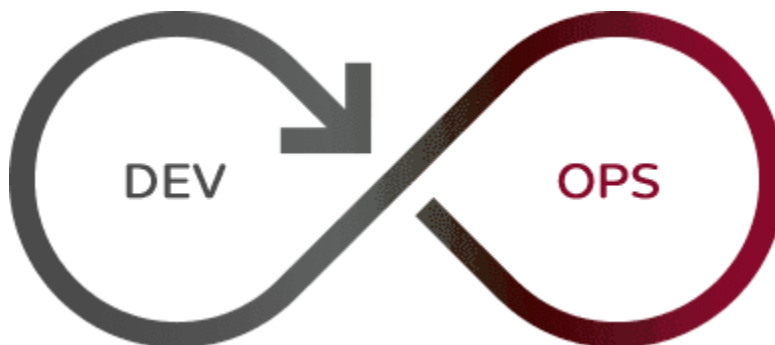




ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ



Ομάδα Χ

Κοκκινογένης Βασίλης - 2021042

Φωτεινόπουλος Παναγιώτης - 2021154

Θέμα Εργασίας

Εργασία στο Μάθημα «**Βασικές έννοιες και εργαλεία DevOps**»

Περιεχόμενα	
Ανάλυση και Σχεδίαση	2
Παραδοχές	2
Ανάλυση	3
Σχεδίαση	5
Τεχνολογίες / Εργαλεία	8
Αρχιτεκτονική Εφαρμογής	8
Σενάρια χρήσης	9
Deployment	52
Ansible	52
Ansible - Docker	52
Kubernetes	52
CI/CD	54
Γενικά Σχόλια/Παρατηρήσεις	54
Με δυσκόλεψε / δεν υλοποίησα	54
Κώδικας	54
Αποθετήρια κώδικα	55
Δοκιμαστικά accounts και urls	55
Url δοκιμαστικού περιβάλλοντος	55
Οδηγίες Χρήσης / Εγκατάστασης	55

Ανάλυση και Σχεδίαση

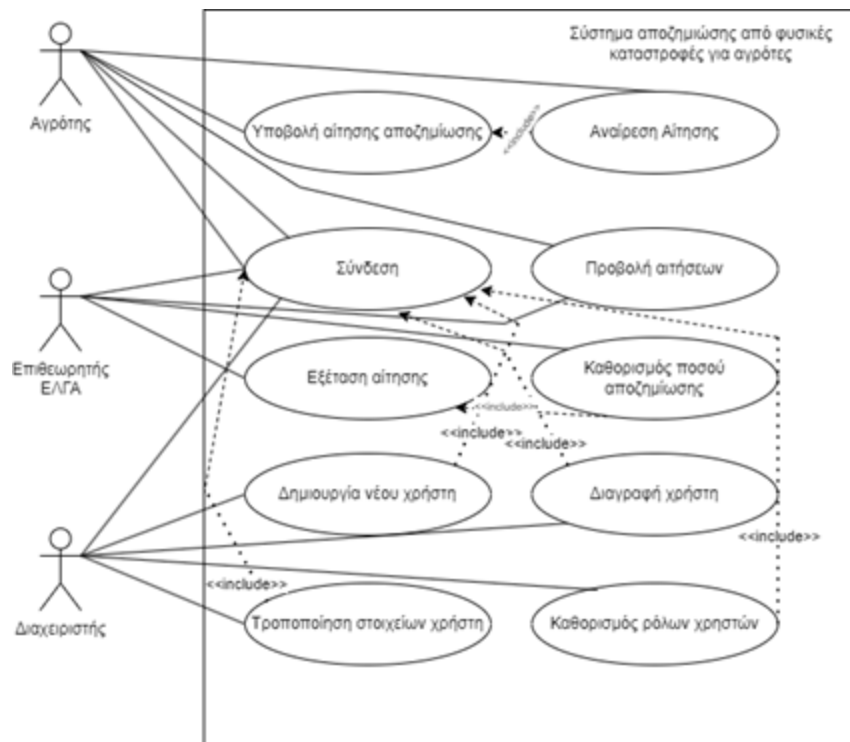
Παραδοχές

1. Η αυθεντικοποίηση εντάσσεται μέσα στο UseCase του Login.
2. Η Απόρριψη/Έγκριση αίτησης συμπεριλαμβάνεται μαζί με εξέταση αίτησης.
3. Αν δεν εγκριθεί το αίτημα, το ποσού αποζημίωσης που θα επιστρέφεται θα είναι

- 0.
4. Θεωρούμε ότι υπάρχει μόνο ένας Admin στο σύστημα και οι αντίστοιχοι ρόλοι (ROLE_ADMIN, ROLE_FARMER, ROLE_INSPECTOR).
5. Εφόσον ο επιθεωρητής ΕΛΓΑ εγκρίνει ή απορρίπτει μια αίτηση ο αγρότης θα μπορεί να συνδεθεί και να δει την απόφαση για την αίτηση του.
6. Θεωρούμε ότι ο διαχειριστής θα μπορεί να δημιουργεί/διαγράφει/τροποποιεί και αναθέτει ρόλους μέσα από το frontend.
7. Θεωρούμε ότι η αποζημίωση καταβάλλεται μόνο όταν η ζημιωμένη παραγωγή ξεπερνά το 40% σε σχέση με την κανονική παραγωγή.
8. Επειδή ο αγρότης μπορεί να παράγει και φρούτα και λαχανικά θεωρούμε ότι η αποζημίωση στα φρούτα είναι 10€ το κιλό και στα λαχανικά 20€ το κιλό. Δηλαδή 10 και 20 ευρώ αντίστοιχα για κάθε κιλό καταστρεμμένης παραγωγής.
9. Θεωρούμε ότι οι αγρότες και οι επιθεωρητές εισάγονται μόνο από τον admin στο σύστημα, δηλαδή δεν μπορούν να κάνουν Signup αλλά μόλις τους προσθέσει ο Admin να κάνουν Login.

Ανάλυση

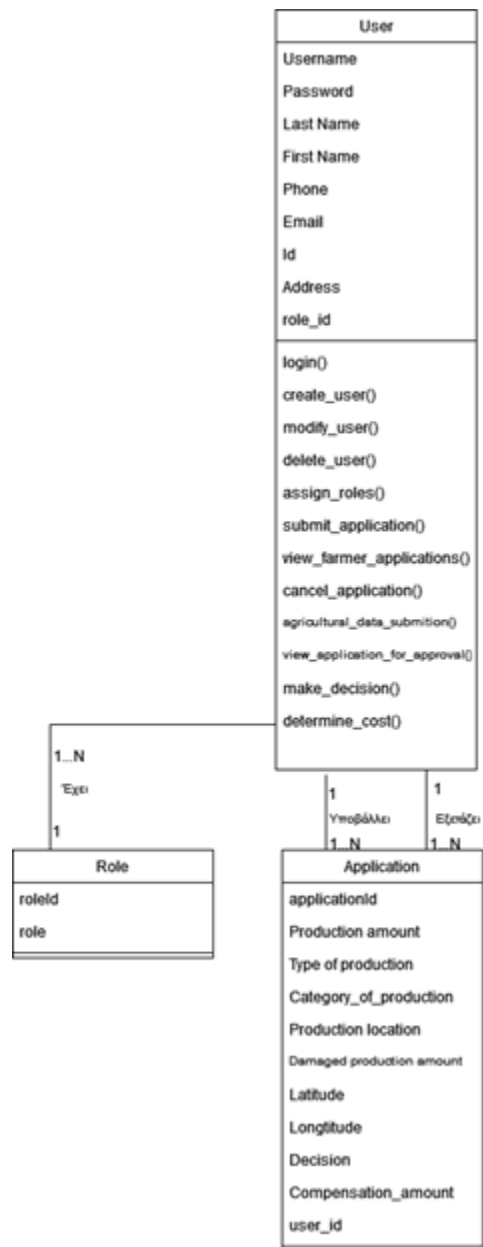
Use Case Διάγραμμα



Επεξήγηση Usecase diagram: Στο usecase diagram φαίνονται οι 3 actors που περιγράφονται και από την εκφώνηση, ο αγρότης, ο επιθεωρητής ΕΛΓΑ και ο admin. Ο αγρότης, μπορεί να υποβάλλει αίτηση αποζημίωσης στην οποία συμπεριλαμβάνεται η καταχώρηση των στοιχείων της αγροτικής του παραγωγής, ενώ παράλληλα μπορεί και να δει τις αιτήσεις που έχει υποβάλλει και την κατάσταση τους (έγκριση-απόρριψη) και τέλος μπορεί και να αναιρέσει την αίτηση του μόνο αν δεν έχει ελεγχθεί (έγκριση-απόρριψη). Ο επιθεωρητής ΕΛΓΑ, μπορεί να εξετάσει αιτήσεις και να καθορίσει τα αντίστοιχα ποσά αποζημίωσης. Ο Admin, μπορεί όπως αναφέρει ξεκάθαρα και η εκφώνηση να δημιουργήσει/διαγράψει ένα χρήστη, να τροποποιήσει τα στοιχεία του και να αναθέσει ρόλους στους χρήστες. Τέλος, για να πραγματοποιηθούν όλες οι παραπάνω λειτουργίες πρέπει οι χρήστες να κάνουν login.

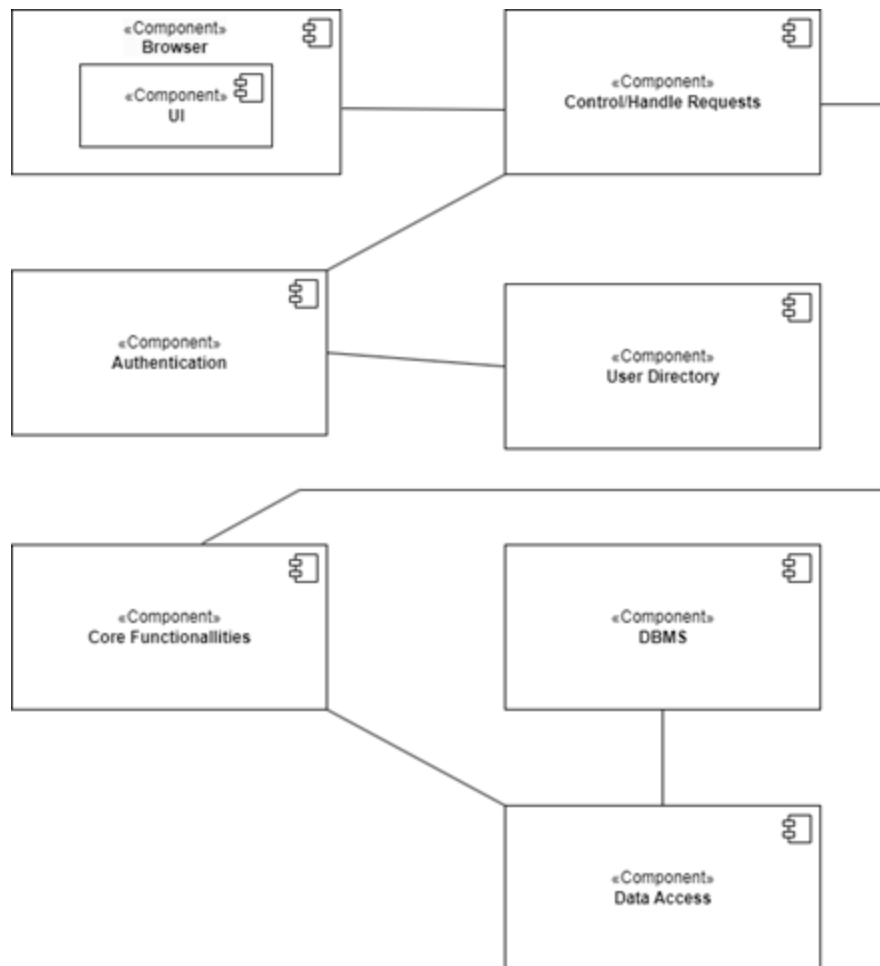
Σχεδίαση

Class διάγραμμα



Επεξήγηση Class diagram: Στο Class διάγραμμα έχουμε 3 κλάσεις. Η κλάση User έχει τα attributes Username, Password, First Name, Last Name κ.ο.κ. Ειδικότερα έχει μεθόδους όπως περιγράφει η εκφώνηση για την δημιουργία/διαγραφή/τροποποίηση χρηστών και ανάθεση ρόλων σε αυτούς και για υποβολή αίτησης αποζημίωσης, αναίρεση της αίτησης αποζημίωσης, προβολή των υποβληθέντων αιτήσεων του αγρότη αλλά και υποβολής των στοιχείων της αγροτικής του παραγωγής. Επιπλέον, έχει μια μέθοδο που αφορά την απόφαση(Έγκριση ή Απόρριψη) για τις υποβληθείσες αιτήσεις, μια άλλη που αφορά την απόφαση/καθορισμό του πόσου αποζημίωσης μιας αίτησης και μια μέθοδο που αφορά την προβολή των αιτήσεων προς έγκριση. Επιπροσθέτως, υπάρχει και μια κλάση Application με τα αντίστοιχα attributes που φαίνονται στο διάγραμμα και τα αντίστοιχα associations μεταξύ της κλάσης User και Application. Τέλος, υπάρχει μια κλάση Role με τα αντίστοιχα attributes που φαίνονται στο διάγραμμα και το αντίστοιχο association μεταξύ της κλάσης User και Role.

