Project Report for PLH513

Microservices Assignment

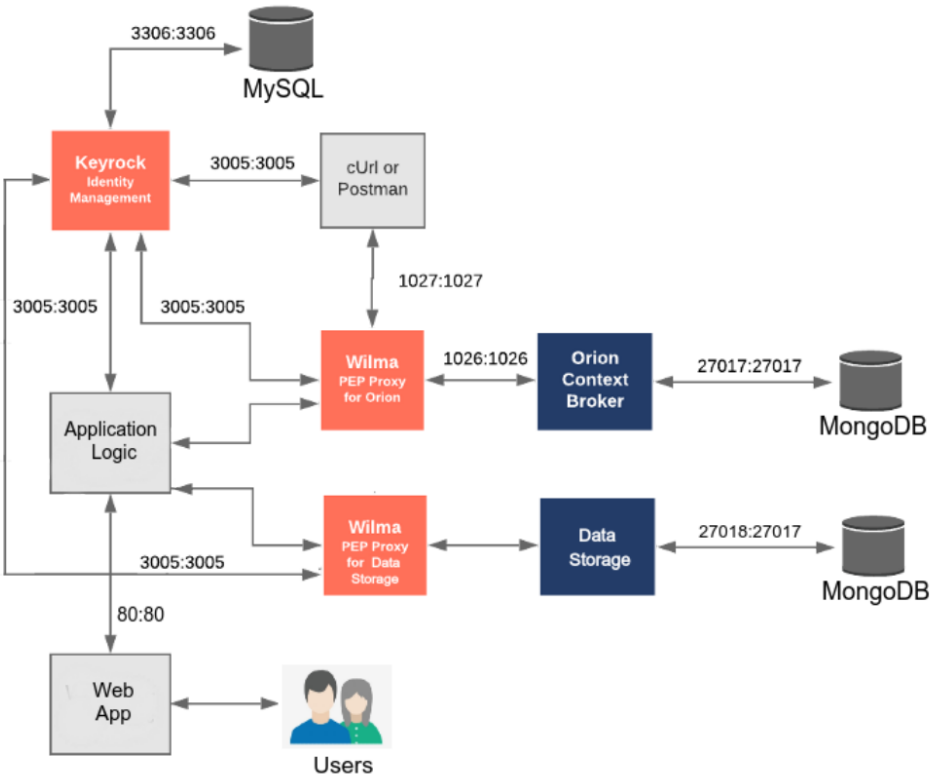
Thanos Delatolas
2016030074

Services in Cloud and Fog

December 5, 2020

Abstract

Κληθήκαμε να υλοποιήσουμε το παρακάτω distributed system.



Το Web-App είναι ένας ξεχωριστός container απο το application logic και στέλνει requests είτε μέσω AJAX είτε μέσω curl στο application logic το οποίο ξέρει και που να τα προωθήσει. Για να γίνεται σωστά η επικοινωνία μεταξύ Web-App και Application logic όλα τα requests περιέχουν ένα flag που δείχνει το λόγο που στέλνεται το request, π.χ. για register το register flag θα έχει τιμή true.

Το σύστημα μπορεί να υλοποιήσει όλες τις ζητούμενες λειτουργίες για τους users και για τους cinema owners. Οι λειτουργίες του admin γίνονται αποκλειστικά απο τον keyrock.

Compilation

Δεν χρειάζεστε τα volumes, οι databases γίνονται restore μέσω του compose. Το σύστημα τρέχει όλο με τα δεδομένα με την εντολή sudo docker-compose up -d.

Ένας μικρός χάρτης του κώδικα

- Το Web-App στέλνει requests μέσω του αρχείου make_request.php και μέσω ajax απο τα javascript files πάντα στο app-logic.
- Το application logic δέχεται requests απο το Web-App στα αρχεία login_req.php και app_logic_req.php.
- Στο αρχείο receive_req.php το Data storage δέχεται requests απο το application logic.

Μπορείτε να βρείτε το request που θέλετε να διορθώσετε παίρνοντας το flag του request από το Web-App (π.χ. Για εισαγωγή στα favorites το flag είναι add_fav) και κάνοντας ctrl+F στα αρχεία login_req.php, app_logic_req.php, receive_req.php.

