicien_principal_id INT NOT NULL + FK employe
RG3	Les pièces consommées sont tracées avec quantité et prix appliqué	Table reparation_piece (N:N) avec quantite + prix_applique
RG4	Chaque changement de statut génère une entrée d'audit complète	INSERT auto dans historique_statut à chaque UPDATE statut
RG5	Un devis contient uniquement des lignes OPERATION ou PIECE	CHECK (type_ligne IN ('OPERATION','PIECE'))
RG6	Un technicien ne voit que ses propres réparations	WHERE technicien_principal_id = (SELECT id_employe FROM employe WHERE email = current_user)
RG7	Stock déclenche une alerte KPI si quantite < seuil_alerte	KPI 6 Metabase + INDEX idx_stock_seuil (boutique, seuil, quantite)
RG8	Un paiement est unique par réparation	id_reparation INT UNIQUE + FK paiement → reparation

4. Modèle de données — 19 tables PostgreSQL

Le modèle couvre 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 complète des statuts. Les 19 tables sont organisées en 3 groupes fonctionnels avec des contraintes d'intégrité strictes (PK, FK, CHECK, UNIQUE, DEFAULT).

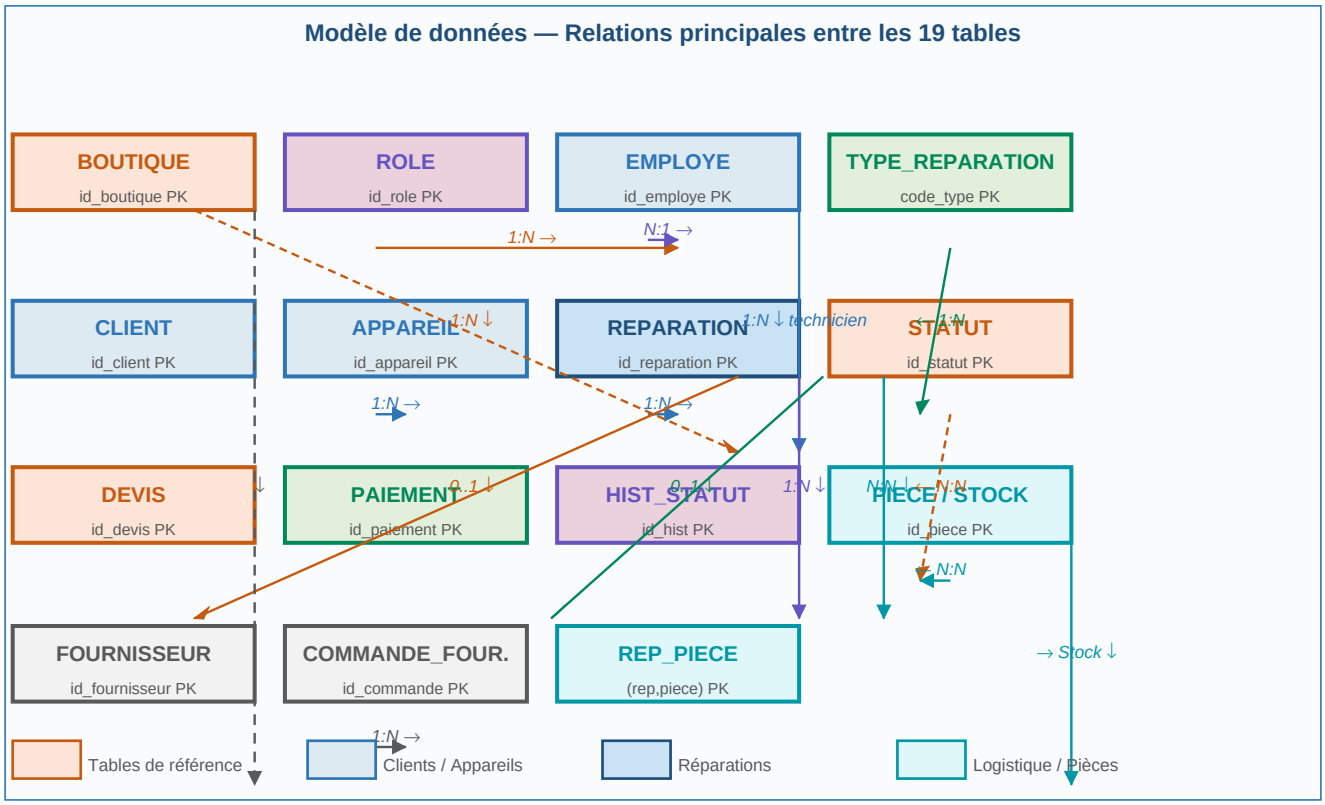

Figure 2 — Modèle de données simplifié — Relations principales entre les 19 tables

