**Seminarul 12 — Partea 3 (Proiecte/teme extinse)**

Partea 3 din Seminarul 12 oferă un set coerent de \*\*45 de proiecte\*\* împărțite pe trei niveluri (L1, L2, L3) care îmbină \*\*tabele\*\* (TanStack Table), \*\*grafice\*\* (Chart.js) și \*\*funcții pure\*\* pentru prelucrarea datelor. Scopul este consolidarea competențelor prin sarcini atomice, verificabile automat (\*\*Vitest\*\* & \*\*Jest\*\*), dar și prin extensii realiste (export, i18n, theming, sincronizări bidirecționale, specificații declarative pentru grafice, crossfilter, ferestre virtualizate, fluxuri în timp real).  
   
\*\*Metodologie didactică.\*\* Fiecare proiect este structurat identic: (1) \*\*Obiectiv educațional\*\* (ce deprindere vizați), (2) \*\*Specificații\*\* (cerințe măsurabile), (3) \*\*Contracte de test\*\* (ce verifică testele, în termeni observabili), (4) \*\*Soluție minimă\*\* (principii algoritmice, nu neapărat cod complet aici), (5) \*\*Erori tipice\*\* (antipatters), (6) \*\*AI‑assist (VSL)\*\*—prompturi utile pentru GitHub Copilot sau un LLM (ChatGPT, Claude, Mistral). Abordarea privilegiază \*\*funcțiile pure\*\* pentru că acestea sunt: (a) deterministe (ușor de testat), (b) ușor de compus, (c) independente de UI.  
   
\*\*Principii transversale.\*\*  
- \*\*Single source of truth.\*\* Datele „raw” sunt unice; derivele (filtrare, agregare, paginate) se calculează „la cerere” prin \*\*selectori\*\* (funcții pure).  
- \*\*Contract over pixels.\*\* Testele privesc contractele: forme de obiect (e.g., `{ labels, datasets }`), efecte observabile (aria‑sort, lungimi, secvențe), nu detalii fragile de implementare (DOM intern al graficului).  
- \*\*Accesibilitate (a11y).\*\* Tabele semantice, `aria-sort`, butoane de sortare, `caption`, focus și tastatură; grafice deterministe, culori lizibile (palete contrastante), alternative textuale (sumare de date).  
- \*\*i18n/Theming.\*\* Formatarea numerelor și datelor prin `Intl.\*`; palete dependente de temă și testate pe canale (sequential/diverging); nu codificați doar cu culoarea.  
- \*\*Performanță.\*\* Debounce/throttle pentru filtre; virtualizare; bugete măsurabile; funcții `downsample`/`rollups` pentru volum mare de date.  
- \*\*Reproducibilitate.\*\* Configurări declarative, \*snapshots\* ale specificațiilor, serializare stabilă (`serializeConfig`), opțiuni pentru export (CSV/PNG).  
- \*\*Siguranță.\*\* `sanitizeCsv` pentru export, evitarea injecțiilor (e.g., formula injection).  
   
\*\*Flux de lucru recomandat.\*\* Alegeți un proiect, rulați `npm test` în `starter/`, completați TODO‑urile din `src/lib/lib.ts` până testele devin verzi, apoi inspectați `solution/` pentru a compara abordarea. Documentația din README indică \*learning goals\* și prompturi de AI. Testele sunt identice (logic) în \*\*Vitest\*\* și \*\*Jest\*\* pentru a vă antrena în ambele ecosisteme.

# L1 — Fundamental

## L1-P01 — Table Basics (sort asc/desc)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Table Basics (sort asc/desc)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Table Basics (sort asc/desc)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Table Basics (sort asc/desc)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Table Basics (sort asc/desc)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P02 — Column Filters (text/debounce)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Column Filters (text/debounce)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Column Filters (text/debounce)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Column Filters (text/debounce)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Column Filters (text/debounce)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P03 — Pagination (client‑side)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Pagination (client‑side)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Pagination (client‑side)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Pagination (client‑side)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Pagination (client‑side)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P04 — Column Visibility & Pinning  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Column Visibility & Pinning” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Column Visibility & Pinning” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Column Visibility & Pinning”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Column Visibility & Pinning\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P05 — Row Selection (single/multi)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Row Selection (single/multi)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Row Selection (single/multi)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Row Selection (single/multi)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Row Selection (single/multi)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P06 — CSV Export (visible rows)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „CSV Export (visible rows)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „CSV Export (visible rows)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „CSV Export (visible rows)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*CSV Export (visible rows)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P07 — Responsive Table (stacked <480px)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Responsive Table (stacked <480px)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Responsive Table (stacked <480px)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Responsive Table (stacked <480px)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Responsive Table (stacked <480px)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P08 — Caption & A11y (roles, aria-sort)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Caption & A11y (roles, aria-sort)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Caption & A11y (roles, aria-sort)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Caption & A11y (roles, aria-sort)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Caption & A11y (roles, aria-sort)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P09 — Bar Chart (static dataset)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Bar Chart (static dataset)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Bar Chart (static dataset)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Bar Chart (static dataset)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Bar Chart (static dataset)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P10 — Line Chart (time series mock)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Line Chart (time series mock)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Line Chart (time series mock)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Line Chart (time series mock)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Line Chart (time series mock)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P11 — Legend Toggle (show/hide series)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Legend Toggle (show/hide series)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Legend Toggle (show/hide series)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Legend Toggle (show/hide series)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Legend Toggle (show/hide series)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P12 — Tooltip Formatting (Intl.\*)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Tooltip Formatting (Intl.\*)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Tooltip Formatting (Intl.\*)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Tooltip Formatting (Intl.\*)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Tooltip Formatting (Intl.\*)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P13 — Colour Palette (light/dark)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Colour Palette (light/dark)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Colour Palette (light/dark)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Colour Palette (light/dark)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Colour Palette (light/dark)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P14 — One‑way Sync (table→chart)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „One‑way Sync (table→chart)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „One‑way Sync (table→chart)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „One‑way Sync (table→chart)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*One‑way Sync (table→chart)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L1-P15 — Smoke Integration (table+chart)  
\*\*Nivel:\*\* L1  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Smoke Integration (table+chart)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Smoke Integration (table+chart)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Smoke Integration (table+chart)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Smoke Integration (table+chart)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

