

ΥΠΗΡΕΣΙΕΣ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ ΚΑΙ ΣΤΗΝ ΟΜΙΧΛΗ

ΑΣΚΗΣΗ - 2 ΑΝΑΦΟΡΑ

ΟΝΟΜΑ: ΠΑΥΛΟΣ
ΕΠΩΝΥΜΟ: ΣΩΦΡΟΝΙΟΥ
ΑΜ: 2015030129

ΕΙΣΑΓΩΓΗ - ΤΕΧΝΟΛΟΓΙΕΣ

Η εφαρμογή που ζητείται για την δεύτερη προγραμματιστική άσκηση έχει υλοποιηθεί στο εικονικό περιβάλλον λειτουργικού συστήματος που προσφέρεται από την πλατφόρμα λογισμικού Docker. Όλες οι υπηρεσίες (services) που χρησιμοποιούνται για την υλοποίηση της άσκησης βρίσκονται μέσα σε software containers τα οποία έχουν προκύψει από την εικονοποίηση των διαφόρων υπηρεσιών. Για να “τρέξει” η παρούσα εργασία απαιτείται από τον χρήστη να έχει εγκατεστημένη την πλατφόρμα Docker, το εργαλείο docker-compose, και να προσθέσει τους φακέλους **project_mongo-db**, **project_mymongoDB_data_container** και **project_mysql-db** στον φάκελο που αποθηκεύει ο Docker τα volumes του (Για Ubuntu είναι /var/lib/docker). Ύστερα απλά κάνει πρέπει να κάνει docker-compose up μέσα στον φάκελο project.

ΕΠΙΣΚΟΠΗΣΗ ΤΩΝ ΥΠΗΡΕΣΙΩΝ - DOCKER

Παρακάτω αναφέρονται οι υπηρεσίες που βρίσκονται μέσα στο “docker-compose.yml” αρχείο και συνθέτουν την αρχιτεκτονική της εφαρμογής.

- **php_logic** : Το στιγμιότυπο που δημιουργείται από μία εικόνα ενός Apache server, αποτελεί το application logic της εφαρμογής, και υπάρχει το frontend (HTML, CSS) μαζί με το backend μέρος (Javascript, PHP).
- **nefosservice** : Το στιγμιότυπο που δημιουργείται από μία εικόνα ενός Apache server μαζί με το extension που απαιτείται για χρήση της βιβλιοθήκης MongoDB για την γλώσσα PHP. Αποτελεί το REST API της εφαρμογής το οποίο επικοινωνεί μαζί με μία βάση Mongo για την δημιουργία/ανάγνωση/ενημέρωση/διαγραφή (CRUD) πληροφοριών της εφαρμογής.
- **mongoDB** : Το στιγμιότυπο που δημιουργείται από μία εικόνα βάσης δεδομένων MongoDB και κρατάει τις πληροφορίες της εφαρμογής, επικοινωνεί με το REST API (nefosservice).
- **nefosservice-proxy** : Το στιγμιότυπο που δημιουργείται από μια εικόνα WILMA fiware/pep-proxy και αποτελεί την proxy υπηρεσία που προστατεύει και ανακατευθύνει τα HTTP requests από το application logic (php_logic) στο REST API της εφαρμογής (nefosservice).
- **orion** : Το στιγμιότυπο που δημιουργείται από μία εικόνα fiware/orion και αποτελεί την υπηρεσία context broker η οποία καθιστά εφικτή την PUB/SUB επέκταση της εφαρμογής για ενημέρωση των ταινιών. Επικοινωνεί μαζί με μία βάση MongoDB στην οποία αποθηκεύονται τα entities, τα subscriptions και οι πληροφορίες που απαιτούνται για τον orion.
- **mongo-db** : Το στιγμιότυπο που δημιουργείται από μία εικόνα βάσης δεδομένων MongoDB και κρατάει τις πληροφορίες του Orion Context Broker (orion) με τον οποίο επικοινωνεί.

- **orion-proxy** : Το στιγμιότυπο που δημιουργείται από μια εικόνα WILMA fiware/per-proxy και αποτελεί την proxy υπηρεσία που προστατεύει και ανακατευθύνει τα HTTP requests από το application logic (php_logic) στον Orion Context Broker (orion).
- **keyrock** : Το στιγμιότυπο που δημιουργείται από μία εικόνα fiware/idm και αποτελεί την υπηρεσία Identity Manager που διαχειρίζεται και αυθεντικοποιεί τους χρήστες της εφαρμογής. Επικοινωνεί μαζί με μία βάση MySQL στην οποία αποθηκεύονται όλες οι απαραίτητες πληροφορίες της υπηρεσίας μαζί με τα δεδομένα των χρηστών της εφαρμογής.
- **mysql-db** : Το στιγμιότυπο που δημιουργείται από μία εικόνα βάσης δεδομένων MySQL και κρατάει τις πληροφορίες της υπηρεσίας Keyrock-IDM (keyrock) με την οποία επικοινωνεί.

