**Seminarul 10 — React: componente și JSX — Partea 2 (Laborator extins)**

\*\*Scopul Părții 2\*\* este să construim, \*end‑to‑end\*, o aplicație didactică „\*\*ClubHub\*\*” în \*\*React 19\*\* cu \*\*Vite\*\* și \*\*TypeScript\*\*, prin exerciții progresive ce ating esența subiectului „Componente și JSX”: \*props/state\*, \*events\*, \*lists & keys\*, \*controlled forms\*, \*composition\*, cu un prim contact disciplinat cu \*transitions\* (non‑blocking updates). Fiecare etapă are: \*\*worksheet (cerință + checklist)\*\*, \*\*cod\*\* (starter + completare), \*\*teste\*\* \*\*Vitest & Jest\*\* (side‑by‑side), și \*\*AI‑assist (VSL)\*\*.

---  
## 0) Setup: proiect Vite + React (TS) și configurarea testelor (Vitest & Jest)  
(Conținut identic cu livrarea anterioară: cerințe, comenzi, explicații, checklist)  
---  
## 1) Componente & JSX (Header, Footer, ClubCard)  
(Conținut identic: cerințe, explicații, teste, AI‑assist)  
---  
## 2) State & Events (SearchBox controlat + filtrare listă)  
...  
---  
## 3) Lists & keys (ClubList cu `key` stabil)  
...  
---  
## 4) Controlled forms (AddClubForm)  
...  
---  
## 5) Composition & children  
...  
---  
## 6) Performance primer (startTransition)  
...  
---  
## 7) Refactoring + a11y pass  
...  
---  
## Codul livrat (în arhiva ZIP)  
...  
---  
## AI‑assist (VSL) — prompts  
...  
---  
## Troubleshooting  
...  
---  
## Extensii  
...

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

{  
 "name": "s10p2-react-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": {  
 "react": "^19.0.0",  
 "react-dom": "^19.0.0"  
 },  
 "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",  
 "typescript": "^5.4.0",  
 "vite": "^5.4.0",  
 "vitest": "^1.6.0"  
 }  
}

\*\*Fragment:\*\* `vitest.config.ts`

import { defineConfig } from 'vitest/config'  
import react from '@vitejs/plugin-react'  
export default defineConfig({ plugins:[react()], test:{ environment:'jsdom', setupFiles:'./tests/setup.ts', css:false } })

\*\*Fragment:\*\* `jest.config.cjs`

module.exports = { testEnvironment:'jsdom', transform:{'^.+\\.(t|j)sx?$':'babel-jest'}, moduleFileExtensions:['ts','tsx','js','jsx'], setupFilesAfterEnv:['<rootDir>/tests/setup.ts'] };

\*\*Fragment:\*\* `babel.config.cjs`

module.exports = { presets: [['@babel/preset-env',{targets:{esmodules:true}}], ['@babel/preset-react',{runtime:'automatic'}], '@babel/preset-typescript'] };

\*\*Fragment:\*\* `src/App.tsx`

import React, { useMemo, useState, startTransition } from 'react'  
import { Club } from './types'  
import { Header } from './components/Header'  
import { Footer } from './components/Footer'  
import { SearchBox } from './components/SearchBox'  
import { ClubList } from './components/ClubList'  
import { AddClubForm } from './components/AddClubForm'  
import { useFilteredClubs } from './hooks/useFilteredClubs'  
const initialClubs: Club[] = [  
 { id: 'tech', name: 'Tech & Coding', category: 'technology' },  
 { id: 'arts', name: 'Arts & Culture', category: 'arts' },  
 { id: 'sport', name: 'Sports Club', category: 'sports' }  
]  
export default function App(){  
 const [query, setQuery] = useState('')  
 const [clubs, setClubs] = useState<Club[]>(initialClubs)  
 const filtered = useFilteredClubs(clubs, query)  
 function addClub(input: Omit<Club, 'id'>){  
 const id = input.name.toLowerCase().replace(/\s+/g,'-')  
 setClubs(prev => [...prev, { id, ...input }])  
 }  
 function onSearchChange(q: string){  
 startTransition(() => setQuery(q))  
 }  
 return (  
 <div className="container stack">  
 <Header title="ClubHub — React + JSX Lab" />  
 <SearchBox value={query} onChange={onSearchChange} placeholder="Search clubs..." />  
 <AddClubForm onAdd={addClub} />  
 <ClubList items={filtered} />  
 <Footer>Built with React 19 + Vite • Lab S10/P2</Footer>  
 </div>  
 )  
}

