Cerințe Workout

Cerințe:

1. Creează o funcție `validateCredentials(username, password)` care returnează un Promise.
2. Implementează `generateToken(user)` care returnează un Promise cu un token de autentificare.
3. Creează `getUserProfile(token)` care returnează profilul utilizatorului pe baza token-ului.
4. Toate funcțiile trebuie să includă simulări de erori și validări.

Cod de pornire: vezi pag 2

Extindere:

Implementează un sistem de expirare pentru token-uri și o funcție de refresh pentru token-uri expirate.

```
// Baza de date simplă de utilizatori
const users = [
  {
    id: 1,
    username: 'ana',
    password: 'parola123',
    name: 'Ana Popescu',
    role: 'admin',
  },
  {
    id: 2,
    username: 'ion',
    password: 'secret456',
    name: 'Ion Ionescu',
    role: 'user',
  },
  {
    id: 3,
    username: 'maria',
    password: 'pass789',
    name: 'Maria Dumitrescu',
    role: 'user',
  },
]

// Funcția pentru validarea credențialelor
function validateCredentials(username, password) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (!username || !password) {
        reject(new Error('Username și password sunt obligatorii'))
        return
      }

      const user = users.find(
        (u) => u.username === username && u.password === password
      )

      if (user) {
        resolve(user)
      } else {
        reject(new Error('Credențiale invalide'))
      }
    }, 1000)
  })
}

// Funcția pentru generarea token-ului
function generateToken(user) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (!user || !user.id) {
        reject(new Error('Utilizator invalid'))
        return
      }

      // Generează un token simplu (în realitate ar trebui să folosești JWT)
      const token = `${user.id}-${Date.now()}-${Math.random()
        .toString(36)
        .substr(2, 9)}`
      resolve(token)
    }, 500)
  })
}

// Funcția pentru preluarea profilului utilizatorului
function getUserProfile(token) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (!token) {
        reject(new Error('Token lipsă'))
        return
      }

      // Extrage ID-ul utilizatorului din token (implementare simplă)
      const userId = parseInt(token.split('-')[0])
      const user = users.find((u) => u.id === userId)

      if (user) {
        resolve({
          id: user.id,
          name: user.name,
          role: user.role,
          lastLogin: new Date().toISOString(),
        })
      } else {
        reject(new Error('Token invalid sau expirat'))
      }
    }, 800)
  })
}

// Testare cu .then() și .catch()
console.log('Testare sistem autentificare...')

validateCredentials('ana', 'parola123')
  .then((user) => {
    console.log('✅ Credențiale valide:', user.name)
    return generateToken(user)
  })
  .then((token) => {
    console.log('✅ Token generat:', token)
    return getUserProfile(token)
  })
  .then((profile) => {
    console.log('✅ Profil utilizator:', profile)
  })
  .catch((error) => {
    console.error('❌ Eroare autentificare:', error.message)
  })
```