Component/Deployment διάγραμμα



Επεξήγηση Component diagram: Το σύστημα μας θα έχει ένα frontend μέρος οπότε για αυτό βάλαμε το component UI, το frontend θα τρέχει πάνω σε ένα browser για αυτό και ενσωματώσαμε το component UI μέσα στο component browser. Επίσης, υπάρχει και ένα component authentication που χρησιμοποιείται για την αυθεντικοποίηση των χρηστών όταν κάνουν login. Το component User directory θα περιέχει όλους τους χρήστες (farmer, inspector, admin), το component Core Functionalities εμπεριέχει όλες τις λειτουργίες που εκτελεί το σύστημα ενώ το component Control/Handle Requests υπάρχει για να χειρίζεται τα σφάλματα. Τέλος, υπάρχουν και άλλα δύο components το component DBMS, που είναι η βάση δεδομένων που αποθηκεύουμε τα αποτελέσματα και ένα component Data Access που

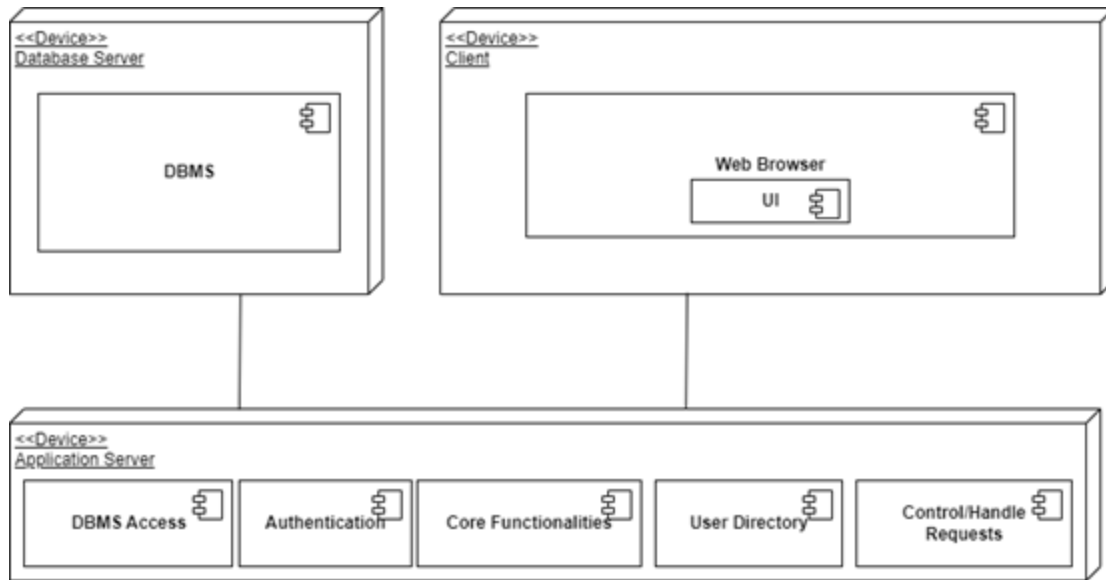
χρησιμοποιείται για την πρόσβαση στην βάση δεδομένων.

Τεχνολογίες / Εργαλεία

Για την εφαρμογή του backend μας χρησιμοποιήθηκε SpringBoot framework γραμμένο σε Java γλώσσα προγραμματισμού. Η βάση δεδομένων μας δουλεύει με Postgres τεχνολογία. Μέσω RestAPI επικοινωνεί το backend με το Frontend μας το οποίο θα λάβει ή θα στείλει πόρους από την βάση δεδομένων. Το Frontend αναπτύχθηκε με τεχνολογία Vue3 framework και JavaScript, CSS, Bootstrap. Επιπλέον, για το deployment χρησιμοποιήθηκαν τεχνολογίες ansible, docker, kubernetes αλλά και Jenkins. Έχουμε δουλέψει πάνω στο σύστημα μας με παρόχους της Google (με gcloud), της Microsoft (με Azure) αλλά και τοπικά με Vagrant.

Αρχιτεκτονική Εφαρμογής

Deployment diagram:



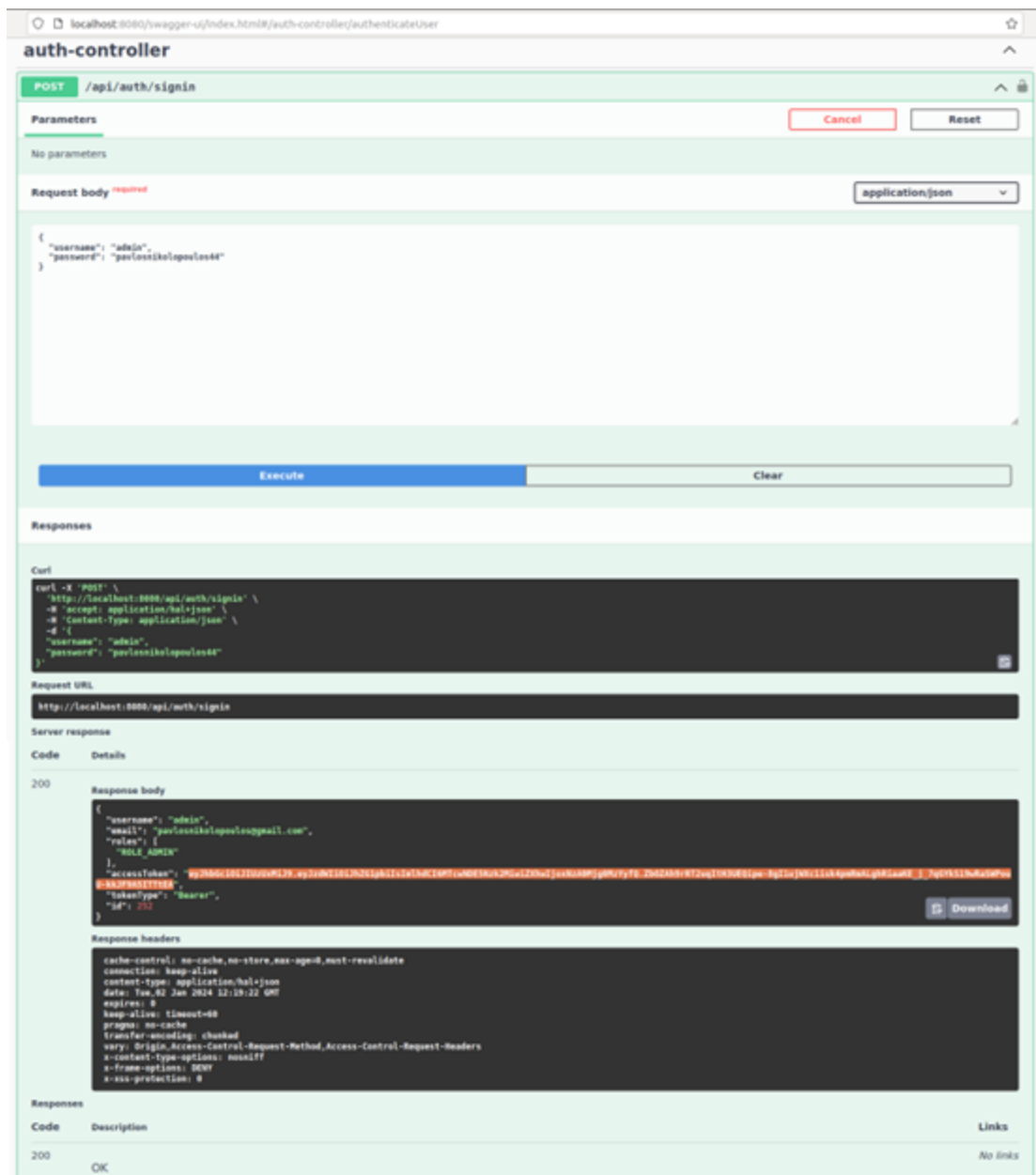
Επεξήγηση Deployment diagram: Έχουμε έναν Database Server στον οποίο τρέχει το Component της βάσης δεδομένων DBMS, έχουμε έναν Client όπου τρέχει ο Browser του χρήστη και στον Browser του, τρέχει το UI της εφαρμογής. Ακόμα, έχουμε έναν ApplicationServer στον οποίο βρίσκονται όλα τα υπόλοιπα Components της εφαρμογής μας. Τέλος, έχουμε και δύο διασυνδέσεις και συγκεκριμένα μια διασύνδεση μεταξύ του Client και του ApplicationServer και μεταξύ του ApplicationServer και του Database Server.

Σενάρια χρήσης

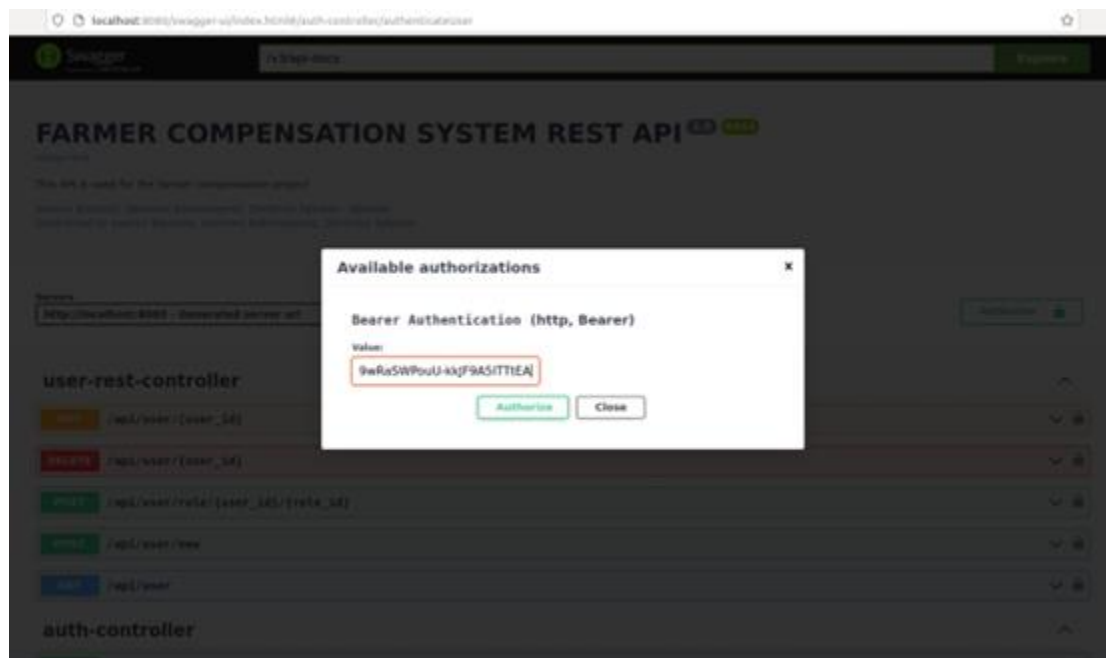
Σενάρια της εφαρμογής

Οδηγίες για τη χρήση της εφαρμογής με τη βοήθεια του swagger

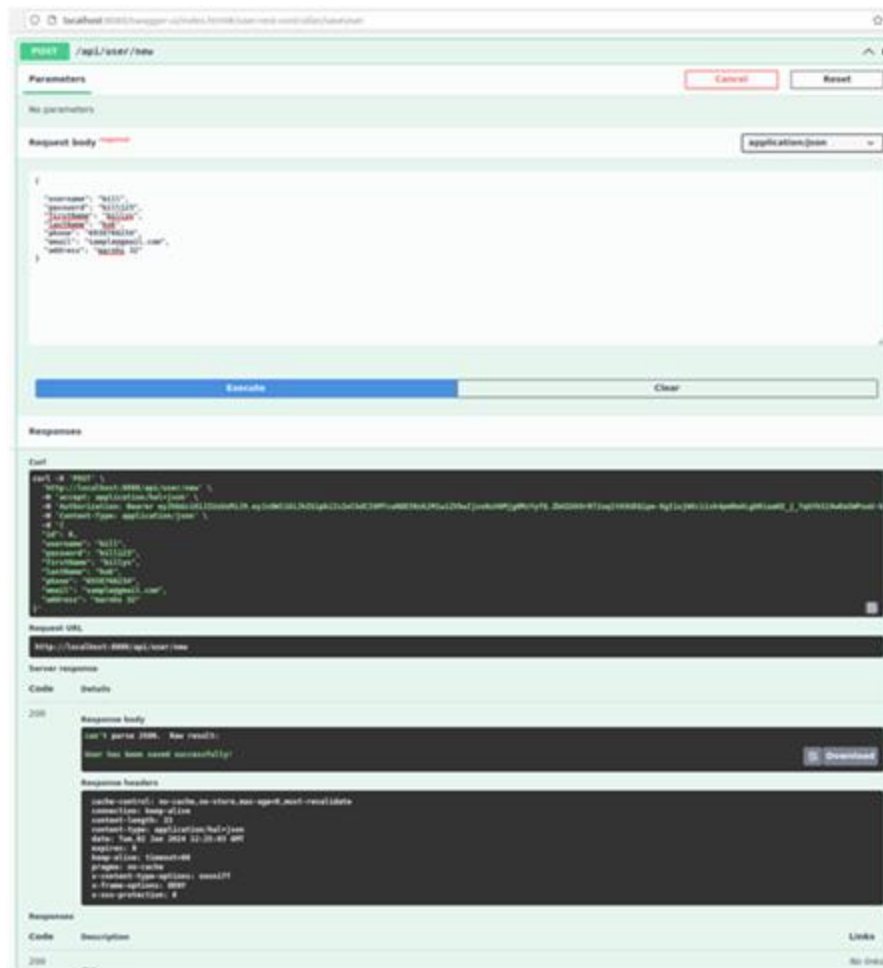
Αρχικά συνδεόμαστε σαν admin με την παρακάτω μέθοδο του AuthController που αφορά την αυθεντικοποίηση χρηστών



