ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ, ΣΧΕΔΙΑΣΜΟΣ & ΑΝΑΠΤΥΞΗ CROSS PLATFORM ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΟΛΟΓΙΩΝ ΔΙΑΔΙΚΤΥΟΥ

Δημητρίου Γεώργιος

Επιβλέπων καθηγητής

Δρ. Χρυσόστομος Στύλιος

Άρτα, Σεπτέμβριος 2020

ΜΕΛΕΤΗ, ΣΧΕΔΙΑΣΜΟΣ & ΑΝΑΠΤΥΞΗ CROSS PLATFORM ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΟΛΟΓΙΩΝ ΔΙΑΔΙΚΤΥΟΥ

RESEARCH, DESIGN & DEVELOPMENT OF A CROSS PLATFORM APPLICATION USING WEB TECHNOLOGIES

Εγκρίθηκε από τριμελή εξεταστική επιτροπή
Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Όνομα Επίθετο,

2. Μέλος επιτροπής

   Όνομα Επίθετο,

3. Μέλος επιτροπής

   Όνομα Επίθετο,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Δημητρίου Γεώργιος

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά τον Δρ. Χρυσόστομο Στύλιο για την υποστήριξη και την καθοδήγηση καθ' όλη τη διάρκεια της συνεργασίας μας, στα πλαίσια εκπόνησης της εν λόγω πτυχιακής εργασίας.

ΠΕΡΙΛΗΨΗ

Αρχικά, γίνεται μια παρουσίαση σε διάφορες τεχνολογίες διαδικτύου της σημερινής εποχής και μια ανάλυση σε τεχνικές ανάπτυξης εφαρμογών. Στη συνέχεια, παρουσιάζονται οι απαιτήσεις της εφαρμογής μαζί με πολλά στοιχεία του σχεδιασμού της, αλλά και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξή της. Τέλος, παρουσιάζεται αναλυτικά η υλοποίηση της εφαρμογής.

ABSTRACT

Initially, there is a presentation of a variety of today's web technologies and an analysis of application development techniques. Subsequently, there is an application's requirements analysis with many of its design's features and a brief presentation of the application's technologies used for its development. Lastly, there is a detailed presentation of the application's implementation.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Στα πλαίσια της Πρακτικής Άσκησης που πραγματοποίησα στο Εργαστήριο Γνώσης και Ευφυούς Πληροφορικής, μου ανατέθηκε να σχεδιάσω και να δημιουργήσω το front-end κομμάτι ενός desktop application με τη χρήση τεχνολογιών ιστού. Αφότου έφτασα στο επιθυμητό σημείο το έργο, το επεξεργάστηκα με τις κατάλληλες τεχνολογίες και το μετέτρεψα σε μια cross-platform desktop εφαρμογή.

1. Τι είναι το cross-platform;

Με τον όρο cross-platform εννοείται η ανάπτυξη ενός λογισμικού ή μιας εφαρμογής σε πολλές υπολογιστικές πλατφόρμες. Για παράδειγμα, μια εφαρμογή είναι cross-platform αν μπορεί να εκτελείται σε πολλές πλατφόρμες όπως Microsoft Windows, Linux και macOS αλλά σε πλατφόρμες στις κινητές συσκευές. Το cross-platform μπορεί να χωριστεί σε desktop συσκευές και σε κινητές συσκευές. Γενικά, με την αλματώδη ανάπτυξη των κινητών συσκευών και άλλων ειδών πλατφόρμες, οι προγραμματιστές στρέφονται όλο και πιο πολύ προς το cross-platform development.

Εικόνα 1: Ανάπτυξη εφαρμογής *[1]*

2. Τεχνικές ανάπτυξης εφαρμογών

Ένα συχνό ερώτημα που υπάρχει ανάμεσα στους προγραμματιστές είναι τι τεχνική να προτιμήσει να χρησιμοποιήσει για να δημιουργήσει μια εφαρμογή. Υπάρχουν λοιπόν οι εγγενείς (Native applications), οι υβριδικές (Hybrid applications) και οι cross-platform εφαρμογές.

Εικόνα 2: Native, Hybrid ή Cross-platform;

2.1 Εγγενείς εφαρμογές (Native applications)

Εγγενής ονομάζεται η εφαρμογή που έχει σχεδιαστεί και δημιουργηθεί για ένα συγκεκριμένο λειτουργικό σύστημα με συγκεκριμένη γλώσσα προγραμματισμού, συνήθως αυτή που απαιτεί τι λειτουργικό σύστημα. Τα Google Maps, LinkedIn είναι παραδείγματα εγγενών εφαρμογών. [2]

Πλεονεκτήματα:

- Είναι πολύ γρήγορες σαν εφαρμογές, καθώς δεν διαθέτουν πολύπλοκο κώδικα και πολλά χαρακτηριστικά είναι ήδη προεγκατεστημένα.
- Είναι πολύ ισχυρές σε σχέση με άλλες εφαρμογές, επειδή χρησιμοποιούν στο έπακρο τους πόρους του hardware μιας συσκευής.
- Μπορούν να χρησιμοποιηθούν χωρίς κανένα πρόβλημα ακόμα και αν ο χρήστης βρίσκεται εκτός σύνδεσης.

Μειονεκτήματα:

- Ο κώδικάς τους είναι μη επαναχρησιμοποιήσιμος. Αυτό σημαίνει πως αν κάποιος προγραμματιστής θέλει να δημιουργήσει μια εφαρμογή και σε άλλη πλατφόρμα, πρέπει να την δημιουργήσει εκ νέου με διαφορετικό κώδικα.
- Το κόστος ανάπτυξης και συντήρησης είναι πολύ μεγάλο.

2.2 Υβριδικές εφαρμογές (Hybrid applications)

Οι υβριδικές εφαρμογές συνδυάζουν τη δύναμη του web development και των πόρων της συσκευής. Αποτελείται από το κομμάτι του backend που δημιουργείται χρησιμοποιώντας γλώσσες όπως JavaScript, HTML και CSS και από ένα εγγενές κέλυφος που εμφανίζει την εφαρμογή. Τα Gmail, Instagram αποτελούν παραδείγματα υβριδικών εφαρμογών. [3]

Πλεονεκτήματα:

- Η υλοποίηση και η συντήρηση είναι απλή και πολύ φθηνή.
- Η ανάπτυξη τους είναι αρκετά ταχύτερη σε σχέση με τις εγγενείς, καθώς υπάρχει μια ενιαία βάση κώδικα.
- Μπορούν να κάνουν χρήση των χαρακτηριστικών μιας συσκευής, όπως ο χώρος αποθήκευσης, το GPS, η κάμερα κλπ.

Μειονεκτήματα:

- Πιο αργές σε σχέση με τις εγγενείς εφαρμογές, καθώς πρέπει να φορτώσουν όλα τα στοιχεία της εφαρμογής.
- Υπάρχει περίπτωση κάποια χαρακτηριστικά να μην υποστηρίζονται σε όλα τα λειτουργικά συστήματα. Αυτό αποφεύγεται βέβαια με συνεχείς ελέγχους και τροποποιήσεις.

Εικόνα 3: Διαφορές μεταξύ Native και Hybrid κατά την ανάπτυξη *[4]*

2.3 Cross-platform εφαρμογές

Οι cross-platform εφαρμογές είναι το άκρως αντίθετο των εγγενών εφαρμογών. Η cross-platform ανάπτυξη περιλαμβάνει τη χρήση ενός κώδικα και σε πολλές πλατφόρμες. Φυσικά δεν απαιτείται μια συγκεκριμένη γλώσσα προγραμματισμού για κάθε λειτουργικό σύστημα. Παραδείγματα cross-platform εφαρμογών είναι τα Facebook, Skype, Slack.

Πλεονεκτήματα:

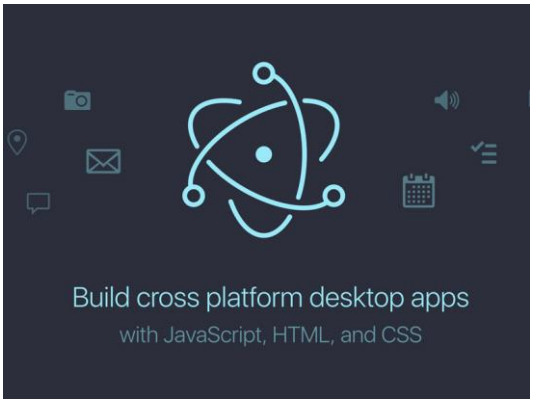
- Είναι πολύ αποδοτικό όσον αφορά την οικονομία, καθώς τα έξοδα που σχετίζονται με τα εργαλεία και τις διαδικασίες ανάπτυξης είναι συγκριτικά λιγότερα.
- Ο κώδικας είναι επαναχρησιμοποιήσιμος και μεταφέρεται σε διαφορετικές πλατφόρμες.
- Δεν υπάρχουν περιορισμοί όσον αφορά την πλατφόρμα που χρησιμοποιούν οι πελάτες. Η εφαρμογή είναι απολύτως κοινή για όλους.

