Resurse suplimentare pe care recomand să le parcurgi

✅ **Exercițiul 1:** Recreează exemplul de gestionare a utilizatorilor și comenzilor din lecția despre Promises, dar folosind async/await.

```
// Funcțiile de exemplu pentru a le implementa cu async/await
async function getUser(id) {
  // Implementează această funcție
}

async function getUserOrders(user) {
  // Implementează această funcție
}

async function calculateTotal(data) {
  // Implementează această funcție
}

async function completeProcessing(id) {
  // Implementează funcția care folosește cele trei funcții anterioare în secvență
}
```

✅ Exercițiul 2: Implementează un sistem de cache pentru operațiuni asincrone

```
// Implementează un wrapper care memorează rezultatele funcțiilor asincrone
// pentru a evita apeluri redundante pentru aceiași parametri
function createAsyncCache(asyncFunction) {
  // Implementează această funcție
}

// Exemplu de utilizare
const cachedGetData = createAsyncCache(async (id) => {
  console.log(`Fetching data for id: ${id}`)
  await delay(1000) // Simulează o operație lentă
  return { id, data: `Data for ${id}` }
})

// Test cache
async function testCache() {
  console.time('Prima cerere')
  await cachedGetData(42)
  console.timeEnd('Prima cerere')

  console.time('Cerere din cache')
  await cachedGetData(42) // Ar trebui să fie instant
  console.timeEnd('Cerere din cache')

  console.time('Cerere diferită')
  await cachedGetData(43) // Ar trebui să fie lentă din nou
  console.timeEnd('Cerere diferită')
}
```

✅ Exercițiul 3: Implementează o funcție care execută operațiuni asincrone în serie, limitând numărul de operațiuni concurente.

```
async function processQueue(items, processFn, concurrencyLimit = 2) {
  // Implementează o funcție care procesează `items` folosind `processFn`
  // dar care nu rulează mai mult de `concurrencyLimit` operațiuni simultan
}

// Utilizare
async function testProcessQueue() {
  const items = [1, 2, 3, 4, 5, 6, 7, 8]

  // Simulează o operațiune asincronă care durează un timp variabil
  async function processItem(item) {
    console.log(`Începe procesarea pentru ${item}`)
    await delay(item * 500) // Durata variază în funcție de item
    console.log(`Procesare finalizată pentru ${item}`)
    return item * 2
  }

  const results = await processQueue(items, processItem, 3)
  console.log('Rezultate:', results)
}
```