Πανεπιστημιο Δυτικης Αττικης
Τμημα Μηχανικων Πληροφορικης και Υπολογιστων

Τεχνικό Έγγραφο

Φοιτητής: ***Ιάκωβος Μαστρογιαννόπουλος***

Μάθημα: ***Ηλεκτρονική Μάθηση*** – Καθηγητής: ***Χρήστος Τρούσσας***

Περίληψη

Αυτό είναι το τεχνικό έγγραφο που ζητήθηκε να εκπονηθεί για το μάθημα «Ηλεκτρονική Μάθηση» του Πανεπιστήμιου Δυτικής Αττικής. Στο συγκεκριμένο έγγραφο, εξηγείται ο τρόπος ανάλυσης και σχεδιασμού της πλατφόρμας, όπως και αναφέρονται οι τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη της. Ο κώδικας της εργασίας βρίσκεται στο **GitHub**.

Περιεχόμενα

Κατάλογος σχημάτων

1. Ανάλυση ζητημάτων

1.1. Παρουσίαση θέματος. Το θέμα της εργασίας είναι ένα λογισμικό ηλεκτρονικής μάθησης που επιτρέπει σε εκπαιδευτικούς να ανεβάσουν υλικό και να δημιουργήσουν διαγωνίσματα για τους μαθητές που συμμετάσχουν στα μαθήματα τους. Ο βασικός σκοπός της εργασίας είναι να παρέχεται μία πλατφόρμα η οποία να έχει καλό σχεδιασμό και έχει καλή εμπειρία ο χρήστης του. Δηλαδή, καλή **Διεπαφή Χρήστη/Εμπειρία Χρήστη (User Interface/User Experience, UI/UX)**.

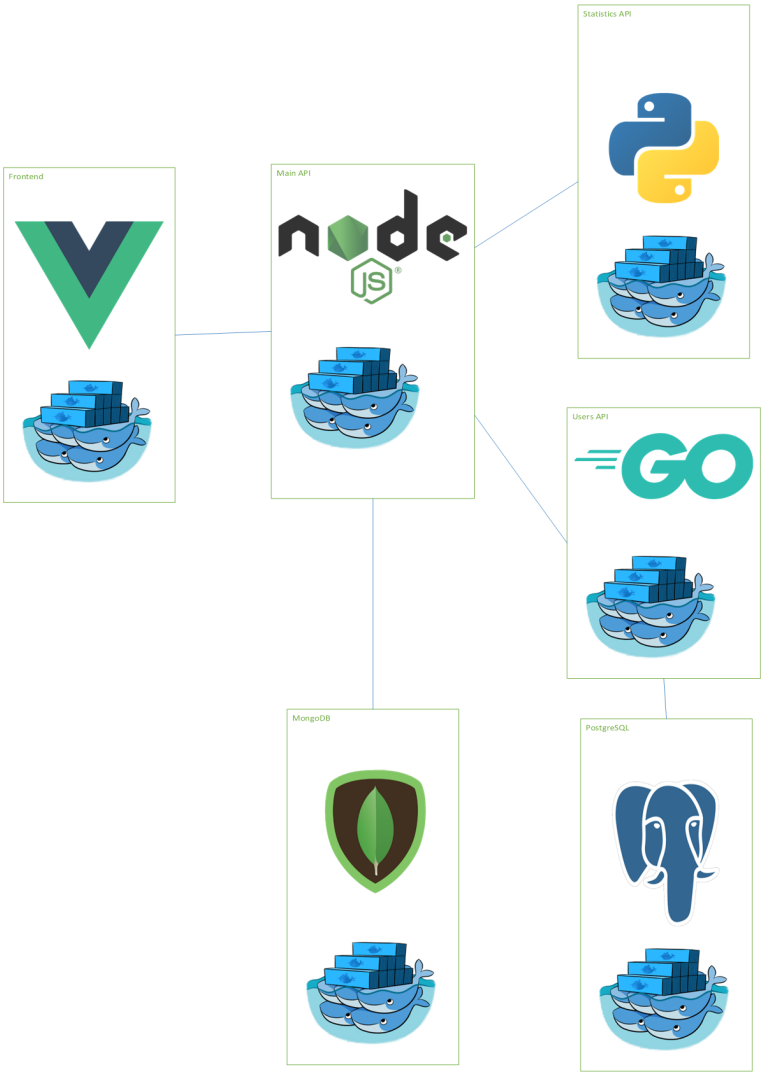
1.2. Λειτουργίες Πλατφόρμας.

