**Seminarul 11 — Redux Toolkit — Partea 2 (Laborator extins)**

\*\*Obiectivul Părții 2\*\* este să construim, end‑to‑end, o aplicație didactică „\*\*ClubHub+ State\*\*” în \*\*React + TypeScript\*\* cu \*\*Redux Toolkit (RTK)\*\*, pentru a exersa: \*slices\* cu `createSlice`, \*entități normalizate\* cu `createEntityAdapter`, \*selectori memoizați\* (Reselect), \*flux asincron\* cu `createAsyncThunk` (și, opțional, \*\*RTK Query\*\*), \*integrare UI\* (React‑Redux), \*persistență minimală\* și \*\*testare duală\*\* (\*\*Vitest\*\* & \*\*Jest\*\*) pe aceeași bază de cod. Fiecare etapă include: \*\*worksheet\*\* (cerință + checklist), \*\*starter code\*\*, \*\*teste\*\*, și \*\*AI‑assist\*\* (prompts concise pentru GitHub Copilot sub VSL).

---

## 0) Setup: proiect Vite + RTK + testare duală + MSW

\*\*Scop.\*\* Inițializăm proiectul cu Vite (React + TS), adăugăm `@reduxjs/toolkit`, `react-redux`, `reselect`, `msw` și configurăm \*\*Vitest\*\* & \*\*Jest\*\* în paralel (JSDOM, RTL).

\*\*Pași (rezumat):\*\*  
- `package.json` conține `test:vitest`, `test:jest` și `test` (ambele).   
- `vitest.config.ts` și `jest.config.cjs` setează `environment: 'jsdom'` și `setupFiles` pentru `@testing-library/jest-dom` și \*\*MSW\*\*.   
- `src/app/store.ts` expune `store`, `RootState`, `AppDispatch`; `src/app/hooks.ts` definește \*\*typed hooks\*\*.

\*\*Worksheet (Cerință):\*\*  
- Rulează `npm i`, apoi `npm test` și `npm run dev`.   
- Confirmă că testele de fum trec și UI afişează headerul.

\*\*Checklist:\*\*  
- [ ] Dev server funcționează (apare „ClubHub+ State”).   
- [ ] `npm test` rulează Vitest și Jest.   
- [ ] `tests/setup.ts` pornește/închide \*\*MSW\*\* corect.

---

## 1) Slice `clubs` cu \*\*entityAdapter\*\*

\*\*Scop.\*\* Modelăm lista de `Club` ca entități normalizate, cu \*CRUD\* eficient și \*\*selectori auto‑gen\*\* (`selectAll`, `selectById`, `selectIds`).

\*\*Pași:\*\*  
- Definim `Club` (`id`, `name`, `category`).   
- `createEntityAdapter<Club>` cu `sortComparer`.   
- Starea include `status`, `error`, `query`, `category`.   
- Reduceri: `upsertMany`, `removeOne`, `setQuery`, `setCategory`.   
- Selectori derivați (Reselect): `selectVisibleClubs` după `query` și `category`.

\*\*Teste (unit):\*\*  
- Reducer: `upsertMany` + `removeOne` actualizează corect.   
- Selector: `selectVisibleClubs` filtrează previzibil.

\*\*AI‑assist (VSL):\*\*   
„Create a `createSlice` with `createEntityAdapter` for `Club` (id/name/category) + selectors and write Vitest & Jest tests for CRUD and filtering.”

---

## 2) Integrare UI (React‑Redux): search & filter controlate + add form

\*\*Scop.\*\* Conectăm UI la store: citire cu `useAppSelector`, acțiuni cu `useAppDispatch`. Construim `SearchBox` (controlled), `CategoryFilter` (controlled) și `AddClubForm` (controlled).

\*\*Pași:\*\*  
- `App.tsx`: legăm `q` și `cat` locale de store (`setQuery`, `setCategory`).   
- `AddClubForm`: la `submit`, emite un obiect nou (slugify pentru `id`) → `upsertMany`.   
- `ClubList`: randare eficientă, `key` stabil (`id`), empty state semantic (`role="status"`).

\*\*Teste (RTL):\*\*  
- „Adaugă Robotics” + verifică afișarea cardului.   
- Tastare în `search-input` → lista se reduce la rezultatele relevante.

\*\*AI‑assist:\*\*   
„Create controlled SearchBox and CategoryFilter bound to RTK store via dispatch, and verify with RTL tests that filtering works.”

