
Project2: Οριζόντιο Repository/Aggregator Ανοικτών Μαθημάτων με React Front-end και Spark για Large-Scale ML

---

1. Σκοπός της Εργασίας

Στόχος είναι οι φοιτητές να υλοποιήσουν ένα **οριζόντιο repository/aggregator** που:

- **Συγκεντρώνει** δεδομένα μαθημάτων από **πολλαπλά εξωτερικά repositories** (π.χ. πλατφόρμες ανοικτών μαθημάτων / MOOCs).
- Προσφέρει **ενιαίο front-end** σε **React** για αναζήτηση, φιλτράρισμα και εξερεύνηση μαθημάτων.
- Χρησιμοποιεί **Apache Spark** για **large-scale επεξεργασία δεδομένων και Machine Learning**, π.χ. συστήματα συστάσεων μαθημάτων, clustering, analytics.

Έτσι, οι φοιτητές θα εξασκηθούν σε:

- Web front-end (React).
- Σχεδίαση REST APIs και διαλειτουργικότητα με τρίτα συστήματα.
- Διαχείριση δεδομένων (ETL/aggregation).
- **Big data processing και ML με Spark.**

---

2. Περιγραφή Σεναρίου

Το σύστημα θα λειτουργεί ως **"οριζόντιο" repository**:

- Δεν αντικαθιστά τα υπάρχοντα repositories, αλλά **"κουμπώνει" πάνω τους** μέσω APIs / εισαγωγής δεδομένων.
- Κάνει **harvest / import** μεταδεδομένα μαθημάτων (τίτλος, περιγραφή, θεματική περιοχή, γλώσσα, επίπεδο κ.λπ.).
- Αποθηκεύει τα δεδομένα σε **δική του βάση** με **ενιαίο σχήμα**.
- Παρέχει **ενιαία αναζήτηση** ανεξάρτητα από την αρχική πηγή.
- Εκτελεί **Spark jobs** για:

- ανάλυση χρήσης/προβολών,
- clustering μαθημάτων,
- απλό σύστημα συστάσεων (π.χ. “similar courses” ή “οι χρήστες που είδαν αυτό, είδαν και…”).

---

3. Τεχνολογίες (Υποχρεωτικές & Προτεινόμενες)

- **Front-end:**
  - React (π.χ. με hooks, React Router κ.λπ.).
- **Back-end / API layer:**
  - Node.js (Express) ή άλλο αντίστοιχο framework (αφήνεται επιλογή, αλλά να τεκμηριωθεί).
- **Βάση δεδομένων:**
  - SQL (π.χ. PostgreSQL/MySQL) ή NoSQL (π.χ. **MongoDB**).
- **Big Data & ML:**
  - Apache Spark (Spark SQL, Spark MLlib, ή **PySpark**).
- **Διαλειτουργικότητα:**
  - REST/JSON APIs για εισαγωγή δεδομένων από άλλα repositories.

---

4. Λειτουργικές Απαιτήσεις

4.1 Aggregation / Harvesting

1. Υποστήριξη **τουλάχιστον 2 διαφορετικών πηγών** μαθημάτων (π.χ. “Repository A”, “Repository B”).
2. Για κάθε πηγή:
   - Υλοποίηση **connector** που:
     - Καλεί το API της πηγής (ή αναλύει export σε JSON/CSV).
     - Μετασχηματίζει τα δεδομένα στο **ενιαίο σχήμα** του δικού σας repository.
   - Δυνατότητα **πλήρους συγχρονισμού** (full import) και **μερικής ενημέρωσης** (π.χ. μόνο νέα/τροποποιημένα μαθήματα).

4.2 Ενιαίο Repository

Το εσωτερικό repository πρέπει να περιλαμβάνει τουλάχιστον για κάθε μάθημα:

- Τίτλο
- Σύντομη περιγραφή
- Λέξεις-κλειδιά / θεματική κατηγορία
- Γλώσσα
- Επίπεδο (π.χ. beginner/intermediate/advanced)
- Πηγή (repository name, URL)
- Link εγγραφής/πρόσβασης
- Ημερομηνία τελευταίας ενημέρωσης

4.3 Front-end (React)

Το web UI πρέπει να επιτρέπει:

1. **Αναζήτηση** με βάση τίτλο/λέξεις-κλειδιά.
2. **Φιλτράρισμα** με βάση:
    - γλώσσα,
    - επίπεδο,
    - πηγή repository,
    - θεματική περιοχή.
3. **Προβολή λίστας** μαθημάτων με:
    - σύντομες πληροφορίες,
    - ένδειξη πηγής,
    - κουμπί “View details”.
4. **Σελίδα λεπτομερειών** μαθήματος με:
    - αναλυτική περιγραφή,
    - link προς το αρχικό repository,
    - προτεινόμενα/σχετικά μαθήματα (μέσω Spark ML – βλ. παρακάτω).
5. **Analytics views (προαιρετικά αλλά επιθυμητά)**:

- βασικά στατιστικά (π.χ. πόσα μαθήματα ανά πηγή, ανά γλώσσα, ανά θεματική).

