Refactoring and Modernization Report — Team P03-08

Date: 2025-10-12

This report documents the current state of our application and recommends prioritized refactoring and operational improvements across backend (Spring Boot), frontend (React + Vite), and infrastructure (Docker/Kubernetes/CI). The aim is to increase security, correctness, maintainability, and developer experience while keeping delivery velocity high.

Table of contents

Executive summary

Priority buckets (P0 = highest impact, lowest risk):

- P0 Security & correctness
  - Replace custom JWT code with Spring Security + a vetted JWT library; centralize authn/authz.
  - Make data seeders dev-only and remove default admin credentials from prod paths.
  - Introduce Jakarta Validation on request DTOs and unify error responses.
  - Add health probes and resource policies in Kubernetes; stop using ephemeral storage for uploads in non-dev.
- P1 Maintainability & performance
  - Move business rules out of controllers into services; reduce duplication around JWT parsing and role checks.
  - Convert string status fields (e.g., approvalStatus) to enums; add DB indexes on hot fields.
  - Add Actuator + structured JSON logging; include request correlation IDs.
  - Harden Dockerfile (non-root, cache layers) and streamline CI caching.
- P2 Developer experience & UX
  - Adopt Java records for DTOs; add types on the frontend (TS or JSDoc) and consistent query key patterns.
  - Code split large routes, add error boundaries, and improve cache invalidation flows.

Current architecture snapshot

- Backend: Spring Boot 3.3.2, Java 21, JPA/Hibernate; REST controllers; custom `JwtUtil` for tokens.
- Frontend: React 19 + Vite 7, React Router 7, TanStack Query 5; modern ESLint config.
- Build: frontend built via `frontend-maven-plugin` and copied into Spring static resources.
- Runtime: Multi-stage Dockerfile (Temurin 21). Kubernetes manifests included; MySQL in compose/k8s.

Backend — detailed recommendations

1) Security: Spring Security + JWT library (P0)

Findings: Controllers perform ad-hoc JWT verification (see `AuthApi`, `EventApi`); `JwtUtil` implements signing/verification manually.

Actions:

- Add Spring Security stateless config and a `BearerTokenAuthenticationFilter` (or resource server path) to validate tokens centrally.
- Use a library such as `io.jsonwebtoken:jjwt-api` (or Auth0) to issue/verify tokens.
- Move role checks to annotations (e.g., `@PreAuthorize("hasAnyRole('ADMIN','ORGANIZER')")`). Controllers retrieve principals via `@AuthenticationPrincipal`.

Benefits: Single source of truth for auth, fewer security footguns, simpler controller code, consistent 401/403 handling.

2) Validation & error handling (P0)

Findings: Manual null/blank checks in controllers; `GlobalExceptionHandler` exists but not wired for validation.

Actions:

- Convert request DTOs to records with bean validation annotations and `@Valid` on controller params.
- Extend `GlobalExceptionHandler` to render validation errors consistently (field → message map).

3) Domain modeling & data access (P1)

Findings: `Event.approvalStatus` is a string; common query filters lack explicit indexes; `list()` builds `Specification` returning `null` on no filters.

Actions:

- Introduce `enum ApprovalStatus { PENDING, APPROVED, REJECTED }` with `@Enumerated(EnumType.STRING)`.
- Add indexes on `events(start_time)`, `events(category)`, and `events(organizer_email)`.
- In specifications, return `cb.conjunction()` when no predicates; prefer `Pageable` + projections for read-heavy endpoints.

4) Service layer & transactions (P1)

Findings: Controllers contain business logic (time validation, ownership, file deletion).

Actions:

- Move rules to services annotated with `@Transactional` where needed.
- Encapsulate file system access in a `StorageService` to prevent path traversal and simplify testing.

5) Observability (P1)

Actions:

- Add `spring-boot-starter-actuator` and enable liveness/readiness endpoints.
- Configure JSON logging; add a servlet filter to attach `X-Request-ID` for traceability.

Frontend — detailed recommendations

1) Routing & performance (P2)

Actions:

- Use `React.lazy` / `Suspense` for large routes (Admin/Organizer dashboards).
- Add an app-level error boundary and a query error boundary for TanStack Query.

2) Data layer hygiene (P2)

Actions:

- Standardize query keys (`['events', id]` , etc.) and consolidate an axios instance with token injection.
- Add optimistic updates and targeted invalidations for RSVP flows.

3) Type safety (P2)

Actions:

- Introduce TypeScript or add JSDoc typedefs; keep ESLint strict and add `eslint-plugin-import` for import order.

Infra & DevOps — detailed recommendations

1) Dockerfile hardening (P1)

Actions:

- Use maven:3.9-eclipse-temurin-21 for build with `--mount=type=cache,target=/root/.m2` .
- Run as non-root in the final image; add a `HEALTHCHECK` (Actuator once enabled) and `.dockerignore` .

2) Kubernetes polish (P0–P1)

Actions:

- Add liveness/readiness probes; set CPU/memory requests/limits.
- Use `PersistentVolumeClaim` for `/app/uploads` outside dev; move secrets (DB creds, JWT) to `Secret` .

3) CI/CD (now and later)

Actions:

- Keep both workflows: GitHub-hosted (when billing is available) and self-hosted (runs now).
- Upload JUnit & coverage artifacts on every run; build Docker with buildx; push images on main.

Testing strategy

- Backend: service unit tests; WebMvc slice tests with validation/security; Testcontainers-MySQL for ITs.
- Frontend: component tests for forms and routing; Playwright/Cypress for happy-path E2E.

Phased roadmap

- Phase 0 (today): Actuator, k8s probes, dev-only seeders, `.dockerignore`.
- Phase 1 (1–2 days): Spring Security + JWT library; DTO validation; controller cleanup.
- Phase 2 (2–4 days): Service layer consolidation; enums + indexes; Docker hardening; CI optimizations.
- Phase 3 (ongoing): Frontend type safety; lazy routes; observability enhancements.

File-by-file highlights (non-exhaustive)

- Backend
  - `util/JwtUtil.java` → Replace with library + filter; remove custom verification from controllers.
  - `controller/*Api.java` → Use `@PreAuthorize`; delegate to services; remove duplicated JWT parsing.
  - `model/Event.java` → Convert approvalStatus to enum; add indexes.
  - `repository/EventRepository.java` → Prefer `cb.conjunction()`; adopt `Pageable` consistently.
  - `config/DataSeeder.java` → Guard with `@Profile("dev")`; remove hardcoded admin password from prod paths.
  - `config/GlobalExceptionHandler.java` → Add validation & security exception mappers.
- Frontend
  - `src/App.jsx` and pages → Lazy load heavy routes; error boundaries; query key normalization.
  - `contexts/AuthContext` / `services/api` → Centralize auth token injection and cache invalidation.
- Infra
  - `Dockerfile` → Multi-stage with Maven builder cache; non-root; healthcheck; `.dockerignore`.
  - `k8s/*` → Probes, resources, securityContext, PVC for uploads, Secrets for sensitive config.

Prepared by: Team P03-08