Τοποθετώντας το bear token που μας επιστρέφει ο AuthController στο πλαίσιο “Authorize” μπορούμε πλέον να εκτελέσουμε τις μεθόδους του αντίστοιχου ρόλου μας (admin)



Η παρακάτω μέθοδος του RestController αφορά την προσθήκη νέων χρηστών



Αν ξαναδημιουργήσουμε τον ίδιο χρήστη (βάζοντας το ίδιο email ή username) :

Η παρακάτω μέθοδος του RestController αφορά τη διαγραφή ενός συγκεκριμένου χρήστη

The image is a screenshot of the Swagger UI for a REST API. The main heading is **DELETE /api/user/{user_id}**. Below this, the **Parameters** section shows a single parameter: **user_id** (integer(11)) with a value of **254**. There are **Execute** and **Clear** buttons. The **Responses** section shows a **200** status code with a **Response body** containing a JSON array of two user objects. The first object has an ID of 252, and the second has an ID of 253. Below the response body, the **Response headers** are listed, including **cache-control**, **connection**, **content-type**, **date**, **expires**, **keep-alive**, **pragma**, **transfer-encoding**, **x-content-type-options**, **x-frame-options**, and **x-xss-protection**. At the bottom, there is a **Responses** table with a single row for status code **200** with the description **OK**.

```
DELETE /api/user/{user_id}
```

Parameters

Name	Description
user_id	integer(11)

Execute Clear

Responses

200

Response body

```
{
  "id": 252,
  "username": "admin",
  "password": "2a128584a5875a11376b0f760ad4b047e7a2a201e0b1a0c56",
  "firstName": "Pavlos",
  "lastName": "Nikolaou",
  "phone": "6942033234",
  "email": "pavlosnikolaou@gmail.com",
  "address": "Kipos 44"
},
{
  "id": 253,
  "username": "billi",
  "password": "2a128584a5875a11376b0f760ad4b047e7a2a201e0b1a0c56",
  "firstName": "Billi",
