ΥΠΗΡΕΣΙΕΣ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ
ΝΕΦΟΣ ΚΑΙ ΤΗΝ ΟΜΙΧΛΗ (ΠΛΗ 513)
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2022-2023

2ο Μέρος Προγραμματιστικής Άσκησης

ΕΠΕΚΤΑΣΗ WEB ΕΦΑΡΜΟΓΗΣ ΜΕ ΧΡΗΣΗ MICROSERVICES ΣΕ ΠΕΡΙΒΑΛΛΟΝ DOCKER ΚΑΙ ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΟ ΠΕΡΙΒΑΛΛΟΝ GCP

Επιμέλεια Εργασίας:
Τζαβάρας Αιμίλιος (email: atzavaras@tuc.gr), Λαζίδης Απόστολος (email: alazidis@tuc.gr)

Καθηγητής:
Πετράκης Ευριπίδης

ΠΕΡΙΓΡΑΦΗ ΑΣΚΗΣΗΣ

- Σκοπός της άσκησης είναι η εξοικείωση με ένα πραγματικό περιβάλλον υπολογιστικού νέφους (Google Cloud Platform) και η ανάπτυξη εφαρμογής με χρήση κιβωτίων (containers).

- Όπως και στην 1η άσκηση, ζητείται η εξοικείωση με *ασύγχρονο* τρόπο ανταλλαγής δεδομένων και η δυναμική ανανέωση του περιεχομένου των σελίδων με χρήση της τεχνολογίας AJAX.

- Επίσης, ζητούμενο της άσκησης είναι η εξοικείωση με την επικοινωνία υπηρεσιών μέσω REST, επεκτείνοντας παράλληλα την λειτουργικότητα της εφαρμογής με έτοιμες υπηρεσίες που σχετίζονται με θέματα ασφάλειας της εφαρμογής.

- Τέλος, θα χρειαστεί να εξοικειωθείτε με τη χρήση ενός publish-subscribe μηχανισμού με σκοπό τη δημιουργία συνδρομών και ειδοποιήσεων για τους χρήστες της πλατφόρμας.

Αναλυτικά, τα ζητούμενα της άσκησης είναι τα εξής:

1) ΑΝΑΠΤΥΞΗ (DEPLOYMENT) ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝ DOCKER

Εγκαταστήστε τις απαραίτητες υπηρεσίες σε μορφή containers. Ενδεικτικά, θα χρειαστείτε μία *MySQL* (για τα δεδομένα των χρηστών της εφαρμογής και του Keyrock), την υπηρεσία *Keyrock IDM*, την υπηρεσία *Orion Context Broker*, την υπηρεσία *PEP Proxy Wilma*, μια *MongoDB* (για το Data Storage service και για τη λειτουργία του Orion CB) και έναν *Apache Server* (ή κάποιον άλλον server) για την λογική της εφαρμογής (*βλ. παρακάτω εικόνα*).
Φροντίστε οι υπηρεσίες να συνδέονται κατάλληλα μέσω του δικτύου, να εκθέτουν εξωτερικά μόνο τις πόρτες που χρειάζεται και να διατηρούν τα δεδομένα τους ακόμα και αν πέσει κάποιο container.
Όλα τα containers της εφαρμογής θα πρέπει να περιγράφονται μέσα σε ένα αρχείο *docker-compose.yml*.

2) ΔΥΝΑΜΙΚΗ ΑΝΑΝΕΩΣΗ ΤΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΤΩΝ ΣΕΛΙΔΩΝ ΜΕΣΩ ΑΣΥΓΧΡΟΝΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ

Για κάθε δημιουργία, επεξεργασία, διαγραφή στη βάση δεδομένων, το ανανεωμένο περιεχόμενο θα πρέπει να εμφανίζεται *δυναμικά* στην αντίστοιχη σελίδα μετά από κάθε ενέργεια, όπως και στην 1η άσκηση. Δηλαδή να μη χρειάζεται να ανανεωθεί η εκάστοτε σελίδα για να δει κάποιος χρήστης τις αλλαγές του.
Για παράδειγμα, ο admin διαγράφει το χρήστη πατώντας ένα κουμπί και ο πίνακας με τους χρήστες που βλέπει ανανεώνεται δυναμικά (αφού σβηστεί ο χρήστης!) χωρίς την ανανέωση της σελίδας αυτής.
Χρησιμοποιήστε AJAX για ασύγχρονη επικοινωνία με τον server της εφαρμογής και Javascript για δυναμική ανανέωση των σελίδων.

3) ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕΣΩ ΤΗΣ ΥΠΗΡΕΣΙΑΣ DATA STORAGE

Τα δεδομένα της εφαρμογής (με εξαίρεση τα προσωπικά στοιχεία των χρηστών, τους ρόλους, τις άδειές τους, κλπ, που μπορούν να βρίσκονται στον Keyrock) πρέπει πλέον - αντί να αποθηκεύονται σε μία βάση δεδομένων MySQL - **να αποθηκεύονται σε μία MongoDB** με χρήση της υπηρεσίας Data Storage Service.

Σχεδιάστε κατάλληλα το REST API της υπηρεσίας η οποία θα πρέπει να μετατρέπει τα αιτήματα REST (HTTP requests) σε κατάλληλα queries προς μία NoSQL βάση δεδομένων MongoDB.

Προστατέψτε και αυτήν την εφαρμογή με χρήση του PEP Proxy Wilma.

4) ΕΠΕΚΤΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕ ΧΡΗΣΗ PUB/SUB ΥΠΗΡΕΣΙΑΣ (ORION CB) ΓΙΑ ΤΗΝ ΕΝΗΜΕΡΩΣΗ ΤΩΝ ΧΡΗΣΤΩΝ ΣΧΕΤΙΚΑ ΜΕ ΝΕΑ ΠΡΟΪΟΝΤΑ ΠΟΥ ΤΟΥΣ ΕΝΔΙΑΦΕΡΟΥΝ Ή ΠΡΟΪΟΝΤΑ ΠΟΥ ΑΠΟΣΥΡΟΝΤΑΙ / ΕΞΑΝΤΛΟΥΝΤΑΙ

Φροντίστε κατά τη δημιουργία ενός *προϊόντος* να καταχωρείται η αντίστοιχη οντότητα (entity) στον Orion Context Broker (FIWARE). Κάθε *οντότητα τύπου "Product"* στον Orion θα αντιπροσωπεύει ένα προϊόν και θα κρατάει πληροφορία για αυτό. Εναλλακτικά, μπορούν να δημιουργούνται οντότητες στον Orion *για κάθε κατάστημα/πωλητή (τύπου Seller),* με τα διαφορετικά προϊόντα που πωλούνται από αυτό το κατάστημα/πωλητή (και τα προϊόντα με τη σειρά τους έχουν τις δικές τους πληροφορίες μέσα στις ίδιες οντότητες). Είστε ελεύθεροι να επιλέξετε/δομήσετε εσείς τις οντότητες που σας εξυπηρετούν και συνεπώς τον τρόπο αποθήκευσης της πληροφορίας στον Orion Context Broker.

Οι απλοί χρήστες (καταναλωτές) θα μπορούν να κάνουν *subscribe* σε προϊόντα και να λαμβάνουν ειδοποίηση **σε τουλάχιστον μία από τις εξής** περιπτώσεις: **α)** όταν ένα υπάρχον προϊόν που τους ενδιαφέρει δεν είναι προς το παρόν διαθέσιμο για αγορά, αλλά γίνεται διαθέσιμο σε διάστημα ημερών που θα θέσετε (*π.χ. άμεση ειδοποίηση στο χρήστη ότι ένα κινητό τηλέφωνο είναι τώρα διαθέσιμο για αγορά!*) και **β)** όταν ένα προϊόν αποσύρεται (με βάση την ημερομηνία απόσυρσής του) ή εξαντλείται (μπορείτε να το ρυθμίσετε εσείς με κάποιο flag με τιμές 0 ή 1). Είστε ελεύθεροι να επιλέξετε και να υλοποιήσετε μία από τις δύο παραπάνω περιπτώσεις **(α ή β)** ή και τις δύο περιπτώσεις.

