**Seminarul 9 — Partea 3: Proiecte/teme (45 proiecte)**

Această parte reunește \*\*45 de proiecte\*\* împărțite pe \*\*L1/L2/L3\*\* (fundamentale/intermediare/avansate) în jurul aceluiași domeniu didactic — \*Student Clubs\* — și al aceluiași toolchain: \*\*Node/Express + SQLite + Sequelize\*\* cu \*\*Vitest\*\* și \*\*Jest\*\* rulate \*\*side‑by‑side\*\*. Fiecare proiect include:  
• \*\*Specificație\*\*, \*\*criterii de acceptare\*\* și \*\*sugestii de testare\*\*,  
• \*\*Starter code\*\* + \*\*Solution\*\* (aplicație completă),  
• \*\*Teste automate\*\* (activabile via `tests/config.json`),  
• \*\*AI‑assist (VSL)\*\* — prompturi scurte, verificabile.

**Cum sunt organizate proiectele**

s9p3-projects/  
 L1/  
 L1-P01/starter|solution/  
 ...  
 L1-P15/  
 L2/  
 L2-P01/ ... L2-P15/  
 L3/  
 L3-P01/ ... L3-P15/  
(monorepo separat): s9p3-monorepo/ (PNPM workspaces; pachete starter)

Toate proiectele pot fi rulate identic: `npm i`, `npm test`, `npm run dev`. Testele sunt comune și se activează în funcție de `tests/config.json`. Varianta \*\*starter\*\* este suficientă pentru a începe exercițiile; varianta \*\*solution\*\* implementează complet validările, hook‑urile, indexul compus pe `Registration(memberId,clubId)`, tranzacțiile și încărcarea `include` (anti‑N+1).

**L1-P01 — P01 — Modele de bază + sync**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P02 — P02 — UNIQUE pe Club.name**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P03 — P03 — Validare email Member**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P04 — P04 — Relație 1-N Club-Event**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P05 — P05 — Relație M-N Member-Club (through Registration)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P06 — P06 — CASCADE la ștergerea Club**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P07 — P07 — Eager loading (anti-N+1)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P08 — P08 — Tranzacție enroll (commit/rollback)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P09 — P09 — Index compus (memberId,clubId)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, indexCompound.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P10 — P10 — Seed minimal deterministic**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, seeders.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P11 — P11 — Hooks (lowercase email)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P12 — P12 — Scopes (active/inactive)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P13 — P13 — Raport simplu (count members)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P14 — P14 — DTO/proiecții selective**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, eager.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L1-P15 — P15 — Refactor layere (db/services)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P01 — P01 — Modele de bază + sync**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, seeders.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P02 — P02 — UNIQUE pe Club.name**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, eager, transaction, indexCompound.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P03 — P03 — Validare email Member**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P04 — P04 — Relație 1-N Club-Event**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, eager, transaction, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P05 — P05 — Relație M-N Member-Club (through Registration)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assocMN, cascade, eager, transaction, indexCompound.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P06 — P06 — CASCADE la ștergerea Club**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, transaction, indexCompound, seeders.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P07 — P07 — Eager loading (anti-N+1)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, indexCompound, seeders.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P08 — P08 — Tranzacție enroll (commit/rollback)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, eager, transaction, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P09 — P09 — Index compus (memberId,clubId)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P10 — P10 — Seed minimal deterministic**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assocMN, cascade, eager, transaction, indexCompound, seeders.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P11 — P11 — Hooks (lowercase email)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, cascade, eager, transaction, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P12 — P12 — Scopes (active/inactive)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, eager, transaction, indexCompound, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P13 — P13 — Raport simplu (count members)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P14 — P14 — DTO/proiecții selective**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assocMN, cascade, eager, transaction, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L2-P15 — P15 — Refactor layere (db/services)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P01 — P01 — Modele de bază + sync**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P02 — P02 — UNIQUE pe Club.name**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P03 — P03 — Validare email Member**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P04 — P04 — Relație 1-N Club-Event**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P05 — P05 — Relație M-N Member-Club (through Registration)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P06 — P06 — CASCADE la ștergerea Club**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P07 — P07 — Eager loading (anti-N+1)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P08 — P08 — Tranzacție enroll (commit/rollback)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P09 — P09 — Index compus (memberId,clubId)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P10 — P10 — Seed minimal deterministic**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P11 — P11 — Hooks (lowercase email)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P12 — P12 — Scopes (active/inactive)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P13 — P13 — Raport simplu (count members)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P14 — P14 — DTO/proiecții selective**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**L3-P15 — P15 — Refactor layere (db/services)**