  "lastName": "Kuk",
  "phone": "6930766234",
  "email": "billi@gmail.com",
  "address": "Kardis 32"
}
```

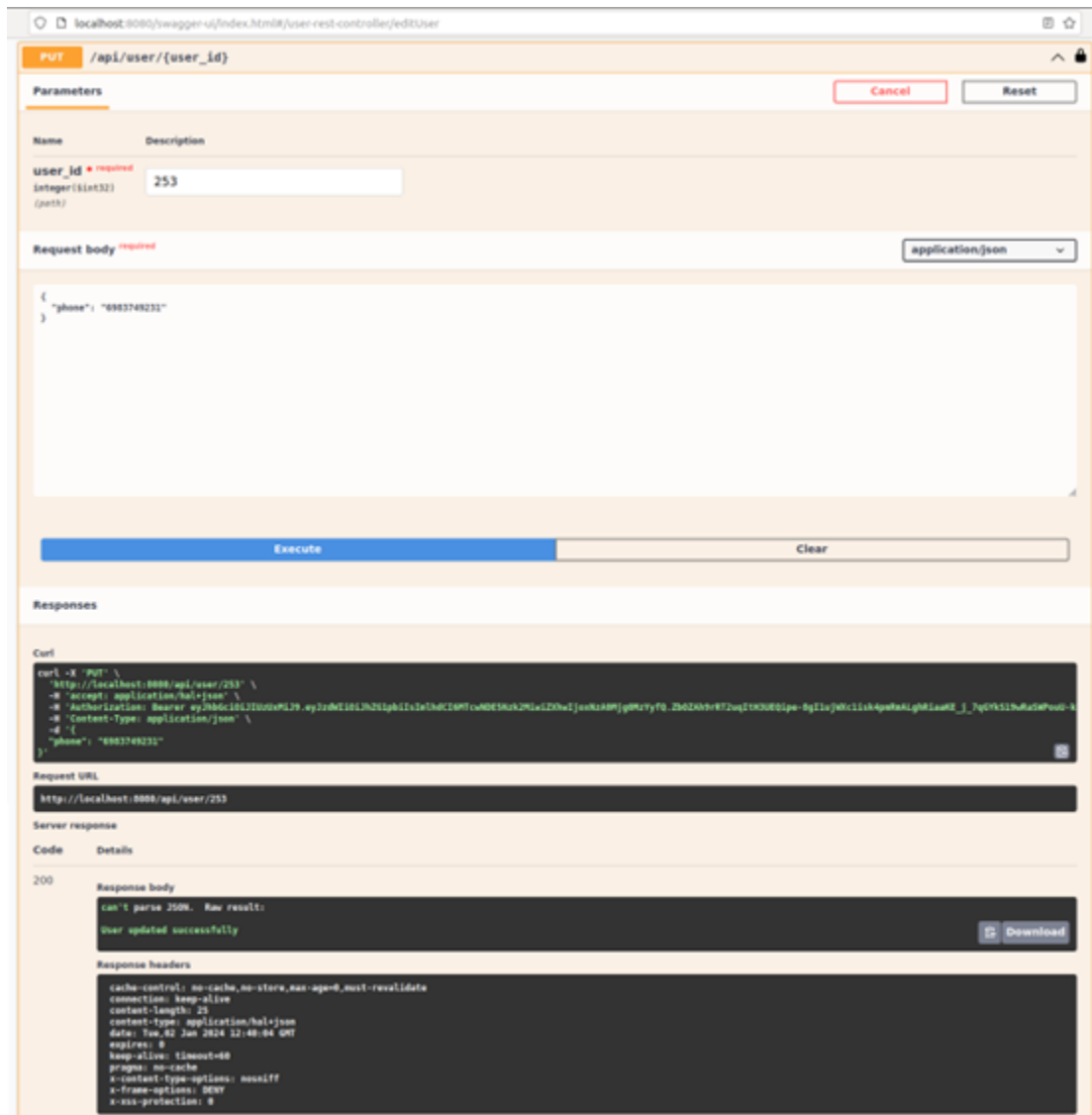
Response headers

```
cache-control: no-cache, no-store, max-age=0, must-revalidate
connection: keep-alive
content-type: application/hal+json
date: Tue, 42 Jan 2024 12:34:13 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

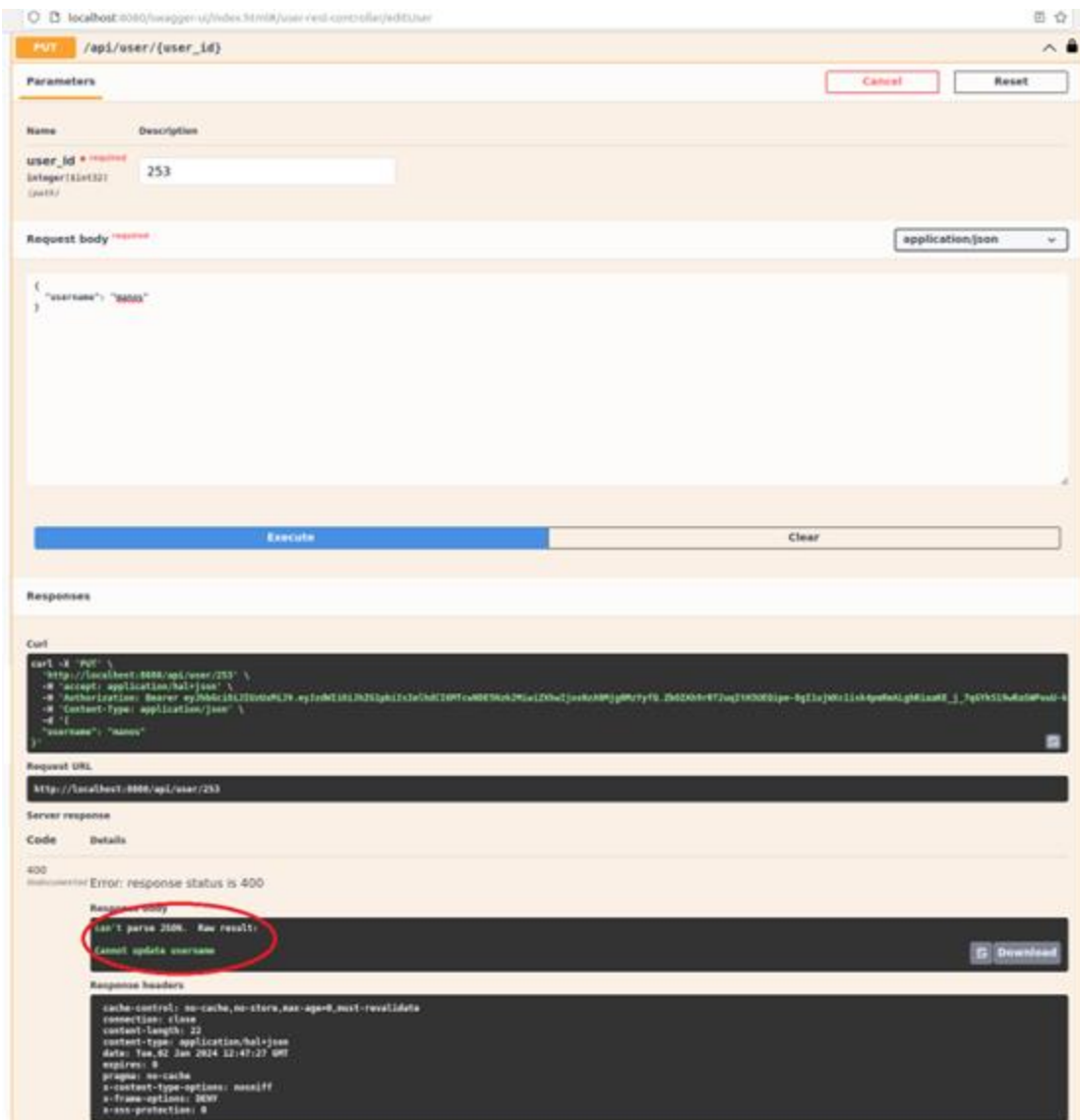
Responses

Code	Description	Links
200	OK	No links

Η παρακάτω μέθοδος του RestController αφορά την τροποποίηση των στοιχείων ενός συγκεκριμένου χρήστη. Επιτρέπεται η τροποποίηση μόνο του αριθμού τηλεφώνου, της διεύθυνσης κατοικίας και του email

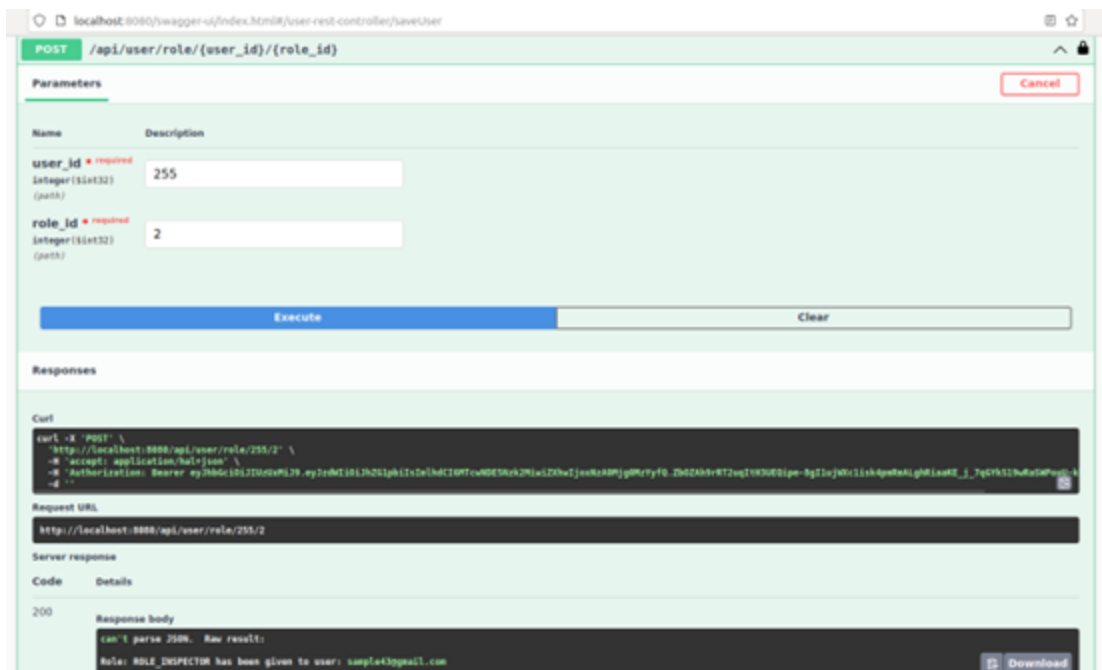


Εδώ π.χ. που προσπαθούμε να τροποποιήσουμε το username, δεν μας το επιτρέπει.

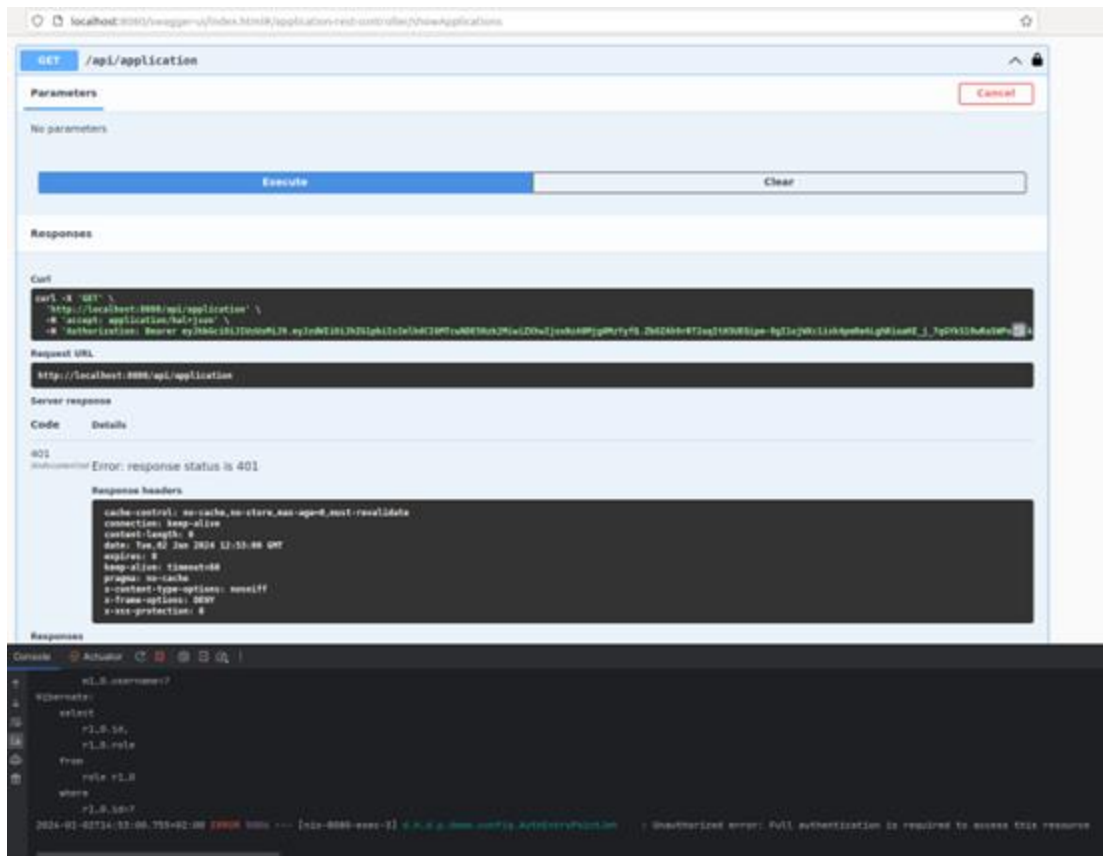


Η παρακάτω μέθοδος του RestController αφορά την ανάθεση ενός ρόλου σε έναν συγκεκριμένο χρήστη. (όσον αφορά το `role_id` έχουμε 1 = `ROLE_FARMER`, 2 = `ROLE_INSPECTOR`, 3 = `ROLE_ADMIN`)

[illegible]

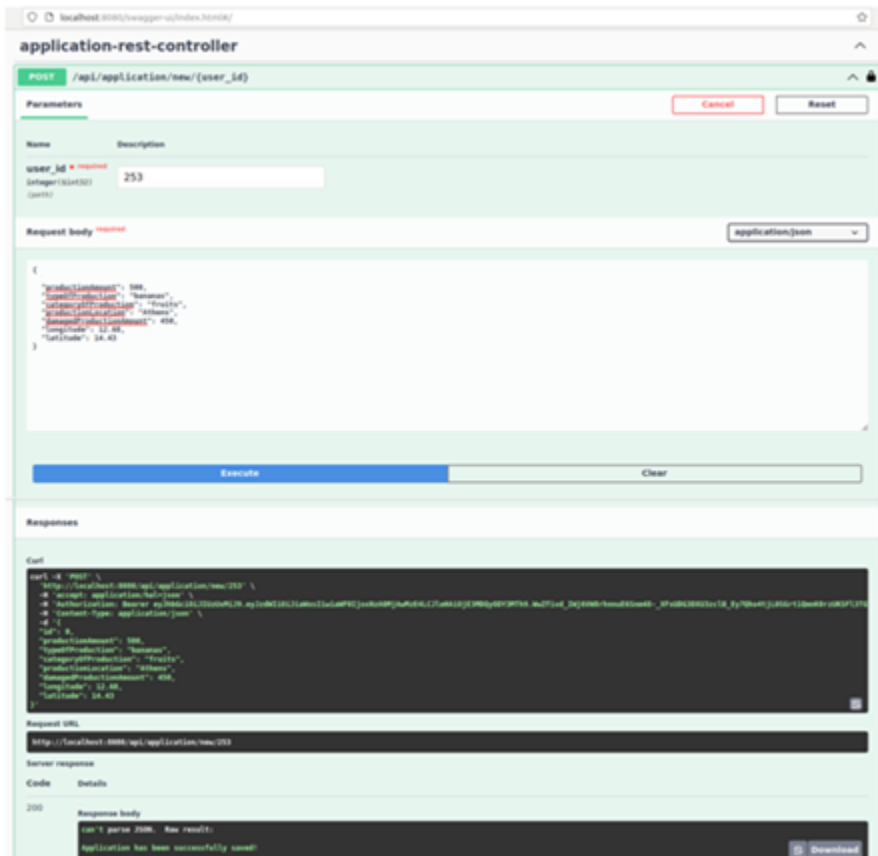


Επειδή δεν έχουμε συνδεθεί σαν inspector αλλά σαν admin δεν έχουμε πρόσβαση στην εξής μέθοδο



Το ίδιο ισχύει και για τις μεθόδους που μπορεί να εκτελέσει μόνο ο farmer. Έχοντας κάνει πλέον login ως farmer (με username bill και password bill123), μπορούμε να εκτελέσουμε τις ακόλουθες μεθόδους:

Η παρακάτω μέθοδος του RestController αφορά τη δημιουργία ενός application από έναν συγκεκριμένο χρήστη



Δεν επιτρέπεται να έχουμε συμπληρωμένα από πριν τα attributes decision και compensationAmount όταν δημιουργούμε νέο application

POST

/api/application/new/{user_id}

Parameters

Cancel

Reset

Name	Description
user_id required	
Integer (32 bit)	253
(path)	

Request body required

application/json