Οι πληροφορίες αυτές θα πρέπει να στέλνονται αυτόματα **τόσο στη βάση δεδομένων** για να προβάλλονται στο χρήστη (καταναλωτή) όταν εισέλθει ξανά στην εφαρμογή **όσο και στο γραφικό περιβάλλον** για ενημέρωση του χρήστη *σε πραγματικό χρόνο* όταν συμβαίνει κάποια ενημέρωση των προϊόντων από τους πωλητές και πρέπει να ειδοποιηθούν για αυτές στο feed τους. Το feed του χρήστη θα πρέπει δηλαδή να προβάλλει πληροφορίες για αλλαγή της κατάστασης ενός προϊόντος και να τις διατηρεί για συγκεκριμένο χρονικό διάστημα. Χρησιμοποιήστε για τα παραπάνω τον μηχανισμό pub/sub που παρέχει ο Orion CB μέσω της REST διεπαφής.

Ο Orion για να δουλέψει απαιτεί μία βάση δεδομένων **MongoDB** στην οποία αποθηκεύει όλα του τα δεδομένα. Για τις λειτουργίες, όμως, του Orion αρκεί να δουλέψετε με το REST API του.

5) ΧΡΗΣΗ ΤΗΣ ΥΠΗΡΕΣΙΑΣ KEYROCK IDM (FIWARE) ΓΙΑ ΑΥΘΕΝΤΙΚΟΠΟΙΗΣΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΤΩΝ ΧΡΗΣΤΩΝ

Αλλάξτε την έως τώρα λογική της εφαρμογής για ταυτοποίηση χρηστών και χρησιμοποιήστε την REST διεπαφή του Keyrock για την αυθεντικοποίησή τους (user

authentication) μέσω του πρωτοκόλλου OAuth 2.0. Θα χρειαστεί επιπλέον να κάνετε register την εφαρμογή σας μέσα από το γραφικό περιβάλλον του Keyrock και να κωδικοποιήσετε το client_id και το client_secret που θα σας δοθούν από τον Keyrock. Η κωδικοποίηση αυτή θα πρέπει να γίνει σε *base64* (*client_id:client_secret*). Το αποτέλεσμα της κωδικοποίησης θα αποτελέσει το Authorization Basic header που απαιτείται στο REST call του Keyrock για το authentication και την παροχή ενός προσωρινού access token στην εφαρμογή.
Ουσιαστικά, μπορούμε να καταργήσουμε την admin page της εφαρμογής μας και όλη η διαχείριση να γίνεται από το dashboard του Keyrock. Μπορεί να γίνεται όμως και από δική σας σελίδα (επιλέγετε εσείς το πώς θα γίνει). Στην αναφορά σας, θα πρέπει να διευκρινιστεί το τι επιλέξατε. Για πρόσβαση στην εφαρμογή, οι χρήστες (καταναλωτές) και οι πωλητές θα πρέπει να έχουν κάνει sign up στον Keyrock και να έχουν επιβεβαιωθεί (να γίνουν enabled) από κάποιον διαχειριστή τόσο για πρόσβαση στον Keyrock όσο και για δικαίωμα πρόσβασης στην εφαρμογή.

Για να έχουν πρόσβαση στην εφαρμογή, οι απλοί χρήστες και οι πωλητές θα πρέπει να έχουν κάνει sign up στον Keyrock και να έχουν *επιβεβαιωθεί* από κάποιον διαχειριστή: **α)** τόσο για πρόσβαση στον Keyrock, όσο και **β)** για δικαίωμα πρόσβασης στην εφαρμογή.

