Πανεπιστήμιο Κρήτης
ΗΥ359 – Διαδικτυακός Προγραμματισμός (Web Programming)
Χειμερινό Εξάμηνο 2021/2022
Διδάσκων: **Μιχαήλ Μουνταντωνάκης**
Υπεύθυνος Βοηθός: **Γ. Στυλιανάκης** (Συντονιστής) και οι υπόλοιποι βοηθοί

3η Σειρά Ασκήσεων (Ατομική)

Διάρκεια: 22/11 – 8/12
Αξία: 12% του τελικού σας βαθμού
(Συνολο ασκήσεων: **105 μονάδες**)
Θεματική ενότητα : **Servlets -AJAX-REST API**

Άσκηση 1. [Personalized-Health] Εγγραφή χρήστη [30%]

Καλείστε να καλύψετε τις ανάγκες εγγραφής ενός χρήστη για την πλατφόρμα υγείας που θα υλοποιήσετε. Η φόρμα εγγραφής θα περιέχει τα πεδία εγγραφής που υλοποιήσατε στην πρώτη άσκηση και στην δεύτερη άσκηση. 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 email και το ΑΜΚΑ.
  - **Γιατροί και Χρήστες θα αποθηκεύονται σε διαφορετικά tables στη βάση**
  - **Δεν επιτρέπεται να έχουν το ίδιο username ή email ή ΑΜΚΑ**
    - **ένας απλός χρήστης με έναν άλλο απλό χρήστη**
    - **ένας γιατρός με έναν άλλο γιατρό**
    - **ένας απλός χρήστης και ένας γιατρός**
  - **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 **(latitude longitude)** της **διεύθυνσης** (από την άσκηση 2), οπότε στέλνετε και αυτά τα έξτρα πεδία.
- 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 επιστρέψει όλα τα στοιχεία που έδωσε ο χρήστης και το μήνυμα
  - «Η εγγραφή σας πραγματοποιήθηκε επιτυχώς» **αν είναι απλός χρήστης**
  - «Η εγγραφή σας πραγματοποιήθηκε, αλλά θα πρέπει να σας πιστοποιήσει ο διαχειριστής», **αν είναι γιατρός**

  - 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 (αμκα, username, email) (12 μονάδες)
- Σωστό registration και σωστή αποθήκευση στη βάση δεδομένων (12 μονάδες)
- Κατάλληλα μηνύματα και status codes για επιτυχία ή αποτυχία των requests και της εγγραφής (6 μονάδες)

Άσκηση 2. [Personalized-Health] Λειτουργίες Εγγεγραμμένου Χρήστη [30%]

Εδώ θα υλοποιήσετε κάποιες λειτουργίες ενός εγγεγραμμένου απλού χρήστη **(όχι γιατρού)**, ιδανικά μέσω ενός single page application και μέσω **HTTPsession API ή Cookies.**
Ένας εγγεγραμμένος χρήστης θα πρέπει να μπορεί: **(12 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 έχει **τις 3 ακόλουθες επιλογές:**

- 1. Να δει τα στοιχεία του συγκεντρωτικά σε μια σελίδα και να μπορεί να τα αλλάξει όλα εκτός από τα username, ΑΜΚΑ **(8 μονάδες)**
  - Αν παει να αλλαξει το email, πρέπει να ελέγχεται ότι δεν υπάρχει σε κάποιον χρήστη η γιατρό
- 2. Να χρησιμοποιεί το API https://rapidapi.com/malaaddincelik/api/fitness-calculator/ έτσι ώστε να μπορεί αν έχει δηλώσει βάρος και ύψος να βλέπει: **(5 μονάδες)**
  - Το BMI του και την κατάσταση της υγείας του (normal, overweight, κλπ)
    - μέσω του **Get bmi**
  - Το ιδανικό βάρος (μέσω της φόρμουλας Devine)
    - μέσω του **Get Ideal Weight**
- 3. Να μπορεί να δει μία λίστα με όλους τους γιατρούς που είναι πιστοποιημένοι (certified), πχ σε κάποιο πίνακα **(5 μονάδες)**
  - Να δείχνει τις εξής πληροφορίες για τον κάθε γιατρό
    - Όνομα, Επώνυμο, Διεύθυνση, Πόλη, Πληροφορίες, Ειδικότητα, Τηλέφωνο
  - Κάποιοι ενδεικτικοί πιστοποιημένοι γιατροί (μη αληθινά δεδομένα και τυχαίες διευθύνσεις στο Ηράκλειο) υπάρχουν στη βάση.

