**Cahier des charges fonctionnel**

**Projet : Plateforme de gestion de la facturation de l'eau et de l'assainissement**

**Objectif global**

Développer une solution modulaire permettant de gérer de bout en bout les processus liés à la distribution d'eau potable et au traitement des eaux usées : abonnements, contrats, interventions, facturation, communication, gestion documentaire, statistiques, sécurité et automatisation des processus (workflow).

.

├── Database

│   ├── config

│   │   ├── config.d.ts

│   │   └── config.json

│   ├── migrations

│   │   ├── 20241207\_add\_type\_code\_to\_contract\_counters.js

│   │   ├── 20241207\_create\_contrat\_audit\_table.js

│   │   └── 20241207\_create\_contrat\_compteur\_historique\_table.js

│   ├── models

│   │   ├── ContractCounter.model.ts

│   │   ├── abonnement.model.ts

│   │   ├── action.model.ts

│   │   ├── adresse.model.ts

│   │   ├── affaire.model.ts

│   │   ├── client.model.ts

│   │   ├── compteur.model.ts

│   │   ├── contrat.model.ts

│   │   ├── contratAudit.model.ts

│   │   ├── contratCompteurHistorique.model.ts

│   │   ├── contratCosignataire.model.ts

│   │   ├── entreprise.model.ts

│   │   ├── enums

│   │   │   └── userRole.enum.ts

│   │   ├── facture.model.ts

│   │   ├── index.js

│   │   ├── index.ts

│   │   ├── intervention.model.ts

│   │   ├── lettrage.model.ts

│   │   ├── ligneFacture.model.ts

│   │   ├── lotFacturation.model.ts

│   │   ├── page.model.ts

│   │   ├── pageAction.model.ts

│   │   ├── paiement.model.ts

│   │   ├── rolePagePermission.model.ts

│   │   ├── tarif.model.ts

│   │   ├── user.d.ts

│   │   ├── user.js

│   │   ├── user.js.map

│   │   ├── user.model.ts

│   │   ├── workflow.model.ts

│   │   ├── workflowEdge.model.ts

│   │   ├── workflowNode.model.ts

│   │   └── workflowRun.model.ts

│   ├── package-lock.json

│   ├── package.json

│   ├── repositories

│   │   ├── abonnement.repository.ts

│   │   ├── adresse.repository.ts

│   │   ├── affaire.repository.ts

│   │   ├── client.repository.ts

│   │   ├── compteur.repository.ts

│   │   ├── contrat.repository.ts

│   │   ├── contratAudit.repository.ts

│   │   ├── contratCompteurHistorique.repository.ts

│   │   ├── contratCosignataire.repository.ts

│   │   ├── entreprise.repository.ts

│   │   ├── facture.repository.ts

│   │   ├── intervention.repository.ts

│   │   ├── ligneFacture.repository.ts

│   │   ├── lotFacturation.repository.ts

│   │   ├── paiement.repository.ts

│   │   ├── tarif.repository.ts

│   │   ├── user.repository.d.ts

│   │   ├── user.repository.ts

│   │   └── workflow.repository.ts

│   ├── seeders

│   ├── sequelize.d.ts

│   ├── sequelize.js

│   ├── sequelize.js.map

│   └── sequelize.ts

├── README.md

├── database\_schema.md

├── docker-compose.yml

├── docs

│   ├── CDC.docx

│   ├── architecture.md

│   ├── deployment-checklist.md

│   ├── doc.docx

│   └── flux.svg

├── domain

│   ├── entité

│   │   ├── contrat.ts

│   │   ├── contratAudit.ts

│   │   ├── contratCompteurHistorique.ts

│   │   ├── cosignataire.ts

│   │   ├── index.ts

│   │   └── user.ts

│   └── errors

│   ├── login.errors.ts

│   └── register.errors.ts

├── frontend

│   ├── Dockerfile

│   ├── cypress

│   │   └── e2e

│   │   ├── auth.cy.ts

│   │   └── profile.cy.ts

│   ├── cypress.config.js

│   ├── env.d.ts

│   ├── index.html

│   ├── nginx.conf

│   ├── package-lock.json

│   ├── package.json

│   ├── postcss.config.js

│   ├── src

│   │   ├── App.vue

│   │   ├── assets

│   │   │   ├── aquaerp\_logo.svg

│   │   │   ├── aquaerp\_logo\_perfect.svg

│   │   │   └── logo.png

│   │   ├── components

│   │   │   ├── clients

│   │   │   │   ├── AddClientModal.vue

│   │   │   │   ├── ClientDetailDrawer.vue

│   │   │   │   └── ClientSearchCard.vue

│   │   │   ├── contrats

│   │   │   │   ├── AddContratModal.vue

│   │   │   │   ├── ContractAuditTrail.vue

│   │   │   │   ├── ContratDetailModal.vue

│   │   │   │   ├── CosignatairesInvitationCard.vue

│   │   │   │   ├── CosignatairesStatus.vue

│   │   │   │   └── SignatureStatusBadge.vue

│   │   │   ├── dashboard

│   │   │   │   ├── DashboardCard.vue

│   │   │   │   ├── DashboardSection.vue

│   │   │   │   ├── RecentActivity.vue

│   │   │   │   ├── StatCard.vue

│   │   │   │   └── UserSummary.vue

│   │   │   ├── layout

│   │   │   │   ├── Layout.vue

│   │   │   │   ├── Navbar.vue

│   │   │   │   └── Sidebar.vue

│   │   │   ├── ui

│   │   │   │   ├── ConfirmationModal.vue

│   │   │   │   ├── DynamicForm.vue

│   │   │   │   ├── Notification.vue

│   │   │   │   ├── NotificationContainer.vue

│   │   │   │   ├── ResourceTable.vue

│   │   │   │   ├── SearchBar.vue

│   │   │   │   ├── button

