**Seminarul 2 — Partea 3 (Proiecte/teme extinse)**

Această parte conține 45 de proiecte (15 × L1/L2/L3), fiecare cu specificații și soluții (idee + algoritm), aliniate la principiile Partelor 1–2: funcții pure, compoziție declarativă, contracte testabile, suite Vitest/Jest în oglindă. Arhivele asociate includ variante starter/solution, o variantă monorepo (PNPM workspaces) și workflows CI (lint, unit-vitest, unit-jest, e2e, workbox-sw). Proiectele L3 includ un viewer web static testat cu Playwright și pregătit pentru PWA prin Workbox.

**Nivel L1 (Fundamental) — L1-01: Top‑K interese (map/flatMap → countBy → sort stabil → slice)**

API: topKInterests(rows, k) → [{key,count}]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-02: Distribuție pe facultăți (countBy)**

API: countByFaculty(rows) → Record<faculty,number>

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-03: Deduplicare după e‑mail (uniqBy)**

API: dedupByEmail(rows) → rows

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-04: Partiționare valid/invalid (partition + predicate)**

API: partitionValid(rows) → { valid, invalid }

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-05: GroupBy (faculty,year) cu cheie compusă**

API: groupByFacultyYear(rows) → Record<'FAC|YEAR',rows[]>

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-06: difference/intersection pe seturi (email)**

API: setOpsByEmail(a,b) → {difference, intersection}

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-07: chunk pentru paginare**

API: paginate(arr, n) → T[][]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-08: zip/unzip pentru aliniere de liste**

API: zipAlign(a,b) / unzipPairs(pairs)

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-09: flatten/flatMap pe interese multiple**

API: flattenInterests(rows) → string[]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-10: Comparator determinist pentru frecvențe**

API: sortByCountThenAlpha(entries) → sorted

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-11: keyBy cu politică last‑wins**

API: keyByEmail(rows) → Record<email,row>

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-12: Pipeline parametric (filters facultate/interes)**

API: filterRows(rows,{faculty,interest}) → rows

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-13: compact/defined pentru curățare listă**

API: compactDefined(arr) → arr

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-14: Sortare cu Intl.Collator pentru text**

API: sortAlphaRO(arr) → arr

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L1 (Fundamental) — L1-15: Lanț map→filter→reduce→slice (rezumat)**

API: pipelineSummary(rows,k) → {topK,total}

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-01: groupBy multi‑cheie + agregare (count)**

API: groupCount(rows, keysFn) → Record<key,count>

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-02: uniqBy cu cheie derivată (emailCanon)**

API: uniqByEmailCanon(rows) → rows

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-03: partition cu predicate compuse (AND/OR)**

API: partitionBy(rows, pred) → [pass, fail]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-04: differenceBy pe obiecte fără id (proiecție)**

API: differenceByProj(a,b, proj) → rows

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-05: reduce cu acumulatoare multiple (sum,min,max,count)**

API: stats(nums) → {sum,min,max,count}

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-06: zip & merge (aliniere și combinare câmpuri)**

API: zipMerge(a,b, key) → merged[]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-07: pipeline parametric (AND/OR builder)**

API: buildPredicate(opts) → (row)=>boolean

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-08: comparator compus (primar/ secundar/ terțiar)**

API: sort3(arr, a,b,c) → sorted

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-09: tap/peek audit (dimensiuni intermediare)**

API: auditPipeline(rows, taps[]) → rows

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-10: flatten profund (nested)**

API: deepFlatten(arr) → flat[]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-11: keyBy cu politică error‑on‑collision**

API: keyByUnique(arr, keyFn) → map

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-12: property‑based ideas fără lib (random cases)**

API: randomCases(n, gen) → cases

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-13: chunk progresiv (ferestre)**

API: windows(arr, n) → T[][]

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-14: „lazy ideas” pe iterabile (wrapper simplu)**

API: lazyMap/ lazyFilter (concept)

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L2 (Intermediar) — L2-15: README cu note de performanță și edge‑cases**

API: docOnly

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-01: Iterabile lazy (generatori) – map/filter wrapper**

API: lazyMap, lazyFilter (generator)

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-02: Procesare linie‑cu‑linie (simulare)**

API: readLinesSim(text) → iterator

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-03: CLI cu subcomenzi (topK | dist)**

API: cli(args) → code

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-04: Clasificarea erorilor + mesaje prietenoase**

API: friendlyError(e) → string

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-05: Micro‑profilare a pașilor**

API: profile(fn, x) → {result,ms}

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-06: Mini‑lib publicabil (API fațadă)**

API: index.js + exports

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-07: Viewer web static (raport JSON → tabel)**

API: web/index.html + script.js

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-08: Playwright e2e (smoke UI)**

API: tests/e2e.spec.ts

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-09: Workbox precache (generateSW)**

API: workbox-build config

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-10: Config mediu (ENV) pentru CLI/Viewer**

API: env handling

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-11: Contracte suplimentare (idempotență)**

API: tests/idempotence.test

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-12: compose extins + currying**

API: curry, composeN

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-13: Plugin strategy (predicates/aggregators)**

API: registerPlugin(name, fn)

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-14: PWA offline (ultimul raport)**

API: cache report.json

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

**Nivel L3 (Avansat) — L3-15: Pachetarea artefactelor (zip outputs)**

API: archive reports

Specificație:

• Date: liste de obiecte (rânduri înscriere). • Cerință: implementează API-ul de mai sus ca funcție pură, cu contracte clare (intrări/ieșiri), fără I/O. • Cazuri-limită: liste goale, duplicate, lipsuri. • Evaluare: teste în oglindă Vitest/Jest, verificând corectitudinea și determinismul.

Soluție (idee):

• Folosește map/flatMap/filter/reduce și utilitare dedicate (countBy, groupBy, uniqBy, predicate-builders). • Definește comparatorii cu tie-breaker alfabetică. • Explică complexitatea (O(n), O(n log n)) și politicile (ex. last-wins).

—

Utilizează arhiva standalone pentru a vedea atât variantele starter cât și solution; monorepo-ul oferă un mod scalabil de lucru pe pachete. În L3-07 găsești un viewer web simplu (JSON→HTML) cu test Playwright și posibilitate de generare Service Worker via Workbox.