

# **ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ**

**Μαριος Τσουλοπουλος  
ics22190**

**Εργασία 3**

**ΘΕΣΣΑΛΟΝΙΚΗ, 2024**

# Backend

## Spring WebFlux

Για την ανάπτυξη του backend της εφαρμογής, επιλέχθηκε το Spring WebFlux, επειδή προσφέρει σημαντικά πλεονεκτήματα για σύγχρονες εφαρμογές που απαιτούν υψηλή απόδοση και κλιμακωσιμότητα. Το WebFlux χρησιμοποιεί ένα μοντέλο προγραμματισμού βασισμένο στην αντιδραστική αρχιτεκτονική(reactive), το οποίο επιτρέπει τη διαχείριση πολλών ταυτόχρονων αιτημάτων με αποτελεσματικό τρόπο, χωρίς να δεσμεύονται threads, χάρη στη χρήση non-blocking IO.

Ένας βασικός λόγος για την επιλογή του ήταν η δυνατότητα διαχείρισης μεγάλου αριθμού χρηστών ταυτόχρονα, καθώς το WebFlux μειώνει την κατανάλωση πόρων και τον χρόνο αναμονής, ειδικά όταν υπάρχουν λειτουργίες που εξαρτώνται από αργές εξωτερικές υπηρεσίες, όπως βάσεις δεδομένων ή API. Επίσης, ευθυγραμμίζεται με τις αρχές σύγχρονων εφαρμογών, όπως οι μικροϋπηρεσίες, και υποστηρίζει τεχνολογίες όπως τα WebSockets και Server-Sent Events, που είναι απαραίτητες για real-time λειτουργίες.

Η χρήση του WebFlux συνδυάζεται άψογα με το R2DBC, το οποίο επιτρέπει αντιδραστική επικοινωνία με τη βάση δεδομένων, εξασφαλίζοντας ομαλή και αποδοτική ροή δεδομένων από την αρχή έως το τέλος της εφαρμογής. Έτσι, το WebFlux αποτέλεσε την ιδανική επιλογή για την υλοποίηση του backend, διασφαλίζοντας ότι η εφαρμογή θα μπορεί να ανταποκρίνεται αποτελεσματικά στις ανάγκες των χρηστών.

## Βιβλιοθήκες

### **io.jsonwebtoken:jjwt-api:0.12.6**

Παρέχει APIs για τη δημιουργία, υπογραφή και επαλήθευση JSON Web Tokens (JWTs) για ασφάλεια και αυθεντικοποίηση.

### **org.projectlombok:lombok:1.18.18**

Προσφέρει annotations που μειώνουν τον boilerplate κώδικα, όπως getters, setters, constructors και άλλες χρήσιμες λειτουργίες για αντικείμενα.

### **org.mapstruct:mapstruct:1.6.2**

Είναι μια βιβλιοθήκη χαρτογράφησης που διευκολύνει τη μετατροπή αντικειμένων μεταξύ διαφορετικών δομών δεδομένων (DTOs, Entities) μέσω annotations.

### **org.springframework.boot:spring-boot-starter-webflux**

Παρέχει όλα τα απαραίτητα για την ανάπτυξη αντιδραστικών (reactive) web εφαρμογών, συμπεριλαμβανομένου του Spring WebFlux και υποστήριξης non-blocking IO.

### **org.springframework.boot:spring-boot-starter-security**

Περιλαμβάνει εργαλεία για την υλοποίηση λειτουργιών ασφάλειας, όπως αυθεντικοποίηση και εξουσιοδότηση, με εύκολο τρόπο στο Spring Boot.

### **org.springframework.boot:spring-boot-starter-data-r2dbc**

Προσφέρει υποστήριξη για αντιδραστική επικοινωνία με βάσεις δεδομένων χρησιμοποιώντας το R2DBC (Reactive Relational Database Connectivity).