Μειονεκτήματα:

- Παρά την δυνατότητα επαναχρησιμοποίησης του κώδικα, η ανάπτυξη μιας cross-platform εφαρμογής θεωρείται λίγο δύσκολη, καθώς πρέπει να δοθεί προσοχή στις μικρές διαφορές μεταξύ των πλατφορμών.
- Η εμπειρία των χρηστών μπορεί να μην είναι τόσο πλούσια όσο σε μια εγγενή εφαρμογή, αλλά είναι σε πολύ καλό επίπεδο.

3. Εργαλεία που χρησιμοποιούνται για την ανάπτυξη cross-platform εφαρμογών

3.1 Electron



Εικόνα 4: Το λογότυπο του Electron *[5]*

Το Electron είναι ένα framework ανοιχτού κώδικα (open-source) που αναπτύχθηκε από το GitHub το 2013. Χρησιμοποιώντας την τελευταία έκδοση του Node.js, επιτρέπει στους προγραμματιστές να αναπτύξουν cross-platform desktop εφαρμογές με τη χρήση δημοφιλών τεχνολογιών διαδικτύου όπως το HTML, CSS και JavaScript. Αρχικά είχε δημιουργηθεί για τον Atom, έναν επεξεργαστή κειμένου και πηγαίου κώδικα που υπάρχει για τα λειτουργικά συστήματα Windows, Linux, macOS. Οι προγραμματιστές το επιλέγουν κυρίως γιατί βασίζεται εξ' ολοκλήρου στα πρότυπα ιστού που γνωρίζουν σχεδόν όλοι και επειδή προσφέρει βασικές λειτουργίες όπως αυτόματη ενημέρωση, δημιουργό installer για desktop εφαρμογές. Γνωστές εφαρμογές που έχουν υλοποιηθεί με Electron είναι τα Visual Studio Code, Atom, GitHub Desktop.

3.2 Ionic

Εικόνα 5: Το λογότυπο του Ionic *[6]*

Το Ionic είναι μια ολοκληρωμένη εργαλειοθήκη ανάπτυξης λογισμικού (Software Development Kit ή SDK) που χρησιμοποιείται για την ανάπτυξη cross-platform εφαρμογών που μοιάζουν με native κυρίως για κινητές συσκευές. Δημιουργήθηκε το 2013 από τους Max

Lynch, Ben Sperry και Adam Bradley της εταιρείας Drifty. Η τελευταία έκδοση έδωσε τη δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν όποιο framework επιθυμούν για να αναπτύξουν τον κώδικα της εφαρμογής όπως η Angular, η React ή Vue.js αλλά και άλλα web components. Επίσης προσφέρει εργαλεία και τεχνολογίες όπως η HTML, CSS και Sass. Η χρήση μιας μόνο βάσης κώδικα και η γενική ευκολία στη χρήση είναι τα χαρακτηριστικά που κάνουν το Ionic ελκυστικό στους προγραμματιστές.

3.3 Qt

Το Qt είναι ένα από τα καλύτερα framework για ανάπτυξη γραφικών διεπαφών χρήστη (GUI) και cross-platform εφαρμογών. Χρησιμοποιεί μία βάση κώδικα ώστε να είναι εύκολη η μεταφορά σε πολλές πλατφόρμες, όπως Linux, Windows, macOS, Android αλλά και σε ενσωματωμένα συστήματα. Υποστηρίζει γλώσσες όπως η C++, Python και η Qt QML, η οποία είναι μια δηλωτική γλώσσα με τη δυνατότητα περιγραφής επιχειρηματικής λογικής με JavaScript. Στις περισσότερες περιπτώσεις το Qt χρησιμοποιείται και συντάσσεται με το περιβάλλον ανάπτυξης Qt Creator.

3.4 Haxe

Το Haxe είναι μία ανοιχτού κώδικα υψηλού επιπέδου cross-platform γλώσσα προγραμματισμού και ένας μεταγλωττιστής (compiler) που έχουν τη δυνατότητα να παράγουν εφαρμογές και πηγαίο κώδικα από μία βάση κώδικα. Ο μεταγλωττιστής μεταγλωττίζει τον πηγαίο κώδικα σε διάφορες γλώσσες, παράγοντας κώδικα για κάθε πλατφόρμα. Κάποιες από τις γλώσσες που υποστηρίζει το Haxe είναι οι C++, JavaScript, PHP, Python, Lua. Αξίζει να σημειωθεί πως το Haxe χρησιμοποιείται όλο και πιο συχνά στην ανάπτυξη video games. [7]

