Eindopdracht Backend
Thomas Oudhoff
2025/02

Medicatie inname App

Inhoudsopgave

Inleiding

Deze backend ondersteunt een medicatie-inname app met drie rollen (ADMIN, VERZORGER, GEBRUIKER). De API faciliteert beheer van gebruikers en medicatie, upload/download van bijsluiters, het vastleggen van toedieningen, beheer van innameschema's en notificatie-instellingen.
De architectuur is gelaagd (Controller → Service → Repository) met Spring Boot 3, Java 21, Spring Security voor authenticatie/autorisatie en Spring Data JPA (Hibernate) op MySQL. Validatie gebeurt met Bean Validation; fouten worden eenduidig gemapt naar HTTP-statuscodes via een @RestControllerAdvice. Voor demo/testdoeleinden is er data.sql, een Postman-collectie en 60/60 geautomatiseerde tests (unit + integratie).

Github Link:

ThomasOudhoff/medicijnInnameApp: Eindopdracht BackEnd NOVI hogeschool

Verantwoording keuzes

Architectuur & taal

- Spring Boot 3 + Java 21: volwassen ecosysteem, moderne LTS-runtime, veel out-of-the-box integraties (security, validation, actuator).
- Gelaagd: controllers dun (I/O + statuscodes), domeinlogica in services (transactiegrenzen), data-toegang via repositories. Dit bevordert testbaarheid en scheiding van verantwoordelijkheden.
- DTO's (Create/Update/Response) i.p.v. entiteiten in de API: voorkomt over-posting, maakt validatie en versiebeheer eenvoudiger.

Domeinmodel

- Rol als enum (ADMIN, VERZORGER, GEBRUIKER) i.p.v. aparte tabel: simpel, voldoende voor de opdracht; minder complex dan M2M user-roles.
- ZorgRelatie als aparte entiteit (M–N) tussen VERZORGER en patiënt (GEBRUIKER): autorisatie voor verzorgers kan hierdoor op patiëntniveau worden afgedwongen.
- MedicatieBijsluiter als 1–1 met gedeelde primary key (medicatie_id): maakt duidelijk dat een bijsluiter *behoort* bij één medicatie.
    - Bewuste trade-off: binaire data (BLOB) in de database voor eenvoud en transactie-consistentie; daarnaast is URL mogelijk wanneer bestanden extern gehost worden.
- InnameSchema en Toediening:
    - Schema is gekoppeld aan Medicatie én Gebruiker: eigenaarschap en filtering per patiënt zijn eenduidig.
    - Toediening verwijst naar Schema én Medicatie voor makkelijke rapportages en integriteitschecks.
- NotificatieInstellingen 1–1 met Gebruiker: één configuratieblok per gebruiker is logisch en eenvoudig.

Security & autorisatie

- HTTP Basic voor eenvoud (demo/POC) + method-level security met @PreAuthorize:
    - Self-service: patiënten mogen alleen eigen resources wijzigen.
    - Verzorgers: read-only voor gekoppelde patiënten (via OwnershipLookup checks).
    - Admin: volledige rechten.
- 403 vs 401: ongeautoriseerd (401) bij ontbreken/onjuiste credentials; 403 wanneer je wel ingelogd bent maar geen toegang hebt. Dit is in de API-handler geborgd.
- Wachtwoorden met {bcrypt} en DelegatingPasswordEncoder: veilig en compatibel met seed-data.

Validatie & foutafhandeling

- Bean Validation (@Valid) op DTO's; veldfouten worden expliciet teruggegeven (fieldErrors) met status 400.
- Domeinspecifieke checks mappen naar 404 (niet gevonden), 409 (conflict, b.v. e-mail al in gebruik), 413 (payload te groot), 415 (unsupported media type), 403 (forbidden).
- Bestandsvalidatie bij upload: verplicht veld, max. 5MB, alleen image/ of application/pdf.

Data & performance

- MySQL met JPA/Hibernate; LAZY waar mogelijk om onnodige data te vermijden.
- Bij download van bijsluiters gebruiken we ByteArrayResource en correcte Content-Type/Content-Disposition.
- Migrations: voor de opdracht volstaat data.sql + ddl-auto=update. (Zie doorontwikkeling.)

Testen

- Unit-tests voor services (o.a. getByIdOr404_found_ok, delete-paden, validaties bij setBijsluiterUrl).
- Integratietests met MockMvc voor controllers (minimaal twee, meer aanwezig), inclusief security-paden (200/401/403/404/409).

Documentatie & tooling

- README met quickstart, endpoints en autorisatiematrix.
- Postman-collectie + environment met volledige demo-flow (Admin, Verzorger, Self-service patiënt).
- Klassendiagram (UML) en twee sequentiediagrammen afgestemd op de daadwerkelijke methoden/URI's.

Mogelijke doorontwikkelingen

Security & beheer

- JWT/OAuth2 in plaats van Basic; refresh-tokens; password-reset en lockout-beleid.
- Fijner-granulaire rechten (b.v. per medicatie of per actie), audit-logging van wijzigingen.

Data & migraties

- Flyway/Liquibase voor versiebeheer van schema's (in plaats van ddl-auto), plus seed via migrations.
- Bestandsopslag naar S3/MinIO met pre-signed URLs; streaming downloads; virus-scan bij upload.

Kwaliteit & betrouwbaarheid

- Testcontainers voor echte MySQL in tests; contract tests voor API backward compatibility.
- CI/CD pipeline met build, tests, Sonar-checks en automatische artefacten.
- Observability: metrics (Micrometer/Prometheus), tracing (OpenTelemetry), gestructureerde logs.

Functionaliteit

- Notificaties echt verzenden (e-mail/push) via async jobs of event-queue (b.v. RabbitMQ/Kafka).
- Planning: reminder-cron op basis van schema; snooze-logica server-side.
- Rapportages: gemist/ingenomen per periode, export CSV/PDF.
- Zoek/sort/paginatie op lijsten; filters (per gebruiker, per medicatie, datumbereik).
- Internationalisatie (i18n) en toegankelijke foutmeldingen.

Veiligheid & privacy

- Encryptie van gevoelige velden (bijv. BLOBs) at rest; ETag/caching headers bij downloads.
- Rate-limiting/throttling op upload/download endpoints.

Deployability

- Docker (app + MySQL), compose voor lokale spin-up; k8s manifests voor productie.

- Feature flags voor gefaseerde uitrol van nieuwe functionaliteit.