### **com.google.code.findbugs:jsr305:3.0.2**

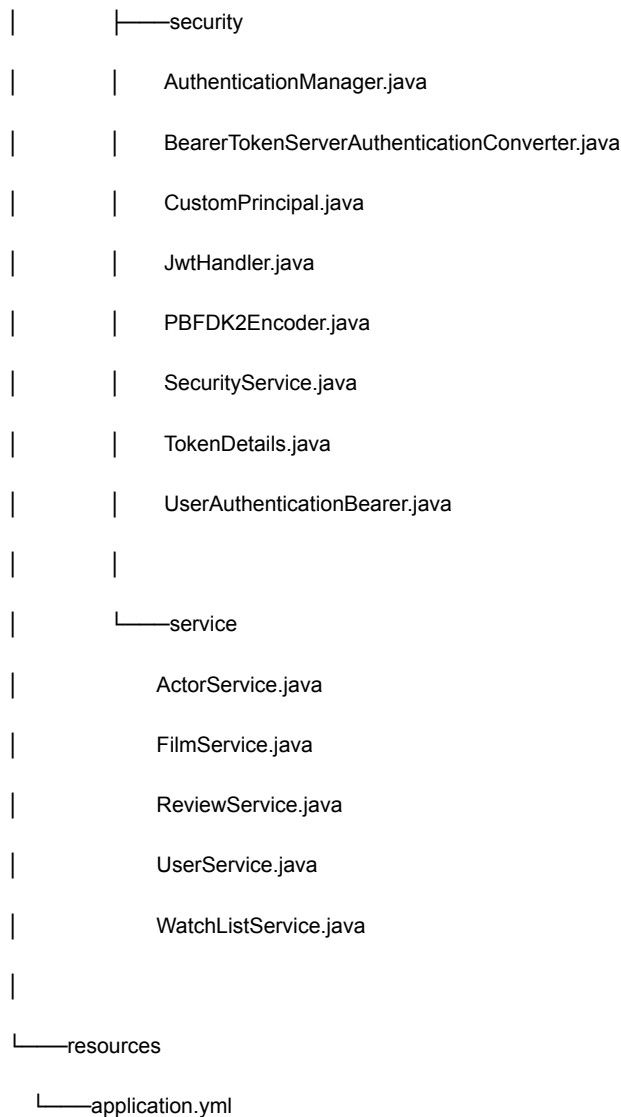
Παρέχει annotations για βελτίωση της ποιότητας του κώδικα, όπως έλεγχος nullability, βοηθώντας στην ανίχνευση σφαλμάτων κατά τη μεταγλώττιση.

## **Tree**

java

```
|  └── gr
|
|  └── cleavest
|
|      └── webfluxfilmsdb
|          |
|          |   WebfluxfilmsdbApplication.java
|          |
|          |
|          |   └── config
|          |       |
|          |       |   WebSecurityConfig.java
|          |       |
|          |       |
|          |       |   └── dto
|          |           |
|          |           |   AuthRequestDTO.java
|          |           |
|          |           |   AuthResponseDTO.java
|          |           |
|          |           |   FilmDTO.java
|          |           |
|          |           |   ReviewDTO.java
|          |           |
|          |           |   ReviewRequestDTO.java
|          |           |
|          |           |   SimpleDTO.java
|          |           |
|          |           |   UserDTO.java
|          |           |
|          |           |   WatchListDTO.java
|          |           |
|          |           |   WatchListResponseDTO.java
|          |           |
|          |           |
|          |           |   └── entity
|          |               |
|          |               |   ActorEntity.java
|          |               |
|          |               |   FilmEntity.java
|          |               |
|          |               |   MovieActorEntity.java
|          |               |
|          |               |   ReviewEntity.java
```

		UserEntity.java
		WatchListEntity.java
		errorhandling
		AppErrorAttributes.java
		AppErrorWebExceptionHandler.java
		exception
		ApiException.java
		AuthException.java
		UnauthorizedException.java
		mapper
		FilmMapper.java
		UserMapper.java
		WatchListMapper.java
		repository
		ActorRepository.java
		FilmRepository.java
		MovieActorRepository.java
		ReviewRepository.java
		UserRepository.java
		WatchListRepository.java
		rest
		AuthRestController.java
		FilmController.java
		ReviewController.java
		WatchListController.java



## application.yml

περιέχει τις βασικές ρυθμίσεις της εφαρμογής, όπως το port στο οποίο λειτουργεί ο server, τα στοιχεία σύνδεσης με τη βάση δεδομένων (όνομα χρήστη, κωδικός πρόσβασης και URL της βάσης) και τις παραμέτρους ασφάλειας για τη διαχείριση JWT (όπως το μυστικό κλειδί και η διάρκεια ζωής των tokens).