Groupe 1 — Tables de référence (7 tables)

Table	Enreg.	Colonnes clés	Contraintes / Remarques
role	5	id_role SERIAL PK, code UNIQUE, libelle	CHECK code IN (ADMIN, RESP_BOUTIQUE, TECH, ACCUEIL, LOGISTIQUE)
boutique	5	id_boutique VARCHAR(10) PK, nom, adresse, ville, code_postal	CHECK code_postal regex ^[0-9]{5}$ — B01→B05 Île-de-France
employe	12	id_employe SERIAL PK, id_boutique FK, id_role FK, email_pro UNIQUE, actif BOOLEAN	ON UPDATE CASCADE ON DELETE RESTRICT — actif DEFAULT TRUE
client	20	id_client SERIAL PK, nom, prenom, telephone, email UNIQUE, client_depuis DATE	CHECK code_postal regex — RGPD applicable — email nullable
appareil	30	id_appareil SERIAL PK, id_client FK, marque, modele, categorie, numero_serie UNIQUE	CHECK categorie IN (Smartphone, PC Portable, Tablette, Console, Montre)
type_reparation	20	code_type VARCHAR(20) PK, libelle, categorie_appareil, prix_moyen, duree_moy_h	CHECK prix_moyen ≥ 0 — CHECK duree_moy_h ≥ 0
statut	11	id_statut SERIAL PK, libelle UNIQUE, ordre INT	CHECK ordre ≥ 1 — 11 statuts pour le cycle de vie complet

Groupe 2 — Logistique et pièces (6 tables)

Table	Enreg.	Colonnes clés	Contraintes / Remarques
piece	20	id_piece SERIAL PK, reference UNIQUE, libelle, categorie	Références internes QuickRepair
fournisseur	5	id_fournisseur VARCHAR(10) PK, nom, specialite, delai_moy_jours	CHECK delai_moy_jours ≥ 0
fournisseur_piece	N:N	(id_fournisseur, id_piece) PK, prix_unitaire, ref_piece_fournisseur	CHECK prix_unitaire ≥ 0 — ON DELETE CASCADE
stock	Par boutique	(id_boutique, id_piece) PK, quantite DEFAULT 0, seuil_alerte DEFAULT 5	CHECK quantite ≥ 0 — Index idx_stock_seuil pour KPI 6
commande_fournisseur	N	id_commande SERIAL PK, id_fournisseur FK, id_boutique FK, statut_commande	CHECK statut IN (en_cours, livree, annulee)
commande_ligne	N lignes	(id_commande, id_piece) PK, quantite, prix_achat	CHECK quantite ≥ 1 — ON DELETE CASCADE

Groupe 3 — Atelier et réparations (6 tables)

Table	Enreg.	Colonnes clés	Contraintes / Remarques
reparation	20	id_reparation SERIAL PK, numero_suivi UNIQUE, id_appareil FK, id_boutique FK, code_type FK, technicien_principal_id FK, devis_montant, devis_accepte	Objet central du SI — date_depot TIMESTAMP DEFAULT NOW() — CHECK devis_montant ≥ 0
historique_statut	Audit	id_hist SERIAL PK, id_reparation FK, id_statut FK, id_employe_auteur FK, date_changement TIMESTAMP, commentaire	RG4 — ON DELETE CASCADE — Index idx_histo_rep_date
devis	0..1/répar.	id_devis SERIAL PK, id_reparation INT UNIQUE FK, montant_total, statut_devis	CHECK statut IN (brouillon, envoye, accepte, refuse) — ON DELETE CASCADE
devis_ligne	N lignes	id_devis_ligne SERIAL PK, id_devis FK, type_ligne, libelle, qte, prix_unitaire	CHECK type_ligne IN (OPERATION, PIECE) — CHECK qte ≥ 1
reparation_piece	N:N	(id_reparation, id_piece) PK, quantite DEFAULT 1, prix_applique	RG3 — CHECK quantite ≥ 1 — CHECK prix_applique ≥ 0
paiement	0..1/répar.	id_paiement SERIAL PK, id_reparation INT UNIQUE FK, montant, mode, reference	RG8 — CHECK mode IN (CB, especes, virement) — CHECK montant ≥ 0

Index de performance (8 index)