│   │   │   │   │   └── Button.vue

│   │   │   │   ├── card

│   │   │   │   │   ├── Card.vue

│   │   │   │   │   ├── CardContent.vue

│   │   │   │   │   └── CardHeader.vue

│   │   │   │   └── input

│   │   │   │   └── Input.vue

│   │   │   └── users

│   │   │   ├── CreateUserModal.vue

│   │   │   ├── UserDetailModal.vue

│   │   │   └── UsersTable.vue

│   │   ├── composables

│   │   │   └── use-websocket.ts

│   │   ├── config

│   │   │   └── formConfigs.ts

│   │   ├── directives

│   │   │   └── has-permission.ts

│   │   ├── docs

│   │   │   └── DynamicForms.md

│   │   ├── lib

│   │   │   └── utils.ts

│   │   ├── main.ts

│   │   ├── router

│   │   │   ├── guards.ts

│   │   │   └── index.ts

│   │   ├── services

│   │   │   ├── api

│   │   │   │   ├── address.service.ts

│   │   │   │   ├── client.service.ts

│   │   │   │   ├── contract.service.ts

│   │   │   │   ├── intervention.service.ts

│   │   │   │   └── meter.service.ts

│   │   │   ├── api.service.ts

│   │   │   ├── authorization.service.ts

│   │   │   ├── client.service.ts

│   │   │   ├── http.interceptor.ts

│   │   │   └── user.service.ts

│   │   ├── stores

│   │   │   ├── auth.store.ts

│   │   │   ├── client.store.ts

│   │   │   ├── contract-draft.store.ts

│   │   │   ├── contract.store.ts

│   │   │   ├── intervention.store.ts

│   │   │   ├── meter.store.ts

│   │   │   ├── notification.store.ts

│   │   │   └── user.store.ts

│   │   ├── style.css

│   │   ├── types

│   │   │   ├── auth.types.ts

│   │   │   ├── client.types.ts

│   │   │   ├── contract.types.ts

│   │   │   ├── intervention.types.ts

│   │   │   └── user.types.ts

│   │   ├── utils

│   │   │   └── debounce.ts

│   │   ├── views

│   │   │   ├── DashboardView.vue

│   │   │   ├── HomeHubView.vue

│   │   │   ├── HomeView.vue

│   │   │   ├── UsersView.vue

│   │   │   ├── admin

│   │   │   │   ├── AdminLayout.vue

│   │   │   │   ├── CaissesView.vue

│   │   │   │   ├── PagesHabilitationView.vue

│   │   │   │   ├── RolesView.vue

│   │   │   │   ├── TarifsAutoUpdateView.vue

│   │   │   │   └── UsersAdminView.vue

│   │   │   ├── auth

│   │   │   │   ├── ActivateView.vue

│   │   │   │   ├── ChangePasswordView.vue

│   │   │   │   ├── ConfirmView.vue

│   │   │   │   ├── ForgotPasswordView.vue

│   │   │   │   ├── InviteUserView.vue

│   │   │   │   ├── LoginView.vue

│   │   │   │   ├── ProfileView.vue

│   │   │   │   ├── RegisterView.vue

│   │   │   │   └── ResetPasswordView.vue

│   │   │   ├── clients

│   │   │   │   ├── ClientDetailView.vue

│   │   │   │   └── ListClientsView.vue

│   │   │   ├── contracts

│   │   │   │   ├── ContractDetailsView.vue

│   │   │   │   ├── ContractSignaturePage.vue

│   │   │   │   ├── CreateContractWizard.vue

│   │   │   │   └── steps

│   │   │   │   ├── WizardStep1.vue

│   │   │   │   ├── WizardStep2.vue

│   │   │   │   ├── WizardStep3.vue

│   │   │   │   ├── WizardStep4.vue

│   │   │   │   ├── WizardStep5.vue

│   │   │   │   ├── WizardStep6.vue

│   │   │   │   ├── WizardStep7.vue

│   │   │   │   ├── WizardStepMeter.vue

│   │   │   │   └── WizardStepPayment.vue

│   │   │   ├── dashboard

│   │   │   │   ├── ClientsView.vue

│   │   │   │   ├── ContratsView.vue

│   │   │   │   ├── DashboardLayout.vue

│   │   │   │   ├── FormExampleView.vue

│   │   │   │   └── UsersView.vue

│   │   │   ├── errors

│   │   │   │   └── AccessDeniedView.vue

│   │   │   └── interventions

│   │   └── vite-env.d.ts

│   ├── tailwind.config.js

│   ├── tsconfig.json

│   └── vite.config.ts

├── infrastructure

│   ├── adapters

│   │   ├── logger.adapter.ts

│   │   └── mail.adapter.ts

│   ├── config

│   │   ├── config.module.ts

│   │   └── config.service.ts

│   ├── events

│   │   ├── event-bus.ts

│   │   └── event.constats.ts

│   ├── guards

│   │   ├── auth.guard.ts

│   │   ├── guards.module.ts

│   │   ├── roles.decorator.ts

│   │   └── roles.guard.ts

│   └── logger

│   ├── logger.module.ts

│   └── logger.service.ts

├── k8s

│   ├── README.md

│   ├── api-gateway.yaml

│   ├── auth-service.yaml

│   ├── configmap-example.yaml

│   ├── contrat-service.yaml

│   ├── database-init.yaml

│   ├── facture-service.yaml

│   ├── frontend.yaml

│   ├── ingress.yaml

│   ├── intervention-service.yaml

│   ├── mailer-service.yaml

│   ├── monitoring.yaml

│   ├── operation-service.yaml

│   ├── secrets-example.yaml

│   └── workflow-service.yaml

├── package-lock.json

├── package.json

├── schema-exemple.png

├── scripts

│   └── deploy.sh

├── services

│   ├── agency-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   └── main.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── api-gateway