## WebSecurityConfig

ρυθμίζει την ασφάλεια της εφαρμογής, επιτρέποντας πρόσβαση σε συγκεκριμένες δημόσιες διαδρομές, υλοποιώντας CORS, και χρησιμοποιώντας ένα φίλτρο αυθεντικοποίησης με Bearer Tokens και JWT. Περιλαμβάνει επίσης μηχανισμούς διαχείρισης μη εξουσιοδοτημένων και απαγορευμένων αιτημάτων.

## DTO(package)

Το πακέτο περιέχει διάφορα DTO (Data Transfer Objects), τα οποία είναι απλά αντικείμενα που χρησιμοποιούνται για την ανταλλαγή δεδομένων μεταξύ του backend και των endpoints της εφαρμογής. Τα DTO λειτουργούν αποκλειστικά ως φορείς δεδομένων και δεν περιέχουν επιχειρηματική λογική ή πολύπλοκες λειτουργίες, οπότε δεν απαιτείται περαιτέρω ανάλυση ή εστίαση σε αυτά.

## **entity(package)**

Το πακέτο περιέχει διάφορα entities, τα οποία αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων και χρησιμοποιούνται για την αποθήκευση και ανάκτηση δεδομένων. Τα entities χρησιμεύουν κυρίως για τη χαρτογράφηση (mapping) των δεδομένων στη βάση και δεν περιέχουν λογική, οπότε δεν απαιτείται περαιτέρω ανάλυση ή λεπτομερής περιγραφή τους.

## **Security**

### **PBKDF2Encoder**

Η κλάση υλοποιεί το PasswordEncoder και περιλαμβάνει δύο μεθόδους: το encode, το οποίο λαμβάνει τον κωδικό και τον κρυπτογραφεί, και το matches, που συγκρίνει δύο κωδικούς για επαλήθευση. Επέλεξα να χρησιμοποιήσω τον αλγόριθμο PBKDF2 αντί για BCrypt, καθώς το BCrypt είναι ευρέως χρησιμοποιούμενο, και ήθελα να δοκιμάσω κάτι διαφορετικό.

### **CustomPrincipal**

Η κλάση CustomPrincipal υλοποιεί το Principal, που αποτελεί το default interface ασφάλειας του Spring και περιέχει το όνομα του χρήστη. Ωστόσο, έχω επεκτείνει τη λειτουργικότητά της, ώστε να περιλαμβάνει και το id του χρήστη για επιπλέον πληροφορίες.

### **TokenDetails**

Η κλάση περιέχει πληροφορίες σχετικά με το token, όπως το userId, το ίδιο το token, την ημερομηνία δημιουργίας του, και την ημερομηνία λήξης του.

### **SecurityService**

Η μέθοδος authenticate δέχεται ως ορίσματα το username και το password. Το password κρυπτογραφείται σε μορφή PBKDF2 με τη βοήθεια του PBKDF2Encoder. Στη συνέχεια, γίνεται έλεγχος στη βάση δεδομένων για την εγκυρότητα των στοιχείων. Εάν είναι σωστά, επιστρέφεται το token, διαφορετικά, αποστέλλεται μήνυμα σφάλματος.

Η μέθοδος generateToken δημιουργεί και υπογράφει ένα JWT χρησιμοποιώντας τα δεδομένα που παρέχονται (claims, subject, expirationDate). Δημιουργεί το token με το Jwts.builder(), το υπογράφει με ένα μυστικό κλειδί και επιστρέφει ένα αντικείμενο TokenDetails που περιλαμβάνει το token, την ημερομηνία έκδοσης και λήξης του. Χρησιμοποιείται για την πιστοποίηση και εξουσιοδότηση χρηστών.