ΕΠΙΠΛΕΟΝ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ DOCKER

- Για την επίτευξη της επικοινωνίας των υπηρεσιών μεταξύ τους (στα σημεία όπου αυτή είναι απαραίτητη) έχει δημιουργηθεί το network - **net1** (με default τιμές και network range subnet τις διευθύνσεις 172.18.1.0/24) στο οποίο εντάσσονται όλες οι υπηρεσίες. Κάθε υπηρεσία κάνει expose τα κατάλληλα ports και επίσης έχουν δοθεί τα ports σε environmental variables (π.χ για proxies). Στην περίπτωση των βάσεων δεδομένων τα ports των containers συνδέονται μαζί με τα ports του host μηχανήματος όπως φαίνεται στο ενδεικτικό διάγραμμα της εκφώνησης.
- Για τις τρεις βάσεις δεδομένων της εφαρμογής (2 MongoDB και 1 MySQL) χρησιμοποιούνται volumes για την αποθήκευση των δεδομένων τους στο host μηχάνημα που τρέχει το Docker για να μην διαγράφονται τα δεδομένα κατά την διαγραφή των containers (docker-compose down).

Υπάρχει και η υπηρεσία **mongo-express** η οποία δεν ήταν ζητούμενο της άσκησης αλλά έχει προστεθεί για σκοπούς εύκολου debugging και διόρθωσης.

AJAX - ΔΥΝΑΜΙΚΗ ΑΝΑΝΕΩΣΗ ΜΕ ΑΣΥΓΧΡΟΝΗ ΕΠΙΚΟΙΝΩΝΙΑ

Για την δυναμική ανανέωση περιεχομένου στην εφαρμογή, χρησιμοποιείται η βιβλιοθήκη της Javascript JQuery η οποία παίρνει τα δεδομένα δυναμικά από τα HTML στοιχεία των ιστοσελίδων (φόρμες - tables κτλ) και με τις συναρτήσεις .ajax που παρέχονται ανταλλάσσονται τα δεδομένα ασύγχρονα σε scripts κώδικα PHP τα οποία εκτελούν HTTP requests με χρήση της βιβλιοθήκης CURL. Για την επικοινωνία των services, στο URL της curl αναγράφεται το όνομα της υπηρεσίας όπως αυτό έχει οριστεί στο docker-compose.yml αρχείο και ακολουθείται από το port στο οποίο “ακούει” και κάνει expose η υπηρεσία, ακολουθούμενο από το ανάλογο endpoint στο οποίο θέλει να επικοινωνήσει.

Τα php scripts που καλούνται δυναμικά μέσω του AJAX και επικοινωνούν με CURL requests στο DataStorage (nefoservice) ακολουθούν την σύμβαση ονόματος “ajax_{FunctionName}.php”. Τα scripts που επικοινωνούν με το REST API του Orion ακολουθούν την σύμβαση ονόματος “keyrock_{FunctionName}.php”, και τέλος αυτά που επικοινωνούν με το REST API του Orion ακολουθούν την σύμβαση ονόματος “orion_{FunctionName}.php”.

DATA STORAGE - REST API

Για τα δύο κύρια Collection της εφαρμογής, έχει οριστεί ένα μοντέλο-κλάση (class) στην οποία ανήκουν όλες οι CRUD συναρτήσεις που εκτελούν queries για το συγκεκριμένο collection στην βάση **mongodb**. Οι κλάσεις αυτές ανήκουν μέσα στον φάκελο models. Το script για την σύνδεση της βάσης μαζί με τα αρχεία του composer βρίσκονται στον φάκελο config. Τα endpoints του REST API φτιάχνουν στιγμιότυπα των κλάσεων ανάλογα με το Collection στο οποίο θέλουν να διαβάσουν/γράψουν. Η χρήση των μοντέλων χρησιμοποιείται ως μία πιο δομημένη κατασκευή του ζητούμενου REST API της άσκησης. Το collection subs δεν έχει μοντέλο καθώς οι πληροφορίες του εξαρτώνται από τον orion.

