Construction Manager API – Technical Specification

1. Technology Stack

Component	Version (as of May 2025)	Notes / Rationale
Java	21 LTS	Latest long-term-support release; records, virtual threads
Spring Boot	3.3.x	Built on Jakarta EE 10; native GraalVM executables
Spring Framework	6.3.x	Reactive support, AOT, declarative HTTP clients
Spring Security	6.3.x	Authorization Manager API, method security on records
Spring Data JPA	3.3.x	Record-based projections, Jakarta Persistence 3.2
Hibernate	6.5.x	SQL 2011 support, multi-tenant enhancements
PostgreSQL	16.x	Primary production database engine
H2 Database	2.2.x	In-memory DB for unit & integration tests
Liquibase	4.24.x	Version-controlled schema migrations
MapStruct	1.6.x	Compile-time DTO mapping, supports Java 21 records
Bucket4j	8.x	Token-bucket rate limiting filter
jjwt	0.12.4	JWT signing (ECDSA) and validation

Password Hashing	Argon2 (spring-security-crypto)	Stronger than BCrypt
Micrometer + Prometheus	1.14.x	Metrics collection & scraping
OpenTelemetry	1.40.x	Distributed tracing (OTLP exporter)
springdoc-openapi	3.2.x	Generates Swagger UI 5
Testcontainers	2.4.x	Dockerised PostgreSQL in integration tests
JUnit Jupiter	5.11.x	Unit & slice tests
WireMock	3.5.x	HTTP stubbing for external integrations
Gradle	8.7	Kotlin DSL build scripts with version catalogs
Docker	25.x	Multi-stage build for slim images
Kubernetes	1.31	Helm charts for dev-test-prod clusters
GitHub Actions		CI/CD: build, test, SBOM, Docker push, Helm deploy

2. Architectural Guidelines

• **Hexagonal ('Ports & Adapters') architecture** – controllers, services, repositories clearly separated; domain layer unaware of Spring.

• **Package-by-feature** under `com.acme.construction`: each feature (project, invoice, report) exposes api, domain, infra sub-packages.

• **Record-based DTOs** – Java 21 records for immutable DTOs/commands; validation via Jakarta Validation 3.1.

• **MapStruct mappers** generate adapters between records and JPA entities.

• **Transactional boundaries** sit at service layer (`@Transactional` on app-service classes).

• **Soft deletes** implemented with `deleted_at` column + Hibernate filters.

• **Multi-tenant ready**: company_id discriminator column prepared for future SaaS mode.

• **Observability first**: Micrometer metrics, OTEL traces, structured JSON logs (logback).

3. Controllers & Endpoint Mapping

AuthController

Method & Path	Purpose	Produces
POST /auth/login	Authenticate user and issue JWT tokens	application/json
POST /auth/refresh	Renew an expired access token	application/json
GET /auth/me	Return profile of current user	application/json

UserController

Method & Path	Purpose	Produces
GET /users	Paginated list of users	application/json
POST /users	Create a new user	application/json
PUT /users/{id}/role	Change role or deactivate user	application/json

ProjectController

Method & Path	Purpose	Produces
GET /projects	List projects	application/json
POST /projects	Create project	application/json
GET /projects/{id}	Project details	application/json
PUT /projects/{id}	Update project	application/json
DELETE /projects/{id}	Archive project	application/json

MilestoneController

Method & Path	Purpose	Produces
GET /projects/{id}/milestones	List milestones	application/json
POST /projects/{id}/milestones	Add milestone	application/json
PATCH /milestones/{id}/status	Update milestone status	application/json

ContractorController

Method & Path	Purpose	Produces
GET /contractors	List contractors	application/json
POST /contractors	Add contractor	application/json
POST /projects/{id}/contractors/{contrId}	Assign contractor to project	application/json

InvoiceController

Method & Path	Purpose	Produces
GET /invoices	Global invoice list	application/json
GET /projects/{id}/invoices	Project invoices	application/json
POST /projects/{id}/invoices	Create invoice	application/json
PATCH /invoices/{id}/approve	Approve invoice	application/json

MaterialPlanController

Method & Path	Purpose	Produces
GET /projects/{id}/material-plans	List material plans	application/json

POST /projects/{id}/material-plans	Create material plan	application/json
POST /material-plans/{id}/items	Add item to material plan	application/json

DailyReportController

Method & Path	Purpose	Produces
GET /projects/{id}/daily-reports	List daily reports	application/json
POST /projects/{id}/daily-reports	Create daily report	application/json
POST /daily-reports/{id}/attachments	Add attachment to report	application/json

IssueController

Method & Path	Purpose	Produces
GET /projects/{id}/issues	List issues/RFIs	application/json
POST /projects/{id}/issues	Create issue/RFI	application/json
PATCH /issues/{id}/resolve	Resolve issue	application/json

AttachmentController

Method & Path	Purpose	Produces
POST /attachments	Upload attachment	application/json
GET /attachments/{id}/download	Download attachment	application/json

ReportController

Method & Path	Purpose	Produces

GET /reports/budget?projectId={id}	Budget vs actual report	application/json
GET /reports/overdue	Overdue milestones report	application/json

JobController

Method & Path	Purpose	Produces
POST /internal/jobs/dispatch-overdue	Trigger overdue alert job	application/json

4. Security & Compliance

• **JWT (ECDSA-256)** – short-lived access token (15 min) + refresh token (7 days).

• **Argon2 password encoding** with 1 GiB memory cost (tuned).

• **Role-based access control** via Spring AuthorizationManager; method-level `@PreAuthorize`.

• **Rate limiting**: Bucket4j filter on `/auth/login` and all mutating endpoints (5 r/s IP).

• **CORS**: allow configured SPA origins only; preflight cached 1 h.

• **CSRF protection** enabled for same-origin browser clients (double-submit cookie).

• **Content Security Policy** headers injected by `SecurityFilterChain`.

• **Audit trail**: JPA auditing (`createdBy`, `modifiedBy`) + entity revision history via Envers.

• **Secrets management**: Spring Cloud Vault (Hashicorp) for DB creds & JWT keys.

5. Database & Persistence

• **PostgreSQL 16** with UUID primary keys (`uuid-ossp`).

• Liquibase changesets stored under `db/changelog` and executed on startup.

• Entities are annotated with Jakarta Persistence 3.2; lazy-loading and fetch joins tuned.

• Read-only queries use Spring Data projections (records) to avoid N+1 issues.

• Full-text search extensions (pg_trgm) prepared for issue descriptions.

6. Build, Test & Deployment Pipeline

1. **Static analysis** – Checkstyle, PMD, SpotBugs, SonarQube quality gate ≥ A.

2. **Unit tests** – JUnit 5, mutation coverage > 70 % (Pitest).

3. **Integration tests** – Spring Boot test slices + Testcontainers.

4. **SBOM** – CycloneDX generated and pushed to artifact registry.

5. **Docker build** – Multi-stage; final image ≤ 180 MB.

6. **Helm chart** – values for dev / staging / prod; blue-green strategy.

7. **Smoke tests** – Post-deployment health & readiness probes.

7. Observability

• Metrics: JVM, DB, HTTP server, custom business KPIs (`invoices_overdue_total`).

• Distributed tracing: OpenTelemetry instrumentation, traces exported to Grafana Tempo.

• Centralised logs: JSON logs shipped via Fluent Bit to OpenSearch.