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── api-gateway

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── affaires

│   │   │   │   └── affaires.controller.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── application

│   │   │   │   └── dtos

│   │   │   │   ├── auth

│   │   │   │   │   ├── auth-response.dto.ts

│   │   │   │   │   ├── change-password.dto.ts

│   │   │   │   │   ├── confirm.dto.ts

│   │   │   │   │   ├── create-agent.dto.ts

│   │   │   │   │   ├── forgot-password.dto.ts

│   │   │   │   │   ├── index.ts

│   │   │   │   │   ├── login.dto.ts

│   │   │   │   │   ├── register.dto.ts

│   │   │   │   │   ├── reset-password.dto.ts

│   │   │   │   │   ├── update-profile.dto.ts

│   │   │   │   │   └── update-status.dto.ts

│   │   │   │   ├── clients

│   │   │   │   │   ├── create-client.dto.ts

│   │   │   │   │   ├── list-clients.dto.ts

│   │   │   │   │   └── update-client.dto.ts

│   │   │   │   ├── contrat

│   │   │   │   │   ├── assign-compteur.dto.ts

│   │   │   │   │   ├── contrat-response.dto.ts

│   │   │   │   │   ├── cosignataire.dto.ts

│   │   │   │   │   ├── create-contrat-draft.dto.ts

│   │   │   │   │   ├── create-contrat.dto.ts

│   │   │   │   │   ├── lien-contrat.dto.ts

│   │   │   │   │   ├── renouvellement-contrat.dto.ts

│   │   │   │   │   ├── resiliation-contrat.dto.ts

│   │   │   │   │   ├── signature-contrat.dto.ts

│   │   │   │   │   ├── suspension-contrat.dto.ts

│   │   │   │   │   └── update-contrat.dto.ts

│   │   │   │   └── users

│   │   │   │   ├── create-user.dto.ts

│   │   │   │   ├── list-users.dto.ts

│   │   │   │   ├── update-user-status.dto.ts

│   │   │   │   └── update-user.dto.ts

│   │   │   ├── auth

│   │   │   │   └── auth.controller.ts

│   │   │   ├── clients

│   │   │   │   └── clients.controller.ts

│   │   │   ├── contrats

│   │   │   │   └── contract.controller.ts

│   │   │   ├── interventions

│   │   │   │   └── interventions.controller.ts

│   │   │   ├── main.ts

│   │   │   └── users

│   │   │   └── users.controller.ts

│   │   ├── test

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── auth-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── application