1. **Εκπαιδευτικό Υλικό** πρέπει να:
   - δίνεται στους χρήστης με πολυμέσα (όπως βίντεο, εικόνες, κείμενο)
   - δίνεται σε κεφάλαια
2. **Αξιολόγηση των μαθητών με διαγωνίσματα** πρέπει να:
   - γίνεται με διαγωνίσματα που θα περιέχουν ερωτήσεις διαφορετικού τύπου (πχ πολλαπλής επιλογής)
   - δίνονται στο τέλος του εξαμήνου, με ένα τελικό διαγώνισμα στο τέλος του μαθήματος
3. **Διατήρηση στατιστικών δεδομένων** για την πρόοδο του μαθητή.
4. **Εμφάνιση μηνυμάτων στους μαθητές** ως συμβουλές ή μηνύματα ανατροφοδότησης

2. Σχεδιασμός πλατφόρμας

2.1. Αρχιτεκτονική Πλατφόρμας. Η αρχιτεκτονική της πλατφόρμας αποφασίστηκε να έχει την λογική των **μικρο-υπηρεσιών (microservices)**, δηλαδή μία σταθερή ιστοσελίδα που δέχεται τα δεδομένα της από ένα κατανεμημένο σύστημα που αποτελείται από πολύ μικρές **Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces, APIs)**. Τα APIs είναι τρία:

- Το κύριο API το οποίο είναι υπεύθυνο να επικοινωνεί με την σταθερή ιστοσελίδα και με τα άλλα δύο APIs. Στην ουσία, είναι ένας μεσολαβητής που ονομάζεται middleware.
- Ένα API που διαχειρίζεται τα δεδομένα της βάσης.
- Και ένα API που θα υπολογίζει στατιστικά δεδομένα για κάθε μαθητή ξεχωριστά.

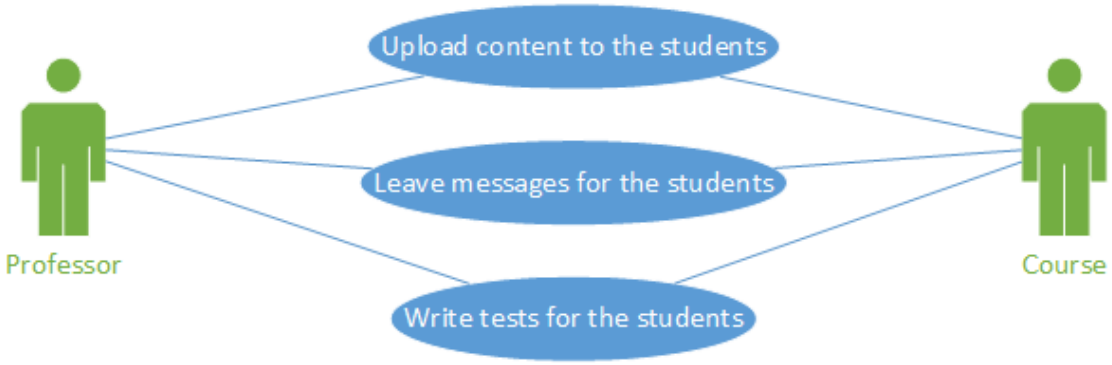


Σχήμα 1: Οι σχέσεις που έχουν μεταξύ τους οι κόμβοι της πλατφόρμας

Τα δεδομένα θα τα παρέχουν οι δύο βάσεις δεδομένων, μία κλασσική SQL και μία μοντέρνα NoSQL. Ο σκοπός των δύο βάσεων είναι ότι μερικά δεδομένα πρέπει να έχουν κάποια δομή από πίσω τους, ενώ κάποια άλλα όπως οι ερωτήσεις των διαγωνισμάτων δεν το χρειάζονται. Όλοι οι κόμβοι της πλατφόρμας είναι φτιαγμένοι πάνω σε ένα εικονικό περιβάλλον δημιουργημένο από το Docker.

