Parfait — j’ai lu tes deux fichiers et j’ai consolidé une **architecture cible** + une **arborescence détaillée avec l’utilité de chaque dossier/fichier**.

Je t’ai tout mis proprement dans un document à côté de la conversation, prêt à imprimer et à partager à l’équipe (modules, endpoints, patterns, check‑list qualité, livrables par jour, etc.).

Tu le vois dans le panneau de droite : **« E COMPTA IA INTERNATIONAL — Architecture & Arborescence projet (contrôle devs) »**.

**🎯 Contexte & objectifs**

Basé sur la feuille de route "E COMPTA IA INTERNATIONAL" (14 jours, backend Spring Boot + PostgreSQL, frontend Angular, i18n, multi‑tenant, standards comptables OHADA/IFRS/GAAP, reporting PDF, fiscalité, assistant IA, SaaS, déploiement Docker/Nginx), voici l’architecture **cible** et l’**arborescence détaillée** des fichiers/folders avec l’utilité de chacun pour contrôler le travail des développeurs.

**🏗️ Architecture cible (vue d’ensemble)**

* **Frontend (Angular)** : App web multi‑langues et multi‑devises, modules métier (auth, tableau de bord, plan comptable, tiers, écritures, journaux, reporting, états financiers, fiscalité, rapprochements, admin/tenant, facturation SaaS), i18n avec @angular/localize + JSON d’assets.
* **Backend (Spring Boot 3)** : API REST, sécurité JWT, multitenant **schema‑per‑tenant** (ou **discriminator** si contrainte d’infra), JPA/Hibernate, patterns **Factory** (standards comptables), **Strategy** (calculs fiscaux/rapprochement), **Rules Engine light** (règles pays), génération PDF, tests.
* **DB (PostgreSQL)** : Instance unique, schémas par tenant (ex : tenant\_bf, tenant\_ci …) ; tables normalisées (entreprise, utilisateur, plan comptable, journaux, écritures, tiers, taxes…).
* **IA (service)** : Intégration API LLM, prompts contextuels par pays/standard, base documentaire (OHADA/IFRS/GAAP) en lecture.
* **DevOps** : Dockerfiles FE/BE, docker-compose (dev/prod), Nginx reverse proxy + SSL, CI/CD (GitHub Actions), monitoring/logs/backup.

**📁 Arborescence proposée (monorepo)**

E-COMPTA-IA/

├─ frontend/ # Angular app (SPA)

│ ├─ src/

│ │ ├─ app/

│ │ │ ├─ core/ # services globaux (auth, http, i18n, interceptors)

│ │ │ │ ├─ services/

│ │ │ │ │ ├─ auth.service.ts # Login/logout, refresh token, profil

│ │ │ │ │ ├─ http.service.ts # Wrapper HttpClient (baseURL, headers)

│ │ │ │ │ ├─ i18n.service.ts # Gestion locale, devise, fuseaux

│ │ │ │ ├─ interceptors/

│ │ │ │ │ ├─ auth.interceptor.ts # Ajout JWT, gestion 401/403

│ │ │ │ │ ├─ tenant.interceptor.ts # Injection X-Tenant-Id ou X-Country-Code

│ │ │ │ ├─ guards/

│ │ │ │ │ └─ auth.guard.ts # Guard routes protégées

│ │ │ │ └─ models/ # Interfaces TypeScript partagées

│ │ │ │ ├─ user.model.ts

│ │ │ │ ├─ tenant.model.ts

│ │ │ │ ├─ accounting.model.ts # Account, CoA, JournalEntry, etc.

│ │ │ │ └─ tax.model.ts

│ │ │ ├─ shared/ # UI components partagés (tables, forms, modals)

│ │ │ ├─ features/

│ │ │ │ ├─ auth/ # Module Auth (login/registration)

│ │ │ │ ├─ dashboard/ # KPI internationaux, sélection pays/standard

│ │ │ │ ├─ accounting-plan/ # Plan comptable dynamique (OHADA/IFRS/GAAP)

│ │ │ │ ├─ parties/ # Tiers (clients/fournisseurs/banques)