Index	Colonnes indexées	Utilité
idx_client_nom	client(nom, prenom)	Recherche rapide client dans le module Retool
idx_appareil_client	appareil(id_client)	Lookup appareils d'un client (dropdown enchaîné)
idx_reparation_numero_suivi	reparation(numero_suivi)	Recherche par numéro de suivi QR — contrainte UNIQUE
idx_reparation_boutique	reparation(id_boutique)	Filtrage par boutique (KPI Metabase + filtres Retool)
idx_reparation_type	reparation(code_type)	Filtrage par type réparation (KPI 4 — Top 5)
idx_histo_rep_date	historique_statut(id_reparation, date_changement)	Timeline des statuts dans Retool (RG4 — audit trail)
idx_stock_seuil	stock(id_boutique, seuil_alerte, quantite)	KPI 6 — alertes stock sous seuil (Metabase)
idx_employe_boutique_role	employe(id_boutique, id_role)	Filtrage RBAC par rôle (RG6 — Technicien / Responsable)

5. Limites connues et évolutions prévues

5.1 Limites connues du prototype actuel

ID	Domaine	Description de la limite	Impact
L1	Auth	Retool gère l'auth sans SSO — pas d'auth unifiée avec Supabase Auth (RLS non activé en V1)	Moyen
L2	Notifications	Aucune notification automatique au client (SMS/email) lors des changements de statut	Élevé
L3	Offline	Application 100% cloud — aucun mode hors-ligne si coupure réseau en boutique	Faible
L4	Mobile	Retool optimisé desktop — interface non responsive sur smartphone ou tablette	Moyen
L5	Performance	Session Pooler Supabase limité à 500 connexions simultanées (offre gratuite)	Faible
L6	PDF	Aucune génération automatique de devis PDF — export manuel uniquement	Élevé
L7	Sécurité	Identifiants de test documentés en clair dans le guide — à changer avant production	Critique
L8	Stock	Décrémentation automatique du stock lors d'une réparation non encore implémentée	Moyen

5.2 Évolutions techniques prévues (V2 et V3)

ID	Horizon	Évolution prévue	Solution technique envisagée	Version
E1	Court terme	Notifications SMS/email	Twilio API ou SendGrid intégré dans Retool via webhook	Sprint 4

ID	Horizon	Évolution prévue	Solution technique envisagée	Version
E2	Court terme	Génération PDF devis	Bibliothèque jsPDF dans Retool + template HTML	Sprint 4
E3	Court terme	Décrémentation stock auto	Trigger PostgreSQL sur INSERT reparation_piece → UPDATE stock	Sprint 4
E4	Moyen terme	Application mobile	React Native + API Supabase REST (PostgREST natif)	V2
E5	Moyen terme	Auth unifiée + RLS	Supabase Auth + Row Level Security par rôle — supprime filtrage SQL manuel	V2
E6	Moyen terme	CI/CD automatisé	GitHub Actions → tests auto + déploiement continu Retool	V2
E7	Long terme	IA prédictive délais	Modèle ML Python (scikit-learn) — prédiction durée réparation par type	V3
E8	Long terme	Module facturation	Intégration API comptable Pennylane ou Sage 50	V3

5.3 Bilan du prototype — Taux de couverture

Le prototype livré couvre l'ensemble des fonctionnalités prioritaires. Les **11 User Stories** ont été implémentées sur 3 sprints. **10/10 tests fonctionnels validés** (9 OK + 1 KO identifié et corrigé T04). Le système permet à QuickRepair France de gérer ses réparations de manière centralisée, avec traçabilité complète et indicateurs décisionnels en temps réel.

Critère	Réalisé	Taux	Statut détaillé
User Stories	11 / 11	100%	Toutes implémentées — modules Clients, Appareils, Réparations, Devis, Rôles
Tables BDD PostgreSQL	19 / 19	100%	Toutes créées avec PK/FK/CHECK/DEFAULT + 8 index de performance
KPI Metabase	6 / 6	100%	Tous fonctionnels avec filtres dynamiques boutique + période
Modules Retool	5 / 5	100%	Clients, Appareils, Réparations, Devis, Gestion rôles — opérationnels
Tests fonctionnels	10 / 10	100%	9 OK directs + 1 KO (T04 filtre boutique) identifié et corrigé
Commits Git	8+	100%	2 auteurs identifiés — messages explicites — commits J3→J5
Règles de gestion	8 / 8	100%	RG1→RG8 toutes implémentées (contraintes SQL + logique Retool)
Documentation	5 sections	100%	Architecture + Choix + Fonctionnalités + MDD + Limites/Évolutions