

Πανεπιστήμιο Κρήτης

HY359 – Διαδικτυακός Προγραμματισμός (Web Programming)

Χειμερινό Εξάμηνο 2024/2025

Διδάσκων: **Μιχαήλ Μουνταντωνάκης**

Υπεύθυνος Βοηθός: **Κώστας Καμπάδας** (Συντονιστής) και οι υπόλοιποι βοηθοί

3η Σειρά Ασκήσεων (Ατομική)

Διάρκεια: Πέμπτη 14/11 – Παρασκευή 6/12

Αξία: 12% του τελικού σας βαθμού

(Συνολο ασκήσεων: **105 μονάδες**)

Θεματική ενότητα : **Servlets - AJAX- REST API**

Άσκηση 1. Εγκατάσταση Tomcat Και Βάσης [15 μονάδες]

Θα πρέπει να εγκαταστήσετε τον Tomcat και να τρέξετε παραδείγματα του μαθήματος, αλλά και τη βάση του μαθήματος. **Σαν παραδοτέο θα έχετε screenshots από την εκτέλεση των παραδειγμάτων**, ενώ και στην εξέταση του μαθήματος θα πρέπει να τρέξετε κάποιο παράδειγμα για να δείξετε ότι δουλεύει ο tomcat και η βάση του μαθήματος, ακόμα και αν δεν έχετε υλοποιήσει κάποια άσκηση από τις υπόλοιπες. Αν χρησιμοποιείται κάποιο άλλο server/βάση, τότε δείχνετε εκείνο τον server/βάση.

Άσκηση 2. [E-199] Εγγραφή χρηστών [30 μονάδες]

Καλείστε να καλύψετε τις ανάγκες εγγραφής ενός χρήστη για την πλατφόρμα E-199, για την καλύτερη αντιμετώπιση συμβάντων από την πυροσβεστική που θα υλοποιήσετε στο project. Η φόρμα εγγραφής θα περιέχει τα πεδία εγγραφής που υλοποιήσατε στην πρώτη και στην δεύτερη σειρά ασκήσεων. Πλέον όμως η εγγραφή θα ολοκληρώνεται με τη δημιουργία ενός χρήστη στη μεριά του server και την αποθήκευση των στοιχείων του στην βάση δεδομένων.

- Μπορείτε να χρησιμοποιήσετε είτε HTML είτε **JSON απαντήσεις από τη μεριά του server (προτείνεται το δεύτερο αφού είναι πιο ευέλικτο).**
- **Κώδικας για τη βάση και οδηγίες σας δίνονται στο elearn**
 - Προσαρμόστε τις φόρμες σας να έχουν τα ίδια ονόματα με τη βάση
 - Στέλνετε json δεδομένα από τον client
- Σε περίπτωση που ο χρήστης δηλώσει ένα username το οποίο υπάρχει ήδη στη βάση, θα πρέπει να τυπωθεί ότι το συγκεκριμένο username χρησιμοποιείται ήδη. Αντίστοιχα και για το τηλέφωνο.
 - **Users και Volunteers θα αποθηκεύονται σε διαφορετικά tables στη βάση**
 - **Δεν επιτρέπεται να έχουν το ίδιο username, email ή κινητό τηλέφωνο**
 - Αυτά θα γίνονται τη στιγμή που συμπληρώνεται το αντίστοιχο πεδίο μέσω AJAX. Άρα εδώ θα χρειαστούν διαφορετικά AJAX requests για να το κάνετε αυτό.

Σημαντικά

- Θα πρέπει να αποθηκεύονται στη βάση και οι συντεταγμένες (lat lon) της **διεύθυνσης** (από την άσκηση 2), οπότε στέλνετε και αυτά τα έξτρα πεδία.
- Για ότι δεν μπορείτε να υλοποιήσετε βάλτε default τιμές (πχ lat lon)
- Θα πρέπει να δουλεύουν όλοι οι έλεγχοι που έχετε κάνει στις προηγούμενες ασκήσεις. Αν για οποιοδήποτε λόγο δε δουλεύουν, μπορείτε να κάνετε τους ελέγχους αυτούς στο server.
- Η μικροϋπηρεσία (servlet) που θα φτιάξετε θα πρέπει σε περίπτωση που υπάρχει κάποιο λάθος από τη μεριά του client θα πρέπει ο **server να επιστρέφει το κατάλληλο status code** και τα κατάλληλα μηνύματα λάθους (π.χ. Status code 403 - το username υπάρχει).
- Εφόσον έχουν συμπληρωθεί όλα τα υποχρεωτικά πεδία με σωστό τρόπο και ο χρήστης κάνει click στο κουμπί “Εγγραφή”, **η διαδικασία εγγραφής πρέπει να ολοκληρωθεί επιτυχώς, αποθηκεύοντας το συγκεκριμένο λογαριασμό στη βάση.**
 - Αν ακολουθήσετε τον κώδικα της βάσης, θα γίνει αυτόματα
- Το servlet θα πρέπει να επιστρέφει όλα τα στοιχεία που έδωσε ο χρήστης και το μήνυμα
 - «Η εγγραφή σας πραγματοποιήθηκε επιτυχώς»
 - Μπορείτε να χρησιμοποιήσετε όποια λύση θέλετε, ενδεικτικά να ανανεώνεται η σελίδα μετά την εγγραφή
 - είτε μέσω JAVA Servlets/AJAX (Single Page Application)
 - είτε να μεταφέρεται σε μία καινούρια σελίδα (πχ μέσω JAVA Servlets.)