│ │ │ │ ├─ entries/ # Saisie d’écritures + contrôles

│ │ │ │ ├─ journals/ # Journaux, lettrage minimal

│ │ │ │ ├─ reporting/ # GL, Balance, PDF, filtres multi‑pays

│ │ │ │ ├─ financial-statements/ # Bilan, CR, Flux de trésorerie

│ │ │ │ ├─ tax/ # Déclarations (TVA, IS…), calculs

│ │ │ │ ├─ reconciliations/ # Rapprochements bancaires & lettrage

│ │ │ │ ├─ admin/ # Gestion tenants, utilisateurs, rôles

│ │ │ │ └─ billing/ # Abonnements, paiements multi‑devises

│ │ │ ├─ app.component.\*

│ │ │ └─ app-routing.module.ts

│ │ ├─ assets/

│ │ │ ├─ i18n/

│ │ │ │ ├─ fr.json

│ │ │ │ ├─ en.json

│ │ │ │ └─ ar.json # autres langues au besoin

│ │ │ └─ templates/ # gabarits PDF, logos

│ │ └─ environments/

│ │ ├─ environment.ts

│ │ └─ environment.prod.ts

│ ├─ angular.json

│ ├─ package.json

│ ├─ tsconfig.json

│ └─ Dockerfile

│

├─ backend/ # Spring Boot API

│ ├─ src/main/java/com/ecomptai/

│ │ ├─ api/ # Contrôleurs REST (HTTP boundaries)

│ │ │ ├─ AuthController.java

│ │ │ ├─ TenantController.java

│ │ │ ├─ ChartOfAccountsController.java

│ │ │ ├─ PartiesController.java

│ │ │ ├─ EntriesController.java

│ │ │ ├─ JournalsController.java

│ │ │ ├─ ReportingController.java # GL, Balance

│ │ │ ├─ FinancialStatementsController.java # Bilan, Compte de résultat, Cashflow

│ │ │ ├─ TaxController.java

│ │ │ └─ ReconciliationsController.java

│ │ ├─ config/

│ │ │ ├─ OpenApiConfig.java # Swagger, OpenAPI 3

│ │ │ ├─ SecurityConfig.java # Spring Security + JWT + CORS

│ │ │ ├─ MultiTenantConfig.java # Résolution tenant (schema/discriminator)

│ │ │ ├─ CountrySpecificRules.java # Règles pays (mapping codes → règles)

│ │ │ └─ PdfConfig.java

│ │ ├─ security/

│ │ │ ├─ JwtAuthFilter.java

│ │ │ ├─ JwtService.java

│ │ │ └─ CustomUserDetailsService.java

│ │ ├─ tenant/

│ │ │ ├─ TenantContext.java

│ │ │ ├─ TenantResolver.java # lit en-tête X-Tenant-Id / X-Country-Code

│ │ │ └─ SchemaRoutingDataSource.java # DataSource multi‑schémas

│ │ ├─ domain/

│ │ │ ├─ common/

│ │ │ │ ├─ BaseEntity.java # id, audit fields (createdAt, etc.)

│ │ │ │ └─ Monetary.java # devise, arrondis, format

│ │ │ ├─ org/

│ │ │ │ ├─ Entreprise.java # pays (ISO 3166-1), standard, devise, locale

│ │ │ │ └─ Utilisateur.java

│ │ │ ├─ accounting/

│ │ │ │ ├─ Account.java # Compte comptable

│ │ │ │ ├─ ChartOfAccounts.java # Plan comptable

│ │ │ │ ├─ Journal.java

│ │ │ │ ├─ JournalEntry.java # Écriture

│ │ │ │ ├─ ThirdParty.java # Tiers

│ │ │ │ └─ CurrencyRate.java

│ │ │ ├─ tax/

│ │ │ │ ├─ TaxRegime.java # TVA, IS, etc.

│ │ │ │ └─ TaxRate.java

│ │ │ └─ reporting/

│ │ │ └─ ReportSnapshot.java

│ │ ├─ repository/

│ │ │ ├─ EntrepriseRepository.java

│ │ │ ├─ UserRepository.java

│ │ │ ├─ AccountRepository.java

│ │ │ ├─ JournalRepository.java

│ │ │ ├─ JournalEntryRepository.java

│ │ │ └─ TaxRateRepository.java

│ │ ├─ service/

│ │ │ ├─ AuthService.java

│ │ │ ├─ TenantService.java

│ │ │ ├─ accounting/

│ │ │ │ ├─ CoAFactory.java # Factory (OHADA/IFRS/GAAP)