### **JwtHandler**

Η κλάση `VerificationResult` περιέχει δύο πεδία: `claims` και `token`.  
Η μέθοδος `check` καλεί τη μέθοδο `verifyToken` εάν όλα είναι εντάξει, αλλιώς επιστρέφει σφάλμα.

Η μέθοδος `verifyToken` ελέγχει αν ο χρόνος ζωής του `token` έχει λήξει. Η μέθοδος `getClaimsFromToken` εξάγει τα `claims` από το `token`.

## AuthenticationManager

Η κλάση `AuthenticationManager` υλοποιεί το `ReactiveAuthenticationManager`. Η μέθοδος `authenticate` ελέγχει αν υπάρχει ο χρήστης. Θα μπορούσαμε να προσθέσουμε στον χρήστη ένα πεδίο `enabled`, το οποίο να είναι `false` κατά τη διάρκεια της εγγραφής και να το ενεργοποιούμε μόλις ολοκληρωθεί η επιβεβαίωση του email του.

## UserAuthenticationBearer

δημιουργεί μια αυθεντικοποίηση χρήστη από τα δεδομένα ενός JWT, εξάγοντας το ID και το όνομα χρήστη από τα `claims` και επιστρέφοντας ένα `UsernamePasswordAuthenticationToken` για τον χρήστη, χωρίς να απαιτεί άλλες εξουσιοδοτήσεις.

## BearerTokenServerAuthenticationConverter

Η κλάση `BearerTokenServerAuthenticationConverter` υλοποιεί το interface `ServerAuthenticationConverter` και χρησιμοποιείται για την εξαγωγή του JWT από το `Authorization header` ενός HTTP αιτήματος. Χρησιμοποιεί τη συνάρτηση `getBearerValue` για να αφαιρέσει το "Bearer " prefix και να πάρει το JWT token, το οποίο στη συνέχεια επαληθεύεται με τον `jwtHandler`. Αν το token είναι έγκυρο, η μέθοδος `UserAuthenticationBearer.create` δημιουργεί το αντικείμενο `Authentication`. Η διαδικασία είναι ασύγχρονη και βασίζεται σε `Mono` για την επιστροφή των αποτελεσμάτων.

## exception(package)

custom exception, ώστε να μπορώ να αναγνωρίζω εύκολα τα δικά μου σφάλματα όταν συμβαίνουν και να ξέρω ακριβώς που να ψάχνω για να τα εντοπίσω.

## mapper(package)

Όλες οι κλάσεις σε αυτό το πακέτο βασίζονται στη βιβλιοθήκη `MapStruct` για την αυτόματη μετατροπή αντικειμένων. Η χρήση της βιβλιοθήκης απλοποιεί τις μετατροπές και εξασφαλίζει τη συνέπεια, καθιστώντας περιττή οποιαδήποτε περαιτέρω ανάλυση.

## Repository(Γενικά)

Στο Spring, το **Repository** είναι υπεύθυνο για την αλληλεπίδραση με τη βάση δεδομένων, παρέχοντας ενσωματωμένες μεθόδους για βασικές CRUD ενέργειες.  
Στο Spring Boot, τα `Repositories` επιτρέπουν τη διαχείριση της αλληλεπίδρασης με τη βάση δεδομένων μέσω του Spring Data JPA. Όταν δημιουργούμε μια μέθοδο σε ένα `Repository interface`, το Spring αναγνωρίζει το όνομα της μεθόδου και αυτόματα δημιουργεί το αντίστοιχο SQL query. Για παράδειγμα, αν δημιουργήσουμε μια μέθοδο με όνομα

findByName, το Spring Data JPA θα δημιουργήσει ένα query που αναζητά εγγραφές στη βάση δεδομένων με το πεδίο name. Παράδειγμα:

```
public interface ProductRepository extends JpaRepository<Product, Long> {  
    List<Product> findByName(String name);  
}
```

Το παραπάνω παράδειγμα δημιουργεί αυτόματα το εξής SQL query: `sql Copy code SELECT * FROM product WHERE name = ?`; Αυτό σημαίνει ότι δεν χρειάζεται να γράφουμε το SQL query χειροκίνητα, το Spring το διαχειρίζεται για εμάς. Επειδή οι μέθοδοι του Repository βασίζονται σε τυποποιημένα ονόματα, δεν έχει νόημα να εξηγώ κάθε κλάση ξεχωριστά, καθώς το Spring δημιουργεί και εκτελεί τα queries αυτόματα.