Στο τέλος της αναφοράς παρουσιάζονται όλα τα flags που χρησημοποιούνται.

1. Keyrock, Login, Register

Έχει υλοποιηθεί και login και register page. Ωστόσο για να συνδεθεί ο admin θα πρέπει να μπεί στο login του keyrock έιναι στη διευθυνση: http://localhost:3005/. Θα ήθελα να υλοποιήσω και admin page έτσι ώστε όταν διαγράφεται ένας χρήστης να διαγράφονται και τα data του στο data storage αλλα και οι subscriptions (άν έχει) από τον orion.

Ο keyrcok βρίσκεται στη διευθυνση: 172.18.1.5

1.1 Login

Το γραφικό περιβάλλον βρίσκεται στη διευθυνση: http://localhost/.

Πριν προχωρήσουμε σε λεπτομέριες είναι σημαντικό πως ο keyrock στέλνει permanent tokens. Για να είναι εφικτό αυτό έθεσα token type: permanent από το γραφικό περιβάλλον του keyrock για την εφαρμογή μου.

Επιπλέον δημιούργησα δυο roles τον CINEMAOWNER και τον USER.

Εφόσον ο χρήστης πατήσει login στέλνονται οι πληροφορίες της φόρμας (email, password) στο **application logic** μέσω του request:

GET /login_req.php?login=login&email=<email>&password=<password>

Στη συνέχεια το application logic και συγκεκριμένα το αρχείο **login_req.php** στέλνει στον keyrock 2 curl requests τα οποία παρουσιάζονται με την αντίστοιχη σειρά:

POST http://172.18.1.5:3005/oauth2/token
Authorization: Basic base64_encode(<client_id> :<client_secret>)
Content-Type: application/x-www-form-urlencoded

grant_type: password
username : $_GET[email]
password : $_GET[password]
scope : permanent

Από το response παίρνουμε access_token και το refresh_token. Σε αυτό το σημείο θα μπορούσε να γίνεται έλεγχος για να δούμε αν το input του χρήστη έχει κάποιο λάθος. Αντίθετα στέλνεται κι άλλο request στον keyrock για να πάρουμε τις πληροφορίες του χρήστη. Σε περίπτωση που ο ρόλος δεν είναι ούτε CINEMAOWNER ούτε USER τότε θα εμφανιστεί μήνυμα λάθους στον χρήστη.

Το request με το access_token που μόλις λάβαμε είναι:

GET http://172.18.1.5:3005/user?access_token=<access_token>

Η απάντηση περιέχει:

- Το id του χρήστη, το οποίο αποθηκεύεται και στη mongo. Θα αναλυθεί στη παρουσίαση του data storage.
- access_token.
- refresh_token.
- email, username, role.

Επιστέφεται στο Web-App όπου αν τα δεδομένα που πήρε είναι null εμφανίζεται μύνημα λάθους στη σελίδα αλλίως ο χρήστης γίνεται redirect στο welcome.php.

1.2 Register

Το γραφικό περιβάλλον βρίσκεται στη διευθυνση: http://localhost/register.php

Εφόσον ο χρήστης πατήσει Register στέλνονται οι πληροφορίες της φόρμας (username, email, password, role) στο **application logic** μέσω του request:

GET /login_req.php?
register=true
username=<username>
email=<email>
password=<password>
role=<role>

Το πρώτο πράγμα που κάνει το application logic είναι να στέλνει request στον keyrock για να πάρουμε ένα X-Auth-Token με τα credentials του admin.

POST http://172.18.1.5:3005/v1/auth/tokens
Content-Type: application/json
name=tdel@test.com
password=1234

Απο το response header παίρνουμε το X-Auth-Token.
Στη συνέχεια στέλνουμε request στον keyrock για να δημιουργήσει ένα account με τα credentials που έδωσε ο χρήστης.