│ │ │ │ ├─ OhadaCoAFactory.java

│ │ │ │ ├─ IfrsCoAFactory.java

│ │ │ │ ├─ GaapCoAFactory.java

│ │ │ │ ├─ EntryService.java # Validation multi‑standards

│ │ │ │ └─ JournalService.java

│ │ │ ├─ reporting/

│ │ │ │ ├─ GeneralLedgerService.java # Grand Livre

│ │ │ │ └─ TrialBalanceService.java # Balance

│ │ │ ├─ financials/

│ │ │ │ ├─ FinancialStatementsFactory.java # Bilan/CR/Flux selon standard

│ │ │ │ └─ MappingTemplates.java # Mapping comptes → rubriques états

│ │ │ ├─ tax/

│ │ │ │ ├─ TaxCalculatorStrategy.java # Strategy (TVA, IS) par pays

│ │ │ │ ├─ BfTaxCalculator.java

│ │ │ │ └─ CiTaxCalculator.java

│ │ │ ├─ reconciliation/

│ │ │ │ ├─ BankReconciliationService.java

│ │ │ │ └─ MatchingEngine.java # Lettrage

│ │ │ ├─ pdf/

│ │ │ │ ├─ PdfRenderer.java # iText/OpenPDF/FlyingSaucer

│ │ │ │ └─ TemplatesLoader.java

│ │ │ └─ ai/

│ │ │ ├─ AssistantService.java # Appels LLM

│ │ │ └─ PromptTemplates.java # Prompts multi‑standards

│ │ ├─ dto/

│ │ │ ├─ AuthDtos.java # LoginRequest, TokenResponse...

│ │ │ ├─ AccountingDtos.java # AccountDto, EntryDto...

│ │ │ └─ TaxDtos.java

│ │ └─ EComptaIaApplication.java

│ ├─ src/main/resources/

│ │ ├─ application.yml # Profils dev/test/prod, CORS, JWT, multitenant

│ │ ├─ messages\_fr.properties # i18n backend

│ │ ├─ messages\_en.properties

│ │ ├─ db/migration/ # Flyway scripts (V1\_\_init.sql, …)

│ │ └─ templates/ # Freemarker/Thymeleaf pour PDF

│ ├─ pom.xml (ou build.gradle)

│ └─ Dockerfile

│

├─ devops/

│ ├─ docker-compose.dev.yml

│ ├─ docker-compose.prod.yml

│ ├─ nginx/

│ │ ├─ nginx.conf # Reverse proxy, SSL, gzip, cache

│ │ └─ sites-enabled/ecompta.conf

│ ├─ ci-cd/

│ │ └─ github-actions.yml # Build, tests, images, déploiement

│ ├─ k8s/ # (optionnel) manifests Helm/Plain YAML

│ └─ scripts/

│ ├─ wait-for-it.sh

│ └─ backup-db.sh

│

├─ docs/

│ ├─ api-contracts/ # OpenAPI YAML/JSON

│ ├─ accounting-standards/ # Mapping OHADA/IFRS/GAAP

│ ├─ rules/ # Règles pays (TVA, IS…)

│ └─ README.md

│

├─ .env.example # Variables (API\_URL, DB, JWT\_SECRET…)

├─ README.md

└─ LICENSE

**🧩 Utilité des dossiers/fichiers (résumé)**

* **frontend/src/app/core/** : couches transverses (auth, http, i18n, interceptors, guards, modèles). Garantit la cohérence des appels API, de la sécurité et de la localisation.
* **frontend/src/app/features/** : *verticals métier* alignés sur la roadmap (plan comptable, écritures, journaux, reporting, états, fiscalité, rapprochements, admin, billing). Chaque module a ses composants, services, routes, state (si NgRx), tests.
* **frontend/assets/i18n/** : fichiers JSON de traduction, clés stables, noms courts.
* **backend/api/** : API controllers REST (limiter à orchestration et validation simple).
* **backend/config/** : configuration technique (sécurité, multi‑tenant, OpenAPI, PDF, règles pays).
* **backend/security/** : JWT, filtres, services d’authentification.
* **backend/tenant/** : résolution du tenant (en‑tête, sous‑domaine), routage DataSource par schéma.
* **backend/domain/** : entités métier (entreprise, utilisateur, comptes, journaux, écritures, taxes…).
* **backend/repository/** : interfaces Spring Data JPA.
* **backend/service/** : logique applicative. **Factory/Strategy** pour standards/impôts.
* **backend/dto/** : objets d’échange (entrées/sorties API), évite l’exposition directe des entités.
* **backend/resources/db/migration/** : scripts Flyway versionnés (création schémas, tables, indexes).
* **devops/** : conteneurisation, reverse proxy, CI/CD, scripts d’exploitation.
* **docs/** : contrats d’API, mappings comptables, règles pays (source de vérité non‑code).