4.4 Spark & Machine Learning

Οι φοιτητές πρέπει να υλοποιήσουν **τουλάχιστον ένα σενάριο ML** με Spark, π.χ.:

1. **Course Similarity / Recommendations:**
   - *Χρήση περιγραφών/λέξεων-κλειδιών για **vectorization** (TF-IDF).*
   - Υπολογισμός similarity (π.χ. cosine similarity) μεταξύ μαθημάτων.
   - Από το front-end: όταν ο χρήστης βλέπει ένα μάθημα, να εμφανίζονται **"Similar courses"** που προκύπτουν από το Spark job.

ή/και

2. **Clustering Μαθημάτων:**
   - Ομαδοποίηση μαθημάτων σε clusters βάσει θεματικού περιεχομένου.
   - Εμφάνιση cluster label/ID στο UI.

ή/και

3. **Χρήση δεδομένων χρήστη (αν προλάβουν):**
   - Καταγραφή προβολών/likes σε τοπικό log.
   - Spark job που μαζεύει τα logs και παράγει personalized suggestions (απλό επίπεδο).

---

5. Μη Λειτουργικές Απαιτήσεις

- **Καθαρή αρχιτεκτονική** (διαχωρισμός front-end / back-end / Spark jobs).
- **API τεκμηρίωση** (π.χ. OpenAPI/Swagger ή έστω αναλυτικό README).
- **Κλιμακωσιμότητα σε επίπεδο σχεδιασμού** (να φαίνεται ότι το σύστημα μπορεί να επεκταθεί με νέους connectors).
- **Κώδικας σε git repository** με σαφή δομή φακέλων.

---

6. Ενδεικτική Αρχιτεκτονική

1. **Front-end (React app):**

- Καλεί το back-end μέσω REST APIs.

2. **Back-end (API Server):**
    - Endpoints για:
        - /courses (list, filters, pagination)
        - /courses/{id} (details)
        - /courses/{id}/similar (Spark-based recommendations)
        - /sync/{source} (trigger harvesting από συγκεκριμένη πηγή)
    - Επικοινωνεί με DB και αποθηκεύει τα aggregated μαθήματα.
3. **Spark Layer:**
    - Job για reading από τη DB (ή exported datasets).
    - Εφαρμογή ML pipelines (feature extraction, training, similarity).
    - Εξαγωγή αποτελεσμάτων (π.χ. πίνακας similar_courses ή cluster assignments) πίσω στη DB ή σε άλλο αποθετήριο που θα διαβάζει το API.

---

7. Βήματα Υλοποίησης (Οδηγός για τους Φοιτητές)

1. **Ανάλυση απαιτήσεων & μοντελοποίηση δεδομένων**
    - Ορισμός ενιαίου σχήματος μαθήματος.
    - ER διάγραμμα ή schema design.
2. **Υλοποίηση Connectors / Harvesters**
    - Scripts ή services που φέρνουν δεδομένα από τις εξωτερικές πηγές.
3. **Υλοποίηση Back-end API**
    - CRUD + search endpoints.
    - Endpoint που σερβίρει δεδομένα για Spark jobs (αν χρειάζεται).
4. **Υλοποίηση Spark ML Pipeline**
    - Προετοιμασία δεδομένων (cleaning, text preprocessing).
    - Επιλογή αλγορίθμου (similarity, clustering κ.λπ.).
    - Export των αποτελεσμάτων για χρήση από το API.
5. **Υλοποίηση Front-end (React)**

- Σελίδες: Αναζήτηση, Λίστα, Λεπτομέρειες, Analytics.
- Κλήσεις API, διαχείριση κατάστασης, βασική UX βελτιστοποίηση.

6. **Testing & Τεκμηρίωση**
    - Unit tests ή έστω βασικά integration tests.
    - README με οδηγίες εγκατάστασης, run, και περιγραφή αρχιτεκτονικής.

---

8. Παραδοτέα

Κάθε ομάδα καταθέτει:

1. **Κώδικα** σε git repository (με tags για βασικά milestones).
2. **Τεχνική αναφορά (8–15 σελίδες)** που περιλαμβάνει:
    - Σκοπό και περιγραφή συστήματος.
    - Αρχιτεκτονική, διαγράμματα (ER, υψηλού επιπέδου).
    - Περιγραφή connectors και API.
    - Περιγραφή Spark ML pipeline (αλγόριθμοι, μετασχηματισμοί, features).
    - Screenshots του front-end.
    - Πειραματικά αποτελέσματα (π.χ. παραδείγματα recommendations, clusters).
3. **Σύντομη παρουσίαση (10–15 λεπτών)** στην οποία θα δείξουν:
    - Live demo του συστήματος.
    - Αρχιτεκτονικά διαγράμματα.
    - Βασικά αποτελέσματα ML.

---

9. Τρόπος Αξιολόγησης (Ενδεικτικά Ποσοστά)

- 25% Front-end (React UI, UX, λειτουργικότητα).
- 25% Back-end & data aggregation (API design, connectors, μοντέλο δεδομένων).

- 30% Spark & ML κομμάτι (σωστή χρήση, ποιότητα αποτελεσμάτων, τεκμηρίωση).
- 20% Τεκμηρίωση & Παρουσίαση (αναφορά, διαγράμματα, καθαρότητα).

---

10. Ενδεικτικές Επεκτάσεις (για bonus)

- Υποστήριξη περισσότερων πηγών repositories.
- User accounts & basic personalization (π.χ. bookmarks, history).
- Πιο προχωρημένα ML μοντέλα (π.χ. collaborative filtering, embeddings).
- Dashboard για admins με monitoring των συγχρονισμών.