4. Τεχνολογίες διαδικτύου

4.1 HTML (HyperText Markup Language)

Το HTML είναι μια γλώσσα σχεδιασμένη για να εμφανίζει ιστοσελίδες σε έναν browser στο διαδίκτυο. Ο browser διαβάζει ένα HTML έγγραφο και το συνθέτει σε μια ιστοσελίδα που κάποιος μπορεί να διαβάσει. Το HTML αποτελείται από στοιχεία με τα οποία χτίζεται μια ιστοσελίδα. Τα στοιχεία αυτά αποτελούνται από ετικέτες, με τις οποίες το δημιουργεί και διαμορφώνει παραγράφους, ενότητες, επικεφαλίδες, συνδέσμους αλλά και να εμφανίσει διαδραστικές φόρμες. Το HTML συνήθως χρησιμοποιείται μαζί με το CSS για να μορφοποιούνται τα αντικείμενα στην ιστοσελίδα και με τη JavaScript για να επηρεάζει τη συμπεριφορά του περιεχομένου της σελίδας.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content=
        "text/html; charset=us-ascii" />
        <script type="text/javascript">
            function reDo() {top.location.reload();}
            if (navigator.appName == 'Netscape') {top.onresize = reDo;}
            dom=document.getElementById;
        </script>
    </head>
    <body>
    </body>
</html>
```

Εικόνα 6: Παράδειγμα υλοποίησης HTML-JavaScript *[8]*

Το HTML εφευρέθηκε το 1991 και η πρώτη του έκδοση κυκλοφόρησε το 1993. Αισίως βρίσκεται στην 5η του έκδοση, όπου προστέθηκαν αρκετές νέες δυνατότητες και στοιχεία σε σχέση με την προηγούμενη.

4.2 CSS

Το CSS (Cascading Style Sheets) είναι μια γλώσσα που χρησιμοποιείται για τη διαμόρφωση μιας ιστοσελίδας. Χρησιμοποιείται δηλαδή για τον έλεγχο εμφάνισης ενός εγγράφου, κυρίως HTML. Μπορεί να διαμορφώσει μια ιστοσελίδα με κάθε δυνατό τρόπο, προσφέροντας χρώματα, στοίχιση σε κείμενα και άλλα πολλά που από μόνο του το HTML

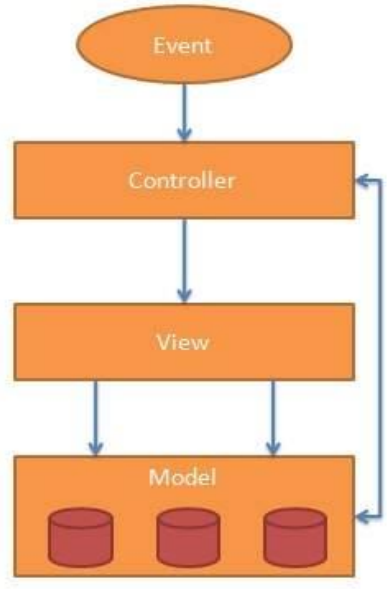
δεν μπορεί να προσφέρει. Όσον αφορά μια σελίδα, η μορφοποίηση της γίνεται συνήθως σε ξεχωριστό αρχείο από αυτό του HTML και έτσι προσφέρεται ευελιξία. Η πρώτη του έκδοση κυκλοφόρησε το 1996. Εκτός από το HTML, το CSS υποστηρίζεται και από άλλες γλώσσες όπως οι XHTML, XML, XUL και SVG.

4.3 Πλαίσια ανάπτυξης τεχνολογιών ιστού (Web Development Frameworks)

4.3.1 Angular

Το Angular είναι ένα ανοιχτού κώδικα framework βασισμένο στη γλώσσα προγραμματισμού TypeScript που χρησιμοποιείται για την κατασκευή δυναμικών web εφαρμογών. Κυκλοφόρησε το 2016 και από τότε συνεχώς ανανεώνεται με νέες λειτουργίες. Επικεφαλής του Angular είναι η Angular Team της Google μαζί με μια κοινότητα από εταιρείες και ιδιώτες. Αυτή τη στιγμή κυκλοφορεί η 10η έκδοσή του.

Μεγάλο προσόν του Angular είναι η μέθοδος ανάπτυξης λογισμικού ονόματι MVC (Model View Controller). Η μέθοδος αυτή χωρίζει την εφαρμογή σε 3 μέρη, στο Model που είναι υπεύθυνο για τη διαχείριση των δεδομένων της εφαρμογής, στο View που είναι υπεύθυνο για την προβολή των δεδομένων με τη βοήθεια του HMTL και του CSS και στο Controller που είναι υπεύθυνο να λαμβάνει είσοδο και να εκτελεί λειτουργίες και τροποποιήσεις στο Model.



Εικόνα 7: Η μέθοδος ανάπτυξης MVC *[9]*

4.3.2 React Native

Το React Native είναι ένα ανοιχτού κώδικα framework που χρησιμοποιείται για την ανάπτυξη εφαρμογών σε κινητές συσκευές και δημιουργήθηκε από το Facebook.. Βασίζεται στη React, η οποία είναι μία JavaScript βιβλιοθήκη. Το συγκεκριμένο framework επιτρέπει στους προγραμματιστές να γράφουν και εγγενή κώδικα σε γλώσσες όπως η Java και η Swift προσφέροντας ευελιξία. Επίσης, το React Native δεν χρησιμοποιεί HTML και CSS, αλλά χρησιμοποιούνται ισοδύναμα στοιχεία που περιέχονται React Native. [10]

4.3.3 Ruby on Rails

Το Ruby on Rails ή αλλιώς Rails είναι ένα server-side framework που χρησιμοποιείται για την ανάπτυξη εφαρμογών και δημιουργήθηκε από τον David Heinemeier Hansson το 2004. Το Rails χρησιμοποιεί τη γλώσσα προγραμματισμού Ruby και σαν μέθοδο ανάπτυξης το MVC , όπως και το Angular. Ενθαρρύνει τη χρήση JSON και XML για μεταφορά δεδομένων και τη χρήση HTML, CSS και JavaScript για διεπαφή χρήστη (User Interface). Έχουν κυκλοφορήσει 6 εκδόσεις του Rails και έχει επηρεάσει πολλά άλλα frameworks.

Οι προγραμματιστές επιλέγουν το Rails κυρίως για την ταχύτητα ανάπτυξης. Δημιουργήθηκε με γνώμονα τον προγραμματιστή και η ταχύτητα οφείλεται στο μοτίβο ανάπτυξης που χρησιμοποιεί ονόματι COC (Convention over Configuration). Με αυτό το μοτίβο, ο προγραμματιστής χρειάζεται να καθορίζει μόνο τις μη συμβατικές πτυχές της εφαρμογής. Για παράδειγμα, αν υπάρχει μια κλάση στο μοντέλο με όνομα «Πωλήσεις», τότε ο αντίστοιχος πίνακας στη βάση δεδομένων ονομάζεται αυτόματα «πωλήσεις». [11]

4.3.4 Vue.js

Εικόνα 8: Το λογότυπο του Vue.js

Το Vue.js ή αλλιώς σκέτο Vue είναι ένα ανοιχτού κώδικα framework σε JavaScript που χρησιμοποιείται για την ανάπτυξη διεπαφών χρήστη και εφαρμογών. Δημιουργός του είναι ο Evan You, ο οποίος εργαζόταν στη Google και χρησιμοποίησε τεχνικές από άλλα frameworks και έτσι κυκλοφόρησε το Vue.js το 2014. Το Vue συνδυάζει το HTML με τη JavaScript για την ανάπτυξη κώδικα και βρίσκεται στην έκδοση 2.6. Δυνατό του σημείο στην προτίμηση των προγραμματιστών είναι η απλότητα στη χρήση, μιας και κάποιος με καλή γνώση σε HTML, CSS και JavaScript μπορεί να ασχοληθεί με το Vue. Επίσης, τα components που γράφονται σε HTML, CSS και JavaScript δε χωρίζονται σε ξεχωριστά αρχεία όπως π.χ. στο Angular. [12]

