**Seminarul 8 — Partea 3: Proiecte/teme (extinsă)**

Această Partea 3 reunește \*\*45 de proiecte\*\* distribuite pe trei niveluri — \*\*L1 (Fundamente)\*\*, \*\*L2 (Intermediar)\*\*, \*\*L3 (Avansat)\*\* — pentru a consolida competențele REST pe \*\*Node/Express\*\*. Fiecare proiect include: \*\*scop didactic\*\*, \*\*specificații\*\*, \*\*criterii de acceptare\*\*, \*\*pași recomandați\*\*, \*\*soluție (rezumat)\*\* și \*\*AI‑assist (VSL)\*\*. La final, ai \*\*arhive ZIP\*\* (starter standalone și monorepo PNPM) și un \*\*ghid DOCX\*\* de utilizare. Testele „smoke” (Vitest & Jest) rulează imediat; testele de „pattern” sunt marcate `todo`, pentru a le activa când vrei să verifici automat prezența conceptelor cheie (ETag, Idempotency‑Key etc.).

**Cum folosești pachetul de proiecte**

1) Alege proiectul (ex. `l2-p02`) și extrage directorul din arhiva „standalone” \*\*sau\*\* lucrează în monorepo (`packages/l2-p02`).   
2) Citește `README.md` (learning goals + specificații) și `tests/config.json` (pattern‑uri pe care le vei activa în testele proprii).   
3) Rulează `npm i && npm run dev` (sau `pnpm` în monorepo) — verifică `GET /ping → pong`.   
4) Implementează rutele/validările/erorile conform specificației; adaptează testele pentru a acoperi scenariile de acceptanță.   
5) Folosește \*\*AI‑assist\*\* pe bucle scurte (Very Short Loop): generează scheletele, rulează testele, rafinează codul, repetă.

**Structura fiecărui proiect (starter)**

<project>/  
 src/app.js # Express app minimal (include /ping)  
 src/server.js # pornește pe PORT  
 tests/  
 vitest/smoke.test.ts  
 vitest/patterns.test.ts # generează `it.todo(...)` din tests/config.json  
 jest/smoke.test.cjs  
 jest/patterns.test.cjs # generează `test.todo(...)` din tests/config.json  
 tests/config.json # pattern‑uri recomandate pentru proiect  
 package.json # scripts: dev, test:vitest, test:jest, test  
 vitest.config.ts, jest.config.cjs  
 README.md # learning goals, specificații, pași, acceptanță

**L1 — Fundamente (15 proiecte)**

**L1-P01. Listare colecție clubs + mapare coduri de bază**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P02. Detaliu club + 404 Not Found**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P03. Creare club (201 + Location) + validări minime**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P04. Ștergere club (204) — idempotent la efect final**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P05. PATCH club — corecții parțiale simple**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P06. Paginare ?page&limit — semantici clare**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P07. Sortare ?sort=name,-createdAt — multi‑cheie**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P08. Filtrare ?category=...**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P09. Validare semantică la create (categorie din set)**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P10. Registrations: POST + mapare 422 pentru câmpuri**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P11. Registrations: GET ?club=...**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P12. Problem Details (RFC 7807) unificat**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P13. CORS minim: allow liste locale**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P14. Request logging + X-Request-Id**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L1-P15. Structură pe layere: routes/controllers/services**

\*\*Scop didactic.\*\* Exersezi maparea corectă între operații, metode și coduri HTTP; formezi reflexe pentru validări elementare și erori standardizate.

\*\*Specificații (rezumat).\*\* Implementează tema indicată de titlu, folosind `application/json` la succes și `application/problem+json` la erori. Respectă codurile: `201/204/404/409/422` conform cazului. Păstrează cheile JSON în `camelCase` și datele în ISO‑8601.

\*\*Criterii de acceptare.\*\* Rutele răspund cu status potrivit; pentru `POST` se setează `Location`; pentru input invalid → `422` cu `errors` granular.

\*\*Pași recomandați.\*\* (1) Adaugă rutele; (2) scrie validatori „pure”; (3) conectează‑i prin middleware; (4) adaugă handler global de erori (RFC 7807); (5) scrie 1–2 teste Supertest pentru succes + 1 negativ.

\*\*Soluție (rezumat).\*\* Controller‑e subțiri care apelează servicii; la create: `201 + Location`; la lipsă: `404`; la ștergere: `204`; la validare: `422` cu harta `errors`.

\*\*AI‑assist (VSL).\*\* „Generează routerul pentru resursa X cu `GET/POST/PATCH/DELETE` și validări minime”; „Compune `problem()` și 4 teste care verifică `type/title/status/detail`.”