---

## 3) Async: `createAsyncThunk` + \*\*MSW\*\* (mock API)

\*\*Scop.\*\* Adăugăm `fetchClubs` (GET `/api/clubs`) cu ciclul `pending/fulfilled/rejected`. UI expune un buton „Load” și stări de `loading`/`error`.

\*\*Pași:\*\*  
- Definim `fetchClubs` în slice; în `extraReducers` setăm `status` și `error`.   
- `tests/msw/handlers.ts`: răspuns JSON stabil.   
- `tests/setup.ts`: pornește/închide serverul (MSW Node).

\*\*Teste:\*\*  
- Thunk: după `dispatch(fetchClubs())`, `status === 'succeeded'`, entitățile sunt populate.   
- UI: `click` pe „Load” → apare „Tech & Coding”.

\*\*AI‑assist:\*\*   
„Write a MSW handler for `/api/clubs`, and a thunk test that asserts `status` transitions and entity population.”

---

## 4) Persistență minimală (opțional extins)

\*\*Scop.\*\* Middleware simplu pentru serializarea subarborelui `clubs` în `localStorage` (exercițiu).

\*\*Sugestie:\*\* implementați un `persistMiddleware` cu debounce de 300 ms, stocând `ids`/`entities`/`query`/`category`, fără `status/error`. Test minim: spy pe `localStorage.setItem`.

---

## 5) RTK Query (opțional în aceeași aplicație)

\*\*Scop.\*\* Demonstrăm separarea \*\*server cache\*\* (RTK Query) de \*\*app state\*\* (slice).

\*\*Pași:\*\*   
- `createApi` + `fetchBaseQuery`, endpoint `getClubs`, `addClub`.   
- Hooks `useGetClubsQuery` / `useAddClubMutation` (doar exemplificate).   
- Discuție: când folosești Query vs. thunks (laboratorul vine cu ambele ca referință).

---

## 6) Refactor & a11y pass, apoi \*\*test suite\*\*

\*\*Checklist final:\*\*  
- [ ] `selectVisibleClubs` nu reconstruiește obiecte (evită ruperea memoizării).   
- [ ] Empty state semantic + `aria` labels coerente.   
- [ ] Testele de reducer, selectors, thunk, App trec în ambele framework‑uri.   
- [ ] (Opțional) middleware persistență funcțional; Query hook valid.

---

## Codul livrat (în arhiva ZIP)

- \*\*RTK core\*\*: `clubsSlice.ts` (adapter + thunk + selectors).   
- \*\*Store\*\*: `app/store.ts`, `app/hooks.ts` (typed hooks).   
- \*\*UI\*\*: `SearchBox`, `CategoryFilter`, `AddClubForm`, `ClubList`.   
- \*\*MSW\*\*: `tests/msw/handlers.ts`, `tests/msw/server.ts`, `tests/setup.ts`.   
- \*\*Teste\*\*: reducer/selectors/thunk/App (Vitest & Jest).   
- \*\*Docs\*\*: `docs/WORKSHEET.md`, `docs/README.md` (pași, criterii).

---

## AI‑assist (VSL): prompts recomandate (copiere directă în IDE)

- „Generate a Redux Toolkit slice with `createEntityAdapter` for clubs (id,name,category) and write Vitest tests for `upsertMany` and `removeOne`.”   
- „Create `selectVisibleClubs(query, category)` using `createSelector` and test recomputation count when irrelevant state changes.”   
- „Implement `createAsyncThunk` `fetchClubs`, write MSW handler for `/api/clubs`, and test status transitions.”   
- „Wire up React components (SearchBox, CategoryFilter, AddClubForm) to Redux store with typed hooks, and verify behaviour with RTL.”

---

## Troubleshooting (tipic)

- \*\*Jest nu transpilează TSX\*\* → verifică `babel-jest` + preseturi; `testEnvironment:'jsdom'`.   
- \*\*Vitest fără JSDOM\*\* → `environment:'jsdom'` + `setupFiles` corecte.   
- \*\*MSW nu răspunde\*\* → import corect `tests/setup.ts`; nu amesteca \*browser\* vs. \*node server\* în teste.   
- \*\*Flaky tests\*\* → folosește `userEvent` și \*await\* pentru UI asincron; izolează testele de stat global persistent.

Succes în laborator!