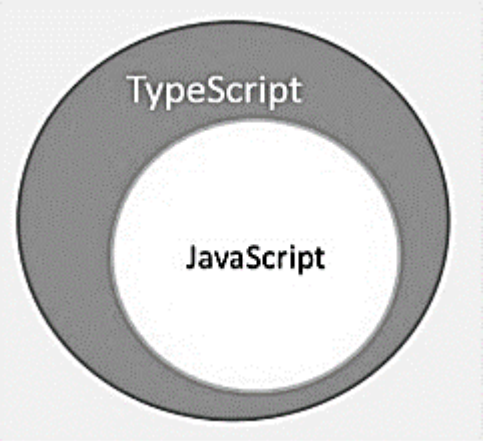
4.4 Γλώσσες προγραμματισμού

4.4.1 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου και μαζί με τα HTML και CSS είναι από τις βασικές τεχνολογίες του παγκόσμιου ιστού. Δημιουργήθηκε από τον Brendan Eich της Netscape Communications Corporation το 1995 με αρχική ονομασία Mocha. Αργότερα μετονομάστηκε σε LiveScript και τελικά κατέληξε στο όνομα JavaScript. Η γλώσσα αυτή συντηρείται και εξελίσσεται σύμφωνα με τα πρότυπα του ECMAScript του οργανισμού ECMA (European Computer Manufacturers Association). Αυτή τη στιγμή κυκλοφορεί η 11^{η} έκδοση του ECMAScript. Η JavaScript παραμένει σταθερά μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού το 2020. [13]

4.4.2 TypeScript

Η TypeScript είναι μια ανοιχτού κώδικα γλώσσα προγραμματισμού που δημιουργήθηκε και συντηρείται από τη Microsoft. Κυκλοφόρησε πρώτη φορά το 2012. Αποτελεί υπερσύνολο της γλώσσας JavaScript. Με άλλα λόγια, η TypeScript είναι η JavaScript με κάποια έξτρα χαρακτηριστικά.



Εικόνα 9: Η TypeScript και η JavaScript σε διάγραμμα Venn *[14]*

Τα κύρια χαρακτηριστικά που κάνει την TypeScript πιο ελκυστική από τη JavaScript είναι τα εξής:

- Υποστηρίζει τον Αντικειμενοστραφή Προγραμματισμό

- Βρίσκει πιθανά σφάλματα και συντακτικά λάθη κατά τη μεταγλώττιση και όχι κατά την εκτέλεση ενός αρχείου
- Έχει τη δυνατότητα να κάνει μεταγλώττιση χωρίς να γνωρίζει τους τύπους των μεταβλητών

4.4.3 Ruby

Η Ruby είναι μια καθαρά αντικειμενοστραφής υψηλού επιπέδου γλώσσα προγραμματισμού που σχεδιάστηκε και αναπτύχθηκε από τον Yukihiro Matsumoto το 1995. Σύμφωνα με το δημιουργό, είναι εμπνευσμένη από τις Smalltalk, Perl, Eiffel, Ada, Basic, Lisp και Python [15]. Κάποια από τα χαρακτηριστικά της Ruby είναι:

- Είναι ανοιχτού κώδικα και διατίθεται ελεύθερα, αλλά υπόκειται σε άδεια.
- Μπορεί να ενσωματωθεί σε HTML αρχεία.
- Έχει εύκολη σύνταξη που επιτρέπει στους νέους προγραμματιστές να μαθαίνουν εύκολα.
- Χάρη στο scalability της, μεγάλες εφαρμογές γραμμένες σε Ruby είναι εύκολα συντηρήσιμα.
- Υποστηρίζει πολλά εργαλεία GUI (Graphics User Interface), όπως τα OpenGL, GTK και Tcl.

4.5 Μορφές δεδομένων (Data formats)

4.5.1 XML

Το XML (Extensive Markup Language) είναι μια γλώσσα σήμανσης βασισμένη σε κείμενο. Οι XML ετικέτες προσδιορίζουν τα δεδομένα και χρησιμοποιούνται για την εμφάνιση των δεδομένων για την αποθήκευση και την οργάνωση των δεδομένων σε αντίθεση με τις ετικέτες του HTML που χρησιμοποιούνται για την εμφάνιση δεδομένων. [16]

Υπάρχουν 3 σημαντικά χαρακτηριστικά που καθιστούν το XML πολύ χρήσιμο:

- Είναι επεκτάσιμο. Επιτρέπει να δημιουργήσει κάποιος τις δικές του περιγραφικές ετικέτες που ταιριάζει στην περίπτωσή του.
- Επιτρέπει την αποθήκευση δεδομένων ανεξάρτητα από το πώς θα παρουσιαστούν.
- Είναι ένα δημόσιο πρότυπο που έχει αναπτυχθεί τον οργανισμό ονόματι W3C (World Wide Web Consortium) και διατίθεται σαν ανοιχτό πρότυπο.

Επίσης το XML έχει διάφορες χρήσεις όπως:

- Μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων και τον κανονισμό των δεδομένων, ανάλογα με το πώς θέλει κάποιος να χειριστεί τα δεδομένα.
- Μπορεί να χρησιμοποιηθεί για το άδειασμα και την επαναφόρτωση βάσεων δεδομένων.
- Οποιοσδήποτε τύπος δεδομένων μπορεί να εκφραστεί σαν έγγραφο XML.

4.5.2 JSON

Το JSON (JavaScript Object Notation) είναι μια μορφή ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάζουν και να το γράφουν. Επεκτάθηκε από γλώσσα JavaScript και ένα μεγάλο του πλεονέκτημα είναι πως είναι μια μορφή δεδομένων ανεξάρτητη κάποιας γλώσσας [17]. Κάποιες από τις χρήσεις του JSON είναι:

- Χρησιμοποιείται κυρίως για τη μεταφορά δεδομένων μεταξύ server και εφαρμογής.
- Χρησιμοποιείται για σειριοποίηση και μεταφορά δεδομένων μέσω διαδικτύου.
- Υπηρεσίες Ιστού και τα API χρησιμοποιούν τη μορφή JSON για την παροχή δημόσιων δεδομένων.
- Μπορεί να χρησιμοποιηθεί με σύγχρονες γλώσσες προγραμματισμού.

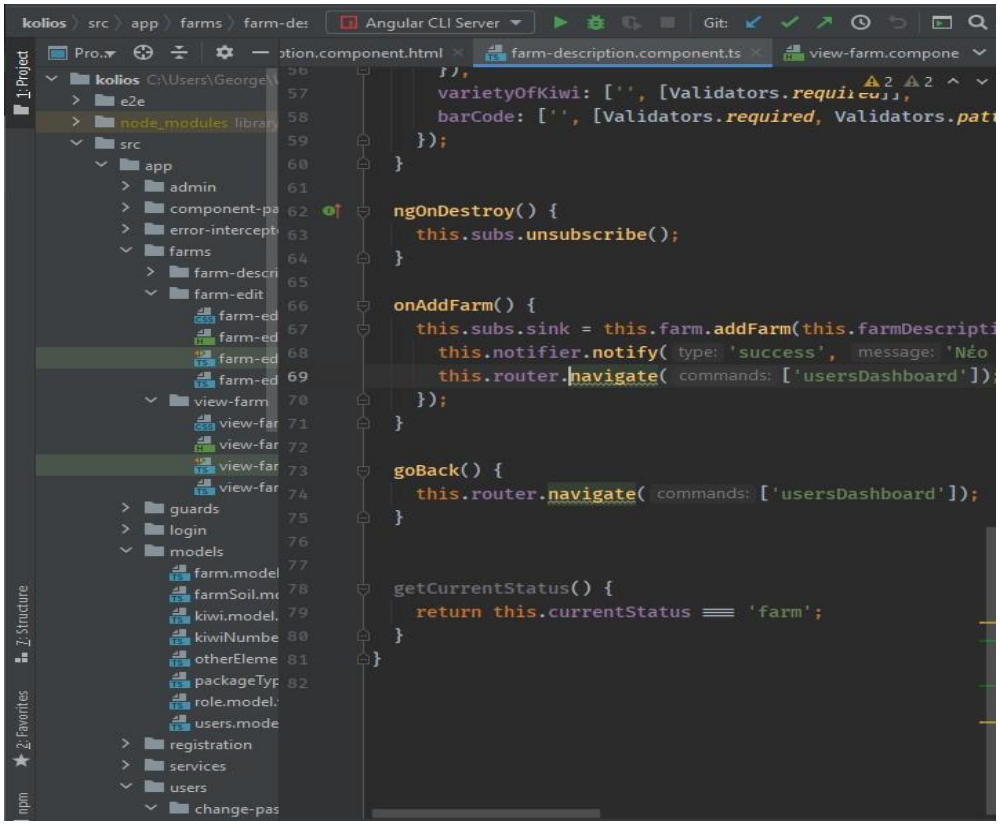
5. Σενάριο εφαρμογής

Έχει δημιουργηθεί το frontend μιας εφαρμογής, η οποία αφορά την ιχνηλασιμότητα των ακτινιδίων. Μέσα από αυτή την εφαρμογή, οι χρήστες μπορούν να εισάγουν δεδομένα σε διάφορα στάδια της αλυσίδας ιχνηλασιμότητας τα οποία είναι: Προσθήκη αγροκτήματος, Λίπανση, Συγκομιδή, Χημικός Έλεγχος, Αποθήκευση, Επεξεργασία, Συσκευασία. Οι χρήστες μπορούν να δουν όλες τις πληροφορίες των σταδίων ιχνηλασιμότητας σε ειδικά διαμορφωμένες κάρτες. Επίσης, υπάρχουν και οι διαχειριστές που φροντίζουν για διάφορες λειτουργίες της εφαρμογής, με βασική τη διαχείριση των χρηστών.

6. Τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής

6.1 JetBrains WebStorm IDE

Το WebStorm είναι ένα ισχυρό ολοκληρωμένο περιβάλλον ανάπτυξης(IDE) για σύγχρονη ανάπτυξη JavaScript που εκδίδεται από την εταιρεία JetBrains. Το WebStorm παρέχει πλήρη υποστήριξη για JavaScript, TypeScript, HTML, CSS καθώς και για frameworks όπως η React, η Angular και η Vue.js. Επίσης, χρησιμοποιείται ευρέως για την κατασκευή mobile και desktop εφαρμογών. [18]



Εικόνα 10: Το User Interface του WebStorm