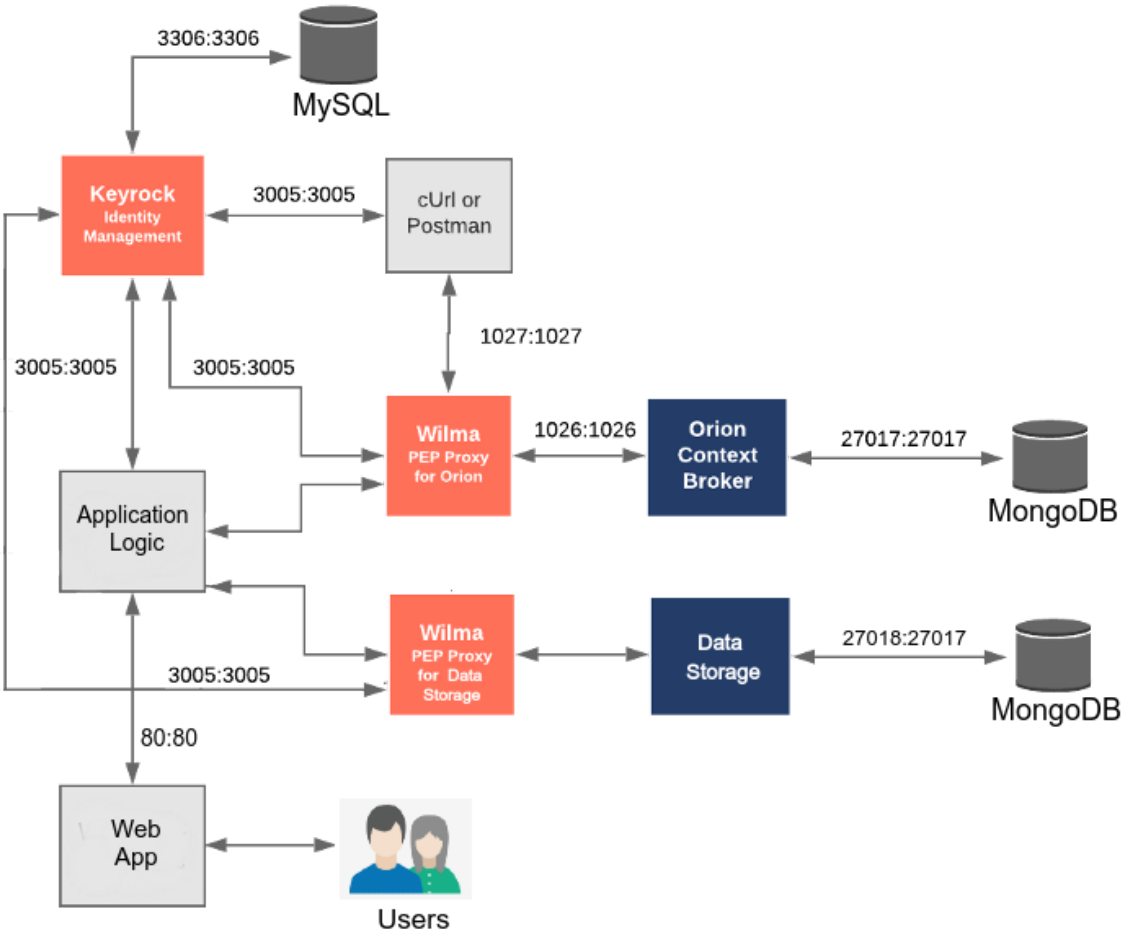
6) ΧΡΗΣΗ ΤΗΣ ΥΠΗΡΕΣΙΑΣ PEP PROXY - WILMA (FIWARE) ΓΙΑ ΠΡΟΣΤΑΣΙΑ ΤΩΝ BACKEND ΕΦΑΡΜΟΓΩΝ ΑΠΟ ΜΗ ΕΞΟΥΣΙΟΔΟΤΗΜΕΝΟΥΣ ΧΡΗΣΤΕΣ

Με την υπηρεσία PEP Proxy, κάθε αίτημα προς μια υπηρεσία (π.χ. Orion Context Broker) ελέγχεται για τον αν θα προωθηθεί ή όχι σε αυτήν. Έτσι, τα αιτήματα προς την υπηρεσία γίνονται μέσω του PEP Proxy, ο οποίος ελέγχει την ορθότητα του Master Key (ή του OAuth 2.0 token) που δίνεται στο header των αιτημάτων και, μόνο αν αυτό είναι σωστό, προωθεί το αίτημα στην υπηρεσία. Με αυτόν τον τρόπο, η υπηρεσία προστατεύεται από κακόβουλους χρήστες και εφαρμογές. Με αυτόν τον τρόπο (χάρη στον PEP Proxy), θα πρέπει να προστατευτεί τόσο ο Orion όσο και η υπηρεσία Data Storage.

