**ЗВІТ: Етап 3 - Розробка веб-орієнтованої версії**

**Варіант:** 88  
**Обрана технологія:** REST API  
**Технологічний стек:** React (веб-браузер) + Spring Boot REST API + PostgreSQL

**1. Вибір технології**

Обрано REST API як основну технологію для веб-версії.

**Чому REST:**

* Простота реалізації та підтримки
* Стандартні HTTP методи (GET, POST, PUT, DELETE)
* JSON формат даних - зручний для JavaScript
* Не потребує додаткових бібліотек на фронтенді
* Легко тестувати (Postman, curl)
* Той самий backend що і для десктопу

**Альтернативи:**

* GraphQL - складніше, overkill для file manager
* gRPC - не підтримується нативно в браузері
* Монолітна (SSR) - гірший UX, full page reload

**2. Реалізація**

**Веб-клієнт (React):**

Браузерний додаток на localhost:3000 (development) який робить HTTP запити до API на localhost:8081.

Той самий код що й у Electron версії, але без:

* Electron preload API
* Синхронізації папок (браузер не має доступу до ФС)
* Fullscreen drag overlay

Залишився весь основний функціонал: login, upload, download, view, delete, replace, sort, filter.

**Backend (Spring Boot REST):**

Той самий що використовується для десктоп-версії. 11 REST endpoints:

**Auth:**

* POST /api/auth/register
* POST /api/auth/login
* GET /api/auth/validate

**Files:**

* POST /api/files/upload
* GET /api/files/list?ascending={bool}&types={array}
* GET /api/files/download/{id}
* GET /api/files/{id}
* PUT /api/files/{id}
* DELETE /api/files/{id}

**Sync:**

* POST /api/sync/compare
* GET /api/sync/remote-files

**HTTP статус коди:**

* 200 OK - успішна операція
* 400 Bad Request - невалідні дані
* 401 Unauthorized - немає/невалідний токен
* 403 Forbidden - немає прав (delete/replace чужого файлу)

**CORS:**

Налаштовано дозвіл cross-origin запитів з localhost:3000 на localhost:8081. Spring Security дозволяє всі методи (GET, POST, PUT, DELETE, OPTIONS) та всі заголовки. Credentials: true для передачі токенів.

**3. Відмінності від десктоп-версії**

**Що однакове:**

* Той самий Spring Boot backend
* Той самий React код (85%)
* Той самий REST API
* Те саме JWT authentication
* Те саме сортування/фільтрація

**Що відрізняється:**

| **Десктоп** | **Веб** |
| --- | --- |
| Electron обгортка | Браузер |
| Доступ до локальної ФС | Тільки через file input |
| Синхронізація папок | Немає |
| Fullscreen drag overlay | Звичайний drag-n-drop |
| Розмір ~180MB | Розмір ~3MB |
| Треба встановлювати | Просто відкрити URL |

**4. Запуск**

**Backend:**

cd backend

mvn spring-boot:run

Запускається на localhost:8081

**Frontend:**

cd frontend

npm start

Запускається на localhost:3000

Браузер автоматично відкриває додаток. Всі запити йдуть на backend через CORS.

**5. Тестування**

REST API протестовано через:

* Postman (всі 11 endpoints)
* curl команди
* Browser DevTools (Network tab)

Всі endpoints повертають коректні HTTP статуси та JSON відповіді згідно специфікації.

**Висновки**

**Реалізовано:** ✓ Веб-клієнт на React  
✓ Повний REST API (11 endpoints)  
✓ Всі операції з файлами через HTTP  
✓ Індивідуальна операція (сортування + фільтрація)  
✓ JWT автентифікація  
✓ CORS для cross-origin запитів

**Переваги REST підходу:**

* Простота
* Стандартизація
* Легке тестування
* Сумісність з будь-якими клієнтами

**Недоліки:**

* Over-fetching (отримуємо всі поля)
* Немає real-time без WebSocket