Έχοντας λοιπόν τόσες πολλές δυνατότητες, αλλά και λόγω ότι είναι ένα «έξυπνο» IDE, όλα τα κομμάτια του κώδικα που αναπτύχθηκαν στην εφαρμογή, έγιναν αποκλειστικά με τη χρήση του συγκεκριμένου ολοκληρωμένου περιβάλλοντος ανάπτυξης.

6.2 Angular

Για την εφαρμογή επιλέχτηκε το Angular framework λόγω κάποιων σημαντικών πλεονεκτημάτων.

- Εκτός του ότι είναι από τα πιο διαδεδομένα frameworks του 2020, θα έχει επίσημα μακροπρόθεσμη υποστήριξη από τη Google.
- Η αρχιτεκτονική του που είναι βασισμένη σε components παρέχει μεγαλύτερης ποιότητας κώδικα. Αυτό συμβαίνει λόγω 1) της δυνατότητας επαναχρησιμοποίησης των components σε διαφορετικά μέρη μιας εφαρμογής, πχ πολύ χρήσιμο σε εφαρμογές μεγάλου βελησνεκούς που εργάζονται πάρα πολλά άτομα 2) της εύκολης δομής τους άρα και εύκολη κατανόηση σε νέους προγραμματιστές 3) της εύκολης συντήρησης τους, αφού τα components μπορούν εύκολα να αντικατασταθούν με καλύτερες, αναβαθμισμένες υλοποιήσεις.
- Είναι ένα πολύ δυνατό οικοσύστημα, καθώς εδώ και χρόνια έχει εμπλουτιστεί από πακέτα, plugins, πρόσθετα και εργαλεία ανάπτυξης, πράγμα που σημαίνει πως στην περίπτωση κάποιου προβλήματος υπάρχει κάποιο εργαλείο ώστε να επιλυθεί.

6.3 Angular Material

Η Angular Material είναι μια βιβλιοθήκη που απευθύνεται σε προγραμματιστές Angular. Διαθέτει μια πληθώρα από components (στοιχεία), τα οποία χρησιμοποιούνται για την κατασκευή ελκυστικών, ταχύτερων, λειτουργικών ιστοσελίδων και εφαρμογών ιστού (web applications). Είναι εμπνευσμένη από τη Google Material Design.

6.4 Bootstrap

Το Bootstrap είναι ένα πολύ δημοφιλές framework ανοιχτού κώδικα (open-source) που χρησιμοποιείται για το σχεδιασμό και την ανάπτυξη ιστοσελίδων ταχύτερα και ευκολότερα. Μεγάλο προσόν του είναι το responsive CSS του, που σημαίνει πως μια ιστοσελίδα που το χρησιμοποιεί, προσαρμόζεται τέλεια σε οποιαδήποτε συσκευή όπως για παράδειγμα κινητά τηλέφωνα, tablets, laptops αλλά και επιτραπέζιους υπολογιστές. Περιλαμβάνει πρότυπα

σχεδίασης (templates) βασισμένα σε HTML, CSS αλλά και JavaScript για φόρμες, πίνακες, κουμπιά κ.ά.

Στην εφαρμογή συνδυάστηκαν σε πολλές περιπτώσεις η Angular Material με το Bootstrap, ώστε να επιτευχθεί ένα αρκετά όμορφο αποτέλεσμα εστιάζοντας στο πλήρες responsiveness του Bootstrap και στην απλότητα της Angular Material.

6.5 jw-paginate

Η jw-paginate είναι μια βιβλιοθήκη που χρησιμοποιείται για να υπάρχει σελιδοποίηση σε σελίδες με πολλά αποτελέσματα. Έχει τη δυνατότητα να μπορεί να χρησιμοποιηθεί και σαν server-side pagination όπου όλη σχεδόν η δουλειά γίνεται από τη μεριά του server στέλνοντας ο ίδιος όσα αποτελέσματα είναι ρυθμισμένος να στέλνει, αλλά και στο frontend ώστε να ρυθμίζεται εκείνη τη στιγμή πόσα αποτελέσματα θα εμφανιστούν.

6.6 angular-notifier

Το angular-notifier είναι μια καλά σχεδιασμένη, πλήρως κινούμενη, εξαιρετικά τροποποιήσιμη και εύκολη στη χρήση βιβλιοθήκη ειδοποιήσεων για εφαρμογές σε Angular. Θεωρείται μία από τις 10 καλύτερες βιβλιοθήκες ειδοποιήσεων αυτή τη στιγμή. Περιλαμβάνει όλων των ειδών τις ειδοποιήσεις, δηλαδή επιτυχίας, πληροφόρησης, προειδοποίησης, κινδύνου.

6.7 RxJS

Το RxJS είναι μια βιβλιοθήκη για reactive programming (αντιδραστικό προγραμματισμό) που χρησιμοποιεί Observables που καθιστούν ευκολότερη την ανάπτυξη ασύγχρονου κώδικα. Παρέχει το Observable, το οποίο είναι μια ροή από δεδομένα (ροή δεδομένων) και επιστρέφεται συνήθως σαν response από τον server. Επίσης παρέχει κάποιες λειτουργίες όταν κάποιος δουλεύει με Observables όπως η σύνθεση πολλαπλών ροών δεδομένων, η

μετατροπή κώδικα για ασύγχρονες λειτουργίες σε Observables και το φιλτράρισμα των ροών δεδομένων.

6.8 ngx-ui-loader

Το ngx-ui-loader είναι ένα πλήρως τροποποιήσιμο περιστρεφόμενο loader για εφαρμογές σε Angular. Χρησιμοποιείται κυρίως σε περιπτώσεις όπου αργεί να έρθει το response από το server, οπότε χρησιμοποιείται για να μην υπάρχει μια κενή σελίδα ώστε να βοηθάει και το χρήστη να καταλάβει τι συμβαίνει.

6.9 Electron

Η τεχνολογία Electron χρησιμοποιήθηκε για την παραμετροποίηση και τη μετατροπή της web εφαρμογής σε desktop εφαρμογή. Αναλυτικές πληροφορίες που αφορούν το σχεδιασμό, θα παρουσιαστούν σε επόμενο κεφάλαιο.

6.10 JSON

Πρέπει να σημειωθεί πως στην εφαρμογή που αναπτύχθηκε, τα δεδομένα που μεταφέρονται από τον server στην εφαρμογή είναι με τη μορφή JSON. Επιλέχτηκε έναντι του XML, λόγω της ευκολότερης κατανόησης και της ευκολότερης διαχείρισης των δεδομένων που μπορεί να προσφέρει.

7. Σχεδιασμός Εφαρμογής

7.1 Δεδομένα εφαρμογής – Βάση Δεδομένων

Τα δεδομένα που εμφανίζονται και διαχειρίζεται η εφαρμογή προέρχονται από μια βάση δεδομένων, ειδικά σχεδιασμένη για αυτό το σενάριο. Τα δεδομένα που καλείται να διαχειριστεί η εφαρμογή παρουσιάζονται στον παρακάτω πίνακα.

Όνομα Μεταβλητής	**Όνομα Δεδομένου**	**Τύπος Μεταβλητής**	**Περιγραφή**
farmerName	Όνομα Παραγωγού	string	
farmCode	Κωδικός Αγροτεμαχίου	string	
farmerContactPhone	Τηλέφωνο Παραγωγού	string	
farmPlantingYear	Έτος Φύτευσης	Date	
farmCoordinates	Συντεταγμένες	Object(number, number)	Γεωγ. μήκος, γεωγ. πλάτος
farmTown	Πόλη Αγροτεμαχίου	string	
farmArea	Περιοχή Αγροτεμαχίου	string	
