Cerințe Workout

Cerințe:

1. Convertește funcția de autentificare din Partea 2 pentru a utiliza async/await.
2. Reescrie funcția de încărcare dashboard din Partea 3 folosind async/await.
3. Implementează gestionarea erorilor cu try/catch.
4. Adaugă funcționalitate de progress tracking pentru operațiuni lungi.

Cod de pornire: vezi pag 2

Extindere:

Implementează un sistem de cache pentru rezultatele operațiunilor asincrone pentru a evita apeluri redundante.

```javascript
// Convertește sistemul de autentificare la async/await
async function authenticateUser(username, password) {
  try {
    console.log(`🔐 Autentificare pentru ${username}...`)

    const user = await validateCredentials(username, password)
    console.log('✅ Credențiale valide:', user.name)

    const token = await generateToken(user)
    console.log('✅ Token generat')

    const profile = await getUserProfile(token)
    console.log('✅ Profil utilizator încărcat:', profile.name)

    return { user, token, profile }
  } catch (error) {
    console.error('❌ Eroare autentificare:', error.message)
    throw error
  }
}

// Convertește încărcarea dashboard-ului la async/await
async function loadDashboardModern() {
  try {
    console.log('🚀 Începe încărcarea dashboard-ului modern...')

    // Încărcăm datele independente în paralel
    const [users, products, orders] = await Promise.all([
      retryOperation(loadUsers),
      retryOperation(loadProducts),
      retryOperation(loadOrders),
    ])

    console.log('✅ Date de bază încărcate cu succes')

    const data = { users, products, orders }

    // Calculăm statisticile secvențial
    const statistics = await calculateStatistics(users, products, orders)
    console.log('✅ Statistici calculate')

    const report = await generateReport(data, statistics)
    console.log('✅ Raport generat')

    return report
  } catch (error) {
    console.error('❌ Eroare la încărcarea dashboard-ului:', error.message)
    throw error
  }
}

// Funcție pentru tracking progres
function createProgressTracker(steps) {
  let currentStep = 0

  return {
    nextStep: function (stepName) {
      currentStep++
      const percentage = Math.round((currentStep / steps.length) * 100)
      console.log(`📊 Progres: ${percentage}% - ${stepName}`)
    },

    complete: function () {
      console.log('✅ Operațiune completă!')
    },
  }
}

// Funcție async cu progress tracking
async function loadDataWithProgress() {
  const steps = [
    'Validare',
    'Autentificare',
    'Încărcare date',
    'Calcul statistici',
    'Generare raport',
  ]
  const progress = createProgressTracker(steps)

  try {
    progress.nextStep('Validare credențiale')
    const authResult = await authenticateUser('ana', 'parola123')

    progress.nextStep('Autentificare completă')
    await new Promise((resolve) => setTimeout(resolve, 500))

    progress.nextStep('Încărcare date dashboard')
    const dashboard = await loadDashboardModern()

    progress.nextStep('Calcul statistici finale')
    await new Promise((resolve) => setTimeout(resolve, 300))

    progress.nextStep('Generare raport final')
    await new Promise((resolve) => setTimeout(resolve, 200))

    progress.complete()

    return {
      auth: authResult,
      dashboard: dashboard,
    }
  } catch (error) {
    console.error('❌ Eroare:', error.message)
    throw error
  }
}

// Testare
async function runTests() {
  console.log('=== Test Autentificare Async/Await ===')
  try {
    await authenticateUser('ana', 'parola123')
  } catch (error) {
    console.error('Test autentificare eșuat:', error.message)
  }

  console.log('\\n=== Test Dashboard Modern ===')
  try {
    const dashboard = await loadDashboardModern()
    console.log('Dashboard încărcat:', dashboard.summary)
  } catch (error) {
    console.error('Test dashboard eșuat:', error.message)
  }

  console.log('\\n=== Test Progress Tracking ===')
  try {
    const result = await loadDataWithProgress()
    console.log('Proces complet finalizat pentru:', result.auth.profile.name)
  } catch (error) {
    console.error('Test progress eșuat:', error.message)
  }
}

runTests()
```