Άσκηση 3. XSS – Cross site scripting και ασφάλεια [10%]

Το **Cross Site Scripting (XSS)** (https://en.wikipedia.org/wiki/Cross-site_scripting) είναι μία μορφή επίθεσης που χρησιμοποιείται κυρίως σε εφαρμογές διαδικτύου. Ένα τυπικό σενάριο τέτοιου είδους επίθεσης είναι το εξής: Μία ιστοσελίδα επιτρέπει την εισαγωγή στοιχείων από το χρήστη (πχ, σχόλια) τα οποία στη συνέχεια δημοσιεύει. Εάν κάποιος χρήστης (ο επιτιθέμενος) εισάγει κάποιο script προς δημοσίευση αντί απλού κειμένου, αυτό τελικά θα αποτελεί μέρος του html και θα εκτελείται κανονικά μετά από κάθε επίσκεψη.

Σε αυτή την άσκηση αρχικά η σελίδα εγγραφής νέου μέλους θα είναι μη ασφαλής και θα κάνετε επίθεση. Συγκεκριμένα, αυτό μπορεί να γίνει με οποιοδήποτε από τα πεδία κειμένου. Ζητείται να:

- **Εφαρμόσετε την επίθεση κάνοντας κάτι από τα εξής (5%)**
    - να αλλάξετε το χρώμα της σελίδας (CSS related) που εμφανίζει τα στοιχεία που έδωσε ο χρήστης
    - ή να εισάγετε ένα button η href στη σελίδα το οποίο στο onclick θα κάνει κάποιο action (π.χ alert ή να στέλνει το χρήστη σε μια εξωτερική σελίδα), και περιγράψτε σύντομα πώς πετύχατε την επίθεση.
        - Προσοχή, οι τωρινές εκδόσεις των browsers έχουν εργαλεία τα οποία προσπαθούν να αποτρέψουν τέτοιου είδους επιθέσεις, οπότε θα πρέπει να απενεργοποιήσετε τη συγκεκριμένη λειτουργικότητα.
        - Προτιμότερο είναι να χρησιμοποιήσετε τον firefox που δεν έχει κάποιο default φιλτρο (αν έχετε όμως το noscript plugin ή άλλα παρεμφερή θα σας ειδοποιήσει για τέτοιου είδους XSS επιθέσεις, τους ελέγχους των οποίων μπορείτε να απενεργοποιήσετε).
- **Διορθώστε την εφαρμογή ώστε να μην είναι πλέον ευάλωτη στην επίθεση (5%)**
- Hint: Δείτε τα σχετικά παραδείγματα που δείξαμε και στο μάθημα

Άσκηση 4. [Personalized-Health] Λειτουργικότητα Εξετάσεων – Ανεξάρτητο REST API [35%]

α) Δημιουργία REST API (23/35)

Υποθέστε ότι ο κάθε ασθενής κάνει εξετάσεις σε διάφορα μικροβιολογικά εργαστήρια, και υπάρχει ένα ανεξάρτητο API για να ανεβαίνουν οι εξετάσεις των ασθενών.
Γι' αυτό το λόγο υπάρχει ανάγκη για ένα καθολικό API, το οποίο είναι υπεύθυνο γι αυτό. 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 εξετάσεων** (POST) **(5 μονάδες)**
    - /newBloodTest/
    - Να λαμβάνει τις εξετάσεις σε ένα JSON (όπως είδαμε στο μάθημα).

        - Να έχει μέσα το JSON τουλάχιστον ΑΜΚΑ, ημερομηνία, όνομα μικροβιολογικού εργαστηριου (δείτε το σχετικό πίνακα στη βάση)
        - Να έχει μία ή παραπάνω μετρήσεις
            - Δεν είναι απαραίτητο να δοθούν όλες οι μετρήσεις (πχ μπορεί να δοθεί μόνο η χοληστερίνη και το ζάχαρο).
            - Θα μπορούν να προστεθούν στο μέλλον μέσω άλλου request έξτρα μετρήσεις
    - **Έλεγχος**
        - η ημερομηνία να μην είναι μελλοντική
        - να έχει δοθεί έστω μία μέτρηση
        - οι μετρήσεις που δόθηκαν να έχουν τιμή >0
        - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
    - **Αποτέλεσμα**
        - Να έχει τα κατάλληλα response codes και μηνύματα ανάλογα με την επιτυχία ή όχι του request
    - Το πρόγραμμα να αποθηκεύει τις μετρήσεις στη βάση, και σε ποια κατηγορία ανήκει η τιμή της (χαμηλή τιμή, φυσιολογική τιμή, υψηλή τιμή)
        - **Δίνεται έτοιμος κώδικας γι αυτό (δείτε τον κώδικα της βάσης)**
    - **Παράδειγμα**
        - /newBloodTest/
            - JSON → {
              "amka":"03069200000",
              "test_date":"2021-10-11",
              "medical_center":"pagni",
              "blood_sugar":"100.0",
              "vitamin_b12":"50.0"
              }