farmAreaToponyms	Τοπωνύμια Περιοχής	string	
farmAcres	Έκταση Αγροτεμαχίου	number	
ph	Ph Εδάφους	number	
electricalConductivity	Ηλεκτρική Αγωγιμότητα Εδάφους	number	
farmSoil	Τύπος Εδάφους	farmSoil	Αμμώδη εδάφη, Αργιλώδη εδάφη, Πηλώδη εδάφη, Οργανικά εδάφη, Αργιλοασβεστώδη εδάφη, Ασβεστώδη εδάφη
farmIrrigationWater	Νερό Άρδευσης	Object(number, number)	(Ph, ηλεκτρική αγωγιμότητα)
varietyOfKiwi	Ποικιλία Ακτινιδίου	Kiwi	Hayward, Soreli, Tsechelidis

barCode	Barcode	string	
nitrogenQuantity	Άζωτο	Array[number, Date]	Ποσότητα, ημερομηνία
phosphorusQuantity	Φώσφορος	Array[number, Date]	Ποσότητα, ημερομηνία
potassiumQuantity	Κάλιο	Array[number, Date]	Ποσότητα, ημερομηνία
otherElements	Άλλα στοιχεία	Array[otherElements, number, Date]	Όνομα, ποσότητα, ημερομηνία
herbicidesExistence	Ύπαρξη ζιζανιοκτόνων	boolean	Ναι/Όχι
sitofexExistence	Ύπαρξη Sitofex	boolean	Ναι/Όχι
typeOfFertilizer	Τύπος Λιπάσματος	TypeOfFertilizer	15-15-15/12-8-16/21-7-14/11-15-15
quantityOfFertilizer	Ποσότητα Λιπάσματος	number	
seasonOfLubrication	Εποχή Λίπανσης	Date	Ημερομηνία
herbicidesName	Ονομασία Ζιζανιοκτόνου	string	
herbicidesDate	Ημερομηνία Ρίψης Ζιζανιοκτόνου	Date	
herbicidesQuantity	Ποσότητα Ζιζανιοκτόνου	number	
hormoneName	Ονομασία Ορμόνης	string	
hormoneDate	Ημερομηνία Ρίψης Ορμόνης	Date	
plantProtectionName	Ονομασία Φυτοπροστατευτικού	string	
plantProtectionDate	Ημερομηνία Ρίψης Φυτοπροστατευτικού	Date	
plantProtectionQuantity	Ποσότητα Φυτοπροστατευτικού	number	

cultivationWorkName	Καλλιεργητικές Εργασίες	string	
cultivationWorkDate	Ημερομηνία Εργασιών	Date	
nameOfHarvestingWorkshop	Όνομα Συνεργείου	string	
harvestDate	Ημερομηνία Συγκομιδής	Date	
opticalCheck	Οπτικός Έλεγχος	boolean	Ναι/Όχι
hardnessOfKiwi	Σκληρότητα Ακτινιδίου	number	
solidSubstanceOfKiwi	Στερεά Ουσία Ακτινιδίου	number	
brixOfKiwi	Σάκχαρα Ακτινιδίου	number	
chemicalCheckDate	Ημερομηνία Χημικού Ελέγχου	Date	
freezer	Ψυκτικός Θάλαμος	string	
temperatureOfFreezer	Θερμοκρασία Θαλάμου	number	
storageDate	Ημερομηνία Αποθήκευσης	Date	
dateOfProcessing	Ημερομηνία Επεξεργασίας	Date	
checkOfHardnessOfKiwi	Έλεγχος Σκληρότητας Ακτινιδίου	number	Στάδιο Επεξεργασίας
checkOfBrix	Έλεγχος Σακχάρων	number	Στάδιο Επεξεργασίας
classOfKiwi	Κλάση Ακτινιδίου	number	1 ή 2
numberOfKiwi	Νούμερο Ακτινιδίου	KiwiNumber	18-20-23-25-27-30-33-36-39-42-46-άποντο
dateOfPackaging	Ημερομηνία Συσκευασίας	Date	
checkOfHardnessOfKiwi	Έλεγχος Σκληρότητας Ακτινιδίου	number	Στάδιο Συσκευασίας

checkOfBrix	Έλεγχος Σακχάρων	number	Στάδιο Συσκευασίας
classOfKiwi	Κλάση Ακτινιδίου	number	1 ή 2
numberOfKiwi	Νούμερο Ακτινιδίου	KiwiNumber	18-20-23-25-27-30-33-36-39-42-46-άποντο
typeOfPackages	Είδος Συσκευασίας	packageType	(10kg loose-6kg loose-3kg tray carton-5kg tray carton-1kgx12-flowpack
packageCode	Κωδικός Παρτίδας Συσκευασίας	string	
name	Όνομα Χρήστη	string	
surname	Επώνυμο Χρήστη	string	
username	Username	string	
password	Κωδικός Πρόσβασης	string	
confirmPass	Επαλήθευση Κωδικού Πρόσβασης	string	
status	Ιδιότητα	string	Farm ή factory
role	Ρόλος - Δικαιώματα	Role	Admin ή user

Πίνακας 1: Ο πίνακας δεδομένων της εφαρμογής

7.2 Απαιτήσεις Εφαρμογής

Οι απαιτήσεις της εφαρμογής είναι οι παρακάτω.

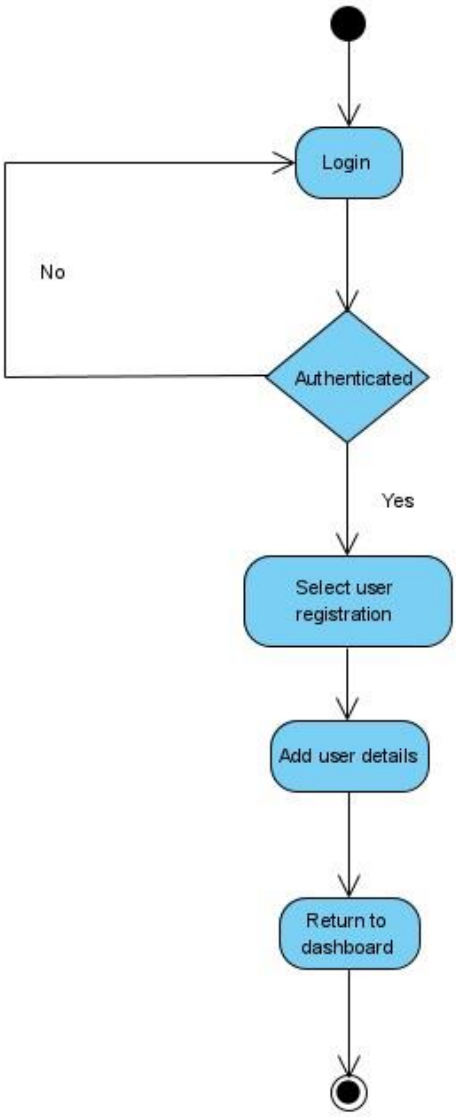
- Θα πρέπει να είναι desktop application
- Μπορούν να έχουν πρόσβαση μόνο συνδεδεμένοι χρήστες
- Υπάρχουν 2 είδη χρηστών, διαχειριστές (admin) και απλοί χρήστες (user)
- Ο user δεν επιτρέπεται να βλέπει τις σελίδες του admin
- Ο admin μπορεί να δημιουργεί και να διαχειρίζεται στοιχεία των users
- Ο user επιτρέπεται να αλλάζει τον κωδικό πρόσβασής του
- Ο user μπορεί να εγγράφει ένα αγρόκτημα και να προσθέτει δεδομένα σε κάθε στάδιο ιχνηλασιμότητας
- Ο user μπορεί να ενημερώνεται με πληροφορίες για όλα τα στάδια ιχνηλασιμότητας των ακτινιδίων
- Ο user έχει τη δυνατότητα να διαγράφει ένα αγρόκτημα

7.3 Ανάλυση σεναρίων / Ροή

7.3.1 Προσθήκη χρήστη (Απλού χρήστη)

Προαπαιτούμενη συνθήκη: Ο χρήστης δεν είναι εγγεγραμμένος και ο συνδεδεμένος χρήστης είναι διαχειριστής

- Ο διαχειριστής επιλέγει να προσθέσει χρήστη.
- Η εφαρμογή ανακατευθύνεται στη φόρμα προσθήκης χρήστη.
- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο διαχειριστής αν τα στοιχεία που εισήχθησαν είναι λάθος.
- Το κουμπί της υποβολής της φόρμας είναι απενεργοποιημένο μέχρι να συμπληρωθούν όλα τα πεδία και να είναι έγκυρα.
- Αν τα πεδία είναι έγκυρα, στέλνονται στο server για επαλήθευση. Αν το username υπάρχει ήδη, εμφανίζεται ειδοποίηση που ενημερώνει για το σφάλμα.
- Αν η επαλήθευση είναι επιτυχής από το server, τότε ο χρήστης προστίθεται επιτυχώς και ο διαχειριστής ανακατευθύνεται στην κεντρική του σελίδα.



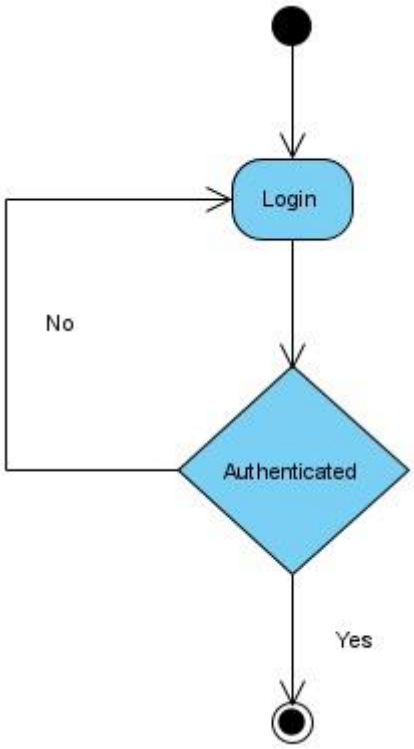
Εικόνα 11: Διάγραμμα ροής «Προσθήκη χρήστη»

7.3.2 Σύνδεση χρήστη (Απλού χρήστη/Διαχειριστή)

Προαπαιτούμενη συνθήκη: Ο χρήστης δεν είναι συνδεδεμένος

- Εμφανίζεται η φόρμα σύνδεσης (αρχική σελίδα).
- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο χρήστης αν τα στοιχεία που εισήχθησαν είναι λάθος.
- Το κουμπί της υποβολής της φόρμας είναι απενεργοποιημένο μέχρι να συμπληρωθούν όλα τα πεδία και να είναι έγκυρα.