```
{  "productionAmount": 400,  "typeOfProduction": "apples",  "categoryOfProduction": "fruits",  "productionLocation": "Athens",  "damagedProductionAmount": 200,  "decision": "Approved",  "longitude": 15.4,  "latitude": 12.8,  "compensationAmount": 5000}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  "http://localhost:8080/api/application/new/253" \  -H 'accept: application/json' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWQ6IiwiaWF0IjoxNzE5MjY4MjY0LCJ1aWQiOiJ1b3RlbnR1eSIsImF1dGUiOiJ1b3RlbnR1eSIsIm5hbWUiOiJ1b3RlbnR1eSIsInVzZXJuYXRzIjoiYXNjaWkiLCJpYXNzLnR5cGU6Ij09eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9' \  -H 'Content-Type: application/json' \  -d '{  "productionAmount": 400,  "typeOfProduction": "apples",  "categoryOfProduction": "fruits",  "productionLocation": "Athens",  "damagedProductionAmount": 200,  "decision": "Approved",  "longitude": 15.4,  "latitude": 12.8,  "compensationAmount": 5000}
```

Request URL

http://localhost:8080/api/application/new/253

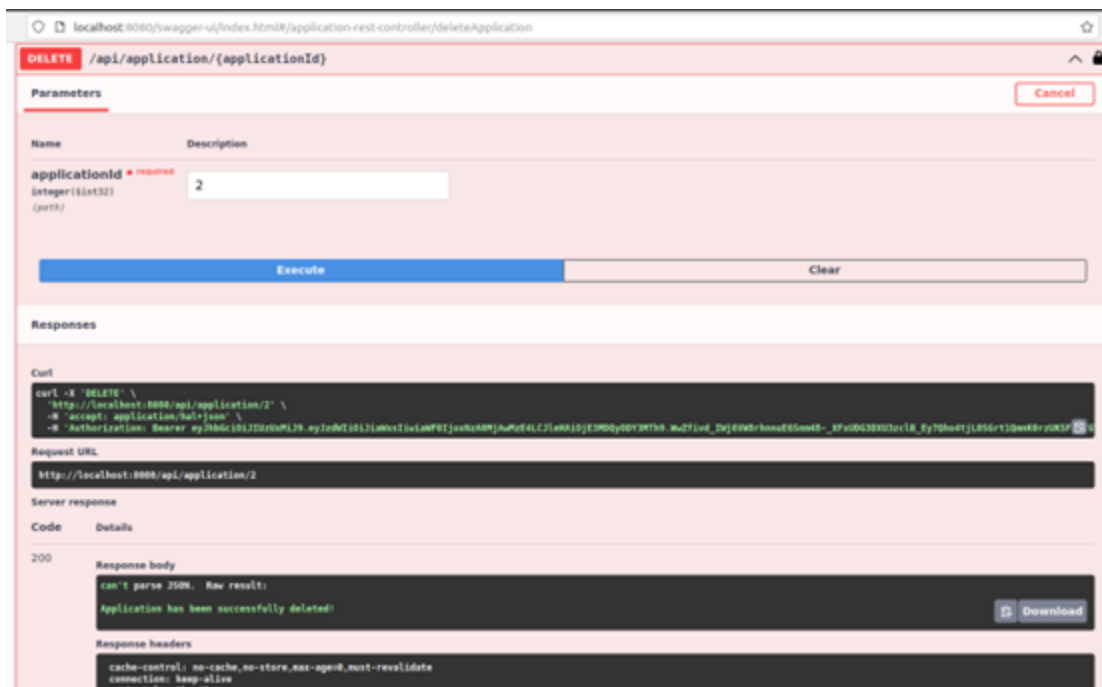
Server response

Code	Details
200	<div>Response body</div> <div>can't parse JSON. Raw result: Fields decision and compensationAmount should not be filled:</div> <div>Download</div>

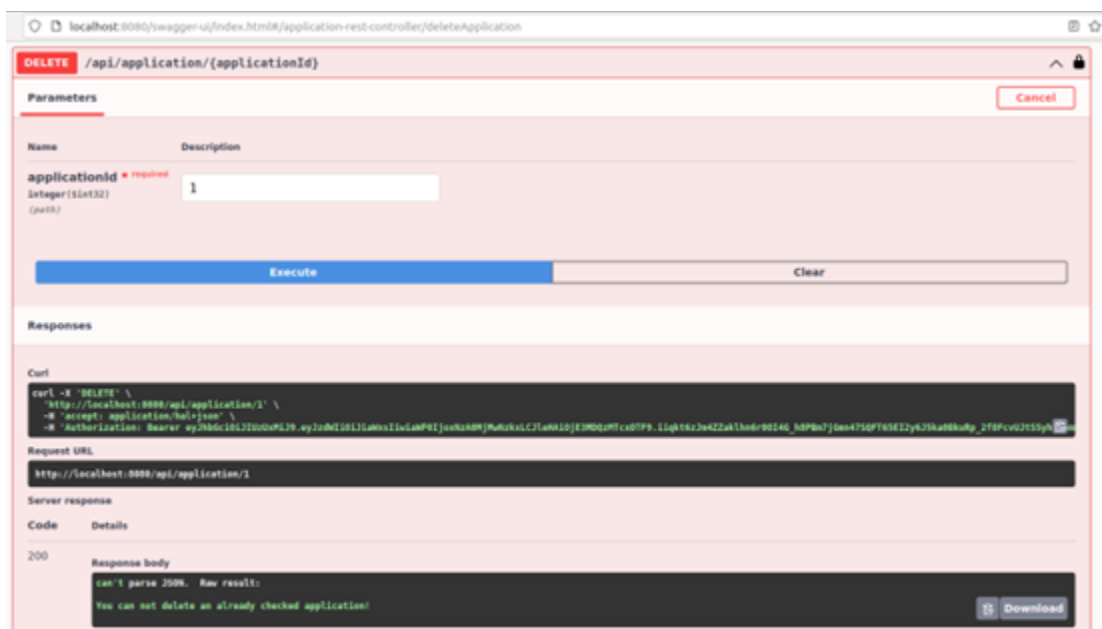
Ούτε μπορούμε να έχουμε categoryOfProduction κάτι άλλο εκτός από “fruits” ή “vegetables”

[illegible]

Η παρακάτω μέθοδος του RestController επιστρέφει όλα τα applications ενός συγκεκριμένου farmer

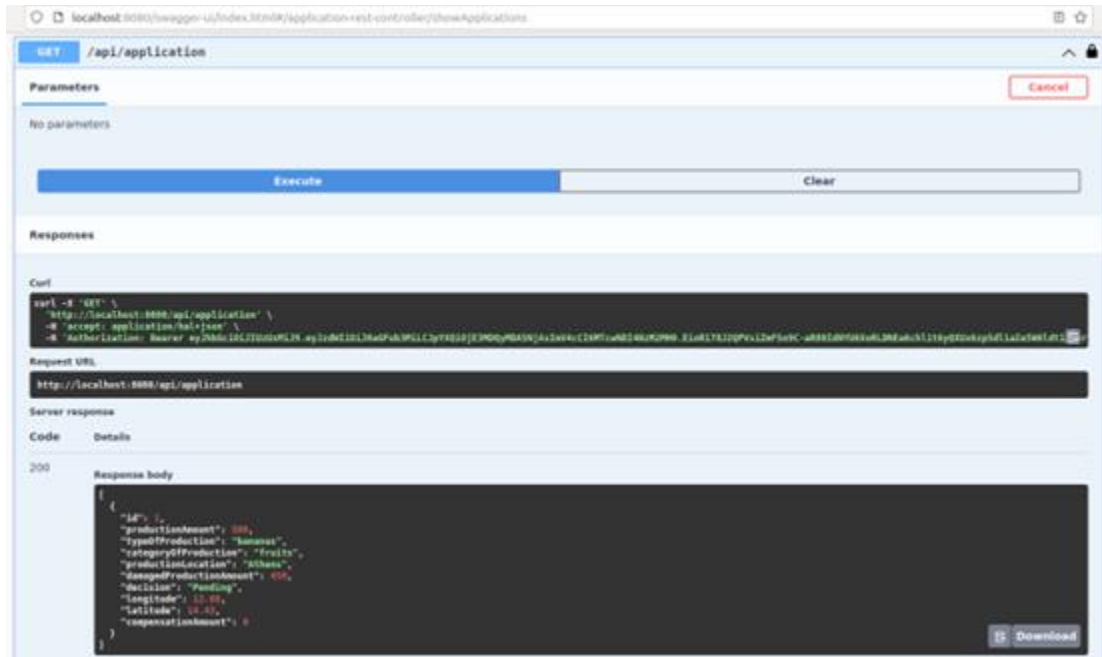


Η διαγραφή μπορεί να πραγματοποιηθεί μόνο αν δεν έχει γίνει ακόμα Approved ή Denied από κάποιον inspector

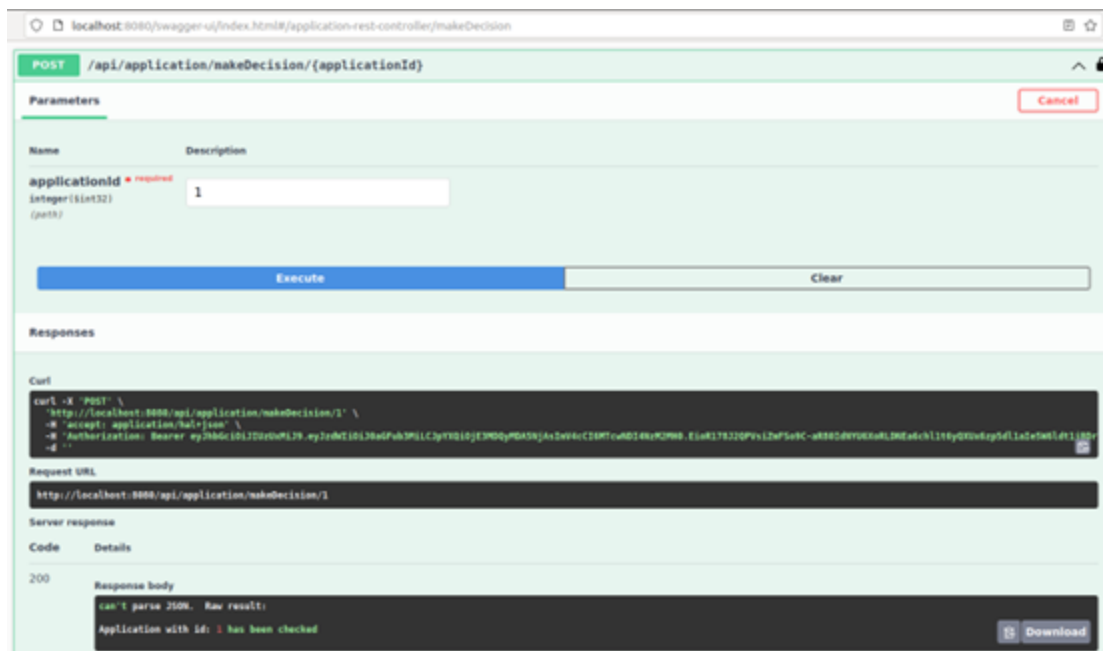


Έχοντας κάνει πλέον login ως inspector (με username thanos και password thanos123), μπορούμε να εκτελέσουμε τις ακόλουθες μεθόδους:

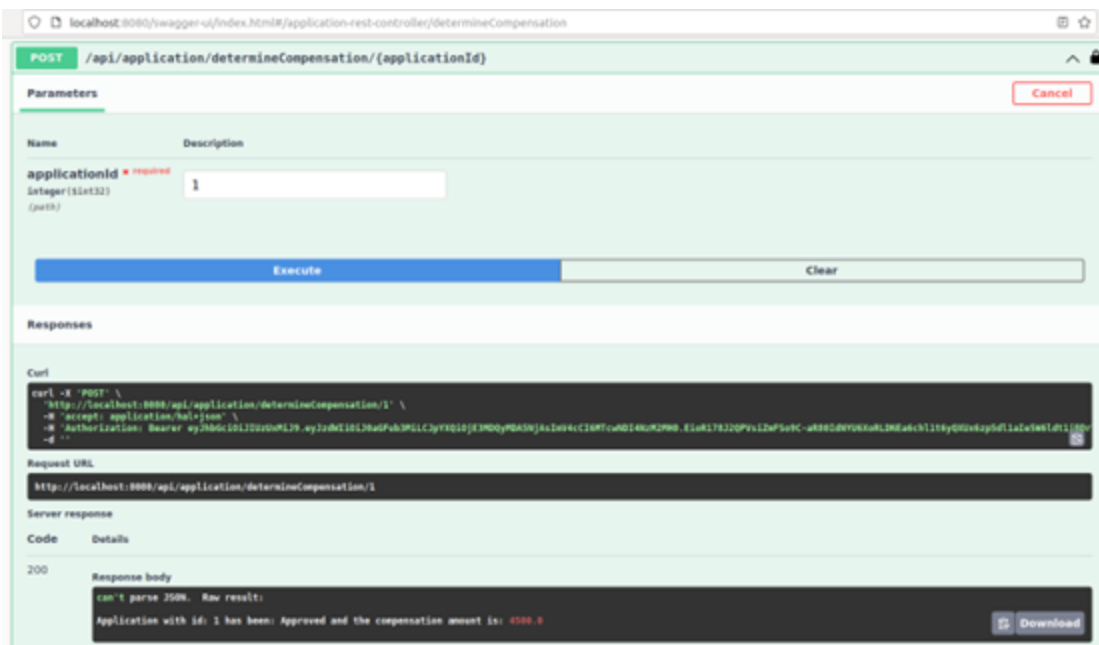
Η παρακάτω μέθοδος του RestController επιστρέφει όλα τα applications όλων των farmer τα οποία έχουν decision “Pending”



Η παρακάτω μέθοδος του RestController αφορά την έγκριση ή μη της αίτησης αποζημίωσης



Αν το ποσοστό $\text{damagedProductionAmount} / \text{productionAmount}$ είναι μεγαλύτερο του 40%, τότε η αίτηση εγκρίνεται και το $\text{compensationAmount}$ καθορίζεται με 10 ευρώ για κάθε κιλό κατεστραμμένης παραγωγής φρούτων ή με 20 ευρώ για κάθε κιλό κατεστραμμένης παραγωγής λαχανικών.



Αν το ποσοστό `damagedProductionAmount / productionAmount` είναι μικρότερο ή ίσο του 40%, τότε η αίτηση απορρίπτεται και το `compensationAmount` είναι 0.

POST /api/application/determineCompensation/{applicationId}

Parameters

Name	Description
applicationId * required <small>Integer (Sint32)</small> <small>(path)</small>	<input type="text" value="3"/>

Execute Clear

Responses