**🔌 Endpoints REST (exemples de contrat)**

* **Auth** : POST /api/auth/login, POST /api/auth/refresh, GET /api/users/me
* **Tenant/Entreprise** : POST /api/tenants, GET /api/tenants/{id}
* **Plan comptable** : GET /api/coa?standard=OHADA&country=BF, POST /api/coa/accounts
* **Tiers** : GET /api/parties, POST /api/parties
* **Écritures** : POST /api/entries, GET /api/entries?journal=VENTES&from=...
* **Journaux** : GET /api/journals
* **Reporting** : GET /api/reporting/gl?period=2025-Q3, GET /api/reporting/trial-balance
* **États financiers** : GET /api/financials/balance-sheet?standard=OHADA&country=BF
* **Fiscalité** : POST /api/tax/calc?country=BF&type=TVA, GET /api/tax/templates?country=BF
* **Rapprochements** : POST /api/reconciliations/bank
* **IA** : POST /api/ai/assist (payload: contexte entreprise/pays/standard + demande)

Tous les endpoints prennent l’en‑tête **X-Tenant-Id** (ou X-Country-Code) et exigent un **JWT**.

**🧠 Patterns & règles clés**

* **Factory** (Plan comptable/États) : CoAFactory → OhadaCoAFactory, IfrsCoAFactory, GaapCoAFactory.
* **Strategy** (Fiscalité, rapprochement) : TaxCalculatorStrategy → implémentations BF/CI/…
* **Multi‑tenant** : SchemaRoutingDataSource + TenantResolver + X-Tenant-Id ; alternatives possibles (colonne discriminante ou DB par client) selon coûts/volumes.
* **Validation multi‑standards** : services EntryService appliquent règles standard + règles pays (CountrySpecificRules).

**⚙️ Config & environnements**

* **application.yml** : profils dev, test, prod (CORS, JWT, DSN Postgres, stratégie multi‑tenant).
* **Variables .env** : DB\_URL, DB\_USER, DB\_PASS, JWT\_SECRET, LLM\_API\_KEY, BASE\_URL.
* **i18n** : langues min. fr, en (ajouter ar si besoin), formats date/nombre par locale.
* **PDF** : modèles Freemarker/HTML + renderer (OpenPDF/FlyingSaucer) ; assets sous resources/templates.

**✅ Contrôles qualité (check‑list pour les devs)**

1. **Tests** : unitaires (services, factories), d’intégration (API), E2E (Cypress côté FE).
2. **Couverture** : >80% services critiques (Entries, Reporting, Tax).
3. **Swagger** disponible en dev : /swagger-ui ; contrat versionné sous docs/api-contracts.
4. **Flyway** : base initialisée automatiquement en dev, migrations idempotentes.
5. **Security** : JWT HS256/RS256, mots de passe hashés (BCrypt), CORS minimal.
6. **Observabilité** : logs structurés JSON, health checks /actuator/health.
7. **CI/CD** : build FE/BE, tests, image Docker, déploiement, tag version (v1.0.0-j14).

**🗂️ Livrables attendus par jour (extrait, aligné sur la roadmap)**

* **J1** : Auth + Sélecteur pays/tenant + Multi‑tenant configuré ; SecurityConfig, MultiTenantConfig, TenantResolver.
* **J2–J3** : CoAFactory + implémentations OHADA/IFRS/GAAP ; EntriesController/Service avec validations.
* **J5** : GeneralLedgerService, TrialBalanceService + endpoints + PDF.
* **J6** : AssistantService (stub prod) + prompts.
* **J8** : FinancialStatementsFactory + mapping comptes→rubriques.
* **J9** : BankReconciliationService + MatchingEngine.
* **J10** : TaxCalculatorStrategy + templates pays.
* **J11** : billing/ (frontend) + endpoints abonnement (backend) ; base multi‑devises.

**🔐 Conventions de nommage**

* **Java** : UpperCamelCase (classes), lowerCamelCase (champs), suffixes clairs (\*Controller, \*Service, \*Repository).
* **TypeScript** : kebab-case fichiers, UpperCamelCase interfaces modèles, dossiers par feature.
* **DB** : snake\_case pour les colonnes, clés primaires id (UUID recommandé), FK explicites.

**📌 Notes d’implémentation rapides**

* **Sélecteur de standard/pays** côté FE (header) → stocké dans state + envoyé en header.
* **Mapping OHADA** minimal viable (PCG SYSCOHADA) pour V1 ; IFRS/GAAP progressifs.
* **Règles pays** chargées au démarrage (cache) + surcharge par tenant.
* **Exports** : PDF + CSV pour GL/Balance.

**🧭 Prochaines étapes**

1. Initialiser le monorepo + squelettes (FE/BE/DevOps).
2. Pousser les contrats OpenAPI et la première migration Flyway.
3. Brancher le sélecteur pays/standard + JWT + multitenant.
4. Attaquer CoAFactory et EntriesService.

*Document prêt pour audit/contrôle quotidien des livrables.*