│   │   │   │   ├── dtos

│   │   │   │   │   ├── auth

│   │   │   │   │   │   ├── auth-response.dto.ts

│   │   │   │   │   │   ├── change-password.dto.ts

│   │   │   │   │   │   ├── confirm.dto.ts

│   │   │   │   │   │   ├── create-agent.dto.ts

│   │   │   │   │   │   ├── forgot-password.dto.ts

│   │   │   │   │   │   ├── index.ts

│   │   │   │   │   │   ├── login.dto.ts

│   │   │   │   │   │   ├── register.dto.ts

│   │   │   │   │   │   ├── reset-password.dto.ts

│   │   │   │   │   │   ├── update-profile.dto.ts

│   │   │   │   │   │   └── update-status.dto.ts

│   │   │   │   │   └── users

│   │   │   │   │   ├── create-user.dto.ts

│   │   │   │   │   ├── list-users.dto.ts

│   │   │   │   │   ├── update-user-status.dto.ts

│   │   │   │   │   └── update-user.dto.ts

│   │   │   │   ├── mappers

│   │   │   │   │   └── user.mapper.ts

│   │   │   │   ├── services

│   │   │   │   │   └── password.service.ts

│   │   │   │   ├── usecases

│   │   │   │   │   ├── auth

│   │   │   │   │   │   ├── activate-email.usecase.ts

│   │   │   │   │   │   ├── activate-user.usecase.ts

│   │   │   │   │   │   ├── change-password.usecase.ts

│   │   │   │   │   │   ├── confirm-invitation.usecase.ts

│   │   │   │   │   │   ├── create-user.usecase.ts

│   │   │   │   │   │   ├── forgot-password.usecase.ts

│   │   │   │   │   │   ├── invite-user.usecase.ts

│   │   │   │   │   │   ├── login

│   │   │   │   │   │   │   └── login.usecase.ts

│   │   │   │   │   │   ├── register

│   │   │   │   │   │   │   ├── register.admin.usecase.ts

│   │   │   │   │   │   │   └── register.interface.usecase.ts

│   │   │   │   │   │   ├── reset-password.usecase.ts

│   │   │   │   │   │   ├── update-password.usecase.ts

│   │   │   │   │   │   ├── update-profile.usecase.ts

│   │   │   │   │   │   └── update-status.usecase.ts

│   │   │   │   │   ├── clients

│   │   │   │   │   │   ├── create-client.usecase.ts

│   │   │   │   │   │   ├── get-client-by-id.usecase.ts

│   │   │   │   │   │   ├── list-clients.usecase.ts

│   │   │   │   │   │   └── update-client.usecase.ts

│   │   │   │   │   └── users

│   │   │   │   │   ├── create-user.usecase.ts

│   │   │   │   │   ├── delete-user.usecase.ts

│   │   │   │   │   ├── list-users.usecase.ts

│   │   │   │   │   ├── resend-invitation.usecase.ts

│   │   │   │   │   ├── update-user-status.usecase.ts

│   │   │   │   │   └── update-user.usecase.ts

│   │   │   │   └── validators

│   │   │   │   ├── email-domain.validator.ts

│   │   │   │   ├── password.validator.ts

│   │   │   │   └── phone-french.validator.ts

│   │   │   ├── auth

│   │   │   │   ├── auth.controller.ts

│   │   │   │   ├── auth.module.ts

│   │   │   │   └── auth.service.ts

│   │   │   ├── clients

│   │   │   │   ├── clients.controller.ts

│   │   │   │   └── clients.module.ts

│   │   │   ├── config

│   │   │   │   └── shared.module.ts

│   │   │   ├── domain

│   │   │   │   ├── entité

│   │   │   │   │   ├── contrat.ts

│   │   │   │   │   └── user.ts

│   │   │   │   ├── enums

│   │   │   │   │   └── user-role.enum.ts

│   │   │   │   └── errors

│   │   │   │   ├── login.errors.ts

│   │   │   │   ├── register.errors.ts

│   │   │   │   └── users.errors.ts

│   │   │   ├── infrastructure

│   │   │   │   ├── adapters

│   │   │   │   │   ├── logger.adapter.ts

│   │   │   │   │   └── mail.adapter.ts

│   │   │   │   ├── config

│   │   │   │   │   ├── config.module.ts

│   │   │   │   │   └── config.service.ts

│   │   │   │   ├── decorators

│   │   │   │   │   └── roles.decorator.ts

│   │   │   │   ├── events

│   │   │   │   │   ├── event-bus.ts

│   │   │   │   │   └── event.constats.ts

│   │   │   │   ├── guards

│   │   │   │   │   ├── auth.guard.ts

│   │   │   │   │   ├── guards.module.ts

│   │   │   │   │   ├── roles.decorator.ts

│   │   │   │   │   └── roles.guard.ts

│   │   │   │   └── logger

│   │   │   │   ├── logger.module.ts

│   │   │   │   └── logger.service.ts

│   │   │   ├── main.ts

│   │   │   ├── metadata.controller.ts

│   │   │   └── users

│   │   │   ├── users.controller.ts

│   │   │   ├── users.module.ts

│   │   │   └── users.service.ts

│   │   ├── test

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.e2e-spec.ts

│   │   │   ├── auth.controller.spec.ts

│   │   │   ├── auth.service.spec.ts

│   │   │   ├── jest-e2e.json

│   │   │   └── users.service.spec.ts

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── contrat-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── README\_NUMBER\_GENERATOR.md