# L2 — Intermediar

## L2-P01 — Server‑backed Table (fetch)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Server‑backed Table (fetch)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Server‑backed Table (fetch)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Server‑backed Table (fetch)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Server‑backed Table (fetch)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P02 — Server Pagination (query params)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Server Pagination (query params)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Server Pagination (query params)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Server Pagination (query params)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Server Pagination (query params)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P03 — Column Grouping (multi‑header)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Column Grouping (multi‑header)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Column Grouping (multi‑header)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Column Grouping (multi‑header)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Column Grouping (multi‑header)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P04 — Row Editing (inline + applyEdit)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Row Editing (inline + applyEdit)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Row Editing (inline + applyEdit)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Row Editing (inline + applyEdit)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Row Editing (inline + applyEdit)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P05 — Virtualized Window (rows)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Virtualized Window (rows)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Virtualized Window (rows)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Virtualized Window (rows)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Virtualized Window (rows)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P06 — Derived Aggregations (groupBy)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Derived Aggregations (groupBy)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Derived Aggregations (groupBy)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Derived Aggregations (groupBy)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Derived Aggregations (groupBy)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P07 — Export PNG (contract)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Export PNG (contract)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Export PNG (contract)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Export PNG (contract)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Export PNG (contract)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P08 — Linked Views (two‑way sync)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Linked Views (two‑way sync)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Linked Views (two‑way sync)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Linked Views (two‑way sync)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Linked Views (two‑way sync)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P09 — Drill‑down (category→table)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Drill‑down (category→table)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Drill‑down (category→table)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Drill‑down (category→table)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Drill‑down (category→table)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P10 — Dual Axis (bar+line)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Dual Axis (bar+line)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Dual Axis (bar+line)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Dual Axis (bar+line)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Dual Axis (bar+line)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P11 — Anomaly Markers (thresholds)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Anomaly Markers (thresholds)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Anomaly Markers (thresholds)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Anomaly Markers (thresholds)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Anomaly Markers (thresholds)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P12 — Internationalisation (RO/EN)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Internationalisation (RO/EN)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Internationalisation (RO/EN)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Internationalisation (RO/EN)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Internationalisation (RO/EN)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P13 — Theming API (variables→palette)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Theming API (variables→palette)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Theming API (variables→palette)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Theming API (variables→palette)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Theming API (variables→palette)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P14 — Accessibility Advanced (grid nav)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Accessibility Advanced (grid nav)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Accessibility Advanced (grid nav)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Accessibility Advanced (grid nav)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Accessibility Advanced (grid nav)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L2-P15 — Contract Tests (config shape)  
\*\*Nivel:\*\* L2  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Contract Tests (config shape)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Contract Tests (config shape)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Contract Tests (config shape)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Contract Tests (config shape)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