POST http://172.18.1.5:3005/v1/users
Content-Type: application/json
X-Auth-token: <X-Auth-token>

$$\textbf{user} = \left\{ \begin{array}{l} \textbf{username: < username >} \\ \textbf{email: < email >} \\ \textbf{password: < password >} \end{array} \right\}$$

Σε περίπτωση που στην απάντηση η μεταβλητή user δεν είναι null ο user έχει δημιουργηθεί. Συνεπώς στέλνουμε ένα τελευταιο request στον keyrock για να δώσουμε role στον χρήστη. Ανάλογα με τον ρόλο που έβαλε στη φόρμα στέλνεται το request: (το client_id είναι ο κώδικος που δίνει ο keyrock στην εφαρμογή)

POST
http://172.18.1.5:3005/v1/applications/<client_id>/user/<user_id>/roles/<role_id>
Content-Type: application/json
X-Auth-token: <X-Auth-token>

Στη συνέχεια στέλνεται θετίκη απάντηση στο Web-App και ο χρήστης γίνεται redirect στη login page.

Στη περίπτωση που ο χρήστης δεν δημιουργηθεί, ελέγχεται αν στην απάντηση του keyrock στο error->message είναι: Email already used, ο χρήστης ενημερώνεται για να να βάλει άλλο email.

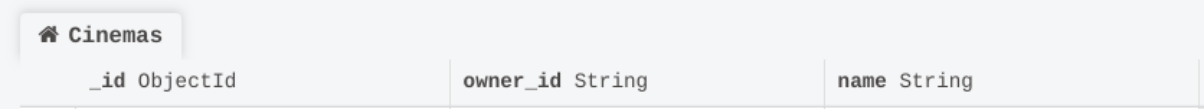
1.3 Τι λείπει;

- Admin page
- Αν ο χρήστης έχει ήδη account στον keyrock αλλά δεν είναι εγγεγραμμένος στην εφαρμογή, δεν θα δουλέψει σωστά η σελίδα. Θα έπρεπε να κάνω άλλη μια σελίδα για τη περίπτωση αυτή. Για τώρα η λύση είναι να μπει ο admin και να τον κάνει authorize.

2. Data storage

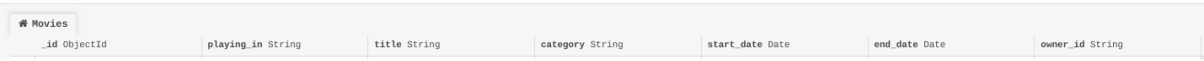
Πριν προχωρήσουμε σε λεπτομέρειες είναι σηματνικό να παρουσιαστούν τα collections της mongo.

Cinemas



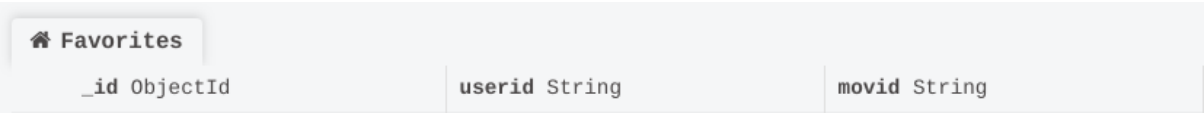
- Το owner_id είναι το id που δίνει ο keyrock στους χρήστες.
- Κάθε owner μπορεί να έχει πολλά cinema

Movies



- Το playing_in είναι το _id του Cinema που παίζεται η ταινία.
- Το owner_id είναι το id που δίνει ο keyrock στους χρήστες, και είναι το id του χρήστη που έχει το cinema που παίζεται η ταινία.

Favorites



- Το userid είναι το id που δίνει ο keyrock στους χρήστες.
- Το movid είναι το _id του collection Movies.

Τα collections: Feed, Subscriptions θα παρουσιαστούν στο κεφάλαιο του orion.

Το Data storage λαμβάνει requests στο αρχείο receive_req.php

Το σενάριο παραμένει ίδιο, το **Web-App** στέλνει AJAX και curl requests στο **application logic** στα οποία προστίθεται το magic_key (θα αναλύθει περεταίρω στο κεφάλαιο για τη wilma) και προωθούνται στο data storage. Το **Web-App** χρείαζεται δεδομένα για τους χρήστες για όλες τις σελίδες εκτός απο login και register.