Βαθμολόγηση

- AJAX requests και έλεγχος για διπλότυπα (username, κινητό, email) **(10 μονάδες)**
- Σωστό registration και σωστή αποθήκευση στη βάση δεδομένων **(15 μονάδες)**
- Κατάλληλα μηνύματα και status codes για επιτυχία ή αποτυχία των requests και της εγγραφής **(5 μονάδες)**

Άσκηση 3. [E-199] Λειτουργίες Εγγεγραμμένου Χρήστη [25 μονάδες]

Εδώ κάνετε την άσκηση μόνο για users, δε χρειάζεται για Volunteers

Εδώ θα υλοποιήσετε κάποιες λειτουργίες ενός εγγεγραμμένου χρήστη ιδανικά μέσω ενός single page application και μέσω **HTTPsession API ή Cookies**.

Ένας εγγεγραμμένος χρήστης θα πρέπει να μπορεί: **(13 μονάδες)**

- να κάνει login, αλλιώς το σύστημα να του πετάει κατάλληλο μήνυμα μη επιτυχημένου login
- να κάνει logout
- να παραμένει συνδεδεμένος μέχρι να τελειώσει η συνεδρία - session (να μη χρειάζεται δηλαδή να κάνει login ξανά μέχρι να κάνει log-out).

Επίσης σε μία μοναδική σελίδα (σε ένα single page application) ο συνδεδεμένος χρήστης πρέπει να δει τα στοιχεία του συγκεντρωτικά σε μια σελίδα και να μπορεί να τα αλλάξει όλα εκτός από τα username, κινητό και ΑΦΜ **(12 μονάδες)**

Άσκηση 4. [E-199] Καταχώρηση Συμβάντος – Ανεξάρτητο REST API [35 μονάδες]

**** Αν δε μπορεί κάποιος να κάνει το REST API και το κάνει μέσω Servlets, θα πάρει max 25/35 για την άσκηση 4**

Μπορείτε να το υλοποιήσετε με όποια βιβλιοθήκη για REST API θέλετε, πχ με την Java Spark που είδαμε στο μάθημα

α) Δημιουργία REST API (25 μονάδες)

Υποθέστε ότι υπάρχει ένα καθολικό API, στο οποίο κάποιος βάζει πληροφορίες για ένα συμβάν. Το REST API, θα πρέπει να έχει τις εξής δυνατότητες (μπορείτε και να τις επεκτείνεται αν σας βολεύει), χωρίς να χρειάζεται κάποιος να συνδεθεί σε αυτό με username/password (θεωρούμε στα πλαίσια της άσκησης ότι είναι προσβάσιμο από όλους):

- **1. Καταχώρηση Συμβάντος (POST) (9 μονάδες)**
 - /incident/
 - Να τοποθετεί ένα καινούριο συμβάν από κάποιο χρήστη ή επισκέπτη μέσω ενός JSON.
 - Να έχει μέσα το JSON τα στοιχεία που χρειάζονται
 - incident_type
 - description
 - user_type
 - user_phone
 - address
 - lat (μπορεί και να μη δοθεί)
 - lon (μπορεί και να μη δοθεί)
 - prefecture (μπορεί και να μη δοθεί)
 - municipality (μπορεί και να μη δοθεί)
 - Έλεγχος
 - το user_phone να είναι 10-14 ψηφία
 - το user_type να είναι guest, admin ή user
 - Το incident type να είναι ένα από τα εξής: fire/accident
 - Το prefecture να είναι ένα από τα εξής: heraklion/chania/rethymno/lasithi
 - Αποτέλεσμα
 - Να έχει τα κατάλληλα response codes και μηνύματα ανάλογα με την επιτυχία ή όχι του request
 - Να βάζει στη βάση το συμβάν μαζί με τις εξής τιμές:
 - status→submitted
 - danger→unknown
 - start_datetime→ βάζει ο κώδικας την τωρινή datetime
 - Το πρόγραμμα να αποθηκεύει το συμβάν στη βάση
 - Παράδειγμα
 - /incident/

- JSON → {


```

      {"incident_type":"fire",
      "description":"Fire in Agia Pelagia",
      "user_type":"guest",
      "user_phone":"6977771111",
      "address":"Agia Pelagia",
      "lat":"X",
      "lon":"Y",
      "prefecture":"Heraklion",
      "municipality":"Malevizi"}
      
```