# L3 — Avansat

## L3-P01 — Crossfilter (table + 2 charts)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Crossfilter (table + 2 charts)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Crossfilter (table + 2 charts)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Crossfilter (table + 2 charts)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Crossfilter (table + 2 charts)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P02 — Column Virtualization (wide)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Column Virtualization (wide)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Column Virtualization (wide)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Column Virtualization (wide)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Column Virtualization (wide)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P03 — Infinite Scroll (cursor merge)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Infinite Scroll (cursor merge)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Infinite Scroll (cursor merge)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Infinite Scroll (cursor merge)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Infinite Scroll (cursor merge)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P04 — Complex Aggregations (roll‑ups)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Complex Aggregations (roll‑ups)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Complex Aggregations (roll‑ups)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Complex Aggregations (roll‑ups)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Complex Aggregations (roll‑ups)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P05 — Vega‑Lite Spec (declarativ)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Vega‑Lite Spec (declarativ)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Vega‑Lite Spec (declarativ)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Vega‑Lite Spec (declarativ)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Vega‑Lite Spec (declarativ)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P06 — Custom Chart Spec (axes/scales)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Custom Chart Spec (axes/scales)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Custom Chart Spec (axes/scales)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Custom Chart Spec (axes/scales)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Custom Chart Spec (axes/scales)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P07 — Brushing & Linking (range)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Brushing & Linking (range)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Brushing & Linking (range)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Brushing & Linking (range)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Brushing & Linking (range)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P08 — Annotations (labels/notes)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Annotations (labels/notes)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Annotations (labels/notes)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Annotations (labels/notes)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Annotations (labels/notes)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P09 — Realtime Merge (sliding window)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Realtime Merge (sliding window)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Realtime Merge (sliding window)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Realtime Merge (sliding window)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Realtime Merge (sliding window)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P10 — Snapshot Reproducibility  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Snapshot Reproducibility” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Snapshot Reproducibility” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Snapshot Reproducibility”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Snapshot Reproducibility\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P11 — Printing & PDF (paged CSS)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Printing & PDF (paged CSS)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Printing & PDF (paged CSS)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Printing & PDF (paged CSS)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Printing & PDF (paged CSS)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P12 — Large Data (downsample)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Large Data (downsample)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Large Data (downsample)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Large Data (downsample)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Large Data (downsample)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P13 — SSR/SSG Serialize/Hydrate  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „SSR/SSG Serialize/Hydrate” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „SSR/SSG Serialize/Hydrate” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „SSR/SSG Serialize/Hydrate”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*SSR/SSG Serialize/Hydrate\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P14 — Security Sanitation (CSV)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Security Sanitation (CSV)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Security Sanitation (CSV)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Security Sanitation (CSV)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Security Sanitation (CSV)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

## L3-P15 — Performance Budget (metrics)  
\*\*Nivel:\*\* L3  
  
\*\*Obiectiv educațional.\*\* Să înțelegi și să aplici corect conceptul „Performance Budget (metrics)” într‑un context de tabel/grafic, operând pe funcții pure unde este posibil pentru a reduce cuplajul și a crește testabilitatea.  
  
\*\*Specificații.\*\*  
1. Implementează funcțiile din `src/lib/lib.ts` relevante pentru „Performance Budget (metrics)” (vezi testele).   
2. Menține determinismul: fără efecte secundare necontrolate, fără dependențe globale ascunse.   
3. Respectă contractele de ieșire (forme de obiect și proprietăți).   
4. Rulează testele în \*\*Vitest\*\* și \*\*Jest\*\* — ambele trebuie să fie verzi.  
  
\*\*Contracte de test (sumar).\*\*  
— Testele verifică proprietăți observabile (e.g., lungimea subsetului, ordinea elementelor, câmpuri ale configurării graficului, etc.).   
— Nu se testează randarea pixel cu pixel; accent pe formele de date și pe semantica DOM acolo unde e cazul.  
  
\*\*Soluție minimă (idee).\*\*  
— Pornește de la o implementare directă: evită premature optimisation.   
— Pentru „Performance Budget (metrics)”, traduce cerința într‑o funcție pură: primește \*input\* (rânduri/parametri), calculează \*output\* (subset/structură/valoare), fără a depinde de context de UI.   
— Adaugă acoperire pentru cazuri de margine (empty arrays, null/undefined inappropriate, parametri eronați).  
  
\*\*Erori tipice.\*\*  
— Confundarea datelor „display” cu date „raw” (e.g., sort pe stringul formatat, nu pe valoarea numerică).   
— Mutarea \*in place\* a array‑urilor partajate (efecte secundare).   
— Lipsa tratării pentru intrări goale; asumarea că ordinea etichetelor coincide mereu cu ordinea valorilor.   
— Contracte „spartane” care nu pot susține extensii (ex.: grafice cu un singur dataset hard‑codat).  
  
\*\*AI‑assist (VSL).\*\*  
- „Write a pure function that implements \*\*Performance Budget (metrics)\*\* with TypeScript types and 5 unit tests (edge cases). Keep IO immutable.”   
- „Refactor the function to avoid O(n²) behaviour where possible and document time complexity.”   
- „Generate a user‑friendly error message for invalid inputs and suggest a fallback.”

---  
## Indicații de evaluare (rubrică)

- \*\*Corectitudine (40%)\*\* — Toate testele verzi; funcțiile respectă contractele; comportament determinist.  
- \*\*Claritate (20%)\*\* — Tipare TS explicite; nume de funcții și variabile semnificative; README actualizat.  
- \*\*Robustețe (20%)\*\* — Tratarea cazurilor de margine; mesaje de eroare utile; lipsa efectelor secundare nedocumentate.  
- \*\*Extensibilitate (10%)\*\* — Structuri de date care permit extinderi; separarea derivatelor prin selectori.  
- \*\*Calitate a11y/i18n/theming (10%)\*\* — Respectarea semantici de tabel; folosirea `Intl.\*`; palete lizibile.