```
Curl
curl -X "POST" \
  http://localhost:8080/api/application/determineCompensation/3 \
  -H "accept: application/json" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTQwLjE0LThlZmFuY3M1CjByYXNpdjEPM0dyPDEBMjpsZW4iOjEPMTcwMDI4eGg5N3B_6zaSeDp5tK1czH4baas_E3YSGcmRr-6201KXNlcxcwbu_Vz4WSPb-X3fZispy7B_Sap0001Q3" \
  -d "{}"
```

Request URL
http://localhost:8080/api/application/determineCompensation/3

Server response

Code	Details
200	Response body <pre>{ "can": true, "percentage": 2500, "rawResult": { "Application with id: 3 has been Denied and the compensation amount is: 0.0" } }</pre> Download

Εγχειρίδιο χρήσης Frontend εφαρμογής:

Με την μετάβαση στον σύνδεσμο localhost:5173 μεταφερόμαστε στην συγκεκριμένη σελίδα.



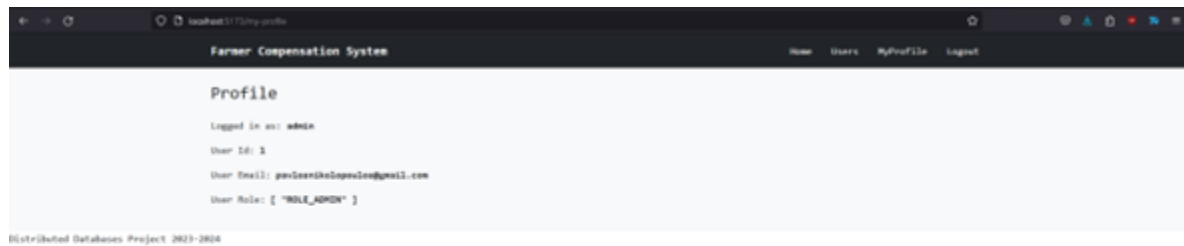
Στην συνέχεια τοποθετούμε τα στοιχεία του admin που υπάρχει ήδη στην βάση (username: admin, password: panlosnikoloroulos44) και πατάμε login.



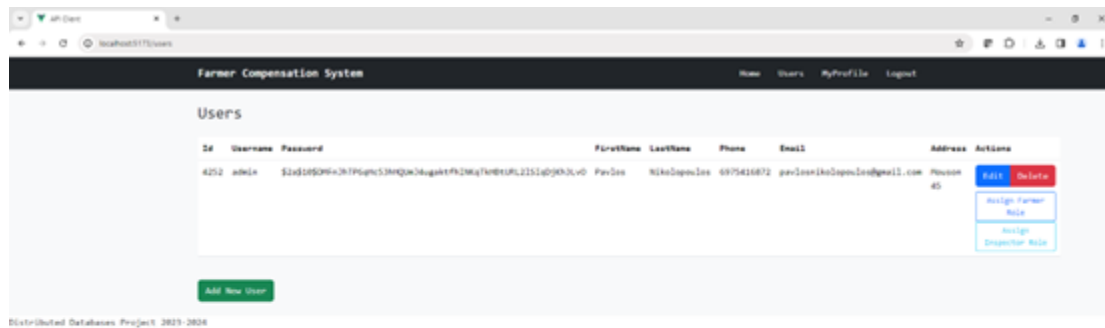
Αφού πατήσουμε login θα εμφανιστούν στο πάνω δεξί μέρος της οθόνης τέσσερα tabs. Το ένα tab λέγεται Users όπου από εκεί χειρίζεται ο admin όλες τις λειτουργίες του, ένα tab MyProfile όπου περιέχει συνοπτικά τα στοιχεία του συνδεδεμένου χρήστη (username,id,email,role) ενώ τέλος υπάρχει ένα tab Logout όπου όταν το πατήσει ο χρήστης θα του εμφανιστεί ένα κουμπί για να κάνει Logout και ένα tab όπου σε περίπτωση που το πατήσει ο χρήστης θα επιστρέψει στην αρχική οθόνη.



MyProfile tab για τον admin:



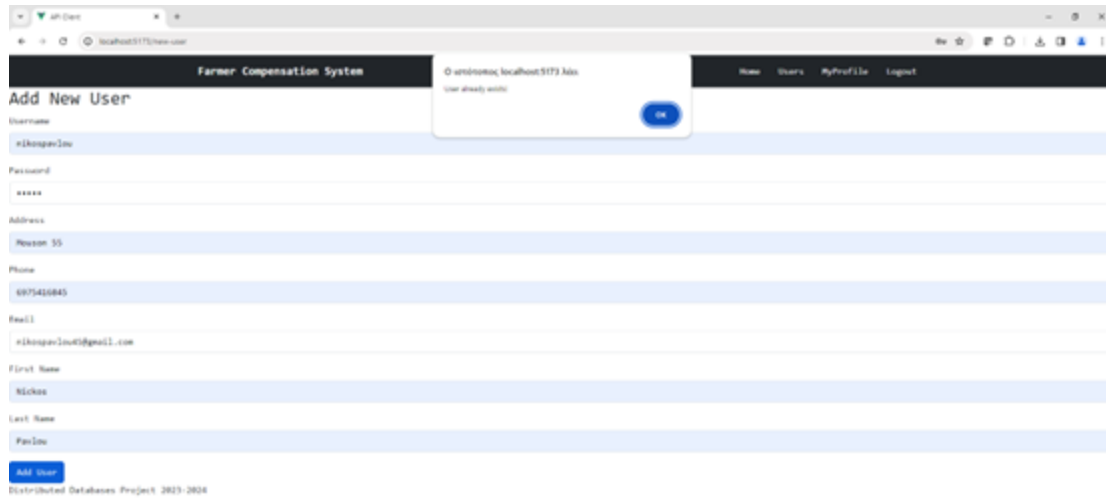
Στο tab Users ο admin θα μπορεί να προσθέτει νέο χρήστη, να αλλάζει κάποια στοιχεία του, να αναθέτει ρόλο σε ένα χρήστη και να διαγράφει έναν χρήστη όπως φαίνεται και από τα κουμπιά παρακάτω.



Μόλις ο χρήστης πατήσει το κουμπί Add New User θα του εμφανιστεί μια φόρμα για την συμπλήρωση των στοιχείων του νέου χρήστη. Αν ο χρήστης δεν υπάρχει (δηλαδή δεν υπάρχει χρήστης με το ίδιο username ή password) τότε εμφανίζεται μήνυμα επιτυχίας και ο χρήστης πατάει Ok και επιστρέφεται αυτόματα πίσω στην σελίδα με τους Users. Σε περίπτωση που υπάρχει χρήστης με το ίδιο username ή email εμφανίζεται μήνυμα σφάλματος και ο χρήστης πρέπει να διορθώσει τα αντίστοιχα πεδία και να προσθέσει τον χρήστη.

The screenshot shows a web browser window with the URL 'localhost:5173/new-user'. The page title is 'Farmer Compensation System'. A dark navigation bar at the top contains the title and a user profile icon with the text 'Ο επόνητος: localhost:5173 kds'. To the right of the navigation bar are links for 'Home', 'Users', 'MyProfile', and 'Logout'. The main content area is titled 'Add New User'. It contains a form with the following fields: Username (filled with 'nikospavlou'), Password (filled with '****'), Address (filled with 'Rouson 46'), Phone (filled with '6075416876'), Email (filled with 'nikospavlou@gmail.com'), First Name (filled with 'Nikos'), and Last Name (filled with 'Pavlos'). At the bottom of the form is a blue 'Add User' button. A white modal dialog box is open in the center of the screen, displaying the message 'Ο επόνητος: localhost:5173 kds' and 'user has been saved successfully!'. The dialog has a blue 'Ok' button.

Εμφανίζεται μήνυμα σφάλματος καθώς υπάρχει χρήστης με username nikospavlou, αν ο χρήστης όμως το διορθώσει ο χρήστης προστίθεται κανονικά.



Farmer Compensation System

0 admin@localhost:5173 http://localhost:5173/new-user

Home Users MyProfile Logout

Add New User

Username
nikos@evlmsi

Password

Address
Piraeus 55

Phone
6975410845

Email
nikos@evlmsi@gmail.com

First Name
Nikos

Last Name
Pavlou

[Add User](#)

Distributed Databases Project 2023-2024

Farmer Compensation System

Home Users MyProfile Logout

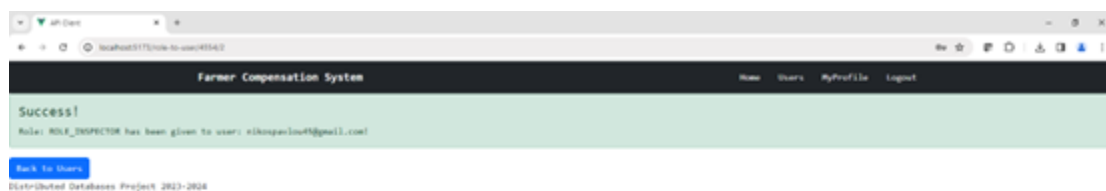
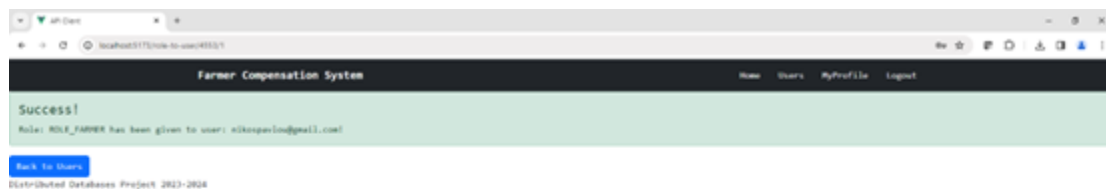
Users

Id	Username	Password	FirstName	LastName	Phone	Email	Address	Actions
4252	admin	\$2a\$08\$W4n7P5qnc53Hq5w3kugak7h2Wq7v88uRzJ15i8j9K3LcD	Pavlos	Nikolaopoulos	6975410872	pavlos@nikolaopoulos@gmail.com	Piraeus 45	Edit Delete Assign Farmer Role Assign Inspector Role
4953	nikos@evlmsi	\$2a\$08\$7r8e8Z7Acf8h3uMPr3u7y5C7w10L4w8t8t8u8e8t8q7C	Nikos	Pavlou	6975410876	nikos@evlmsi@gmail.com	Piraeus 46	Edit Delete Assign Farmer Role Assign Inspector Role
4954	nikos@evlmsi	\$2a\$08\$38v7M7P4u7r8u8q8dP7K78u7y7K7r887p7r8u788K312D	Nikos	Pavlou	6975410845	nikos@evlmsi@gmail.com	Piraeus 55	Edit Delete Assign Farmer Role Assign Inspector Role

[Add New User](#)

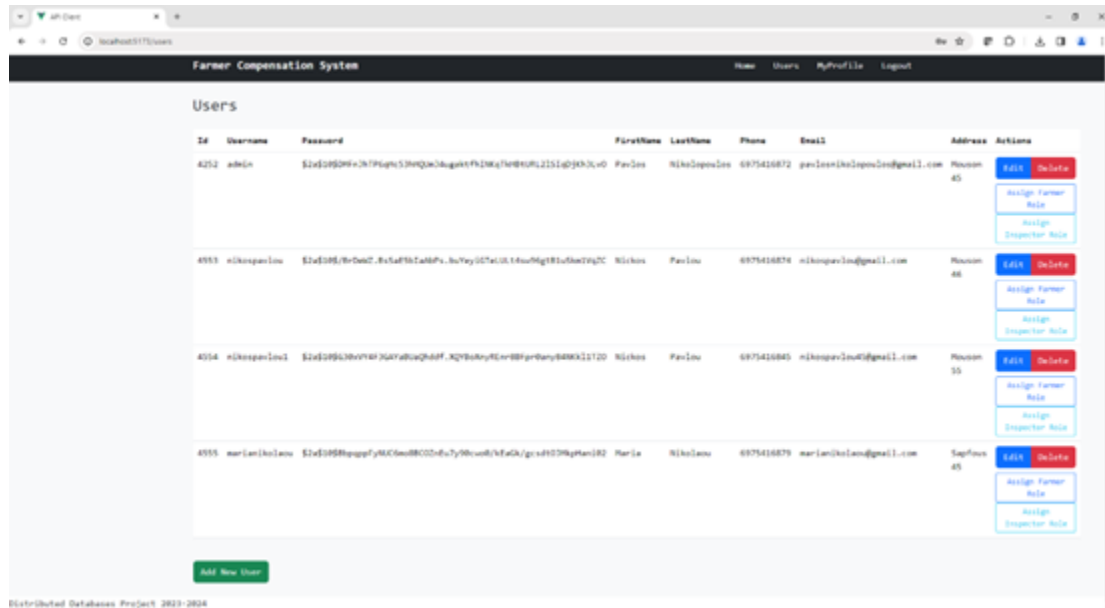
Distributed Databases Project 2023-2024

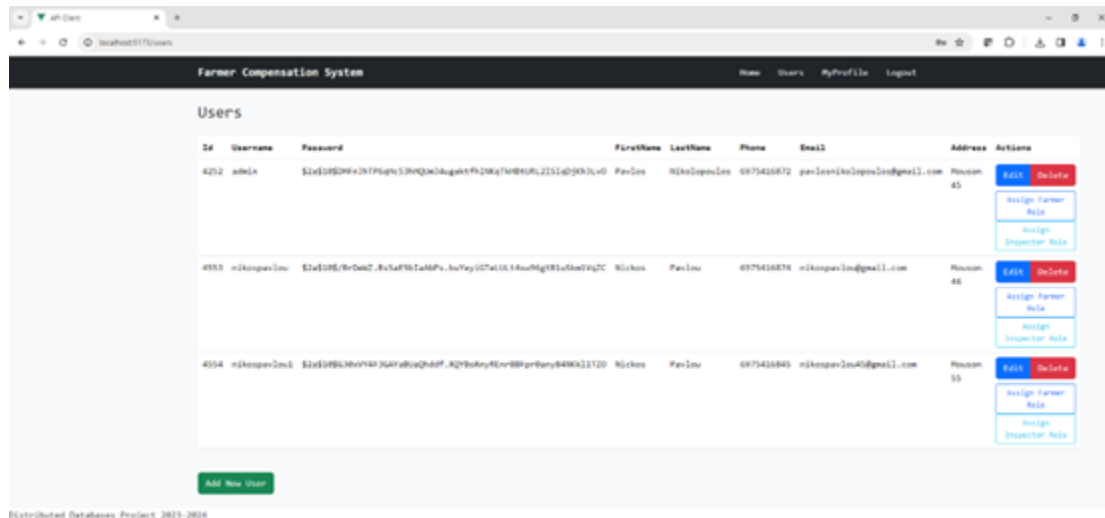
Με το πάτημα των κουμπιών Assign Farmer Role και Assign Inspector Role ο admin μπορεί να αναθέσει στον χρήστη τους αντίστοιχους ρόλους. Εμφανίζεται συγκεκριμένα ένα μήνυμα που εμφανίζει ποιος ρόλος ανατέθηκε σε ποιον χρήστη.



Ο admin πατώντας το αντίστοιχο κουμπί Delete μπορεί να διαγράψει έναν χρήστη. Μόλις πατήσει το κουμπί θα εμφανιστεί μια σελίδα που τον ρωτάει αν όντως θέλει να διαγράψει τον

χρήστη ή να επιστρέψει πίσω. Ενδεικτικά διαγράφουμε τον χρήστη με username marianikolaou.

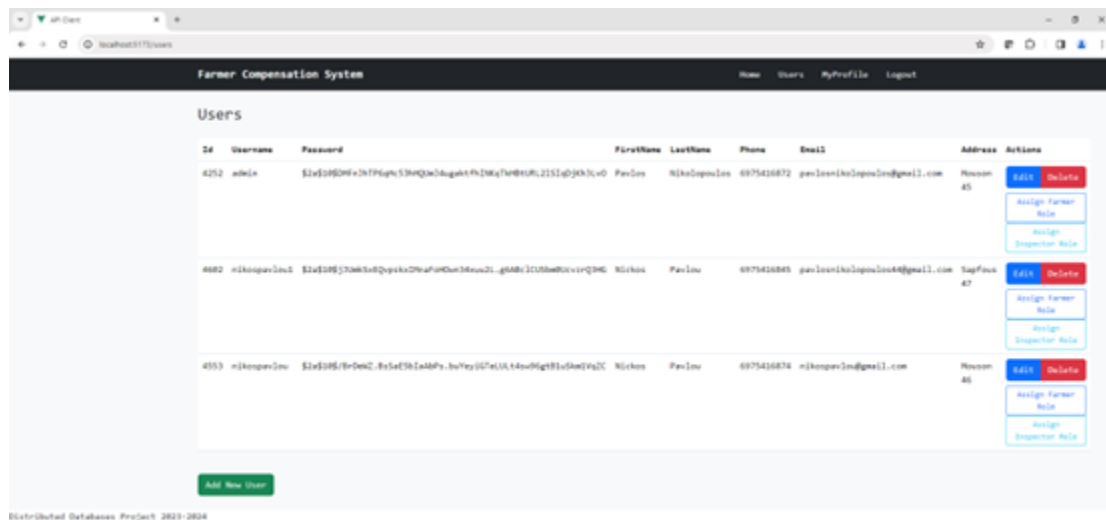
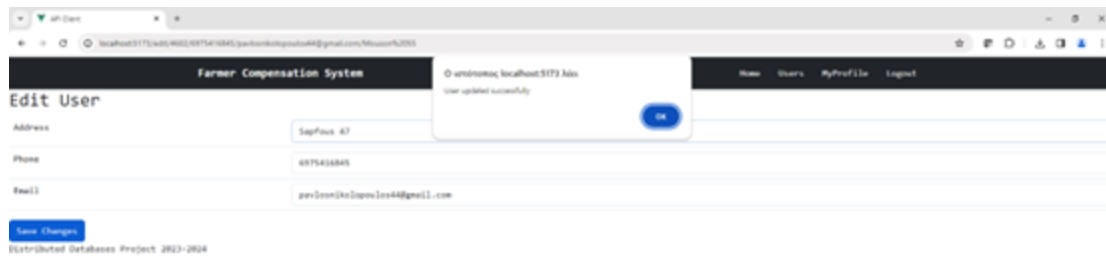




Τέλος, ο admin μπορεί να αλλάξει κάποια στοιχεία των χρηστών και αυτά είναι το τηλέφωνο, η διεύθυνση και το email. Ο χρήστης μπορεί να αλλάξει ένα στοιχείο ή όλα από τα 3. Σε περίπτωση που βάλει email που υπάρχει ήδη εκτός από το δικό του (δηλαδή email άλλου χρήστη) τότε εμφανίζεται αντίστοιχο μήνυμα.

The screenshot shows a web browser window with the title 'Farmer Compensation System'. The browser's address bar displays a local IP address: 'localhost:5172'. The page has a dark header with the system name and navigation links: 'Home', 'Users', 'MyProfile', and 'Logout'. The main content area is titled 'Edit User' and contains three input fields: 'Address' (with the value 'Sapfous 43'), 'Phone', and 'Email'. Below these fields is a blue button labeled 'Save Changes'. At the bottom left of the page, there is a footer that reads 'Distributed Databases Project 2023-2024'.