**L2 — Intermediar (15 proiecte)**

**L2-P01. JSON Merge Patch (RFC 7386) — PATCH coerent**

\*\*Scop didactic.\*\* Introduci politici robuste: json merge patch (rfc 7386) — patch coerent.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P02. ETag + If-None-Match → 304 Not Modified**

\*\*Scop didactic.\*\* Introduci politici robuste: etag + if-none-match → 304 not modified.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: ETag, If-None-Match).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P03. If-Match la update → 409/412 (conceptual)**

\*\*Scop didactic.\*\* Introduci politici robuste: if-match la update → 409/412 (conceptual).

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P04. Idempotency-Key la POST /registrations (demo)**

\*\*Scop didactic.\*\* Introduci politici robuste: idempotency-key la post /registrations (demo).

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: Idempotency-Key).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P05. Paginare cu meta {page,limit,total} + validări**

\*\*Scop didactic.\*\* Introduci politici robuste: paginare cu meta {page,limit,total} + validări.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P06. Validare output (shape) — filtrare câmpuri interne**

\*\*Scop didactic.\*\* Introduci politici robuste: validare output (shape) — filtrare câmpuri interne.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P07. Coduri 4xx distincte: 400 vs 422**

\*\*Scop didactic.\*\* Introduci politici robuste: coduri 4xx distincte: 400 vs 422.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P08. Înregistrări duplicate — 409 Conflict**

\*\*Scop didactic.\*\* Introduci politici robuste: înregistrări duplicate — 409 conflict.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P09. CORS preflight (OPTIONS) — antete corecte**

\*\*Scop didactic.\*\* Introduci politici robuste: cors preflight (options) — antete corecte.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: OPTIONS, Access-Control-Allow-Methods).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P10. Cache-Control pentru GET listă/detaliu**

\*\*Scop didactic.\*\* Introduci politici robuste: cache-control pentru get listă/detaliu.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P11. Rate-limit (demo) → 429 Too Many Requests**

\*\*Scop didactic.\*\* Introduci politici robuste: rate-limit (demo) → 429 too many requests.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: 429, Too Many Requests).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P12. Logger structurat (JSON) + redactare PII**

\*\*Scop didactic.\*\* Introduci politici robuste: logger structurat (json) + redactare pii.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P13. Probleme aggregate în 422.errors (multi‑câmp)**

\*\*Scop didactic.\*\* Introduci politici robuste: probleme aggregate în 422.errors (multi‑câmp).

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P14. Validări pe query params (page/limit)**

\*\*Scop didactic.\*\* Introduci politici robuste: validări pe query params (page/limit).

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L2-P15. Testare HTTP robustă (Supertest) — suite negative**

\*\*Scop didactic.\*\* Introduci politici robuste: testare http robustă (supertest) — suite negative.

\*\*Specificații (rezumat).\*\* Extinde baza L1; implementează conceptul‑cheie anunțat (vezi și `tests/config.json`, patterns: —).

\*\*Criterii de acceptare.\*\* Demonstrezi comportamentul prin răspunsuri măsurabile: ex. `If-None-Match` → `304`, `Idempotency-Key` → răspuns repetabil, `OPTIONS` pentru preflight etc.

\*\*Pași recomandați.\*\* (1) Adaugă utilitarele (ex. `calcEtag`, `idemStore`); (2) instrumentează controller‑ele; (3) confirmă prin teste Supertest; (4) opțional, activează testele de pattern‑uri.

\*\*Soluție (rezumat).\*\* Pentru ETag, calculezi hash pe reprezentare și setezi antetul; la `If-None-Match` identic → `304`. Pentru `Idempotency-Key`, păstrezi rezultatul pentru cheie în fereastra curentă. Pentru preflight, răspunzi cu antetele `Access-Control-Allow-\*`.

\*\*AI‑assist (VSL).\*\* „Scrie `calcEtag()` și un test care verifică `304` la `If-None-Match` identic”; „Generează un demo `Idempotency-Key` și teste care confirmă același `id` la repetare.”