│   │   ├── docs

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── application

│   │   │   │   ├── dtos

│   │   │   │   │   ├── assign-compteur.dto.ts

│   │   │   │   │   ├── contrat-response.dto.ts

│   │   │   │   │   ├── cosignataire.dto.ts

│   │   │   │   │   ├── create-contrat-draft.dto.ts

│   │   │   │   │   ├── create-contrat.dto.ts

│   │   │   │   │   ├── lien-contrat.dto.ts

│   │   │   │   │   ├── renouvellement-contrat.dto.ts

│   │   │   │   │   ├── resiliation-contrat.dto.ts

│   │   │   │   │   ├── signature-contrat.dto.ts

│   │   │   │   │   ├── suspension-contrat.dto.ts

│   │   │   │   │   └── update-contrat.dto.ts

│   │   │   │   ├── errors

│   │   │   │   │   └── contrat.errors.ts

│   │   │   │   ├── mappers

│   │   │   │   │   └── contrat.mapper.ts

│   │   │   │   ├── services

│   │   │   │   │   ├── audit.service.ts

│   │   │   │   │   ├── contrat-command.service.ts

│   │   │   │   │   ├── contrat-query.service.ts

│   │   │   │   │   ├── contrat.service.ts

│   │   │   │   │   ├── inter-service.service.ts

│   │   │   │   │   ├── multi-tenant.service.ts

│   │   │   │   │   ├── notification.service.ts

│   │   │   │   │   ├── number-generator.service.spec.ts

│   │   │   │   │   ├── number-generator.service.ts

│   │   │   │   │   └── workflow.service.ts

│   │   │   │   ├── usecases

│   │   │   │   │   ├── create-contrat.usecase.ts

│   │   │   │   │   ├── delete-contrat.usecase.ts

│   │   │   │   │   ├── finalize-contrat.usecase.ts

│   │   │   │   │   ├── get-contrat.usecase.ts

│   │   │   │   │   └── update-contrat.usecase.ts

│   │   │   │   └── validators

│   │   │   │   └── contrat.validator.ts

│   │   │   ├── contrat

│   │   │   │   ├── contrat.controller.ts

│   │   │   │   ├── contrat.microservice.ts

│   │   │   │   └── contrat.module.ts

│   │   │   ├── domain

│   │   │   │   ├── entité

│   │   │   │   │   ├── contrat.ts

│   │   │   │   │   ├── contratAudit.ts

│   │   │   │   │   ├── contratCompteurHistorique.ts

│   │   │   │   │   ├── cosignataire.ts

│   │   │   │   │   ├── index.ts

│   │   │   │   │   └── user.ts

│   │   │   │   └── errors

│   │   │   │   └── contrat.errors.ts

│   │   │   ├── infrastructure

│   │   │   │   ├── adapters

│   │   │   │   │   ├── logger.adapter.ts

│   │   │   │   │   └── mail.adapter.ts

│   │   │   │   ├── config

│   │   │   │   │   ├── config.module.ts

│   │   │   │   │   └── config.service.ts

│   │   │   │   ├── database

│   │   │   │   │   └── database.module.ts

│   │   │   │   ├── events

│   │   │   │   │   ├── event-bus.ts

│   │   │   │   │   └── event.constats.ts

│   │   │   │   ├── guards

│   │   │   │   │   ├── auth.guard.ts

│   │   │   │   │   ├── guards.module.ts

│   │   │   │   │   ├── roles.decorator.ts

│   │   │   │   │   └── roles.guard.ts

│   │   │   │   ├── logger

│   │   │   │   │   ├── logger.module.ts

│   │   │   │   │   └── logger.service.ts

│   │   │   │   └── repositories

│   │   │   │   ├── compteur.repository.ts

│   │   │   │   ├── contrat.repository.ts

│   │   │   │   └── cosignataire.repository.ts

│   │   │   ├── main.ts

│   │   │   └── metadata.controller.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── test-number-generator.js

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── facture-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── application

│   │   │   │   ├── controllers

│   │   │   │   │   ├── facture-message.controller.ts

│   │   │   │   │   ├── facture.controller.ts

│   │   │   │   │   ├── lot-facturation.controller.ts

│   │   │   │   │   └── rapports.controller.ts

│   │   │   │   ├── dtos

│   │   │   │   │   ├── create-facture.dto.ts

│   │   │   │   │   ├── ligne-facture.dto.ts

│   │   │   │   │   ├── lot-facturation.dto.ts

│   │   │   │   │   └── paiement.dto.ts

│   │   │   │   ├── mappers

│   │   │   │   │   └── facture.mapper.ts

│   │   │   │   ├── services

│   │   │   │   │   ├── audit.service.ts

│   │   │   │   │   ├── facture.service.ts

│   │   │   │   │   ├── inter-service.service.ts

│   │   │   │   │   ├── lot-facturation.service.ts

│   │   │   │   │   ├── multi-tenant.service.ts

│   │   │   │   │   └── pdf-generator.service.ts

│   │   │   │   └── validators

│   │   │   ├── domain

│   │   │   │   └── enums

│   │   │   │   ├── facture-statut.enum.ts

│   │   │   │   ├── ligne-facture-type.enum.ts

│   │   │   │   └── paiement-type.enum.ts

│   │   │   ├── infrastructure

│   │   │   │   ├── guards

│   │   │   │   │   ├── jwt-auth.guard.ts

│   │   │   │   │   └── tenant.guard.ts

│   │   │   │   ├── messaging

│   │   │   │   │   └── rabbitmq.module.ts

│   │   │   │   └── repositories

│   │   │   │   ├── facture.repository.adapter.ts

│   │   │   │   ├── ligne-facture.repository.adapter.ts

│   │   │   │   ├── lot-facturation.repository.adapter.ts

│   │   │   │   └── paiement.repository.adapter.ts

│   │   │   ├── interfaces

│   │   │   │   └── controllers

│   │   │   └── main.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── intervention-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── health.controller.ts

│   │   │   ├── intervention

│   │   │   │   ├── dto

│   │   │   │   │   ├── create-intervention.dto.ts

│   │   │   │   │   └── finish-intervention.dto.ts

│   │   │   │   ├── intervention.controller.ts

│   │   │   │   ├── intervention.microservice.ts

│   │   │   │   ├── intervention.module.ts

│   │   │   │   └── intervention.service.ts

│   │   │   ├── main.ts

│   │   │   └── metadata.controller.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── mailer-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── services

│   │   │   └── mailer-service

│   │   ├── src

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.module.ts

│   │   │   ├── events

│   │   │   │   └── facture.events.ts

│   │   │   ├── mailer

│   │   │   │   ├── dto

│   │   │   │   │   └── send-mail.dto.ts

│   │   │   │   ├── mailer.controller.spec.ts

│   │   │   │   ├── mailer.controller.ts

│   │   │   │   ├── mailer.module.ts

│   │   │   │   ├── mailer.service.spec.ts

│   │   │   │   ├── mailer.service.ts

│   │   │   │   └── templates

│   │   │   │   ├── confirmation-mail.pug

│   │   │   │   ├── contract-creation.pug

│   │   │   │   ├── contract-finalized.pug

│   │   │   │   ├── contract-signature-invitation.pug

│   │   │   │   ├── contract-signature-rejected.pug

│   │   │   │   ├── contract-signatures-complete.pug

│   │   │   │   ├── meter-installation.pug

│   │   │   │   ├── registration-link.pug

│   │   │   │   └── welcome.pug

│   │   │   ├── main.ts

│   │   │   ├── templates

│   │   │   │   ├── facture-emission.pug

│   │   │   │   ├── facture-paiement.pug

│   │   │   │   └── facture-relance.pug

│   │   │   └── utils

│   │   │   └── format-utils.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   ├── operation-service

