# W3 Suite – Specifiche Tecniche, Architettura & Stack (Technical Canvas)

## 1) Stack e versioni proposte

* **Linguaggio**: TypeScript **5.x**
* **Backend (Suite & Brand)**: **NestJS 10+**, Node.js **20 LTS**
* **Frontend**: React **18.x**, Vite, Tailwind CSS **3.x** + **shadcn/ui**, Lucide, Framer Motion (animazioni leggere)
* **DB OLTP**: PostgreSQL **16** + **Drizzle ORM** + **RLS**
* **Cache/Queue**: Redis **7** + BullMQ
* **Auth**: OIDC/OAuth2 (Keycloak/Identity Provider), **MFA** (TOTP/WebAuthn)
* **Storage**: S3/MinIO per asset & documenti; opzionale **pgvector** per embeddings (RAG)
* **Analytics/DWH**: Postgres (schema DWH) o BigQuery in futuro; ETL (workers)
* **Observability**: OpenTelemetry, Prometheus/Grafana, ELK/Vector
* **Testing**: Vitest/Jest, Supertest (API), Playwright (e2e)
* **CI/CD**: GitHub Actions; Docker; pnpm workspaces (+ Turborepo opz.)

## 2) Monorepo – struttura definitiva

w3-suite/  
├─ apps/  
│ ├─ frontend/  
│ │ ├─ web/ # SPA Suite (tenant)  
│ │ │ └─ src/  
│ │ │ ├─ app/ # router, providers (Auth, Query), theme  
│ │ │ ├─ layout/ # AppShell, Header, Sidebar (glass)  
│ │ │ └─ features/ # feature-first (cassa, magazzino, crm, settings, gare, report, hr, cms)  
│ │ └─ brand-web/ # SPA Brand Interface (HQ)  
│ │ └─ src/  
│ │ ├─ app/  
│ │ ├─ layout/  
│ │ ├─ core/ # BrandAccessProvider, guards, scope switcher  
│ │ └─ features/ # sales, marketing, ops, amministrativo, dev-tools  
│ └─ backend/  
│ ├─ api/ # NestJS – Suite (OLTP)  
│ │ └─ src/  
│ │ ├─ core/ # auth, rbac, tenancy, audit, settings-base  
│ │ ├─ db/  
│ │ │ ├─ schema/ # Drizzle schema per modulo  
│ │ │ └─ migrations/ # SQL con RLS & indici  
│ │ └─ modules/ # DDD-lite per dominio  
│ │ └─ <modulo>/{domain,application,infra,presentation}  
│ ├─ brand-api/ # NestJS – Brand (DWH + orchestrazione)  
│ │ └─ src/{auth,rbac,analytics,pricing,campaigns,templates,propagation}  
│ ├─ workers/  
│ │ └─ brand-propagation/ # consumer eventi brand→tenant (BullMQ)  
│ └─ cms-render/ (opz) # Astro/Workers per landing pubbliche  
├─ packages/  
│ ├─ ui/ # design system (glass, componenti)  
│ ├─ tokens/ # Tailwind preset + CSS vars brand W3  
│ ├─ sdk/ # client TS: sdk.api (suite), sdk.brand (brand-api)  
│ ├─ dwh/ # query DWH tipizzate (zod/types)  
│ ├─ cms-core/ # schema/logic CMS (sites/pages/forms, publish)  
│ ├─ cms-render/ # blocchi render (Astro/React)  
│ └─ agents/  
│ ├─ runner/ # agent runner (codegen/qa/docs) → PR  
│ └─ prompts/ # prompt templates e convenzioni  
├─ db/  
│ ├─ oltp/ # migrazioni OLTP (Suite)  
│ ├─ brand/ # migrazioni Brand API (cataloghi, deployments)  
│ └─ dwh/ # schema DWH (read-only)  
├─ configs/ # eslint, prettier, tsconfig, vite, jest  
├─ .github/workflows/ # CI separata per ogni app + agent orchestrator  
├─ docker/ # compose dev: postgres, redis, keycloak  
├─ pnpm-workspace.yaml | turbo.json | tsconfig.base.json | package.json