### AuthRestController

Έχει 3 endpoints για register, για login και για πληροφορίες για τον λογαριασμό.

Τα διαθέσιμα endpoints είναι τα εξής: POST /register για την εγγραφή μιας νέας ταινίας, λαμβάνοντας τα δεδομένα ταινίας (FilmDTO) και επιστρέφοντας τα καταχωρημένα δεδομένα

### FilmController

Τα διαθέσιμα endpoints είναι τα εξής: /register για εγγραφή ταινίας, /{id} για προβολή ταινίας με το συγκεκριμένο id, /page και /pages για σελιδοποιημένες ταινίες, /search για αναζήτηση ταινιών, / για όλες τις ταινίες και /{id}/actors για τους ηθοποιούς μιας ταινίας.

### ReviewService

Τα διαθέσιμα endpoints είναι: /film/{id} για την επιστροφή κριτικών μιας ταινίας, /add για την προσθήκη κριτικής, /has για έλεγχο αν ο χρήστης έχει ήδη υποβάλει κριτική, και POST / για τη δημιουργία νέας κριτικής.

### WatchListController

Τα διαθέσιμα endpoints είναι: /check/{filmId} για έλεγχο αν μια ταινία υπάρχει στη λίστα παρακολούθησης του χρήστη, /toggle/{filmId} για την προσθήκη ή αφαίρεση μιας ταινίας από τη λίστα παρακολούθησης, και / για την επιστροφή όλων των ταινιών στη λίστα παρακολούθησης του χρήστη.

## FrontEnd

### NextJS

Το Next.js είναι ένα δημοφιλές framework για εφαρμογές React, που παρέχει δυνατότητες όπως server-side rendering (SSR) και static site generation (SSG) για τη βελτίωση της απόδοσης και της SEO φιλικότητας. Επιπλέον, υποστηρίζει εύκολη διαχείριση routing, API routes, και ενσωματωμένη βελτιστοποίηση για σύγχρονες web εφαρμογές.