- **2. Ανάκτηση εξετάσεων ασθενή** (GET) **(5 μονάδες)**
    - /bloodTests/{AMKA}
    - **Path parameter:** {AMKA}
    - **Optional Parameters**: fromDate (optional), toDate (optional)

    - **Έλεγχος**
        - Αν δοθούν, το fromDate να είναι πάντα ημερολογιακά πριν από το toDate
            - Αν δοθεί μόνο το ένα από τα 2, δεν υπάρχει πρόβλημα
        - Να ελέγχει αν υπάρχει το ΑΜΚΑ στον πίνακα bloodtest
        - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
    - **Αποτέλεσμα**: Ένα JSON που περιέχει όλες τις εξετάσεις για τον ασθενή, και για κάθε εξέταση θα έχει όλες τις πληροφορίες
        - Μπορεί να δίνει προαιρετικά τις παραμέτρους fromDate και toDate για να περιορίζει το χρονικό διάστημα
    - **Παραδείγματα**:
        - /bloodTestMeasure/03069200000
        - /bloodTestMeasure/03069200000?fromDate=2021-01-01&toDate=2021-11-11

- **3. Προβολή συγκεκριμένης μέτρησης ασθενή** (GET) **(4 μονάδες)**
  - /bloodTestMeasure/{AMKA}/{Measure}
  - **Path parameter:** {AMKA}
  - **Path parameter:** {Measure}
    - cholesterol, blood_sugar,...
  - **Έλεγχος**
    - Να υπάρχει το AMKA στον πίνακα bloodtest
    - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
  - **Αποτέλεσμα**: Ένα JSON με όλες οι τιμές που έχουν γίνει για τον ασθενή αυτό για τη συγκεκριμένη μέτρηση, πχ χοληστερίνη (δηλαδή μετρήσεις σε διαφορετικές εξετάσεις)
    - Θα περιέχει την ημερομηνία (test_date), το μικροβιολογικό εργαστήριο, την τιμή της και την κατηγορία
  - **Παραδείγματα**
    - /bloodTestMeasure/03069200000/cholesterol
    - /bloodTestMeasure/03069200000/iron

- **4. Ανανέωση πληροφοριών εξέτασης** (PUT) **(5 μονάδες)**
  - /bloodTest/{bloodTestID}/{measure}/{value}
  - **Path parameter:** {bloodTestID}
  - **Path parameter:** {measure}
  - **Path parameter:** {value}
  - **Έλεγχος**
    - για σωστές μεταβλητές (να υπάρχει το bloodTestID και το measure να αντιστοιχεί σε μέτρηση που υποστηρίζεται, πχ cholesterol)
    - Να είναι το value >0
    - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
  - **Αποτέλεσμα**
    - Να μπορεί να ανανεώσει την τιμή μιας συγκεκριμένης μέτρησης ή να βάλει τιμή σε μία μέτρηση που δεν είχε δοθεί κάποια τιμή πριν, και να στέλνει το σχετικό μήνυμα και κωδικό επιτυχίας (200)
  - **Παραδείγματα**
    - /bloodTest/1/cholesterol/210
    - /bloodTest/1/blood_sugar/98

- **5. Διαγραφή εξέτασης** (DELETE) **(4 μονάδες)**
  - /bloodTestDeletion/{bloodTestID}
  - **Path parameter:** {bloodTestID}
  - **Έλεγχος**
    - για σωστές μεταβλητές (να υπάρχει το bloodTestID)
      - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
  - **Αποτέλεσμα**
    - Αν όλα πάνε καλά να επιστρέφει status 200 και σχετικό μήνυμα
  - **Παράδειγμα**
    - /bloodTestDeletion/1

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 αρνητικές τιμές σε εξέταση)

- Δείτε τα Response.Status Codes του API

β) Τεστάρισμα REST API (12/35)

Θα πρέπει να τεστάρετε το REST API σας με αυτούς τους δύο τρόπους (όπως είδαμε στο μάθημα)

- **1**) **(Plain JAVA)** Με έναν java client για να τεστάρετε όλα τα σενάρια για τις εξετάσεις, χωρίς τη χρήση κάποιου γραφικού περιβάλλοντος **(5 μονάδες)**

- **2) (AJAX+HTML).** Μέσω απλού γραφικού HTML και AJAX requests στο REST API, που να μπορεί να τεστάρει όλα τα σενάρια **(7 μονάδες)**
  - βοηθητικό και 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 σαν IDE. 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