\*\*Fragment:\*\* `package.json`

{  
 "name": "s11p2-rtk-lab",  
 "version": "1.0.0",  
 "private": true,  
 "type": "module",  
 "scripts": {  
 "dev": "vite",  
 "build": "vite build",  
 "preview": "vite preview",  
 "test:vitest": "vitest run --reporter verbose",  
 "test:jest": "jest --runInBand",  
 "test": "npm run test:vitest && npm run test:jest"  
 },  
 "dependencies": {  
 "@reduxjs/toolkit": "^2.2.0",  
 "react": "^19.0.0",  
 "react-dom": "^19.0.0",  
 "react-redux": "^9.1.2",  
 "reselect": "^5.1.0"  
 },  
 "devDependencies": {  
 "@babel/preset-env": "^7.25.0",  
 "@babel/preset-react": "^7.24.0",  
 "@babel/preset-typescript": "^7.24.0",  
 "@testing-library/jest-dom": "^6.4.2",  
 "@testing-library/react": "^14.2.1",  
 "@testing-library/user-event": "^14.5.2",  
 "@types/jest": "^29.5.12",  
 "@types/node": "^20.11.17",  
 "@types/react": "^18.3.3",  
 "@types/react-dom": "^18.3.0",  
 "@vitejs/plugin-react": "^4.3.0",  
 "babel-jest": "^29.7.0",  
 "jest": "^29.7.0",  
 "jsdom": "^24.1.0",  
 "msw": "^2.3.1",  
 "typescript": "^5.4.0",  
 "vite": "^5.4.0",  
 "vitest": "^1.6.0"  
 }  
}

\*\*Fragment:\*\* `src/app/store.ts`

import { configureStore } from '@reduxjs/toolkit'  
import clubsReducer from '../features/clubs/clubsSlice'  
import { clubsApi } from '../services/clubsApi'  
  
export const store = configureStore({  
 reducer: {  
 clubs: clubsReducer,  
 [clubsApi.reducerPath]: clubsApi.reducer,  
 },  
 middleware: (getDefault) => getDefault().concat(clubsApi.middleware),  
 devTools: true  
})  
  
export type RootState = ReturnType<typeof store.getState>  
export type AppDispatch = typeof store.dispatch

\*\*Fragment:\*\* `src/features/clubs/clubsSlice.ts`

import { createSlice, createEntityAdapter, createAsyncThunk, createSelector } from '@reduxjs/toolkit'  
import type { RootState } from '../../app/store'  
  
export type Club = { id: string; name: string; category: string }  
  
export const fetchClubs = createAsyncThunk<Club[]>('clubs/fetchClubs', async () => {  
 const res = await fetch('/api/clubs')  
 if(!res.ok) throw new Error('Network error')  
 return await res.json() as Club[]  
})  
  
const clubsAdapter = createEntityAdapter<Club>({  
 sortComparer: (a, b) => a.name.localeCompare(b.name)  
})  
  
const initial = clubsAdapter.getInitialState({  
 status: 'idle' as 'idle'|'loading'|'succeeded'|'failed',  
 error: null as string|null,  
 query: '',  
 category: 'all' as 'all'|'technology'|'arts'|'sports'  
})  
  
const clubsSlice = createSlice({  
 name: 'clubs',  
 initialState: initial,  
 reducers: {  
 setQuery(state, action: { payload: string }) { state.query = action.payload },  
 setCategory(state, action: { payload: 'all'|'technology'|'arts'|'sports' }) { state.category = action.payload },  
 upsertMany: clubsAdapter.upsertMany,  
 removeOne: clubsAdapter.removeOne,  
 },  
 extraReducers: builder => {  
 builder  
 .addCase(fetchClubs.pending, (state) => { state.status = 'loading'; state.error = null })  
 .addCase(fetchClubs.fulfilled, (state, action) => {  
 state.status = 'succeeded'  
 clubsAdapter.setAll(state, action.payload)  
 })  
 .addCase(fetchClubs.rejected, (state, action) => {  
 state.status = 'failed'  
 state.error = String(action.error.message || 'error')  
 })  
 }  
})  
  
export const { setQuery, setCategory, upsertMany, removeOne } = clubsSlice.actions  
export default clubsSlice.reducer  
  