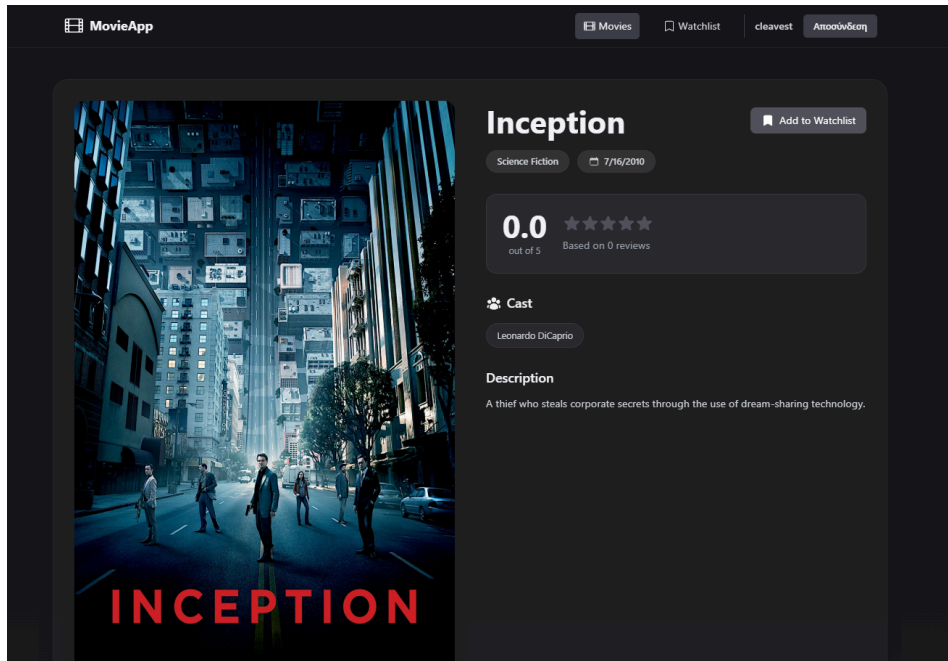
### Routing

Στο Next.js, το routing βασίζεται στη δομή των αρχείων μέσα στον φάκελο app. Κάθε αρχείο αντιστοιχεί σε μια διαδρομή (route), ενώ τα δυναμικά routes ορίζονται με αγκύλες, όπως [id]. Όταν επισκέπτεται κάποιος μια διαδρομή όπως /product/123, το [id]/page.js διαβάζει το id από το URL μέσω του useRouter ή της getStaticProps/getServerSideProps



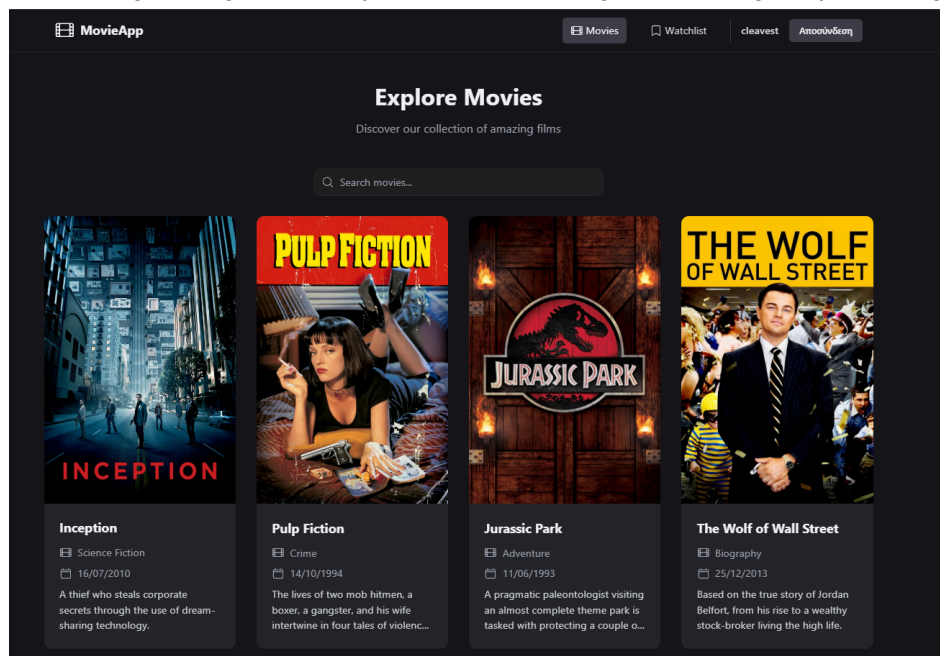
## films/[id]/page

εμφανίζει πληροφορίες για μια συγκεκριμένη ταινία, συμπεριλαμβανομένων των ηθοποιών, κριτικών και βαθμολογιών. Επιτρέπει στους χρήστες να προσθέσουν κριτικές και να προσθέσουν/αφαιρέσουν την ταινία από τη λίστα παρακολούθησης του