Μόλις ο χρήστης πατήσει Save Changes εμφανίζονται τα στοιχεία των υπόλοιπων πεδίων που δεν έχει αλλάξει και έχουν μείνει ίδια και μήνυμα επιτυχίας. Στην συνέχεια μόλις ο χρήστης (admin) πατήσει Ok μεταφέρεται στην σελίδα των Users.

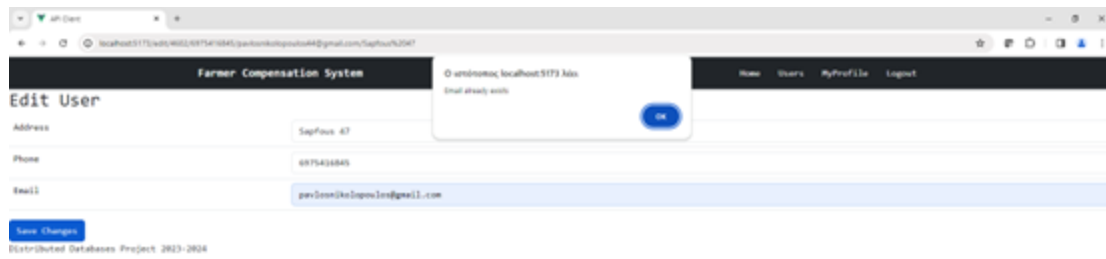


Σε περίπτωση που ο χρήστης (admin) επιχειρήσει να αλλάξει το email και να βάλει ένα email

που έχει ένας άλλος χρήστης τότε εμφανίζεται κατάλληλο μήνυμα. Σε αυτήν την περίπτωση ο χρήστης πατάει Ok και θα πρέπει να αλλάξει το email.



The screenshot shows a web browser window with the URL `localhost:5172/edit/4002/6875476845/pavloskokkinogennas44@gmail.com/5ap8ou%2067`. The page title is "Farmer Compensation System". The main heading is "Edit User". There are three input fields: "Address", "Phone", and "Email". The "Email" field contains the text `pavloskokkinogennas44@gmail.com` and is highlighted with a blue border. Below the input fields is a blue button labeled "Save Changes". At the bottom left, there is a footer that reads "Distributed Databases Project 2023-2024".



Για την δημιουργία μιας νέας αίτησης ο Farmer πατάει το κουμπί Add New Application όπου του εμφανίζεται μια φόρμα που συμπληρώνει τα αντίστοιχα πεδία. Εφόσον, η συμπλήρωση της αίτησης ήταν επιτυχής εμφανίζεται κατάλληλο μήνυμα επιτυχίας και ο Farmer ανακατευθύνεται πίσω στο tab MyApplications.

The screenshot shows a web browser window with the URL 'localhost:5173/new-application'. The page title is 'Farmer Compensation System'. The main heading is 'Add New Application'. The form fields and their values are as follows:

Field	Value
Production Amount	500
Type Of Production	Apples
Category Of Production	Fruits
Production Location	Macedonia
Damaged Production Amount	400
Longitude	52
Latitude	22

At the bottom left of the form, there is a blue button labeled 'Add Application'. Below the button, the text 'Distributed Database Project 2023-2024' is visible.

Farmer Compensation System

0 semroutos,localhost:5173/homeApplication has been successfully saved

HomeMyApplicationsMyProfileLogout

Add New Application

Production Amount

500

Type of Production

Apples

Category of Production

Fruits

Production Location

RecedonLa

Damaged Production Amount

450

Longitude

12

Latitude

22

Add Application

Distributed Database Project 2023-2024

Farmer Compensation System

HomeMyApplicationsMyProfileLogout

Applications

Application Id	Production Amount	Type of Production	Category of Production	Production Location	Damaged Production Amount	Decision	Longitude	Latitude	Compensation Amount	
0002	500	Apples	Fruits	RecedonLa	450	Pending	12	22	0	Delete

Add New Application

Distributed Database Project 2023-2024

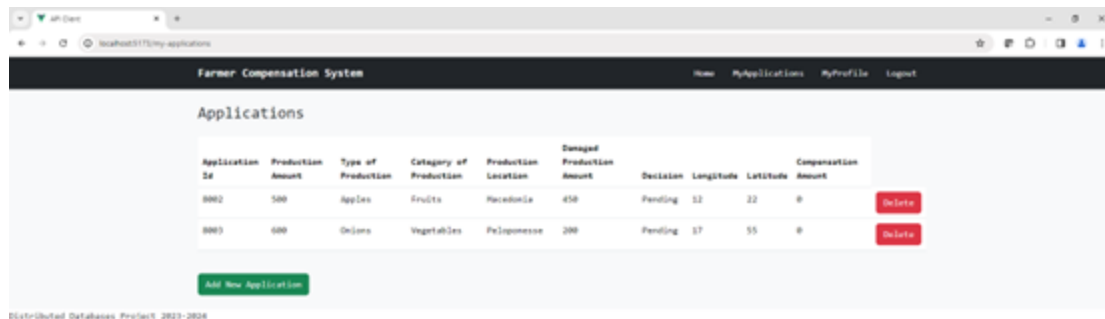
Σε περίπτωση που ο Farmer βάλει σαν ποσότητα ζημιωμένης παραγωγής μεγαλύτερη από την παραχθείσα παραγωγή εμφανίζεται μήνυμα σφάλματος και ο χρήστης θα πρέπει να το διορθώσει. Αντίστοιχα, αν ο Farmer βάλει στο Category of Production

άλλες τιμές πλην των τιμών Fruits ή Vegetables εμφανίζεται επίσης μήνυμα σφάλματος.

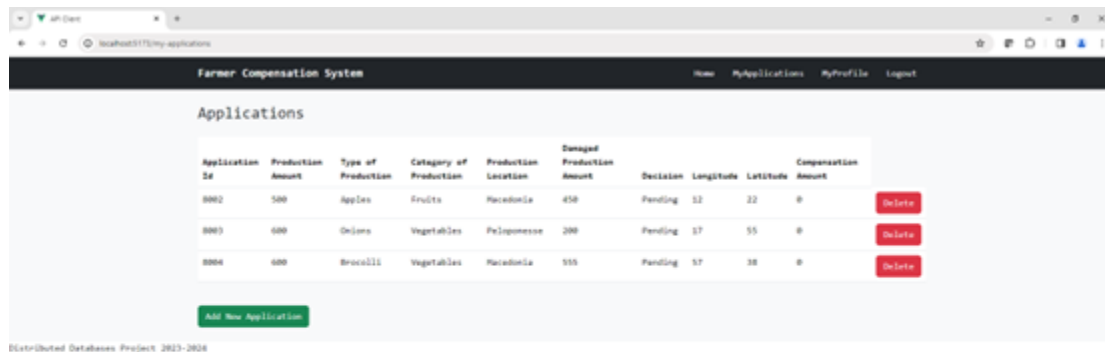
The screenshot shows a web browser window with the URL 'localhost:5175/new-application'. The page title is 'Farmer Compensation System'. The navigation bar includes links for 'Home', 'MyApplications', 'MyProfile', and 'Logout'. The main heading is 'Add New Application'. The form contains the following fields and values:

- Production Amount: 600
- Type of Production: Onions
- Category of Production: Vegetables
- Production Location: Fesposmesse
- Damaged Production Amount: 750
- Longitude: 17
- Latitude: 35

A blue button labeled 'Add Application' is located at the bottom left of the form. Below the button, the text 'Distributed Database Project 2023-2024' is visible.

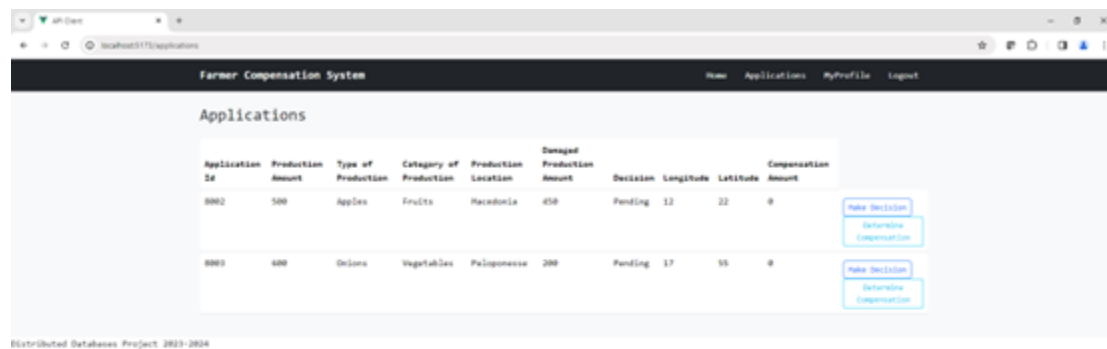


Ο χρήστης μπορεί να διαγράψει μια αίτηση που έχει κάνει. Μόλις πατήσει το κουμπί Delete εμφανίζεται κατάλληλο προειδοποιητικό μήνυμα και ο χρήστης μπορεί να επιβεβαιώσει την διαγραφή και να οριστικοποιήσει την απόφαση του ή να πατήσει Cancel και να επιστρέψει πίσω στο tab MyApplications. Ενδεικτικά φαίνονται εικόνες για την διαγραφή της αίτησης με id 8004.



Ο Farmer τελείωσε μπορεί να κάνει Logout πατώντας όπως ο admin το αντίστοιχο κουμπί.

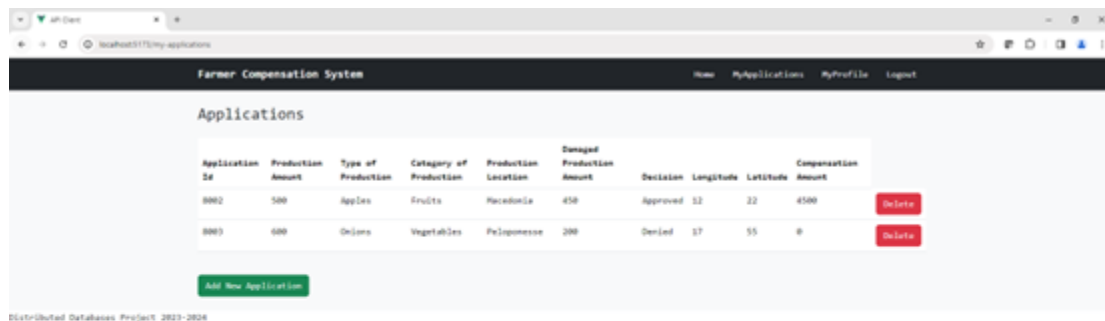
Ο Inspector στην συνέχεια τοποθετεί τα διαπιστευτήρια του και θα του εμφανιστούν 4 tabs, το home, το MyProfile, το Logout και ένα tab Applications που περιέχει τις αιτήσεις των αγροτών προς έλεγχο και αποζημίωση. Συγκεκριμένα, το tab Applications εμφανίζει τις αιτήσεις των αγροτών προς έγκριση (κατάσταση Pending) και κάθε αίτηση έχει δύο κουμπιά δίπλα της το κουμπί MakeDecision ώστε να πάρει απόφαση για την αίτηση και ένα κουμπί DetermineCompensation για να υπολογιστεί το ποσό της αποζημίωσης.



Application Id	Production Amount	Type of Production	Category of Production	Production Location	Damaged Production Amount	Decision	Longitude	Latitude	Compensation Amount
0002	500	Apples	Fruits	Macedonia	450	Pending	12	22	0
0003	400	Onions	Vegetables	Peloponnese	200	Pending	17	35	0

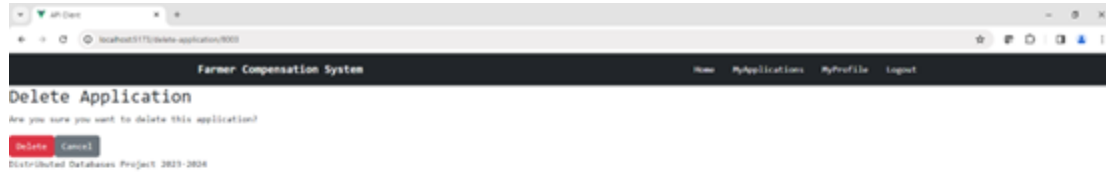
Ο Inspector θα εγκρίνει ή θα απορρίψει αιτήσεις.

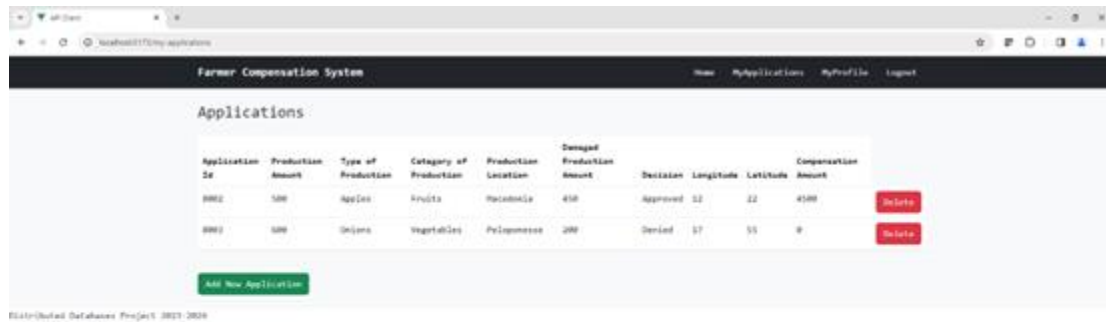
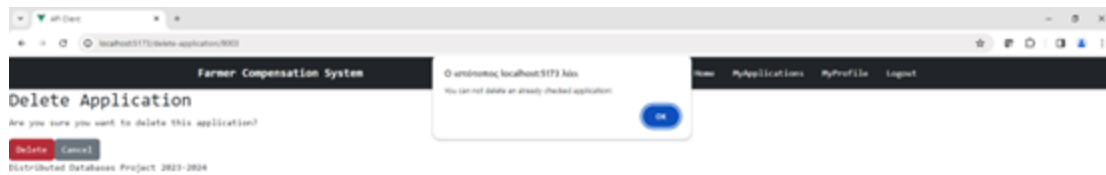
Ο Inspector τώρα μπορεί να κάνει Logout και να συνδεθεί ο Farmer για να δει τις αποφάσεις για τις αιτήσεις του.



Σε περίπτωση που ο Farmer επιχειρήσει να διαγράψει μια αίτηση που είναι ήδη ελεγμένη δεν θα

του επιτραπεί αυτή η ενέργεια και θα εμφανιστεί κατάλληλο μήνυμα. Παρακάτω ο χρήστης επιχειρεί να διαγράψει την αίτηση με id 8003.





Τέλος, ο Farmer κάνει Logout.

Deployment

[Περιγράψτε εδώ το deployment περιβάλλον σας, απο τι αποτελείται (vms, κτλ)]