**Path alias (tsconfig.base):** @ui/\*, @tokens/\*, @sdk/\*, @dwh/\*, @cms-core/\*.

## 3) Sicurezza & Identità

* **OIDC** con flusso Authorization Code + PKCE; refresh via cookie httpOnly, access JWT in memoria.
* **MFA** obbligatoria (TOTP/WebAuthn); **step-up MFA** su operazioni sensibili (billing, deploy brand, export dati).
* **RBAC**:
  + Tenant: roles, role\_perms, user\_assignments (con scope tenant/RS/store), user\_extra\_perms (grant/revoke).
  + Brand: ruoli per **settore** (marketing/sales/ops/amministrativo) + scope rete (all|region|RS|store).
* **Audit**: ogni cambio permessi, publish CMS, propagazione brand, esportazioni.
* **Rate limiting** & **IP allowlist** per integrazioni esterne.

## 4) Database OLTP (PostgreSQL + Drizzle + RLS)

### 4.1) Principi

* **DB unico** per tutti i tenant → ogni tabella multi-tenant ha tenant\_id.
* **RLS obbligatoria**:

ALTER TABLE <table> ENABLE ROW LEVEL SECURITY;  
CREATE POLICY rls\_<table> ON <table>  
 USING (tenant\_id = current\_setting('app.tenant\_id')::uuid);

* Middleware applicativo imposta SET app.tenant\_id = '<uuid>' per richiesta.

### 4.2) Tabelle core (estratto)

* tenants(id, name, ...)
* legal\_entities(id, tenant\_id, name, vat, billing\_profile\_id, ...)
* stores(id, tenant\_id, legal\_entity\_id, name, address, region, ...)
* users(id, tenant\_id, email, ... )
* roles(id, tenant\_id, name) ; role\_perms(id, role\_id, perm)
* user\_assignments(id, user\_id, role\_id, scope\_type, scope\_id)
* user\_extra\_perms(id, user\_id, perm, mode)
* **Billing**: rs\_billing\_profiles, store\_billing\_overrides

### 4.3) CRM

* leads(id, tenant\_id, store\_id, source, campaign\_id, status, score, ...)
* lead\_events(id, lead\_id, channel, payload, created\_at)
* customers(id, tenant\_id, ... ), contacts(...)
* campaigns(id, tenant\_id, name, status, ... )
* campaign\_channels(id, campaign\_id, type, template\_id, ... )
* channel\_templates(id, tenant\_id, type: waba|email|sms|telegram|messenger, status, content)
* messages(id, tenant\_id, channel, to, status, template\_id, campaign\_id, ...)

### 4.4) Cassa & Magazzino

* products(id, tenant\_id, sku, name, price\_cents, tax\_rate, ...)
* orders(id, tenant\_id, store\_id, customer\_id, total\_cents, status, ...)
* order\_items(id, order\_id, product\_id, qty, price\_cents)
* payments(id, order\_id, method, amount\_cents, status, receipt\_no)
* stock\_items(id, tenant\_id, store\_id, product\_id, qty)
* stock\_movements(id, stock\_item\_id, type, delta, reason, ref\_id)
* transfers(id, tenant\_id, from\_store\_id, to\_store\_id, status); transfer\_items(...)

### 4.5) Gare & Commissioni

* tenders(id, tenant\_id, ...), lots(...), applications(...), awards(...)
* commission\_rulesets(id, tenant\_id, version, active); commission\_rules(...)

### 4.6) HR

* courses(id, tenant\_id, title), sessions(course\_id, store\_id, date), enrollments(user\_id, session\_id, status)

### 4.7) Document Intelligence

* documents(id, tenant\_id, type, storage\_url, hash, ...)
* extracted\_fields(id, document\_id, key, value, confidence)

### 4.8) CMS Store (vetrina)

* cms\_sites(id, tenant\_id, legal\_entity\_id, store\_id, subdomain, custom\_domain)
* cms\_pages(id, tenant\_id, site\_id, slug, draft, published, is\_published, published\_at)
* cms\_forms(id, tenant\_id, site\_id, fields, campaign\_id, on\_submit)

**Indici consigliati:** (tenant\_id, created\_at), (tenant\_id, store\_id), (tenant\_id, status), full-text su contenuti CMS/Docs.