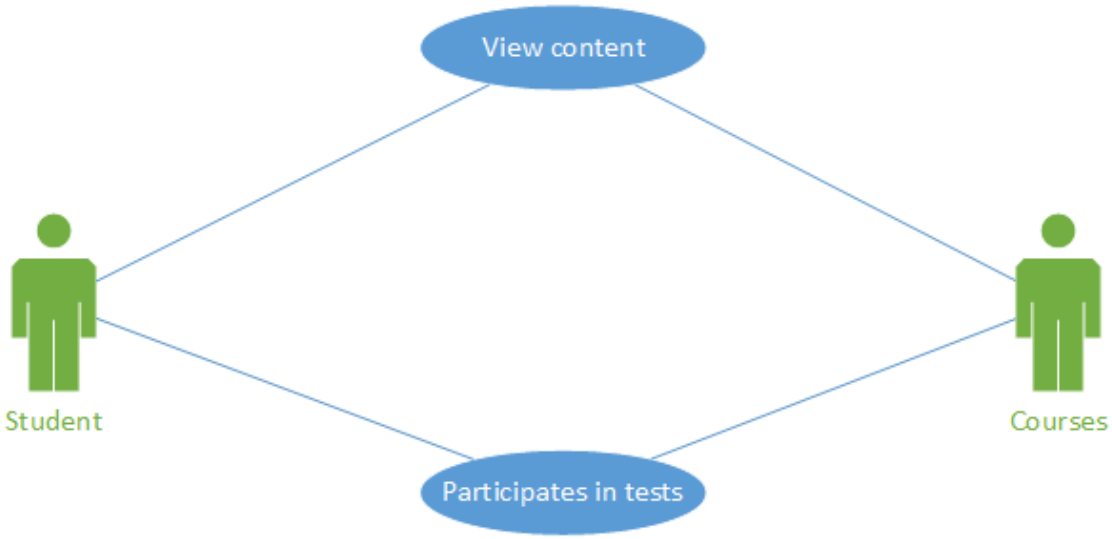
2.2. Περιπτώσεις χρήσεων. Οι περιπτώσεις χρήσεων χωρισμένες σε δύο κατηγόριες:

- Μία για τον καθηγητή



Σχήμα 2: Περιπτώσεις χρήσεων για τον καθηγητή

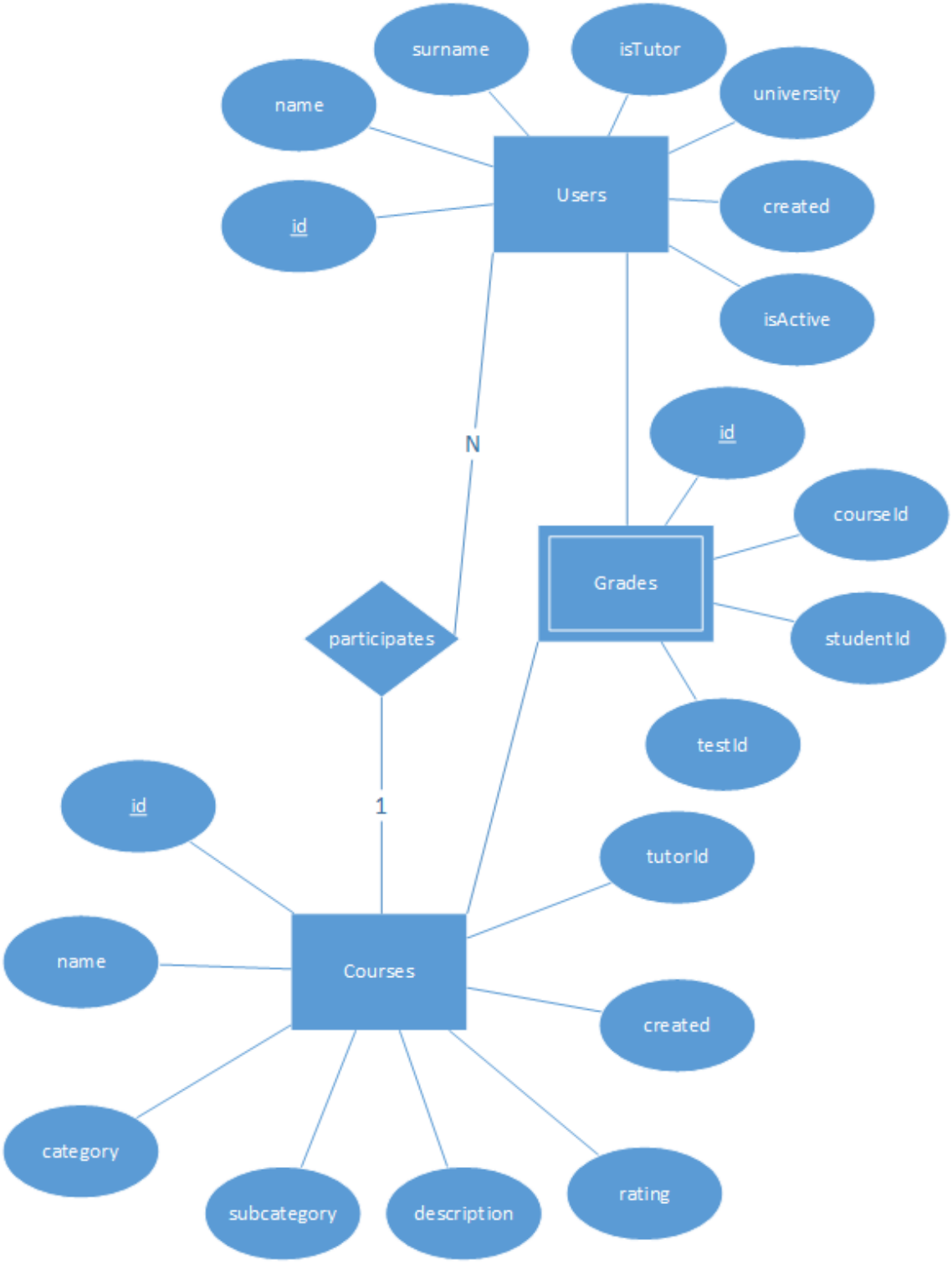
- Μία για τον φοιτητή



Σχήμα 3: Περιπτώσεις χρήσεων για τον φοιτητή

2.3. Σχεδιασμός της βάσης δεδομένων.

2.4. SQL βάση. Η SQL βάση δεδομένων υλοποιήθηκε με την postgres. Στην postgres γίνεται η αρχικοποίηση του ID των χρηστών και των μαθημάτων, τα οποία χρησιμοποιεί και η NoSQL βάση αργότερα.