│   │   ├── Dockerfile

│   │   ├── README.md

│   │   ├── nest-cli.json

│   │   ├── package-lock.json

│   │   ├── package.json

│   │   ├── src

│   │   │   ├── affaire

│   │   │   │   ├── affaire.controller.ts

│   │   │   │   ├── affaire.entity.ts

│   │   │   │   ├── affaire.microservice.ts

│   │   │   │   ├── affaire.model.ts

│   │   │   │   ├── affaire.module.ts

│   │   │   │   ├── affaire.service.ts

│   │   │   │   └── dto

│   │   │   │   ├── create-affaire.dto.ts

│   │   │   │   └── validate-affaire.dto.ts

│   │   │   ├── app.controller.spec.ts

│   │   │   ├── app.controller.ts

│   │   │   ├── app.module.ts

│   │   │   ├── app.service.ts

│   │   │   ├── health.controller.ts

│   │   │   ├── main.ts

│   │   │   └── metadata.controller.ts

│   │   ├── test

│   │   │   ├── app.e2e-spec.ts

│   │   │   └── jest-e2e.json

│   │   ├── tsconfig.build.json

│   │   └── tsconfig.json

│   └── workflow-service

│   ├── Dockerfile

│   ├── README.md

│   ├── nest-cli.json

│   ├── package-lock.json

│   ├── package.json

│   ├── src

│   │   ├── app.controller.spec.ts

│   │   ├── app.controller.ts

│   │   ├── app.module.ts

│   │   ├── app.service.ts

│   │   ├── catalog

│   │   │   ├── catalog.controller.ts

│   │   │   └── catalog.service.ts

│   │   ├── database

│   │   │   ├── sequelize.providers.ts

│   │   │   └── workflow.models.ts

│   │   ├── infrastructure

│   │   │   └── database

│   │   │   └── database.module.ts

│   │   ├── main.ts

│   │   └── workflow

│   │   ├── dto

│   │   │   └── create-workflow.dto.ts

│   │   ├── workflow.controller.ts

│   │   ├── workflow.event-listener.ts

│   │   └── workflow.service.ts

│   ├── test

│   │   ├── app.e2e-spec.ts

│   │   └── jest-e2e.json

│   ├── tsconfig.build.json

│   └── tsconfig.json

├── tsconfig.base.json

└── tsconfig.json

**1. Gestion des contrats et abonnements**

**Objectif**

Permettre la gestion complète du cycle de vie des contrats clients, en intégrant des logiques personnalisées de tarification, types de clients et gestion multi-compteurs.

**Fonctionnalités**

* Création de contrats de différents types : individuels, collectifs, résidentiels, agricoles, commerciaux, industriels.
* Prise en charge des multi-cosignataires (colocation, copropriété).
* Limitation à un seul cosignataire pour les contrats individuels.
* Cycle de vie complet : demande, validation, activation, modification, suspension, résiliation.
* Association flexible à un ou plusieurs compteurs (eau, assainissement).
* Tarification dynamique : selon consommation, période, profil client.
* Modèles de documents personnalisés pour chaque entreprise.

**Interfaces principales**

* **Liste des contrats** : filtre par type, état, client ; actions rapides.
* **Page de création/édition de contrat** : formulaire dynamique selon type ; ajout de compteurs ; sélection tarif ; gestion des co-signataires.
* **Historique et actions sur contrat** : journal des modifications, export PDF, duplication, suspension/réactivation, résiliation.

**2. Sécurité, habilitations et conformité réglementaire**

**Objectif**

Protéger les données personnelles et garantir une gestion rigoureuse des accès, en conformité avec le RGPD.

**Fonctionnalités**

* Gestion avancée des rôles utilisateurs, incluant : super administrateur, administrateur, chef d’équipe, technicien, agent de saisie, agent de terrain, superviseur, gestionnaire contentieux, conseiller clientèle, auditeur externe, etc.
* Possibilité pour un super administrateur de créer, modifier ou supprimer des rôles dynamiquement.
* Définition granulaire des droits d’accès pour chaque page ou module de l’application.
* Interface de paramétrage des droits d’accès par page : gestion des autorisations par rôle (CRUD, actions spécifiques, accès en lecture seule, export, etc.).
* Journalisation complète des actions (logs).
* Respect du RGPD : anonymisation, consentement, droit à l’oubli.
* Authentification sécurisée (JWT, MFA, SSO).

**Interfaces principales**

* **Gestion des utilisateurs** : tableau de bord pour :
  + Création, activation, verrouillage et modification de comptes,
  + Affectation des rôles et agences,
  + Réinitialisation de mot de passe.
* **Gestion des rôles et permissions** : interface de :
  + Création/suppression/modification de rôles,
  + Association par rôle à des pages/modules,
  + Attribution fine des droits par action (CRUD, export, visualisation, validation).
* **Matrice des droits par page** : écran de paramétrage dédié pour :
  + Visualisation des rôles croisés avec les pages de l’application,
  + Autorisations différenciées selon le contexte (ex : lecture seule pour les techniciens sur les contrats, écriture pour les agents de gestion).
* **Journal d’audit** : tableau listant :
  + Actions utilisateurs (connexion, modification de données sensibles…),
  + Date, utilisateur concerné, page ou module affecté,
  + Filtres par période ou type d’action. : création de comptes, activation, verrouillage, changement de rôle.
* **Gestion des rôles et permissions** : création de rôles, assignation par module et action (lecture, écriture, suppression, export, etc.).
* **Journal d’audit** : tableau des actions utilisateur horodatées, filtre par page ou utilisateur.