Ξεκινάμε την ανάλυση μας από τους USERS. Σε περίπτωση που συνδεθεί κάποιος USER γίνεται redirect στον **welcome.php** όπου έχει το feed του. Το feed έχει άμεση σχέση με τον orion και για αυτό το λόγο θα αναλυθεί στο κεφάλαιο του orion. Ο USER έχει ακόμη τη σελίδα movies.php, η οποία έχει όλες τις ταινίες όλων των cinema και δίπλα απο κάθε ταινία υπάρχει ενα button σε σχήμα καρδιάς που στέλνει ajax request για εισαγωγή στα favorites ή διαγραφή απο τα favorites αν είναι κόκκινο.

Όταν φορτώνει η σελίδα στέλνονται 2 requests: το get_movies και το get_favorites απο το Web-App στο Application logic τα οποία προωθούνται στο Data storage. Το Data storage επικοινωνεί με την mongo και επιστρέφει τις ταινίες και τα favorites του χρήστη αντίστοιχα.

Μέσω AJAX το αρχείο movies_jquery.js στέλνει request στο Application logic για εισαγωγή και διαγραφή από τα favorites. Το Application logic πέρα από να προωθεί το request στο data storage στέλνει και στον orion.

Οι CINEMAOWNERS έχουν τρεις σελίδες τις welcome.php, owner.php, owner_add_rem.php. Οι σελίδες αυτές στέλνουν requests στο data storage μέσω του app logic. Για την διαγραφή, αλλαγή και εισαγωγή στοιχείων (movies, cinema) γίνονται αρκετοί έλεγχοι απο το data storage με άσχημο τρόπο (no joins but instead for-loops :(Sorry! :)).Θεωρώ πως δεν έχει νόημα να παρουσιαστουν, αυτό που έχει νόημα είναι να προχωρήσουμε στον orion!

Τι λείπει όμως;

- Διαγραφή Cinema (υπάρχει διαγραφή Movie)

3. Orion

Ως entity έχει θεωρηθεί κάθε ταινία. Το _id του entity δημιουργείται με βάση το movie_id ως εξής:

- Type: Movie
- id: movie_id

Συνεπώς όταν διαγράφουμε μια ταινία μπορούμε με το id της να να διαγράψουμε και το αντίστοιχο entity.

Τα attributes του entity είναι:

- soon (boolean : θα αναλυθεί η σημασία του.)
- stop_playing (boolean : θα αναλυθεί η σημασία του.)
- cin_name
- start_date
- end_date
- title

Όλα τα requests προς τον orion βρίσκονται στο αχρείο orion_requests.php του application logic.

Κατά τη δημιουργεία ταινίας και εφόσον το Data storage επιστρέψει θετική απάντηση για τη δημιουργεία της ταινίας, το application logic στέλνει request στον orion για τη δημιουργεία entity.

POST http://orion_proxy:1027/v2/entities/
Content-Type: application/json
X-Auth-Token: magic_key
id=mov_id
Type=Movie
soon=< soon >
stop_playing=< stop_playing >
title=< title >
cin_name=< cin_name >
category=< category >
start_date=< start_date >
end_date=< end_date >

Ο CINEMAOWNER μπορεί ακόμη να διαγράψει ή και να κάνει update μια ταινία. Το αντίστοιχο πρέπει να γίνει και στον orion αλλά είναι αρκετά ευκολο να γίνει εφόσον το id της entity είναι το id της Movie.

- Εφόσον γίνει update μια ταινία απο το Data storage το Application logic στέλνει το ακόλουθο request στον orion:

PATCH
http://orion_proxy:1027/v2/entities/mov_id/attrs?options=keyValues
Content-Type: application/json
X-Auth-Token: magic_key
soon=< soon >
stop_playing=< stop_playing >
title=< title >
cin_name=< cin_name >
category=< category >
start_date=< start_date >
end_date=< end_date >

- Εφόσον γίνει delete μια movie απο το ο Data storage το Application logic στέλνει request για unsubscribe (θα αναλυθεί παρακάτω) για κάθε subscription της ταινίας

και στη συνέχεια στέλνεται το request για τη διαγραφή του entity:

DELETE
http://orion_proxy:1027/v2/entities/mov_id/attrs?options=keyValues
X-Auth-Token: magic_key

Τα πεδία soon, stop_playing

Όταν γίνεται εισαγωγή ή update (στα πεδία start_date, end_date) σε μια ταινία και κατ΄ επέκταση σ΄ ένα orion entity υπολογίζονται με βάση την current date και τις start_date, end_date της ταινίας οι τιμές: soon, stop_playing.