Σχήμα 4: Σχεδιασμός SQL Βάσης

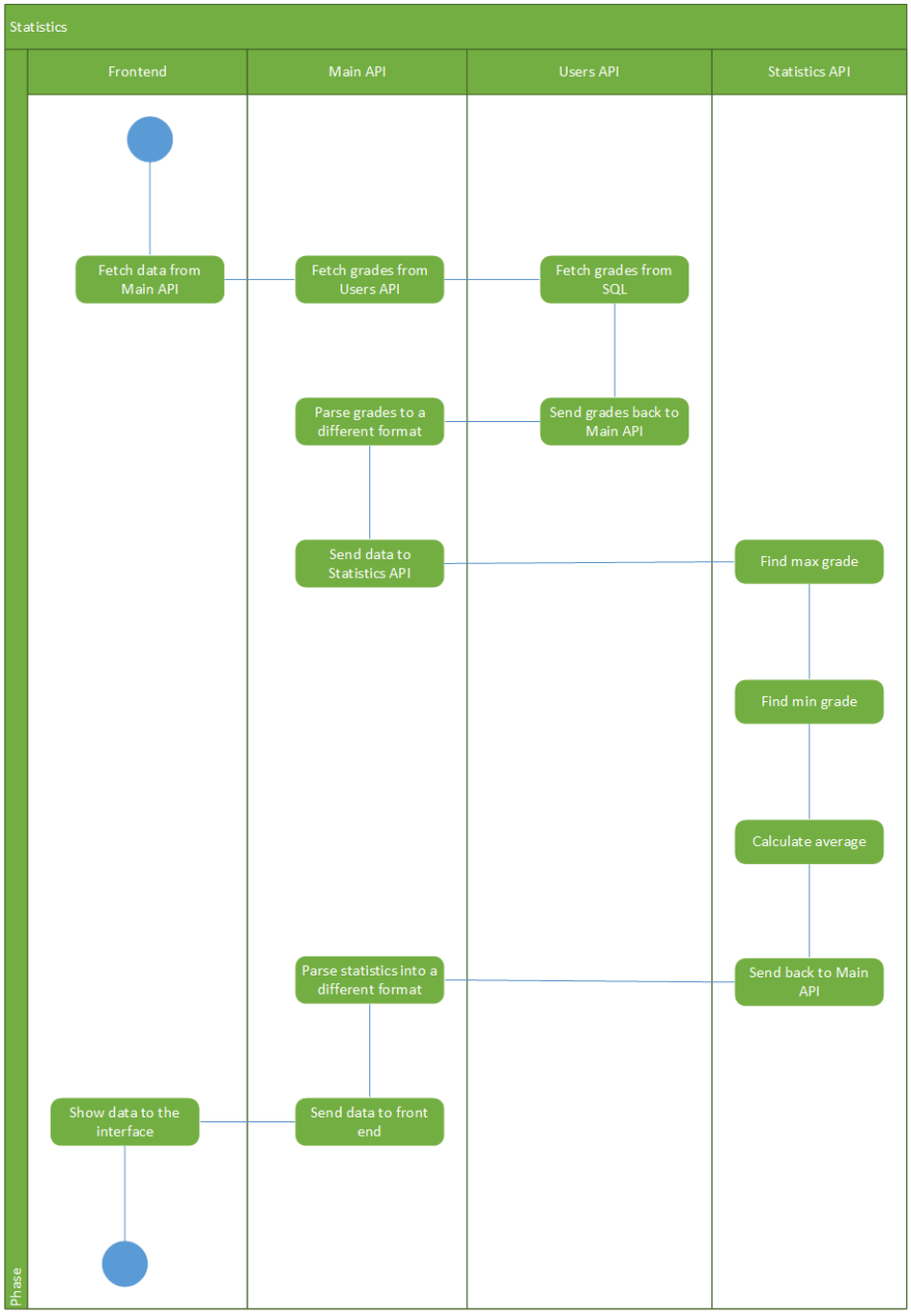
2.5. NoSQL βάση. Η NoSQL βάση δεδομένων υλοποιήθηκε με την MongoDB. Η συγκεκριμένη βάση είναι αδόμητη, όποτε δεν σχεδιάστηκε κάποιο διάγραμμα για να δείξει την μορφή των δεδομένων. Ένα παράδειγμα σε JSON μίας πιθανής εγγραφής μαθημάτων είναι:

```
{
  _id: '0',
  chapters: [
    {
      id: 1,
      files: ['path_to file'],
      test: [
        {
          _id: '1',
          question: 'text',
          question_type: 'multiple',
          answer_1: 'a',
          answer_2: 'b',
          answer_3: 'c',
          answer_4: 'd',
          correct: 'answer_1',
          grade_weight: 30 // In percentage
        }
      ],
      grade_weight: 20
    }
  ]
}
```

Κάθε index του πίνακα chapters παρέχει και ένα διαφορετικό κεφάλαιο, ενώ στον πίνακα test κάθε index είναι και μία ερώτηση.

3. Υλοποίηση πλατφόρμας

3.1. Παράδειγμα: Παρουσίαση Στατιστικών Δεδομένων. Θα ακολουθήσει παράδειγμα της υλοποίησης του πιο περίπλοκου function της πλατφόρμας.



Σχήμα 5: Διάγραμμα δραστηριότητας στατιστικών δεδομένων

3.2. Υλοποίηση του backend.

3.2.1. API των χρηστών. Το API των χρηστών έχει γραφτεί στην γλώσσα προγραμματισμού golang και ενώνεται με την postgres για να πάρει τις βαθμολογίες του χρήστη και να τις επιστρέψει πίσω.

```
type Grade struct {
  Id       int64   `db:"id" json:"id"`
  CourseId int64   `db:"courseId" json:"course_id"`
  UserId   int64   `db:"studentId" json:"student_id"`
  TestId   int64   `db:"testId" json:"test_id"`
  Grade    float32 `db:"grade" json:"grade"`
}

func getGrades(c *gin.Context) {
  id := c.Params.ByName("id")
  var grades []Grade
  _, err := dbmap.Select(&grades, "SELECT * FROM grades WHERE studentId=$1", id)
  if err != nil {
    c.JSON(500, gin.H{"error": err.Error()})
    return
  }
  c.JSON(200, grades)
}
```