**3. Système de workflow**

**Objectif**

Automatiser les processus métiers via des plans d’action multi-étapes orchestrant tous les services de l’application.

**Fonctionnalités**

* Création automatique ou manuelle de plans d’action.
* Chaque plan d’action est composé d’étapes successives (steps).
* Étapes personnalisables : formulaire/questionnaire, action système, intervention, décision manuelle.
* Conditionnalité et logique d'enchaînement dynamique entre les étapes.
* Exemple : Workflow "Résiliation contrat" → création affaire, vérification compteur, intervention si compteur mécanique, génération facture finale, notification client.
* Visualisation et suivi de chaque plan d’action.
* Intégration avec les autres microservices via événements RabbitMQ.

**Interfaces principales**

* **Catalogue de workflows** : définition des types de plans d'action (résiliation, création contrat, etc.).
* **Éditeur de plan d'action** : création ou modification d’un plan d’action via une interface en drag-and-drop avec :
  + Définition de chaque étape (step), de son type (formulaire, système, intervention, validation manuelle).
  + Ordre des étapes, dépendances conditionnelles, branchements logiques.
  + Assignation d’un responsable par étape (profil, rôle ou utilisateur spécifique).
* **Gestion des questionnaires dynamiques** : page dédiée à la création de blocs de questions dynamiques (texte, listes, cases à cocher, conditions de visibilité, dépendances).
  + Ces questionnaires sont stockés et versionnés dans une bibliothèque centrale.
  + Chaque question est paramétrable avec des règles (ex : « afficher si consommation > 0 », « obligatoire si coché »).
* **Assignation de questionnaires aux étapes** : dans l’éditeur de workflow, chaque étape de type "formulaire" peut être liée à un questionnaire prédéfini.
* **Suivi des workflows actifs** : tableau des workflows en cours/terminés avec état d’avancement par étape.
  + Vue détaillée d’un workflow instancié (workflow exécuté pour un client, contrat ou intervention).
  + Historique des actions effectuées, utilisateur concerné, date, statut.

**Détails techniques supplémentaires**

* Les questions dynamiques sont enregistrées dans une base de données modulaire avec moteur de rendu frontend.
* Les étapes d’un plan d’action incluent : type, ordre, conditions de déclenchement, questionnaire associé, API cible ou action système.
* Chaque instance de workflow est historisée, traçable, et auditable.
* Déclenchement : automatique (via événements RabbitMQ comme contract.created, facture.failed, etc.) ou manuel (depuis une fiche affaire, client, contrat).
* Des modèles de workflows sont duplicables ou déployables par agence ou typologie de client. : définition des types de plans d'action (résiliation, création contrat, etc.).
* **Éditeur de plan d'action** : ajout/modification des étapes, conditions, questionnaires dynamiques.
* **Suivi des workflows actifs** : tableau des workflows en cours/terminés avec état d’avancement par étape.

**4. Gestion opérationnelle et interventions techniques**

**Objectif**

Permettre le suivi complet des opérations de terrain (relevés, coupures, réparations, installations).

**Fonctionnalités**

* Planification intelligente des interventions.
* Génération automatique de bons d’intervention.
* Suivi des incidents (fuites, urgences).
* Intégration IoT pour le relevé à distance.

**Interfaces principales**

* **Planning des interventions** : vue calendrier avec affectation des techniciens.
* **Fiche d’intervention** : type, lieu, statut, technicien, rapport d’exécution.
* **Gestion des incidents** : tableau des signalements avec priorisation.

**5. Facturation et suivi financier**

**Objectif**

Fournir un système de facturation automatisé, fiable, multi-services et compatible avec les pratiques comptables modernes.

**Fonctionnalités**

* Deux modes de facturation :
  + **Manuelle** : déclenchée à tout moment par un agent opérationnel selon un besoin ponctuel (ex. : régularisation, départ locataire).
  + **Automatique** : intégrée dans les plans d’action (workflows) ou programmée périodiquement (mensuelle, bimestrielle, etc.).
* Facturation conditionnée par la consommation réelle ou estimée, type de service ou événements (ex. intervention, résiliation).
* Prise en charge de multi-services : eau potable, assainissement, services complémentaires (ex. entretien, raccordement).
* Gestion des paiements : carte bancaire, SEPA, mobile, espèces.
* Gestion des relances : email, SMS, courrier papier, notifications automatisées.
* Échéanciers de paiement, acomptes, avoirs, pénalités de retard.
* Suivi des impayés et des plans de recouvrement.
* Gestion des dossiers contentieux avec historique des actions.
* Génération de duplicatas, justificatifs et attestations.
* Possibilité de simulation de facture pour devis ou prévisionnel.