**L3 — Avansat (15 proiecte)**

**L3-P01. Contract‑first (OpenAPI 3.1) — schemă & validare**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: contract‑first (openapi 3.1) — schemă & validare.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: OpenAPI, schema.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P02. Sub‑resurse: /clubs/:id/members (design)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: sub‑resurse: /clubs/:id/members (design).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P03. Idempotency avansat cu TTL (concept)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: idempotency avansat cu ttl (concept).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P04. HATEOAS light (links) pentru navigare**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: hateoas light (links) pentru navigare.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P05. Import/Export NDJSON — loturi**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: import/export ndjson — loturi.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: NDJSON.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P06. Webhooks (simulare) pentru events**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: webhooks (simulare) pentru events.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P07. SSE (Server‑Sent Events) — notificări**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: sse (server‑sent events) — notificări.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: text/event-stream.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P08. Observabilitate: metri + corelații (cid)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: observabilitate: metri + corelații (cid).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P09. i18n pentru problem+json (RO/EN)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: i18n pentru problem+json (ro/en).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P10. Cursor‑based pagination (design)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: cursor‑based pagination (design).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: cursor.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P11. Testare contract (schema vs. response)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: testare contract (schema vs. response).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P12. Versionare API v1→v2 (plan de migrare)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: versionare api v1→v2 (plan de migrare).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P13. Audit trail (rudimentar)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: audit trail (rudimentar).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P14. Consistență cross‑resursă (rapoarte)**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: consistență cross‑resursă (rapoarte).

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: —.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**L3-P15. E2E smoke (concept) cu Playwright**

\*\*Scop didactic.\*\* Explorezi extensii arhitecturale și contractuale: e2e smoke (concept) cu playwright.

\*\*Specificații (rezumat).\*\* Elaborezi designul și un prototip minimal, cu accent pe contract și mesurabilitate. Patterns relevante în `tests/config.json`: Playwright.

\*\*Criterii de acceptare.\*\* Documentezi deciziile (README) și livrezi rute/artefacte minime demonstrabile (ex. endpoint SSE, fișier OpenAPI).

\*\*Pași recomandați.\*\* (1) Adaugă artefactul (ex. `/openapi.json`); (2) expune ruta sau mecanismul; (3) testează smoke + notează pașii de verificare manuală.

\*\*Soluție (rezumat).\*\* Contractul rămâne clar și extensibil; pentru i18n la `problem+json`, folosești mapări RO/EN pe `title/detail`; pentru SSE, setezi `Content-Type: text/event-stream` și trimiți evenimente `data:`.

\*\*AI‑assist (VSL).\*\* „Generează un schelet OpenAPI 3.1 pentru resursele `clubs`/`registrations`”; „Scrie un endpoint SSE demo (`/events`) și un client minimal de test.”

**Schiță generică de rezolvare — exemplu de controller (rezumat)**

// exemplu generic (pseudo)  
app.get('/api/clubs/:id', (req,res)=>{ /\* 200 + ETag sau 404 \*/ });  
app.post('/api/registrations', (req,res)=>{ /\* 201; cu Idempotency-Key \*/ });  
// la erori: res.status(x).type('application/problem+json').send(problemObj)

**AI‑assist (VSL) — prompturi utile pentru toate proiectele**

• „Generează middleware `validateBody(validator)` care mapează erorile la `422` (RFC 7807).”   
• „Scrie teste Supertest pentru scenariile: 201/204/404/409/422/304.”   
• „Propune un `Idempotency-Key` storage in‑memory + TTL de 10 minute.”   
• „Compune un `problem()` factory cu `errors` (harta câmpurilor) și teste pentru serializare.”   
• „Adaugă SSE `/events` (demo) + client de linie de comandă (curl).”

**Referințe (APA 7, DOI real)**

Fielding, R., Nottingham, M., & Reschke, J. (2022). HTTP Semantics (RFC 9110). RFC Editor. https://doi.org/10.17487/RFC9110  
Fielding, R., Nottingham, M., & Reschke, J. (2022). HTTP Caching (RFC 9111). RFC Editor. https://doi.org/10.17487/RFC9111  
Bray, T. (2017). The JavaScript Object Notation (JSON) Data Interchange Format (RFC 8259). RFC Editor. https://doi.org/10.17487/RFC8259  
Nottingham, M., & Wilde, E. (2016). Problem Details for HTTP APIs (RFC 7807). RFC Editor. https://doi.org/10.17487/RFC7807  
Barth, A. (2011). The Web Origin Concept (RFC 6454). RFC Editor. https://doi.org/10.17487/RFC6454  
Klyne, G., & Newman, C. (2002). Date and Time on the Internet: Timestamps (RFC 3339). RFC Editor. https://doi.org/10.17487/RFC3339  
Hunt, P., Bryan, P., & Nottingham, M. (2015). JSON Merge Patch (RFC 7386). RFC Editor. https://doi.org/10.17487/RFC7386  
Berners‑Lee, T., Fielding, R., & Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax (RFC 3986). RFC Editor. https://doi.org/10.17487/RFC3986