- Αν τα πεδία είναι έγκυρα, στέλνονται στο server για επαλήθευση. Αν το username ή ο κωδικός πρόσβασης είναι λανθασμένο, εμφανίζεται ειδοποίηση που ενημερώνει το χρήστη για το σφάλμα.
- Αν η επαλήθευση είναι επιτυχής από το server, ο χρήστης συνδέεται επιτυχώς και ανακατευθύνεται στο αντίστοιχο dashboard ανάλογα με το ρόλο του.



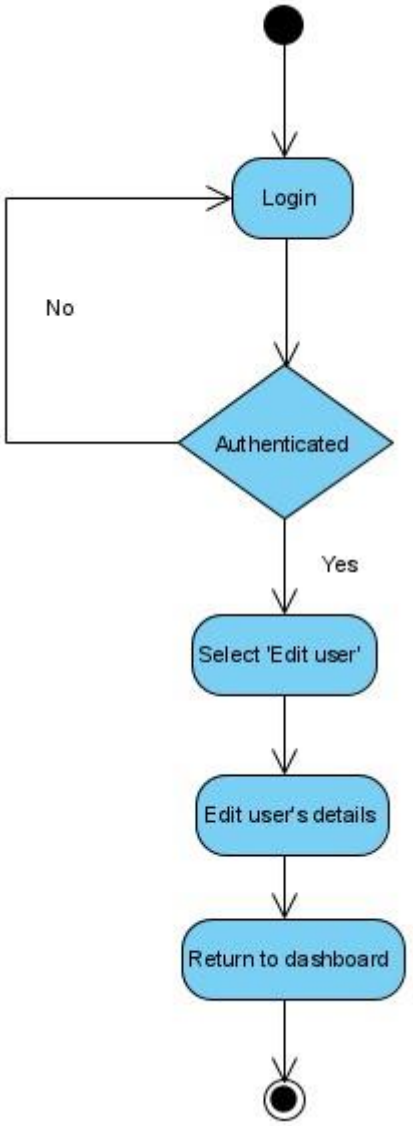
Εικόνα 12: Διάγραμμα ροής «Είσοδος χρήστη»

7.3.3 Επεξεργασία χρήστη (Ενημέρωση χρήστη)

Προαπαιτούμενη συνθήκη: Ο χρήστης είναι συνδεδεμένος και είναι διαχειριστής

- Ο διαχειριστής επιλέγει να επεξεργαστεί έναν χρήστη.
- Γίνεται ανακατεύθυνση στη φόρμα ενημέρωσης χρήστη.
- Ο χρήστης μπορεί να επιλέξει όποιο πεδίο ενδιαφέρεται να ενημερώσει ή και όλα.
- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο χρήστης αν τα στοιχεία που εισήχθησαν είναι λάθος.

- Το κουμπί της υποβολής της φόρμας μένει απενεργοποιημένο αν τα πεδία δεν είναι έγκυρα.
- Αν τα πεδία είναι έγκυρα, στέλνονται στο server και επαληθεύονται. Ο χρήστης έχει ενημερωθεί και ο διαχειριστής ανακατευθύνεται στην κεντρική του σελίδα.



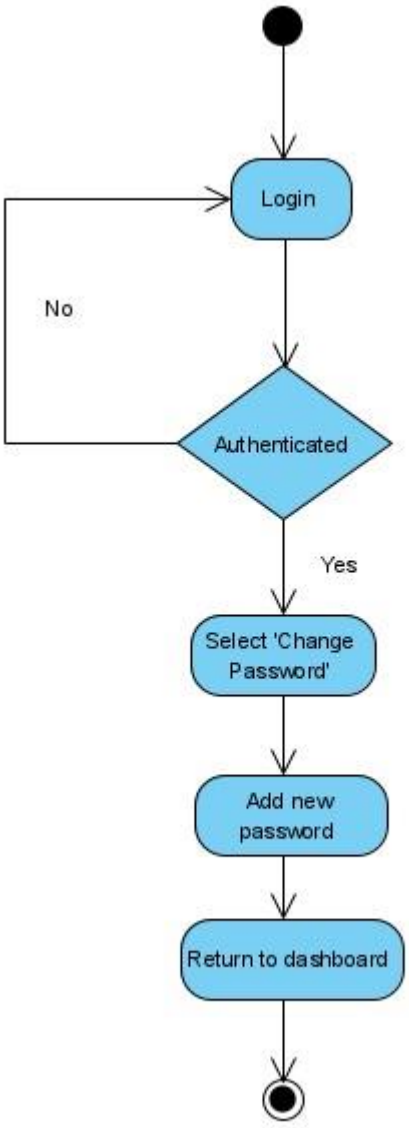
Εικόνα 13: Διάγραμμα ροής «Επεξεργασία χρήστη»

7.3.4 Αλλαγή κωδικού πρόσβασης (Απλού χρήστη)

Προαπαιτούμενη συνθήκη: Ο χρήστης είναι συνδεδεμένος και είναι απλός χρήστης

- Ο χρήστης επιλέγει να αλλάξει κωδικό πρόσβασης.
- Γίνεται ανακατεύθυνση στη φόρμα αλλαγής κωδικού πρόσβασης.

- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο χρήστης αν τα στοιχεία που εισήχθησαν είναι λάθος.
- Το κουμπί της υποβολής της φόρμας είναι απενεργοποιημένο μέχρι να συμπληρωθούν όλα τα πεδία και να είναι έγκυρα.
- Αν τα πεδία είναι έγκυρα, στέλνονται στο server για επαλήθευση. Αν ο παλιός κωδικός πρόσβασης είναι λανθασμένος, εμφανίζεται ειδοποίηση που ενημερώνει το χρήστη για το σφάλμα.
- Αν η επαλήθευση είναι επιτυχής από το server, ο κωδικός πρόσβασης έχει αλλάξει και ο χρήστης ανακατευθύνεται στην κεντρική του σελίδα.

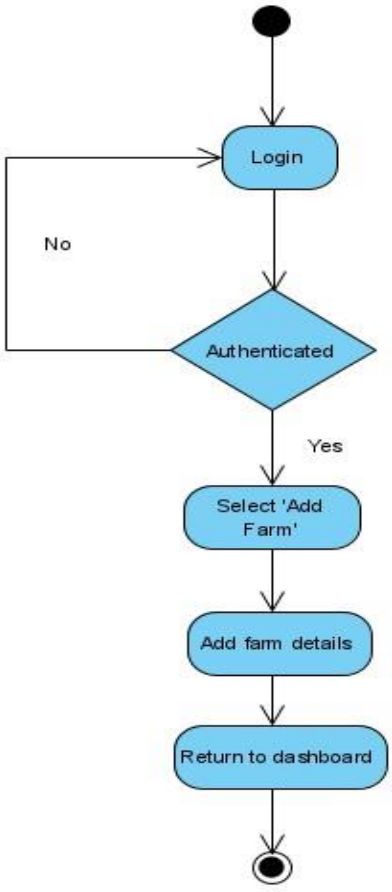


Εικόνα 14: Διάγραμμα ροής «Αλλαγή κωδικού πρόσβασης»

7.3.5 Προσθήκη αγροκτήματος

Προαπαιτούμενη συνθήκη: Ο χρήστης είναι συνδεδεμένος και είναι απλός χρήστης

- Ο χρήστης επιλέγει να προσθέσει ένα αγρόκτημα
- Γίνεται ανακατεύθυνση στη φόρμα προσθήκης αγροκτήματος.
- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο χρήστης αν τα στοιχεία που εισήχθησαν είναι λάθος.
- Το κουμπί της υποβολής της φόρμας είναι απενεργοποιημένο μέχρι να συμπληρωθούν όλα τα πεδία και να είναι έγκυρα.
- Αν τα πεδία είναι έγκυρα, στέλνονται στο server για επαλήθευση. Αν ο κωδικός αγροτεμαχίου ή το barcode υπάρχουν, εμφανίζεται ειδοποίηση που ενημερώνει το χρήστη για το σφάλμα.
- Αν η επαλήθευση είναι επιτυχής από το server, το αγρόκτημα έχει προστεθεί και ο χρήστης ανακατευθύνεται στην κεντρική σελίδα, όπου μπορεί να το δει στον πίνακα.

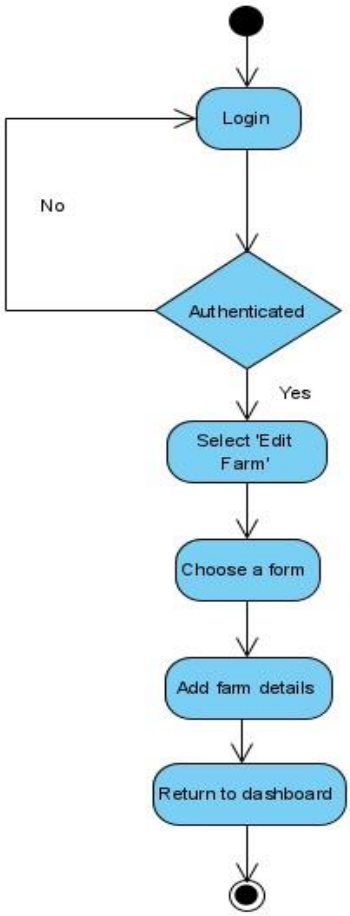


Εικόνα 15: Διάγραμμα ροής «Προσθήκη αγροκτήματος»

7.3.6 Επεξεργασία αγροκτήματος - Προσθήκη στοιχείων για στάδια ιχνηλασιμότητας (πολλές φόρμες)

Προαπαιτούμενη συνθήκη: Ο χρήστης είναι συνδεδεμένος και είναι απλός χρήστης

- Ο χρήστης επιλέγει να επεξεργαστεί ένα αγρόκτημα.
- Γίνεται ανακατεύθυνση στις φόρμες των σταδίων ιχνηλασιμότητας.
- Ο χρήστης επιλέγει οποιαδήποτε από τις φόρμες θέλει να συμπληρώσει.
- Η εφαρμογή ελέγχει δυναμικά την είσοδο των στοιχείων και εμφανίζει μηνύματα λάθους, αλλά και μηνύματα για το τι πρέπει να πληκτρολογήσει ο χρήστης αν τα στοιχεία που εισήχθησαν είναι λάθος.
- Το κουμπί της υποβολής της φόρμας είναι απενεργοποιημένο μέχρι να συμπληρωθούν όλα τα πεδία και να είναι έγκυρα.
- Αν τα πεδία είναι έγκυρα, στέλνονται στο server και επαληθεύονται. Η προσθήκη των στοιχείων είναι επιτυχής και φόρμα γίνεται reset για πιθανή νέα προσθήκη στοιχείων.



Εικόνα 16: Διάγραμμα ροής «Επεξεργασία αγροκτήματος»

7.4 Μοντέλα φορμών

Το μοντέλο είναι μια διεπαφή (interface) που περιέχει τις μεταβλητές-χαρακτηριστικά που περιέχονται σε μια οντότητα, όπως π.χ. σε αυτή την περίπτωση των χρηστών. Επιβάλλεται η ονομασία των μεταβλητών του μοντέλου αλλά και ο τύπος τους να είναι ίδια και στο backend κομμάτι, ώστε να υπάρχει σωστή επικοινωνία frontend – backend.

Βάσει των απαιτήσεων της εφαρμογής, έχουν δημιουργηθεί 17 φόρμες, εκ των οποίων οι 12 αφορούν τα στάδια ιχνηλασιμότητας των ακτινιδίων. Παρακάτω παρουσιάζονται τα μοντέλα τους.

Φόρμα Εισόδου Χρήστη		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
username	Username χρήστη	Input type text
password	Κωδικός Πρόσβασης	Input type password

Πίνακας 2: Μοντέλο φόρμας εισόδου χρήστη

Φόρμα Εγγραφής Χρηστών		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
name	Όνομα χρήστη	<Input type text>
surname	Επώνυμο χρήστη	<Input type text>
username	Username χρήστη	<Input type text>
password	Κωδικός Πρόσβασης	<Input type password>
confirmPass	Επαλήθευση Κωδικού Πρόσβασης	<Input type password>
status	Ιδιότητα χρήστη	<select>
role	Ρόλος-Δικαιώματα χρήστη	<select>

Πίνακας 3: Μοντέλο φόρμας εγγραφής χρηστών

Φόρμα Ενημέρωσης Χρήστη		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
name	Όνομα χρήστη	<Input type text>