## 5) API & Integrazioni

### 5.1) Auth & Profilo

* POST /auth/login (OIDC redirect), GET /me (capabilities + scope aggregato: ruolo + extra grant/revoke). ### 5.2) Moduli (pattern REST)
* GET/POST/PATCH/DELETE /<modulo> con guard Can('<namespace>.<action>') e filtro scope (RS/store). ### 5.3) Webhook Hub (multi-tenant)
* Endpoint unico /webhooks/<provider> (Meta, WhatsApp, Google Leads, Email events):
  + **Verifica firma** (provider-specific), **idempotenza** (event\_id), **enqueue** (BullMQ), **retry**
  + **Routing** per page\_id/phone\_number\_id/google\_customer\_id → tenant/RS/store ### 5.4) Canali comunicazione
* **WhatsApp Business API** (HSM/template, session, media)
* **Email** (SMTP/Mailgun/SendGrid), **SMS** (Twilio/Skebby)
* **Telegram/Messenger** (bot/webhook) ### 5.5) GTM/GA4 & Meta CAPI
* Server-side events su lead/ordini; mapping UTM; deduplica.

## 6) AI Layer

### 6.1) Predittivo

* **Lead Scoring**: endpoint /ai/predict/lead-score → restituisce score + **top-factors** (explainable). Feature store per tenant; modelli baseline (LR/XGBoost) + calibrazione per tenant.
* **Next Best Offer/Action**: API su contesto lead/cliente; cache 5–15m. ### 6.2) Forecast & Simulazioni (Gare/Commissioni)
* /ai/forecast/sales (per store/sku), /ai/simulate/commission con rulesetId + assumptions (what-if). Output tabellare + serie. ### 6.3) Document Intelligence
* /docs/upload → OCR/estrazione → extracted\_fields + confidenza; validazione e linking a entità CRM. ### 6.4) Chat Assistente
* /ai/chat con **tools** sicure (db.readonly, crm.getLead, orders.byCustomer, kb.search). RAG su kb (pgvector/Weaviate), citazioni.

**Sicurezza AI**: RLS applicata ai tools, PII masked nei log, audit per prompt/risposte, quota per tenant.

## 7) Brand Interface (HQ)

* **Brand API** (Nest separato): legge **solo DWH**, scrive/propaga via **eventi**.
* **Moduli brand**: analytics (KPI rete), pricing (listini/versioni), campaigns, templates (WABA/email/SMS), propagation (publish → tenant).
* **RBAC settori**: marketing/sales/ops/amministrativo + scope rete.
* **DWH Schema** (base):
  + **Fact**: fact\_sales, fact\_leads, fact\_stock, fact\_messages
  + **Dimension**: dim\_time, dim\_store, dim\_region, dim\_legal\_entity, dim\_product
* **Propagation flow**: evento brand.<topic>.<action> → worker brand-propagation → chiamate **Tenant API** (mTLS/JWT audience tenant-internal) → audit deployment.

## 8) Frontend

* **Design System**: packages/ui + packages/tokens (palette W3: arancio/viola/nero; glassmorphism; shadows; radius).
* **Access provider**: /me (suite) e /brand/me (brand) → menu/rotte dinamiche per **capabilities** e **scope**.
* **CMS Editor**: feature cms in apps/frontend/web; blocchi (Hero, FeatureList, Pricing, Form, CTA); publish flow.
* **Brand Dev Tools**: pagina unica con tab **Codegen/QA/Docs** che triggera workflow GitHub (orchestratore OpenAI).