Τα Collections της βάσης που αναπαρίστανται με μοντέλα στην άσκηση είναι τα εξής:

- **movies**
- **favorites**
- **subs** (εδώ αποθηκεύονται οι πληροφορίες από τα subscriptions alerts του Orion)

Το collection cinemas θεωρείται ότι εισάγεται από τον Super Admin μία φορά κατά της αρχικοποίηση της εφαρμογής οπότε δεν αλλάζουν οι πληροφορίες του (Δεν έχει CRUD rest Api).

- **cinemas** (Το collection θεωρείται στατικό και περιέχει τα cinema με τους αντίστοιχους owners)

DATA STORAGE REST API REFERENCE MANUAL:

Για τις ταινίες της εφαρμογής:

GET /api/movie/get_single.php?title=**TITLE**

GET /api/movie/get_all.php

GET /api/movie/get_all_owner.php

GET /api/movie/search.php?search=**SEARCH**¶m=**PARAM**&datestart=**DATESTART**&dateend=**DATEEND**

POST /api/movie/create.php με POST body:

```
{
    "title": "TITLE",
    "category": "CATEGORY",
    "cinemaname": "CINEMANAME",
    "startdate": "STARTDATE",
    "enddate": "ENDDATE"
}
```

DELETE /api/movie/delete.php?title=**TITLE**&cinemaname=**CINEMANAME**

PUT /api/movie/update.php

```
{
    "oldtitle": "OLDTITLE"
    "title": "TITLE",
    "category": "CATEGORY",
    "cinemaname": "CINEMANAME",
    "startdate": "STARTDATE",
    "enddate": "ENDDATE"
}
```

Για τα favorites της εφαρμογής:

GET /api/favorite/get_single.php?title=**TITLE**&cinemaname=**CINEMANAME**&username=**USERNAME**

GET /api/favorite/get_all.php?username=**USERNAME**

POST /api/favorite/create.php με POST body:

```
{
    "username": "USERNAME",
    "title": "TITLE",
    "category": "CATEGORY",
    "cinemaname": "CINEMANAME",
    "startdate": "STARTDATE",
    "enddate": "ENDDATE"
}
```

DELETE /api/favorite/delete.php?title=**TITLE**&cinemaname=**CINEMANAME**&username=**USERNAME**

Για τα subs της εφαρμογής:

GET /api/orion/get_subAlert.php?username=**USERNAME**

POST /api/orion/get_subs.php με POST body:

```
{
    "category": "CATEGORY",
    "cinemaname": "CINEMANAME",
}
```

ORION CONTEXT BROKER

Με τον Orion Context Broker ο χρήστης μπορεί να λαμβάνει ειδοποιήσεις για τις ταινίες που έχει προσθέσει στα favorites του. Για παράδειγμα αν κάποιος χρήστης προσθέσει μια ταινία στα αγαπημένες τους και έπειτα ο αντίστοιχος CinemaOwner της ταινίας αυτής αλλάξει την ημερομηνία προβολής της, τότε ο χρήστης θα λάβει ενημέρωση με την αλλαγή που έγινε στο feed του. Στην υλοποίηση της εφαρμογής, για κάθε favorite που προστίθεται δημιουργείται ένα αντίστοιχο entity και subscription. Όταν ο χρήστης αφαιρέσει την ταινία από τα αγαπημένα του διαγράφεται το entity και subscription που δημιουργήθηκαν για το favorite αυτό.

Το entity που αναπαριστά κάθε favorite και έχει την εξής δομή:

```
{
    "id": "TitleCinemanameUsername",
    "type": "Movie",
    "user": {
        "type": "text",
        "value": "USERNAME"
    },
    "movie": {
        "type": "text",
        "value": "TITLE"
    },
    "startdate": {
        "type": "Date",
        "value": "STARTDATE"
    },
    "enddate": {
        "type": "Date",
        "value": "ENDDATE"
    }
}
```

Το subscription ID αποθηκεύεται στην MongoDB βάση της εφαρμογής στο Collection favorites και όταν γίνει κάποια αλλαγή από τον CinemaOwner της ταινίας θα εισαχθεί ένα document στο Collection Subs με τις αλλαγές. Το feed του κάθε χρήστη για τα μηνύματα από τα subscription βρίσκεται στην welcome.php σελίδα. Το feed στέλνει αίτημα στην βάση δεδομένων για νέες ειδοποιήσεις από τα subscriptions με Javascript ανά 5 δευτερόλεπτα.