\*\*Fragment:\*\* `src/components/SearchBox.tsx`

type Props = { value:string; onChange:(v:string)=>void; placeholder?:string }  
export function SearchBox({ value, onChange, placeholder }: Props){  
 return (  
 <label aria-label="search">  
 <span>Search:</span>  
 <input aria-label="search-input" value={value} placeholder={placeholder||'Search...'} onChange={e=>onChange(e.target.value)} />  
 </label>  
 )  
}

\*\*Fragment:\*\* `src/components/ClubCard.tsx`

import { Club } from '../types'  
export function ClubCard({ club }: { club: Club }){  
 return (  
 <article className="card" aria-label="club-card">  
 <h3>{club.name}</h3>  
 <p><strong>Category:</strong> {club.category}</p>  
 </article>  
 )  
}

\*\*Fragment:\*\* `src/components/ClubList.tsx`

import { Club } from '../types'  
import { ClubCard } from './ClubCard'  
export function ClubList({ items }: { items: Club[] }){  
 if(items.length === 0){ return <p role="status">No clubs found.</p> }  
 return <section className="grid" aria-label="club-list">{items.map(c => <ClubCard key={c.id} club={c} />)}</section>  
}

\*\*Fragment:\*\* `src/components/AddClubForm.tsx`

import { useState } from 'react'  
type Props = { onAdd: (club: { name:string; category:string }) => void }  
export function AddClubForm({ onAdd }: Props){  
 const [name, setName] = useState(''); const [category, setCategory] = useState('technology')  
 function submit(e: React.FormEvent){ e.preventDefault(); if(!name.trim()) return; onAdd({ name: name.trim(), category }); setName('') }  
 return (  
 <form onSubmit={submit} aria-label="add-club-form">  
 <label><span>Name</span><input aria-label="name-input" value={name} onChange={e=>setName(e.target.value)} /></label>  
 <label><span>Category</span>  
 <select aria-label="category-select" value={category} onChange={e=>setCategory(e.target.value)}>  
 <option value="technology">technology</option><option value="arts">arts</option><option value="sports">sports</option>  
 </select>  
 </label>  
 <button type="submit">Add</button>  
 </form>  
 )  
}

\*\*Fragment:\*\* `src/hooks/useFilteredClubs.ts`

import { useMemo } from 'react'  
import type { Club } from '../types'  
export function useFilteredClubs(clubs: Club[], query: string){  
 return useMemo(()=>{  
 const q = query.trim().toLowerCase(); if(!q) return clubs  
 return clubs.filter(c => c.name.toLowerCase().includes(q) || c.category.toLowerCase().includes(q))  
 }, [clubs, query])  
}

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

import { render, screen } from '@testing-library/react'  
import userEvent from '@testing-library/user-event'  
import App from '../../src/App'  
test('renders header and initial clubs', async () => {  
 render(<App />)  
 expect(screen.getByRole('heading', { name: /clubhub/i })).toBeInTheDocument()  
 expect(screen.getAllByLabelText('club-card').length).toBeGreaterThan(0)  
})  
test('filters by query and adds a new club', async () => {  
 render(<App />)  
 const user = userEvent.setup()  
 await user.type(screen.getByLabelText('search-input'), 'tech')  
 expect(screen.getAllByLabelText('club-card').length).toBe(1)  
 await user.clear(screen.getByLabelText('search-input'))  
 await user.type(screen.getByLabelText('name-input'), 'Chess Club')  
 await user.click(screen.getByRole('button', { name: /add/i }))  
 expect(screen.getByText(/chess club/i)).toBeInTheDocument()  
})

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

import { render, screen } from '@testing-library/react'  
import userEvent from '@testing-library/user-event'  
import App from '../../src/App'  
test('renders header and initial clubs', async () => {  
 render(<App />)  
 expect(screen.getByRole('heading', { name: /clubhub/i })).toBeInTheDocument()  
 expect(screen.getAllByLabelText('club-card').length).toBeGreaterThan(0)  
})  
test('filters by query and adds a new club', async () => {  
 render(<App />)  
 const user = userEvent.setup()  
 await user.type(screen.getByLabelText('search-input'), 'tech')  
 expect(screen.getAllByLabelText('club-card').length).toBe(1)  
 await user.clear(screen.getByLabelText('search-input'))  
 await user.type(screen.getByLabelText('name-input'), 'Chess Club')  
 await user.click(screen.getByRole('button', { name: /add/i }))  
 expect(screen.getByText(/chess club/i)).toBeInTheDocument()  
})