// selectors  
const base = (s: RootState) => s.clubs  
export const clubsSelectors = clubsAdapter.getSelectors<RootState>((s) => s.clubs)  
export const selectQuery = (s: RootState) => base(s).query  
export const selectCategory = (s: RootState) => base(s).category  
export const selectStatus = (s: RootState) => base(s).status  
export const selectError = (s: RootState) => base(s).error  
  
export const selectVisibleClubs = createSelector(  
 [clubsSelectors.selectAll, selectQuery, selectCategory],  
 (list, q, cat) => {  
 const query = q.trim().toLowerCase()  
 return list.filter(c => {  
 const okQ = !query || c.name.toLowerCase().includes(query) || c.category.toLowerCase().includes(query)  
 const okC = cat === 'all' || c.category === cat  
 return okQ && okC  
 })  
 }  
)

\*\*Fragment:\*\* `src/services/clubsApi.ts`

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'  
import type { Club } from '../features/clubs/clubsSlice'  
  
export const clubsApi = createApi({  
 reducerPath: 'clubsApi',  
 baseQuery: fetchBaseQuery({ baseUrl: '/api' }),  
 tagTypes: ['Clubs'],  
 endpoints: builder => ({  
 getClubs: builder.query<Club[], void>({  
 query: () => '/clubs',  
 providesTags: (result) => result  
 ? [...result.map(({ id }) => ({ type: 'Clubs' as const, id })), { type: 'Clubs', id: 'LIST' }]  
 : [{ type: 'Clubs', id: 'LIST' }]  
 }),  
 addClub: builder.mutation<Club, Partial<Club>>({  
 query: (body) => ({ url: '/clubs', method: 'POST', body }),  
 invalidatesTags: [{ type: 'Clubs', id: 'LIST' }]  
 })  
 })  
})  
  
export const { useGetClubsQuery, useAddClubMutation } = clubsApi

\*\*Fragment:\*\* `tests/msw/handlers.ts`

import { http, HttpResponse } from 'msw'  
  
export const handlers = [  
 http.get('/api/clubs', async () => {  
 return HttpResponse.json([  
 { id: 'tech', name: 'Tech & Coding', category: 'technology' },  
 { id: 'arts', name: 'Arts & Culture', category: 'arts' },  
 { id: 'sport', name: 'Sports Club', category: 'sports' }  
 ])  
 }),  
]

\*\*Fragment:\*\* `tests/setup.ts`

import '@testing-library/jest-dom'  
import { server } from './msw/server'  
  
beforeAll(() => server.listen())  
afterEach(() => server.resetHandlers())  
afterAll(() => server.close())

\*\*Fragment:\*\* `tests/vitest/thunk.test.ts`

import { fetchClubs } from '../../src/features/clubs/clubsSlice'  
import { configureStore } from '@reduxjs/toolkit'  
import clubsReducer from '../../src/features/clubs/clubsSlice'  
  
test('fetchClubs fulfills and sets items', async () => {  
 const store = configureStore({ reducer: { clubs: clubsReducer } })  
 await store.dispatch<any>(fetchClubs())  
 const state = store.getState().clubs as any  
 expect(state.status).toBe('succeeded')  
 expect(Object.keys(state.entities).length).toBeGreaterThan(0)  
})

\*\*Fragment:\*\* `tests/vitest/App.test.tsx`

import { render, screen } from '@testing-library/react'  
import userEvent from '@testing-library/user-event'  
import { Provider } from 'react-redux'  
import { configureStore } from '@reduxjs/toolkit'  
import clubsReducer from '../../src/features/clubs/clubsSlice'  
import App from '../../src/App'  
  
function setup(){  
 const store = configureStore({ reducer: { clubs: clubsReducer } })  
 render(<Provider store={store}><App/></Provider>)  
 return { store }  
}  
  
test('adds a club and filters', async () => {  
 setup()  
 const user = userEvent.setup()  
 await user.type(screen.getByLabelText('name-input'), 'Robotics')  
 await user.click(screen.getByRole('button', { name: /add/i }))  
 expect(screen.getByText(/Robotics/i)).toBeInTheDocument()  
 await user.type(screen.getByLabelText('search-input'), 'robo')  
 expect(screen.getAllByLabelText('club-card').length).toBe(1)  
})  
  
test('loads clubs via thunk', async () => {  
 setup()  
 const user = userEvent.setup()  
 await user.click(screen.getByLabelText('load-btn'))  
 expect(await screen.findByText(/Tech & Coding/i)).toBeInTheDocument()  
})