\*\*Scop didactic.\*\* Consolidarea: modelsBasic, uniqueClub, emailValidation, assoc1N, assocMN, cascade, eager, transaction, indexCompound, seeders, hooksScopes.  
\*\*Specificație.\*\* Implementează cerințele corespondente și asigură tezele de integritate: UNIQUE, validări, FK, CASCADE, tranzacție atomică (dacă este cazul), eager loading pentru a evita N+1.  
\*\*Criterii de acceptare.\*\* Testele din proiect (Vitest & Jest) trebuie să treacă; pe `eager`, numărul de interogări cu `include` să fie strict mai mic decât varianta \*lazy\*.  
\*\*Soluție (ghid).\*\* Pornește de la `src/db/index.js` → definește modele, asocieri, apoi completează piesele lipsă (`Registration` index compus, hooks/scopes în `Member`, tranzacția în `services/enroll.js`). Folosește `seedMinimal` pentru date deterministe.  
\*\*AI‑assist (VSL).\*\* Prompt: „Generează în Sequelize un \*composite unique index\* pe `registrations(memberId,clubId)` și completează testul pentru duplicat.”

**Exemple de Acceptanță (unificate)**

• \*\*UNIQUE Club.name\*\*: inserarea unui club cu același nume eșuează (Sequelize aruncă).   
• \*\*Email valid\*\*: `Member(email)` respinge `bad`, acceptă `ana@uni.ro`.   
• \*\*1‑N\*\*: `Club.findByPk(id, { include: 'events' })` returnează cel puțin un eveniment.   
• \*\*M‑N\*\*: `Club.findByPk(id, { include: 'members' })` conține membri asociați.   
• \*\*CASCADE\*\*: `destroy()` pe `Club` elimină înregistrările corelate din `Registration`.   
• \*\*Eager < Lazy\*\*: numărul de interogări pentru `include` este mai mic.   
• \*\*Tranzacție\*\*: `enrollMember` mărește `memberCount` la commit; la eșec (email invalid) rămâne neschimbat.   
• \*\*Index compus\*\*: a doua înregistrare cu același `(memberId,clubId)` eșuează.   
• \*\*Hooks/Scopes\*\*: e‑mailul se normalizează (lowercase), „inactive” exclude membri marcați `active=false`.

**Sugestii AI‑assist (VSL)**

• „Scrie `Member` cu `defaultScope` și `scope('inactive')` și un `beforeValidate` care face emailul lowercase.”   
• „Adaugă `indexes: [{ unique: true, fields: ['memberId','clubId'] }]` în `Registration`.”   
• „Transformă testul `lazy` într‑un \*for\* care folosește `getMembers()` și \*loghează\* SQL‑urile.”   
• „Completază `enrollMember` să ruleze într‑o tranzacție `sequelize.transaction(...)`.”

**Referințe (APA 7, DOI)**

Codd, E. F. (1970). A relational model of data for large shared data banks. \*Communications of the ACM, 13\*(6), 377–387. https://doi.org/10.1145/362384.362685  
Gaffney, K. P., Hipp, D. R., Kennedy, W. A., Mistachkin, J., Nagan, D., & Owens, R. (2022). SQLite: Past, present, and future. \*PVLDB, 15\*(12), 3535–3548. https://doi.org/10.14778/3554821.3554842  
Härder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. \*ACM Computing Surveys, 15\*(4), 287–317. https://doi.org/10.1145/289.291  
Klyne, G., & Newman, C. (2002). Date and time on the Internet: Timestamps (RFC 3339). RFC Editor. https://doi.org/10.17487/RFC3339  
Xue, Y., Kolekar, S., Liu, Y., Gupta, N., & Ryoo, J. (2022). Reformulator: Automated refactoring of the N+1 problem in database-backed applications. \*ASE 2022\*. https://doi.org/10.1145/3551349.3556911