- **2. Ανάκτηση συμβάντος συγκεκριμένου τύπου και κατάστασης (GET) (8 μονάδες)**

- ο /incidents/:type/:status
- ο **Path parameter:** type, status
- ο **Optional Parameters:** municipality
- ο **Έλεγχος**
 - Μπορείτε να θεωρήσετε ότι αν δώσει type=all και status=all λαμβάνει υπόψιν όλα τα περιστατικά
 - Μπορεί να δώσει προαιρετικά δήμο
- ο **Αποτέλεσμα:** Ένα JSON που περιέχει όλα τα περιστατικά που ταιριάζουν και τις πληροφορίες για αυτό.
- ο **Παραδείγματα:**
 - /incidents/fire/running
 - /incidents/all/all?municipality=Heraklion

- **3. Ανανέωση Κατάστασης Συμβάντος (PUT) (4 μονάδες)**

- ο incidentStatus/:incident_id/:status
- ο **Path parameter:** incident_id, status
- ο **Έλεγχος**
 - Να αλλάζει το status ενός incident
 - Από submitted → fake ή running
 - Από running → finished
- ο **Αποτέλεσμα**
 - Αν όλα πάνε καλά να επιστρέφει status 200 και σχετικό μήνυμα
 - Στο finished να βάζει στη βάση και το end_datetime (αυτόματη μερα/ωρα από τον κώδικα)
- ο **Παράδειγμα**
 - incidentStatus/:incident_id/running

- **4. Διαγραφή Συμβάντος (DELETE) (4 μονάδες)**

- ο /incidentDeletion/:incident_id
- ο **Path parameter:** incident_id
- ο **Έλεγχος**
 - για σωστές μεταβλητές (να υπάρχει το incident id)
 - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
- ο **Αποτέλεσμα**

- Αν όλα πάνε καλά να επιστρέφει status 200 και σχετικό μήνυμα
- ο **Παράδειγμα**
 - /incidentDeletion/2

Η υλοποίησή σας ιδανικά θα παίζει με JSON και θα κάνει σωστή διαχείριση λαθών (σκεφτείτε τη χρήση των status codes). Για παράδειγμα

- 200 όταν όλα είναι καλά
- 403 forbidden (όταν είναι λάθος κάποιο id)
- 406 Not Acceptable (πχ αν δοθούν λάθος τιμές, πχ στο status κλπ)

β) Τεστάρισμα REST API (10 μονάδες)

Θα πρέπει να τεστάρετε το REST API **με έναν από τους παρακάτω τρόπους** (όπως είδαμε/θα δούμε στο μάθημα). Δε χρειάζεται να κάνετε όλους τους τρόπους.

- **1) (Plain JAVA)** Με έναν java client για να τεστάρετε όλα τα σενάρια, χωρίς τη χρήση κάποιου γραφικού περιβάλλοντος
- **2) Μέσω Postman**, όπου θα πρέπει να δείξετε screenshots, αλλά και να το χρησιμοποιήσετε στην εξέταση της άσκησης
- **3) (AJAX+HTML).** Μέσω απλού γραφικού HTML και AJAX requests στο REST API, που να μπορεί να τεστάρει όλα τα σενάρια
 - βοηθητικό για το project
 - καλύτερα να το ανοίξετε από browser με απενεργοποιημένο το CORS

Βοηθητικά

Δείτε τις διαφάνειες και τα παραδείγματα. Για οτιδήποτε ζητάει η άσκηση, έχουμε δείξει έστω ένα μικρό παράδειγμα. Μη ξεχνάτε τη χρήση του "use strict"; για την JavaScript.

Το parse ενός JSON string γίνεται με χρήση της JSON.parse(str) που επιστρέφει το javascript object που αντιστοιχεί στο str, δεδομένου ότι το str είναι μια σωστή αναπαράσταση JSON.

Μπορείτε να χρησιμοποιήσετε κάποιον online linter όπως ο jshint <http://jshint.com/> για να βελτιώσετε την ποιότητα του js κώδικά σας.

Προτείνεται η χρήση του netbeans 12.4 σαν IDE και για τα servlets και για το Spark REST API. Επίσης προσπαθήστε να οργανώσετε με όμορφο τρόπο τα servlets που θα χρησιμοποιήσετε.

Τρόπος Παράδοσης

Η Παράδοση θα γίνεται μέσω elearn. Θα πρέπει να παραδώσετε ένα zip που να περιέχει ένα φάκελο A3_AM, όπου AM ο αριθμός μητρώου σας.

Μπορείτε να έχετε ένα φάκελο για τις 2 πρώτες ασκήσεις (είτε και για τις 3 πρώτες), και ένα φάκελο για τις υπόλοιπες.

Εκπρόθεσμες ασκήσεις **δεν θα γίνονται δεκτές**.

Για σοβαρά θέματα που δεν επιτρέπουν την εμπρόθεσμη παράδοση των ασκήσεων, στείλτε email στον διδάσκοντα.

Αντιγραφή

Σε περίπτωση αντιγραφής θα μηδενίζονται άμεσα οι εργασίες όλων των εμπλεκόμενων.

Καλή εργασία