## 9) Performance & Caching

* **DB**: indici per (tenant\_id, created\_at), partizioni per dati voluminosi (eventi, logs).
* **Cache**: Redis per config, feature flags, lookup mapping webhook; HTTP caching per CMS published.
* **Rate limit**: per IP/tenant su webhook e API pubbliche.
* **Pooling**: pg-pool condiviso, statement timeout, circuit breaker (p50/p95 target).

## 10) Observability

* **Log** strutturati (requestId, tenantId, userId, scope) → ELK/Vector.
* **Metrics** (Prometheus): RPS, latency, error rate, queue depth, webhook success, AI latency.
* **Tracing** (OpenTelemetry): span per modulo; link PR number nelle release.

## 11) CI/CD & Quality Gates

* **Workflows** separati: FE suite, FE brand, API suite, Brand API, Workers.
* **Quality gates**: lint, typecheck, unit, integration, e2e (se toccate rotte), **QA Agent** (test auto), **security scan** (Snyk/Dependabot).
* **Release**: semantic versioning per app; changelog automatico; feature flags.

## 12) Sicurezza applicativa

* CSP stretta, X-Content-Type-Options, HSTS, SameSite/Lax cookies, CSRF per form non OIDC.
* Secret management (Vault/SSM), rotazione chiavi, least-privilege su IAM/DB.
* Dati sensibili cifrati at-rest (S3 SSE) e in-transit (TLS 1.2+).

## 13) Testing strategy

* **Unit** per services/controller/utility.
* **Integration** su DB ephemeral (schema isolato per test).
* **E2E**: Supertest (API) + Playwright (FE).
* **Fixture**: seed 2 tenant (A/B) per verificare isolamento RLS.

## 14) Environments & Config

* dev (Docker Compose: Postgres, Redis, Keycloak), staging, prod.
* Env keys per app: DATABASE\_URL, REDIS\_URL, OIDC\_\*, WABA\_\*, MAIL\_\*, SMS\_\*, S3\_\*, AI\_\*.
* Feature flags (moduli/esperimenti AI) per tenant/brand.

## 15) Agenti di coding (Dev Productivity)

* **Runner interno** (OpenAI) + adapters (Lovable/Replit/Cursor) → PR su path **whitelist**.
* **Orchestratore** (GitHub Actions): pipeline **Codegen → QA → Docs**, gating CI e review umana.
* **Spec YAML** esempi (mini):

module: customers  
entity: { Customer: { id: uuid, tenantId: uuid, firstName: string, lastName: string, email: string(unique) } }  
api: { basePath: /api/v1/customers, crud: true }  
ui: { pages: [CustomersList, CustomerForm] }  
permissions: { namespace: settings.customers, defaults: { roles: [admin, manager\_rs] } }

## 16) SLO/Piani di capacità

* **API OLTP**: p95 < 250ms, errore < 0.5%; **Webhook**: ack < 200ms (enqueue), success > 99.9%.
* **FE**: TTI < 3s su 4G; Lighthouse > 90.
* **AI**: p95 predict < 150ms (cache), extract < 5s (OCR).

## 17) Appendice – Migrazione RLS tipo

-- esempio generico per tabella multi-tenant  
CREATE TABLE leads (  
 id uuid primary key default gen\_random\_uuid(),  
 tenant\_id uuid not null,  
 store\_id uuid not null,  
 source text not null,  
 status text not null,  
 created\_at timestamptz default now()  
);  
ALTER TABLE leads ENABLE ROW LEVEL SECURITY;  
CREATE POLICY rls\_leads ON leads  
 USING (tenant\_id = current\_setting('app.tenant\_id')::uuid);  
CREATE INDEX idx\_leads\_tenant\_created ON leads(tenant\_id, created\_at);

## 18) Glossario

* **Tenant**: organizzazione/cliente.
* **RS**: ragione sociale del tenant.
* **Store**: punto vendita assegnato a una RS.
* **RLS**: Row Level Security di Postgres.
* **DWH**: data warehouse per analisi cross-tenant (Brand).
* **WABA**: WhatsApp Business API.
* **MFA**: multi-factor authentication (TOTP/WebAuthn).