3.2.2. API των στατιστικών δεδομένων. Το API των στατιστικών έχει μία συνολική λειτουργία, αυτή να υπολογίζει τον μέγιστο βαθμό, τον ελάχιστο και τον μέσο όρο που έχει προς το παρόν ο φοιτητής. Είναι γραμμένο στην Python.

```
@app.route('/', methods=['POST'])
@cross_origin()
def index():
  response_object = {}
  data = request.get_json()

  if len(data['grades']) != 0:
    return jsonify({})

  response_object['max'] = max(data['grades'])
  response_object['min'] = min(data['grades'])
  response_object['total'] = len(data['grades'])

  sum = 0.0
  for grade in data['grades']:
    sum += grade
  response_object['average'] = sum / len(data['grades'])

  return jsonify(response_object)
```

3.2.3. Κεντρικό API. Το κεντρικό API είναι γραμμένο σε NodeJS και επικοινωνεί με όλους τους κόμβους και φιλτράρει τα δεδομένα κατάλληλα ώστε να επιστρέφει τις επιθυμητές πληροφορίες. Σε διαφορετικά endpoints, συνδέεται αυτόματα με την MongoDB, αλλά στο συγκεκριμένο παράδειγμα δεν υπάρχει κάποια χρήση της.

```
user_router.post('/:id/grades/statistics', async (req, res) => {
  axios.get(`${user_api_url}/grades/${req.params.id}`)
    .then(async (response) => {
      const tests = response.data
      let grade_array = []

      for (let i = 0; i < tests.length; i++) {
        grade_array.push(tests[i].grade)
      }

      const statistics = await axios.post(
        'http://statistics:5000/',
        { 'grades': grade_array }
      )
      res.status(200).send(statistics.data)
    })
    .catch((error) => {
      res.status(400).send({ error: error })
    })
})
```

3.3. Υλοποίηση του frontend. Το frontend της εφαρμογής υλοποιήθηκε σε VueJS χρησιμοποιώντας διάφορες λειτουργίες της, όπως το Vue Router για να γίνεται διαχείριση των routes και το VueX για να παρέχει έναν state manager.

3.3.1. VueX.

```
import router from "@/router"
import { user_service } from "@/services/user.service"

const user = JSON.parse(localStorage.getItem('user'))
const grades = JSON.parse(localStorage.getItem('grades'))
const statistics = JSON.parse(localStorage.getItem('statistics'))
const state = user ? ...: ...

const actions = {
  ...
  getStatistics({ commit }) {
    user_service.getStatistics(user.id, grades)
      .then(
        query => {
          localStorage.setItem('statistics', JSON.stringify(query.data))
          commit('getStatisticsSuccess', query.data)
        }
      )
  }
}

const mutations = {
  ...
  getStatisticsSuccess(state, statistics) {
    state.statistics = statistics
  }
}

export const account = {
  namespaced: true,
  state,
  actions,
  mutations
}
```

3.3.2. Vue κώδικας.

```
<div>
  ...
  <b-card
    style="padding: 5rem;"
    v-if="!user.is_tutor && statistics !== undefined"
  >
    <div class="row">
      <div class="col-sm-4">
        <h5>Statistics</h5>
        <p>
          <b>Passed tests:</b> {{ statistics.total }}<br>
          <b>Average score:</b> {{ statistics.average }}<br>
          <b>Best score:</b> {{ statistics.max }}<br>
          <b>Worst score:</b> {{ statistics.min }}<br>
        </p>
      </div>
      <div class="col-sm-8">
        <h5 style="text-align: center;">Your Grades</h5>
        <grades-list/>
      </div>
    </div>
  </b-card>
</div>
```

```
import { mapActions, mapState } from 'vuex'
import CoursesList from '@/components/CoursesList.vue'
import GradesList from '@/components/GradesList.vue'

export default {
  components: {
    CoursesList,
    GradesList
  },
  computed: {
    ...mapState('account', ['user', 'statistics'])
  },
  methods: {
    ...mapActions('account', ['getData', 'getStatistics', 'getGrades'])
  },
  created() {
    let user = JSON.parse(localStorage.getItem('user'))
    console.log(user);
    if (user.name === undefined)
      this.getData(user.id)

    if (!this.user.is_tutor) {
      this.getGrades()
      this.getStatistics()
    }
  }
}
```