Το deployment περιβάλλον έχει χρησιμοποιήσει GoogleCloud πάροχο, Azure πάροχο αλλά και τοπικά vagrant.

Ansible

Το αρχικό μας deployment έχει χρησιμοποιήσει ansible δημιουργώντας playbooks τα οποία στην αρχή θα δημιουργήσουν την βάση δεδομένων της postgres (ansible-playbookplaybooks/postgres.yaml -I <πάροχος-vm>). Επιπλέον, η θα κάνουμε deployment την δεύτερη εφαρμογή μας του backend όπου μέσω του repository μας θα εφαρμόσει ότι χρειάζεται για να τρέξουμε την εφαρμογή (ansible-playbookplaybooks/spring.yaml -I <πάροχος-vm>). Τέλος, θα τρέξουμε και την τρίτη εφαρμογή του frontend μας όπου πάλι μέσω του frontend repo μας θα εφαρμόσει το κατάλληλο σενάριο για το τρέξιμο της εφαρμογής (ansible-playbookplaybooks/vuejs.yaml -I<πάροχος-vm>).

Ansible - Docker

Σε αυτό το το deployment δημιουργήσαμε ένα ansible playbook το οποίο σηκώνει το dockercompose τις 3 εφαρμογές μας σε ένα vm (είτε με vagrant, είτε με παρόχους). Συνοπτικά, τα βήματα είναι ότι φτιάχνουμε το playbook κάνοντας install το docker και dockercompose, γίνεται gitclone το repository και έτσι θα κάνουμε start το τα Docker Compose services για να τρέξουμε τις εφαρμογές.

Kubernetes

[Περιγράψτε εδώ τις οντότητες που δημιουργήσατε στο kubernetes]

Στο project μας, δημιουργήσαμε και διαχειριστήκαμε τις οντότητες στο Kubernetes

χρησιμοποιώντας την εντολή `kubectl apply -f`. Μέσω του `kubectl` εφαρμόσαμε τις δηλώσεις των πόρων που ορίζονται μέσα από τα `yaml` αρχεία μας, τα οποία περιέχουν τις ρυθμίσεις για τις υπηρεσίες και εφαρμογές στο Kubernetes `deploy` που υλοποιήσαμε. Το αρχείο `cert-issuer.yaml` περιέχει την οντότητα `ClusterIssuer` όπου ρυθμίζει τη διαδικασία έκδοσης πιστοποιητικών. Στο φάκελο `postgres` του `k8s` φακέλου έχουμε το `deployment` την βάσης δεδομένων `postgres` και με το αρχείο `postgres-deployment.yaml` θα αναπτύξουμε την `PostgreSQL` βάση μας με την εντολή `kubectl apply -f postgres-deployment.yaml`. Το αρχείο `postgres-pvc.yaml` ουσιαστικά είναι το `persistent volume claim` που θα χρησιμοποιήσει η `PostgreSQL` για να εξασφαλίσουμε την μόνιμη αποθήκευση των δεδομένων μας στην βάση. Το `service` (`postgres-svc.yaml`) θα προσθέσει την βάση στο εσωτερικό του `Cluster` και πάλι με `Kubectl` θα ρυθμιστεί η πρόσβαση στην υπηρεσία. Από την άλλη, η `spring` εφαρμογή με το αρχείο `spring-deployment.yaml` θα ρυθμίσουμε την ανάπτυξη της `backend` εφαρμογής και εμπεριέχεται το `liveness` και `readiness probes`. Ο `Ingress` με το `spring-ingress-tls.yaml` θα ρυθμίσει την πρόσβαση της εφαρμογής μέσω `ingress` με `TLS`. Το αρχείο `spring-ingress.yaml` ορίζει τις βασικές `ingress` ρυθμίσεις. Τέλος, το `spring-svc.yaml` δημιουργεί την υπηρεσία της εφαρμογής `Spring`.

Η `frontend` εφαρμογή `vuejs` πάλι την διαχειριστήκαμε με `Kubectl` κάνοντας `kubectl apply -f` τις οντότητες μας. Στο αρχείο `vue-deployment.yaml` θα γίνει η ανάπτυξη της `frontend` εφαρμογής όπου θα διαχειριστούμε την `frontend` εφαρμογή και θα τρέξουμε την εντολή `kubectl apply -f vue-deployment.yaml`. Το αρχείο `vue-ingress-tls.yaml` καθορίζει την πρόσβαση στην εφαρμογή `Vue.js` μέσω `Ingress` με `TLS`, χρησιμοποιώντας το `Let's Encrypt` για την έκδοση πιστοποιητικών. Η εντολή `kubectl apply -f vue-ingress-tls.yaml` ρυθμίζει την ασφαλή πρόσβαση στην εφαρμογή μέσω `HTTPS`. Το αρχείο `vue-ingress.yaml` δημιουργεί βασικές ρυθμίσεις `Ingress` για την εφαρμογή `Vue.js`, επιτρέποντας την πρόσβαση μέσω `HTTP`. Το αρχείο `vue-svc.yaml` δημιουργεί μια υπηρεσία που εκθέτει την εφαρμογή `Vue.js` στο εσωτερικό του `cluster` `Kubernetes`. Η εντολή `kubectl apply -f vue-svc.yaml` ρυθμίζει αυτό το `service`, επιτρέποντας την επικοινωνία μεταξύ των `Pods` και την εξωτερική πρόσβαση μέσω των `Ingress` ρυθμίσεων. Σαν δοκιμή κάναμε με `kubectl port-forward` ένα `testing` του `deployment` για το `spring` (`spring-deployment`). Επίσης, μέσω του `No-IP` όπου είναι ένα `ddns` (`dynamicdns`) κάνουμε `expose` το `frontend`.

CI/CD

Δημιουργήσαμε ένα Jenkins pipeline (Jenkinsfile) όπου θα εκτελέσει συγκεκριμένα stages για την αυτοματοποίηση της εφαρμογής μας. Αρχικά, κάνουμε checkout τον πηγαίο κώδικα από το branch devops-backend του repository μας στο Github. Μετά θα εκτελέσουμε δοκιμές με maven command ./mvnwtest για τα tests μας. Θα ξεκινήσουμε ένα άλλο job για να γίνει run η ansible. Ακόμη, θα κάνουμε Install τα ansible prerequisites όπως κάποια απαραίτητα dependencies χρησιμοποιώντας ansible-galaxy για το role geerlingguy.postgresql. Μετά από αυτό το stage θα κάνουμε install την postgres όπου θα εκτελεστεί ένα ansible playbook για την εγκατάσταση της PostgreSQL από έναν άλλον απομακρυσμένο server. Επιπλέον, κάνουμε deploy το frontend μας τροποποιώντας ρυθμίσεις στον frontend server και θα εκτελεστεί το ansible playbook για την frontend εφαρμογή μας με Vue.js.

Επίσης, έχουμε υλοποιήσει ένα docker.Jenkinsfile όπου ορίζει πάλι ένα Jenkins pipeline για την αυτοματοποίηση με Docker και Ansible. Αρχικά, κάνουμε Checkout τον κώδικα μας στο branch devops-backend. Θα κάνουμε παρόμοια διαδικασία με testing της εφαρμογής εκτελώντας ./mvntest. Επιπροσθέτως, θα πρέπει να κάνουμε docker build and push για την δημιουργία docker image μέσω του nonroot.Dockerfile, και ανεβαίνει στο Docker registry ghri.io με τα κατάλληλα tags. Ύστερα, θα κάνουμε run το ansible pipeline. Τέλος, θα γίνει Install το project μας με dockercompose μέσω ενός ansible playbook όπου έχουμε δημιουργήσει.

Γενικά Σχόλια/Παρατηρήσεις

Με δυσκόλεψε / δεν υλοποίησα

Κώδικας

Αποθετήρια κώδικα

- https://github.com/VasileiosKokki/FarmerCompensation_University.git στα branch:devops-backend και branch:devops-frontend
- https://github.com/VasileiosKokki/Devops_University.git στο branch: main έχουμε τα playbooks της ansible και τον φάκελο vagrant

Δοκιμαστικά accounts και urls

- username: admin
- password: pavlosnikolopoulos44
- role: ADMIN

Url δοκιμαστικού περιβάλλοντος

Οδηγίες Χρήσης / Εγκατάστασης

Έχουμε README.md αρχεία μέσα στα παραπάνω αποθετήρια.