KEYROCK IDM

Κάθε νέος χρήστης μπορεί να κάνει signup από το GUI του keyrock ή και από την σελίδα signup.php η οποία στέλνει HTTP αίτημα στο REST API του KEYROCK (δημιουργείται στο backend ένα X-Auth-Token με τα στοιχεία του admin και προστίθεται στα headers του αιτήματος). Έπειτα για να έχει πρόσβαση στην εφαρμογή θα πρέπει ο admin να τον κάνει enable και Authorize αναθέτοντάς του ρόλο (User ή CinemaOwner) στο application από το GUI του Keyrock. Κατά το Login του χρήστη από την index.php σελίδα, εφόσον τα στοιχεία είναι σωστά δημιουργείται ένα OAuth2 token από τα στοιχεία που κατέθεσε και προστίθεται στις SESSION μεταβλητές για να μπορεί να έχει πρόσβαση σε όλες τις σελίδες της εφαρμογής. Επιπρόσθετα στις SESSION μεταβλητές ανατίθενται τιμές που και για τα υπόλοιπα στοιχεία του χρήστη (όπως το username του ή τον ρόλο του) που λαμβάνονται από με αιτήματα στο REST API του keyrock τα οποία είναι απαραίτητα για τις άλλες υπηρεσίες που προσφέρει η εφαρμογή.

FIWARE WILMA PEP PROXY

Στην εφαρμογή υπάρχουν 2 WILMA Proxy υπηρεσίες. Η μία προστατεύει την υπηρεσία Orion και η άλλη το DataStorage. Η εισαγωγή των proxies επιτεύχθηκε δίνοντας τις κατάλληλες παραμέτρους στις environmental μεταβλητές στον ορισμό των υπηρεσιών στο yml (docker-compose) αρχείο ώστε να ανακατευθύνουν στις εφαρμογές που προστατεύουν. Επιπρόσθετα χρειάστηκε να δοθεί και το APP_ID και APP_PASSWORD που χαρακτηρίζουν την εφαρμογή στον keyrock. Τέλος στο Application Logic της εφαρμογής τα Curl αιτήματα κατευθύνονται στις proxy διευθύνσεις που προστατεύουν τις υπηρεσίες και όχι στις υπηρεσίες καθ' εαυτές. Στα headers των Curl αιτημάτων προστίθεται το OAuth2 token που υπάρχει σε SESSION μεταβλητή κατά την είσοδο του χρήστη.

ΣΧΟΛΙΑ - ΠΑΡΑΤΗΡΗΣΕΙΣ

-Στον φάκελο webdev βρίσκονται όλα τα αρχεία του Application Logic (php_logic). Στον φάκελο data_service_api βρίσκονται όλα τα αρχεία του DataStorage (nefoservice).

-Το migration στο Google Cloud δεν πραγματοποιήθηκε.





-Μέσα στο project, στο directory του REST API υπάρχει το μοντέλο User και τα endpoints για users. Δεν χρησιμοποιείται καθόλου πίνακας Users μέσα στην MongoDB απλά μέσα στο μοντέλο User υπάρχει η συνάρτηση που επιστρέφει το όνομα του cinema που ανήκει σε ένα Username (ενός CinemaOwner). Στην συνάρτηση το query γίνεται στο collection cinemas και όχι σε κάποιο collection user όπως φαίνεται να υπονοεί το όνομα του endpoint. Δεν το άλλαξα λόγω χρόνου.

-Ο orion δουλεύει υπό την προϋπόθεση του ότι ο CinemaOwner δεν θα τροποποιήσει το όνομα της ταινίας καθώς το όνομα της ταινίας χρησιμοποιείται στο id πεδίο του Entity.

-Οι ειδοποιήσεις από τα subscriptions του orion δεν έχουν ημερομηνία διαγραφής, διαγράφονται μόνο όταν διαγραφεί η ταινία από τα favorites.

-Κάθε CinemaOwner έχει μόνο ένα Cinema.

-Ένας CinemaOwner για να τροποποιήσει την ταινία του, πρέπει να πατήσει το edit button (το μολύβι που είναι highlighted). Αφού το πατήσει τα πεδία της ταινίας θα γίνουν editable οπότε θα μπορεί να αλλάξει τις πληροφορίες όπως αυτός επιθυμεί. Για την οριστικοποίηση των αλλαγών πατάει πάλι στο ίδιο κουμπί το οποίο έχει αλλάξει και έχει γίνει ένα tik.

300	300
REX	REX
Action	Action
2020-12-16	2020-12-16
2020-12-30	2020-12-30
 	 

Λογαριασμοί που υπάρχουν ήδη για δοκιμή της εφαρμογής:

Admin:

email: admin@test.com

password: 1234

CinemaOwner:

email: alex@mail.com

password: 123

email: maria@mail.com

password: 123

User:

email: lewis@mail.com

password: 123

email: carlos@mail.com (δεν έχει γίνει enabled όμως)

password: 123