## films/page

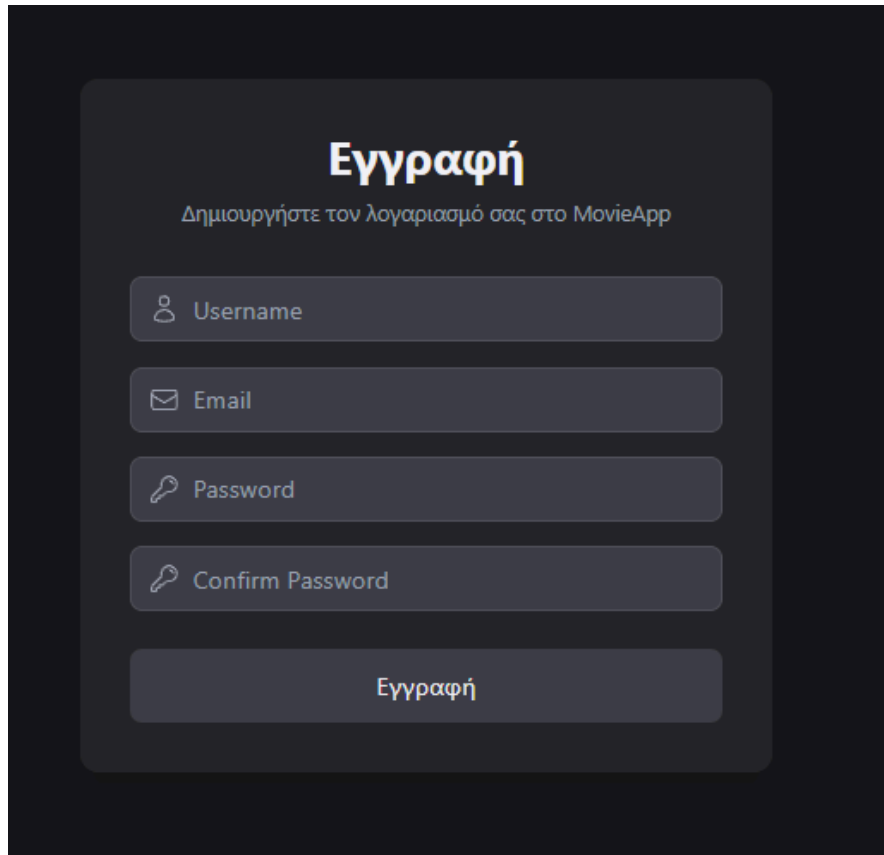
υλοποιεί μια σελίδα εξερεύνησης ταινιών όπου οι χρήστες μπορούν να δουν όλες τις διαθέσιμες ταινίες σε μορφή καρτών με εικόνες και βασικές πληροφορίες.



## register/page

σελίδα εγγραφής χρήστη με φόρμα που περιλαμβάνει πεδία για username, email και κωδικό πρόσβασης. Διαθέτει έλεγχο εγκυρότητας για τους κωδικούς, διαχείριση σφαλμάτων, και

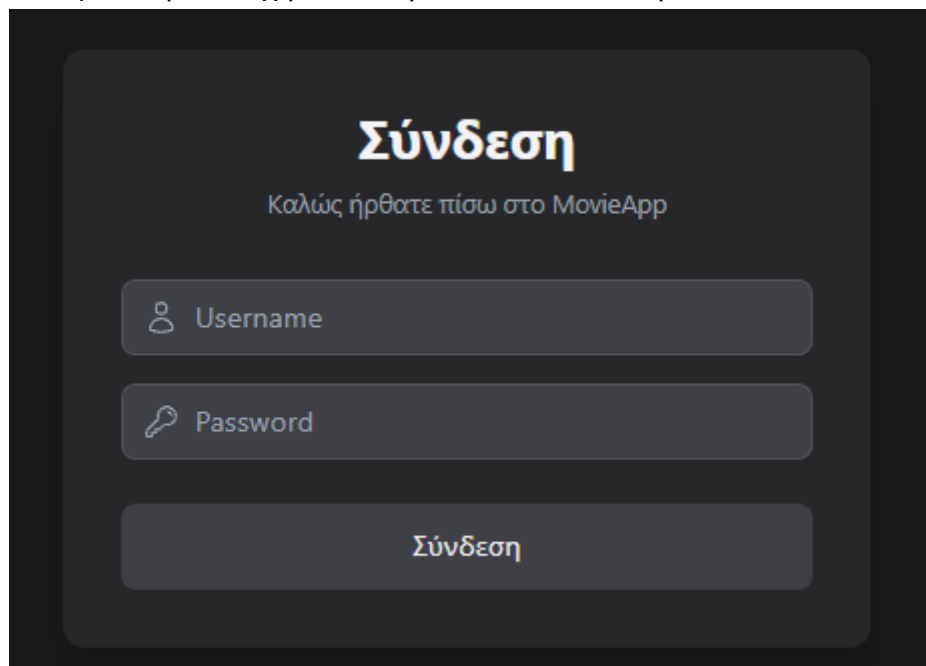
animations για καλύτερη εμπειρία χρήστη, ενώ μετά την επιτυχή εγγραφή ανακατευθύνει στη σελίδα σύνδεσης.