**Interfaces principales**

* **Accès aux factures par le client** : l’entrée dans la facturation s’effectue exclusivement via un accès ciblé :
  + Depuis la fiche client : bouton "Factures", avec affichage filtré par contrat lié.
  + Depuis un compteur : historique de facturation associé à ce compteur (incluant consommations et événements).
  + Depuis un cosignataire : affichage limité aux factures partagées ou où la personne est identifiée comme co-responsable.
  + Aucun accès global ou "select all" n’est autorisé pour des raisons de volumétrie.
* **Page de génération manuelle** : formulaire complet de création d’une facture ponctuelle :
  + Sélection du contrat (filtrage par client),
  + Période concernée, compteur sélectionné,
  + Données de consommation manuelle ou automatique,
  + Lignes de facturation personnalisées (services, pénalités, remises).
* **Planification automatique** : page dédiée à la configuration des règles périodiques :
  + Paramétrage par type de contrat, agence ou profil client,
  + Choix du mode de consommation (estimée/réelle),
  + Fréquence de facturation,
  + Association à un modèle de facture.
* **Suivi des paiements** : accessible uniquement depuis la fiche client :
  + Visualisation des règlements associés aux factures générées,
  + Suivi des échéanciers, pénalités, avoirs,
  + Historique des relances.
* **Gestion des contentieux** : onglet spécifique dans la fiche client :
  + Création de litige sur une facture,
  + Suivi des actions (relances, communications, procédures juridiques),
  + Documents justificatifs associés. : filtre par client, statut, période, type ; export PDF/Excel ; accès rapide aux actions (relancer, dupliquer, annuler, télécharger).
* **Page de génération manuelle** : formulaire complet de création d’une facture ponctuelle :
  + Sélection du client et du contrat,
  + Saisie ou import de la consommation,
  + Ajout de lignes de services ou pénalités,
  + Validation et enregistrement.
* **Planification automatique** : écran de configuration des règles périodiques :
  + Choix des contrats concernés,
  + Fréquence (mensuelle, bimestrielle…),
  + Règles de consommation estimée ou réelle,
  + Activation/désactivation automatique.
* **Suivi des paiements** : page client regroupant :
  + Liste des factures,
  + Statuts de paiement,
  + Avoirs et échéanciers,
  + Relances envoyées et dates de contact.
* **Tableau de bord financier** : suivi consolidé :
  + Montants facturés, encaissés, en retard,
  + Alertes sur dépassements de seuils,
  + Indicateurs de performance financière.
* **Gestion des contentieux** : section dédiée avec :
  + Création de dossier litige,
  + Historique des communications,
  + Export pour service juridique ou recouvrement. : filtre par client, statut, période, type ; export PDF/Excel.
* **Facture manuelle** : formulaire complet (contrat, compteur, période, consommation, type de service).
* **Planification automatique** : tableau de configuration des règles de facturation périodique.
* **Suivi des paiements** : état des règlements, impayés, relances.
* **Historique et contentieux** : fiches client avec actions et justificatifs liés.

**6. Relation client et communication**

**Objectif**

Permettre au client de suivre son contrat, sa consommation et d'interagir avec le service gestionnaire.

**Fonctionnalités**

* Portail client : factures, historiques, contrats, interventions.
* Dépôt de réclamations et demandes spécifiques.
* Notifications automatisées personnalisées.
* Enquêtes de satisfaction avec analyse automatique.

**Interfaces principales**

* **Portail client (frontend dédié)** : vue tableau de bord personnel, consommation graphique, téléchargement factures.
* **Centre de messages** : historique des notifications reçues.
* **Formulaire de contact/réclamation** : motif, catégorie, pièce jointe.
* **Interface d’enquête de satisfaction** : réponse en ligne, analyse des tendances.

**7. Statistiques, reporting et aide à la décision**

**Objectif**

Offrir des outils décisionnels pour le suivi de la performance technique, financière et commerciale.

**Fonctionnalités**

* Tableaux de bord sectorisés.
* Rapports : consommation, incidents, finances, satisfaction client.
* Détection de fuites et fraudes.
* Prédiction de consommation.

**Interfaces principales**

* **Dashboard global** : indicateurs clés (conso moyenne, taux de facturation, incidents).
* **Rapports détaillés** : exportables en PDF/Excel, filtres avancés.
* **Outils d’analyse prédictive** : visualisation des tendances, alertes seuils.

**8. Gestion documentaire centralisée (GED)**

**Objectif**

Centraliser tous les documents et garantir leur traçabilité et validité juridique.

**Fonctionnalités**

* Archivage numérique sécurisé.
* Signature électronique intégrée (Yousign, DocuSign).
* Historique des versions et modifications.

**Interfaces principales**

* **Bibliothèque documentaire** : recherche par type, date, client.
* **Vue contrat/document** : affichage et téléchargement.
* **Suivi de signature** : statut des documents en attente ou complétés.

**9. Administration et paramétrage**

**Objectif**

Permettre aux administrateurs de configurer facilement tous les éléments fonctionnels du système.

**Fonctionnalités**

* Paramétrage des workflows, règles métier, modèles de documents.
* Configuration des grilles tarifaires sans intervention technique.
* Interface d’administration complète.

**Interfaces principales**

* **Configuration des workflows** : sélection des étapes, conditions, actions.
* **Modèles de documents** : édition et prévisualisation.
* **Paramétrage tarifaire** : grille dynamique par tranche, client, saison.
* **Journal système** : suivi des actions administratives.

**Technologies proposées**

* Microservices NestJS (backend principal).
* Spring Boot pour facturation intensive.
* RabbitMQ pour la communication interservices.
* PostgreSQL pour la base de données principale.
* React pour l’interface d’administration et le portail client.
* Intégrations : IoT (via MQTT ou HTTP), Mailer (SendGrid/Mailjet), Signature électronique (Yousign).

**Exemple de processus guidé via workflow**

**Fermeture d’un compteur**

1. Création de l’affaire
2. Détection du type de compteur
3. Intervention terrain si compteur mécanique
4. Clôture du contrat
5. Génération de la facture finale
6. Notification du client
7. Archivage du dossier

Chaque étape est automatisée ou nécessite une action validée, le tout traçable et historisé dans le plan d’action.