- soon = true, όταν η ταινία βγαίνει σε κυκλοφορία σε 30 μέρες απο σήμερα
- stop_playing = true, όταν οι ταινία δεν παίζεται πια.

Subscriptions, notifications

Όταν ένα χρήστης κάνει εισαγωγή στα favorites στέλνεται request στον orion για να δημιουργήσει ένα subscription στη ταινία που έβαλε στα favorites. Σε αυτό το σημείο, έκανα κάτι που μάλλον δεν είναι πολύ ασφαλές αλλά ήταν ο μόνος τρόπος που σκέφτηκα για να μπορώ να χρησημοποιώ το notification του orion. Στο description του notification έβαλα το id του user που απευθύνεται το subscription. Πιο συγκεκριμένα, το request που στέλνει το Application logic εφόσον το Data storage βάλει στα favorites την ταινία είναι (βρίσκεται στο αρχείο orion_requests στη συνάρτηση subscribe):

POST http://orion_proxy:1027/v2/subscriptions
Content-Type: application/json
X-Auth-Token: magic_key
description=user_id

$$\textbf{subject} = \left\{ \begin{array}{l} \textbf{entities=} \left\{ \begin{array}{l} \textbf{id=} \ < \textbf{movie\_id} > \\ \textbf{Type=Movie} \end{array} \right\} \\ \textbf{condition=} \left\{ \begin{array}{l} \textbf{soon} \\ \textbf{stop\_playing} \end{array} \right\} \end{array} \right\}$$

$$\textbf{notification} = \left\{ \begin{array}{l} \textbf{url= http://172.18.1.8/orion\_notification.php} \\ \textbf{attrs} = \left\{ \begin{array}{l} \textbf{start\_date} \\ \textbf{end\_date} \\ \textbf{title} \\ \textbf{cin\_name} \\ \textbf{soon} \\ \textbf{stop\_playing} \end{array} \right\} \end{array} \right\}$$

Συνεπώς, λαμβάνουμε ειδοποίηση μόνο άμα αλλάξει κάποια από τα πεδία soon, stop_playing.

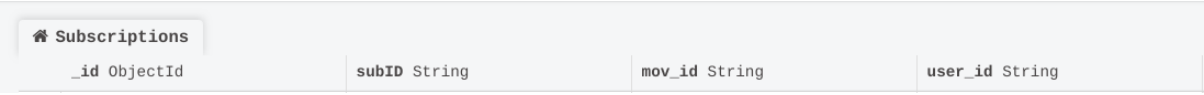
Τι γίνεται στο αρχείο orion_notification.php

Το orion_notification.php δέχεται notifications απο τον orion όταν αλλάζουν τα πεδία soon και stop_playing. Συγκεκριμένα όταν έρχεται ένα notification παίρνουμε το subsciption id και τα δεδομένα, τα οποία είναι τα attrs του notification. Για να μάθουμε και για ποιό user είναι στέλνουμε στον orion το request:

GET http://orion_proxy:1027/v2/subscriptions/< subscriptionId >
X-Auth-Token: magic_key

Έφοσον λάβουμε την έχουμε όλα τα δεδομένα και στέλνεται request στο Data storage να τα αποθηκεύσει.

Για κάθε χρήστη αποθηκεύονται τα Subscriptions και το Feed του.



