**ЗВІТ: Етап 2 - Розробка десктоп-версії**

**Варіант:** 88  
**Технологічний стек:** Electron + React (frontend), Spring Boot + PostgreSQL (backend)

**1. Розробка**

Реалізовано десктоп-додаток який складається з двох частин:

**Frontend (Electron + React):**

* Electron обгортка для доступу до файлової системи
* React інтерфейс для роботи з файлами
* HTTP клієнт для взаємодії з backend

**Backend (Spring Boot):**

* REST API контролери (Auth, Files, Sync)
* Сервіси для бізнес-логіки
* JWT автентифікація
* PostgreSQL для метаданих
* File storage на локальному диску

**Реалізований функціонал:**

Обов'язковий:

* Авторизація на віддаленому сервері (register/login)
* Перегляд файлів з атрибутами (name, created, modified, uploader, editor)
* Завантаження файлів
* Вивантаження файлів
* Видалення файлів (тільки автор)
* Перегляд вмісту .kt та .jpg файлів
* Перемикання видимості колонок (окрім Name)

Індивідуальна операція (варіант 88):

* Сортування за датою модифікації (зростання/спадання)
* Фільтр за типом файлу (.kt, .jpg)

Бонуси:

* Синхронізація локальної папки з сервером (+5 балів)
* Drag-n-drop завантаження (+5 балів)

**Технічні рішення:**

Спільний робочий простір - всі користувачі бачать файли всіх інших. Видалення та заміна дозволені тільки автору файлу.

JWT токени зберігаються в localStorage та передаються у заголовку Authorization для кожного запиту.

Файли зберігаються як userId\_filename на диску сервера. Метадані у PostgreSQL таблиці file\_metadata з індексом на modified\_date для швидкого сортування.

Electron preload API експонує функції writeFile, joinPath, getFolderPath через contextBridge для безпечного доступу до файлової системи з React коду.

**2. Unit-тестування**

Реалізовано 29 тестів з використанням JUnit 5 + Mockito.

**AuthServiceTest (8 тестів):**

* Реєстрація користувача (успішна та з дублікатом username)
* Логін (успішний, невірний username, невірний password)
* Валідація JWT токену
* Витягнення даних з токену

**FileServiceTest (9 тестів):**

* Завантаження різних типів файлів
* Перевірка дублікатів
* Видалення файлів
* Заміна файлів
* Зміна типу файлу при заміні

**SortFilterServiceTest (7 тестів) - ІНДИВІДУАЛЬНА ОПЕРАЦІЯ:**

* Сортування за modified\_date ascending
* Сортування за modified\_date descending
* Фільтрація за типом .kt
* Фільтрація за типом .jpg
* Фільтрація за кількома типами
* Комбінація сортування + фільтрація
* Запит файлів конкретного користувача

**SyncControllerTest (5 тестів):**

* Порівняння локальних і віддалених файлів
* Визначення файлів для upload
* Визначення файлів для download
* Синхронізація у спільному робочому просторі

**Результат:** Всі 29 тестів пройдено успішно. Coverage критичних методів 100%.

**3. Забезпечення GUI**

Графічний інтерфейс реалізовано на React з inline styles.

**Екрани:**

1. **Login** - форма входу (username, password)
2. **Register** - форма реєстрації (username, email, password)
3. **FileList** - головний екран з таблицею файлів
4. **FileViewer** - модальне вікно для перегляду .kt (текст) та .jpg (зображення)
5. **Sync** - діалог синхронізації локальної папки (тільки в Electron)

**Головний екран FileList:**

Drag-n-drop зона для завантаження файлів.

Таблиця з колонками: Name, Created Date, Modified Date, Uploader, Editor, Actions. Всі окрім Name можна приховати через чекбокси.

Кнопки сортування: Ascending, Descending, Clear. Активна кнопка підсвічується.

Кнопки фільтрації: .kt, .jpg. Можна вибрати обидві. Активні підсвічуються.

Для кожного файлу: View (якщо .kt або .jpg), Download, Replace, Delete (тільки автор).

Кнопка Sync Folder (тільки в Electron версії).

**Особливості Electron:**

Fullscreen overlay для drag-n-drop з провідника. Повноекранний невидимий file input активується під час dragging для перехоплення drop подій.

Синхронізація: користувач вибирає локальну папку через webkitdirectory input, програма порівнює з сервером та автоматично завантажує/вивантажує файли.

**4. Текстовий опис роботи системи**

**Запуск:**

Backend запускається на localhost:8081 через mvn spring-boot:run. Frontend запускається через npm run electron:dev - автоматично стартує React dev server на localhost:3000 та відкриває Electron вікно.

**Авторизація:**

Користувач вводить username та password. Frontend відправляє POST /api/auth/login. Backend перевіряє через BCrypt, генерує JWT токен. Frontend зберігає токен у localStorage. При наступних запусках автоматично логінить якщо токен валідний.

**Робота з файлами:**

Список файлів: GET /api/files/list отримує всі файли всіх користувачів. Frontend відображає у таблиці.

Сортування: Натискання кнопки → GET /api/files/list?ascending=true. Backend викликає findAllByOrderByModifiedDateAsc() через JPA. PostgreSQL використовує index на modified\_date.

Фільтрація: Вибір типів → GET /api/files/list?types=kt&types=jpg. Backend фільтрує через Stream API.

Завантаження: Drag-n-drop або file input → POST /api/files/upload з multipart/form-data. Backend зберігає як userId\_filename, створює запис у БД.

Перегляд: Клік View → GET /api/files/download/{id}. Для .kt - відображення тексту у pre, для .jpg - відображення img.

Видалення: Клік Delete (тільки для автора) → window.confirm → DELETE /api/files/{id}. Backend перевіряє uploaderId === userId з токену, видаляє з диску та БД.

**Синхронізація:**

Користувач вибирає локальну папку. Frontend читає список файлів через File API, відправляє POST /api/sync/compare з масивом імен.

Backend повертає toUpload (файли які є локально але немає на сервері) та toDownload (файли які є на сервері але немає локально).

Frontend завантажує файли з toUpload через POST /api/files/upload.

Frontend вивантажує файли з toDownload через GET /api/files/download/{id} та зберігає на диск через window.electronAPI.writeFile().

Відображається результат: кількість uploaded, downloaded, skipped файлів.

**Висновки**

Реалізовано повнофункціональний десктоп-клієнт з усіма вимогами завдання.

**Виконано:**

* Розробка Electron + Spring Boot додатку
* 29 unit-тестів (включно з тестами індивідуальної операції)
* Графічний інтерфейс
* Текстовий опис роботи

**Бонуси:**

* Синхронізація локальної папки (+5 балів)
* Drag-n-drop (+5 балів)