7) MIGRATION ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΣΕ INSTANCE ΤΟΥ GOOGLE CLOUD PLATFORM

Δημιουργήστε ένα VM instance με λειτουργικό Ubuntu, εγκαταστήστε το Docker (και το docker-compose!) σε αυτό και σηκώστε εκεί την εφαρμογή.
Θα χρειαστεί να μεταφέρετε στην εικονική μηχανή (VM) τα αρχεία του κώδικά σας και τα αντίστοιχα Dockerfiles, docker-compose, κλπ.
Θα χρειαστεί επίσης να “ανοίξετε” έξω από το VM (expose) τις πόρτες που αφορούν τον server της εφαρμογής και τον Keyrock για πρόσβαση στα Web UI τους.

Ακολουθεί ένα βοηθητικό/ενδεικτικό διάγραμμα της αρχιτεκτονικής που θα συνθέτουν οι παραπάνω υπηρεσίες.



Ενδεικτικό διάγραμμα αρχιτεκτονικής υπηρεσιών

ΧΡΗΣΙΜΑ LINKS:

https://fiware-orion.readthedocs.io/en/master/
https://fiware-pep-proxy.readthedocs.io/en/latest/
https://fiware-idm.readthedocs.io/en/latest/index.html
https://keyrock.docs.apiary.io/#
https://documenter.getpostman.com/view/513743/RWMLLRui
https://documenter.getpostman.com/view/513743/RWaHxUgP
https://documenter.getpostman.com/view/513743/fiware-getting-started/RVu5kp1c
https://www.tutorialspoint.com/mongodb/mongodb_php.htm
https://www.php.net/manual/en/mongodb.tutorial.library.php

ΣΗΜΕΙΩΣΗ:

- Φροντίστε ώστε η εφαρμογή να είναι όσο γίνεται ευχάριστη και εύκολη στη χρήση (εικόνες, εύχρηστο μενού, γραμματοσειρές, χρώματα, έλεγχος εγκυρότητας δεδομένων εισόδου στις φόρμες, κτλ.).

- Είστε ελεύθεροι να χρησιμοποιήσετε **οποιοδήποτε** Web Framework σας εξυπηρετεί τόσο στη δημιουργία του γραφικού περιβάλλοντος (frontend) όσο και στην υλοποίηση της λογικής της εφαρμογής (backend), αρκεί να μπορείτε να το υποστηρίξετε.
- Φροντίστε κατά το migration της εφαρμογής σε άλλο host μηχάνημα να *διατηρούνται* τα δεδομένα των βάσεων δεδομένων.

ΠΑΡΑΔΟΤΕΑ:

- Scripts κώδικα με επεξηγηματικά σχόλια.
- Αναφορά με παρατηρήσεις για τη δουλειά σας: πώς κάνατε το migration στο GCP, τι δικτύωση υπάρχει μεταξύ των υπηρεσιών και πώς αυτές συνεργάζονται, τεχνολογίες και frameworks που χρησιμοποιήθηκαν και πιθανές ελλείψεις της εφαρμογής που δεν υλοποιήθηκαν ή υλοποιήθηκαν μερικώς, τον τρόπο αναπαράστασης των δεδομένων στον Orion, τι θεωρείτε ως entity και τι attributes έχει, το REST API reference του Data Storage το οποίο σχεδιάσατε και υλοποιήσατε.
- Αρχείο .sql (backup αρχείο της mysql βάσης).

“First, solve the problem. Then, write the code.” - John Johnson

Καλή επιτυχία!