Και το feed:

```
_id: ObjectId("5fc8d9948caaa77356151c44")
subID: "5fc8d9941d3c2bebd5291210"
mov_id: "5fc8d700e8d89934a1128119"
start_date: "2020-04-26"
end_date: "2020-09-06"
cin_name: "Matadero Cineteca"
soon: "0"
stop_playing: "1"
title: "Braveheart"
user_id: "0b10e960-70d4-4318-b6d0-e9132430ab09"
received: "2020-12-03 02:27:00pm"
read: true
```

Λιγα λόγια για το Feed

Έχει υλοποιηθεί με την συνάρτηση setInterval. Διαβάζει το Collection Feed ολόκληρο τη πρώτη φορά που φορτώνει σελίδα και μετά μέσω AJAX requets διαβάζει μόνο τα documents

του χρήστη μέ τιμή στο πεδίο read false.

Unsubscribe

Όταν διαγράφεται κάποια ταινία στέλνονται request στον orion για κάθε subscription που έχει. Επιπλέον αν κάποιος χρήστης βγάλει κάποια ταινία απο τα Favorites στέλνετε ένα request για διαγραφή του αντίστοιχου Subscription. Το subscription id είναι αποθηκευμένο στο Data storage όποτε το παίρνουμε απο εκεί και στέλνουμε στον orion:

DELETE http://orion_proxy:1027/v2/subscriptions/< subscriptionId >
X-Auth-Token: magic_key

4. Wilma

Και τα δυο pep-proxy έχουν το ίδιο PEP_PROXY_MAGIC_KEY που είναι magic_key. Η wilma του orion ακούει στο port 1027 ενώ η wilma του Data sotage στο port 1028.

5. GCP

Το link είναι: http://34.65.233.169/. Παρακαλώ στείλτε μου email να ανοίξω τους container στο adelatolas@isc.tuc.gr

6. Logins

- user1@test.com
- user2@test.com
- user3@test.com
- owner1@test.com
- owner2@test.com
- owner3@test.com

Όλοι έχουν κωδικό 1234. Φυσικά μπορείτε να φτιάξετε και δικού σας χρήστες. Για να δείτε τα notifications μπορείτε να βάλετε το Braveheart του owner3 σε ημερομηνία που να απέχει λιγότερο απο 29 μέρες απο την σημερινή. Θα δείτε το notification στον user1.

7. Requests

Some facts...

- Τα request για το login και register βρίσκοτναι στο πρώτο κεφάλαιο γιατί ήθελα να τα αναλύσω.
- Δεν περιέχονται ούτε τα requests προς τον orion γιατί ήθελα επίσης να τα αναλύσω στο αντίστοιχο κεφάλαιο.
- Η ip 172.18.1.8 είναι του Application logic.
- Πολλά απο τα παρακάτω προωθούνται στο Data Storage.

API time!

Για εύκολο έλεγχο με τον κώδικα: Μπορείτε να πάρετε απο το εκάστοτε request το flag που είναι true και να πάτε με ctrl+f στο αρχείο που στέλνετε.

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_movies:

GET http://172.18.1.8/app_logic_req.php
get_movies=true

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_favorites:

GET http://172.18.1.8/app_logic_req.php
get_favorites=true
user_id = <user_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag add_fav:

POST http://172.18.1.8/app_logic_req.php
add_fav=true
user_id = <user_id>
mov_id = <mov_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag remove_fav:

POST http://172.18.1.8/app_logic_req.php
remove_fav=true
user_id = <user_id>
mov_id = <mov_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_owner_data:

GET http://172.18.1.8/app_logic_req.php
get_owner_data=true
user_id = <user_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_cinemas:

GET http://172.18.1.8/app_logic_req.php
get_cinemas=true
owner_id = <user_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_last_movie:

GET http://172.18.1.8/app_logic_req.php
get_last_movie=true
owner_id = <user_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag get_feed:

GET http://172.18.1.8/app_logic_req.php
get_feed=true
user_id = <user_id>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag modify_movie:

POST http://172.18.1.8/app_logic_req.php
modify_movie=true
mov_id = <mov_id>
playing_in = <playing_in>
title = <title>
category = <category>
start_date = <start_date>
end_date = <end_date>

- Στέλνεται απο το Web-App στο Application logic το οποίο το προωθεί στο αρχείο receive_req.php του Data storage με flag del_movie:

POST http://172.18.1.8/app_logic_req.php
del_movie=true
mov_id = <mov_id>