The registration form is titled "Εγγραφή" (Registration) and includes the subtitle "Δημιουργήστε τον λογαριασμό σας στο MovieApp". It features four input fields: "Username" with a person icon, "Email" with an envelope icon, "Password" with a key icon, and "Confirm Password" with a key icon. A "Εγγραφή" (Register) button is positioned at the bottom.

## login/page

σελίδα σύνδεσης χρήστη με φόρμα που περιλαμβάνει πεδία για username και κωδικό πρόσβασης. Χρησιμοποιεί next-auth για την αυθεντικοποίηση, διαχειρίζεται σφάλματα, και μετά την επιτυχή σύνδεση ανακατευθύνει στη σελίδα ταινιών.



The login form is titled "Σύνδεση" (Login) and includes the subtitle "Καλώς ήρθατε πίσω στο MovieApp". It features two input fields: "Username" with a person icon and "Password" with a key icon. A "Σύνδεση" (Login) button is positioned at the bottom.

## Auth.js(<https://authjs.dev/>)

Η επιλογή του Auth.js με Credentials provider ήταν στρατηγική καθώς επιτρέπει τη δημιουργία ενός προσαρμοσμένου συστήματος αυθεντικοποίησης με username/password, διατηρώντας πλήρη έλεγχο της λογικής πιστοποίησης. Παράλληλα, προσφέρει ενσωματωμένη ασφάλεια και διαχείριση sessions, αποφεύγοντας την ανάγκη για χειροκίνητη υλοποίηση πολύπλοκων λειτουργιών όπως JWT tokens και cookie handling, ενώ παραμένει πλήρως συμβατό με το Next.js framework.

## Middleware

Λειτουργεί ως φύλακας (guard) για συγκεκριμένες διαδρομές της εφαρμογής, συγκεκριμένα για τα paths '/films', και '/watchlist' (και όλα τα υπο-paths τους). Όταν ένας χρήστης προσπαθήσει να προσπελάσει οποιαδήποτε από αυτές τις προστατευμένες διαδρομές, το middleware ελέγχει αν είναι συνδεδεμένος - αν δεν είναι, ανακατευθύνεται αυτόματα στη σελίδα '/login'. Αυτό εξασφαλίζει ότι μόνο πιστοποιημένοι χρήστες μπορούν να έχουν πρόσβαση σε αυτές τις σελίδες.

## Docker

### Nextjs

```
FROM node:latest
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .|
CMD ["npm", "run", "dev"]
```

### Spring

```
FROM openjdk:17-jdk-slim-buster
WORKDIR /app
COPY /build/libs/webfluxfilmsdb-0.0.1-SNAPSHOT.jar /app/webflux.jar
EXPOSE 8084
ENTRYPOINT ["java", "-jar", "webflux.jar"]
```

## docker-compose

```
version: '2.32.0'

services:
  db_movie:
    image: postgres:14.15-alpine3.21
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: postgres
    container_name: db_movie
    ports:
      - 5433:5432
    volumes:
      - ./database.sql:/docker-entrypoint-initdb.d/database.sql
  webflux:
    image: webfluxfilmsdb
    container_name: webfluxfilms
    depends_on:
      - db_movie
    ports:
      - 8084:8084
  movie-front:
    image: movie-front
    container_name: front
    depends_on:
      - webflux
    ports:
      - 3000:3000
    environment:
      - NEXT_PUBLIC_API_URL=http://localhost:8084
      - NEXT_INTERNAL_API_URL=http://webflux:8084
      - NEXTAUTH_URL=http://localhost:3000
      - NEXTAUTH_SECRET=b5f59337a612a2a7dc07328f3e7d1a04722967c7f06df20a499a7d3f91ff2a7e
```

## cmds

```
docker build -t webfluxfilmsdb .
docker build -t movie-front .
docker-compose up
```