surname	Επώνυμο χρήστη	<Input type text>
username	Username χρήστη	<Input type text>
status	Ιδιότητα χρήστη	<select>

Πίνακας 4: Μοντέλο φόρμας ενημέρωσης χρήστη

Φόρμα Αλλαγής Κωδικού Πρόσβασης		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
oldPassword	Παλιός Κωδικός Πρόσβασης	<Input type password>
newPassword	Νέος Κωδικός Πρόσβασης	<Input type password>
confirmPass	Επαλήθευση Κωδικού Πρόσβασης	<Input type password>

Πίνακας 5: Μοντέλο φόρμας αλλαγής κωδικού πρόσβασης

Το μοντέλο του αγροκτήματος σχεδιάστηκε σε πολλά «κομμάτια», καθώς έχει πάρα πολλά χαρακτηριστικά – μεταβλητές. Το διαφορετικά «κομμάτια» τα οποία σχεδιάστηκαν αφορούν: τα χαρακτηριστικά κατά την προσθήκη του αγροκτήματος και τα χαρακτηριστικά για την επεξεργασία του. Παρακάτω θα παρουσιαστεί το μοντέλο σε πολλούς πίνακες, οι οποίες αντιστοιχούν στις φόρμες, ώστε να υπάρχει μια καλύτερη κατανόησή του.

Φόρμα Προσθήκης Αγροκτήματος		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
farmerName	Όνομα Παραγωγού	<input type text>
farmCode	Κωδικός Αγροτεμαχίου	<input type text>
farmerContactPhone	Τηλέφωνο Παραγωγού	<input type text>
farmPlantingYear	Έτος Φύτευσης	<input matDatePicker>
farmCoordinates (latitude, longitude)	Συντεταγμένες (Γεωγραφικό μήκος και πλάτος)	<input type text>
farmTown	Πόλη Αγροτεμαχίου	<input type text>
farmArea	Περιοχή Αγροτεμαχίου	<input type text>
farmAreaToponyms	Τοπωνύμια της Περιοχής	<input type text>
farmAcres	Έκταση Αγροτεμαχίου	<input type text>
ph	Ph Εδάφους	<input type text>
electricalConductivity	Ηλεκτρική Αγωγιμότητα Εδάφους	<input type text>
farmSoil	Τύπος Εδάφους	<select>
farmIrrigationWater (ph, ec)	Νερό Άρδευσης (Ph και ηλεκτρική αγωγιμότητα)	<input type text>
varietyOfKiwi	Ποικιλία Ακτινιδίου	<select>

barCode	Barcode	<input type text>

Πίνακας 6: Μοντέλο φόρμας προσθήκης αγροκτήματος

Φόρμα Προσθήκης Αζώτου		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
quantity	Ποσότητα	<input type text>
date	Ημερομηνία Ρίψης	<input matDatePicker>

Πίνακας 7: Μοντέλο φόρμας προσθήκης αζώτου

Φόρμα Προσθήκης Φωσφόρου		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
quantity	Ποσότητα	<input type text>
date	Ημερομηνία Ρίψης	<input matDatePicker>

Πίνακας 8: Μοντέλο φόρμας προσθήκης αζώτου

Φόρμα Προσθήκης Καλίου		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
quantity	Ποσότητα	<input type text>
date	Ημερομηνία Ρίψης	<input matDatePicker>

Πίνακας 9: Μοντέλο φόρμας προσθήκης καλίου

Φόρμα Προσθήκης Άλλων Στοιχείων		
Ονομασία Πεδίου	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
name	Όνομα στοιχείου	<select>
quantity	Ποσότητα	<input type text>
date	Ημερομηνία ρίψης	<input matDatePicker>

Πίνακας 10: Μοντέλο φόρμας προσθήκης άλλων στοιχείων

Φόρμα Προσθήκης Στοιχείων Λίπανσης		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
typeOfFertilizer	Τύπος Λιπάσματος	<select>
quantityOfFertilizer	Ποσότητα Λιπάσματος	<input type text>
seasonOfLubrication	Εποχή Λίπανσης	<input matDatePicker>
herbicidesName	Ονομασία Ζιζανιοκτόνου	<input type text>
herbicidesDate	Ημερομηνία Ρίψης Ζιζανιοκτόνου	<input matDatePicker>
herbicidesQuantity	Ποσότητα Ζιζανιοκτόνου	<input type text>
hormoneName	Ονομασία Ορμόνης	<input type text>
hormoneDate	Ημερομηνία Ρίψης Ορμόνης	<input matDatePicker>
plantProtectionName	Ονομασία Φυτοπροστατευτικού	<input type text>

plantProtectionDate	Ημερομηνία Ρίψης Φυτοπροστατευτικού	<input matDatePicker>
plantProtectionQuantity	Ποσότητα Φυτοπροστατευτικού	<input type text>
cultivationWorkName	Καλλιεργητικές Εργασίες	<input type text>
cultivationWorkDate	Ημερομηνία Εργασιών	<input matDatePicker>

Πίνακας 11: Μοντέλο Φόρμας Προσθήκης Στοιχείων Λίπανσης

Φόρμα Προσθήκης Στοιχείων Συγκομιδής		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
nameOfHarvestingWorkshop	Όνομα Συνεργείου	<input type text>
harvestDate	Ημερομηνία Συγκομιδής	<input matDatePicker>
opticalCheck	Οπτικός Έλεγχος	<input type radio>

Πίνακας 12: Μοντέλο Φόρμας Προσθήκης Στοιχείων Συγκομιδής

Φόρμα Προσθήκης Στοιχείων Χημικού Ελέγχου		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
hardnessOfKiwi	Σκληρότητα Ακτινιδίου	<input type text>
solidSubstanceOfKiwi	Στερεά Ουσία Ακτινιδίου	<input type text>
brixOfKiwi	Σάκχαρα Ακτινιδίου	<input type text>
chemicalCheckDate	Ημερομηνία Χημ. Ελέγχου	<input matDatePicker>

Πίνακας 13: Μοντέλο Φόρμας Προσθήκης Στοιχείων Χημικού Ελέγχου

Φόρμα Προσθήκης Στοιχείων Αποθήκευσης		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
freezer	Ψυκτικός Θάλαμος	<input type text>
temperatureOfFreezer	Θερμοκρασία Θαλάμου	<input type text>
storageDate	Ημερομηνία Αποθήκευσης	<input matDatePicker>

Πίνακας 14: Μοντέλο Φόρμας Προσθήκης Στοιχείων Αποθήκευσης

Φόρμα Προσθήκης Στοιχείων Επεξεργασίας		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
dateOfProcessing	Ημερομηνία Επεξεργασίας	<input matDatePicker>
checkOfHardnessOfKiwi	Έλεγχος Σκληρότητας Ακτινιδίου	<input type text>
checkOfBrix	Έλεγχος Σακχάρων	<input type text>
classOfKiwi	Κλάση Ακτινιδίου	<input type text>
numberOfKiwi	Νούμερο Ακτινιδίου	<select>

Πίνακας 15: Μοντέλο Φόρμας Προσθήκης Στοιχείων Επεξεργασίας

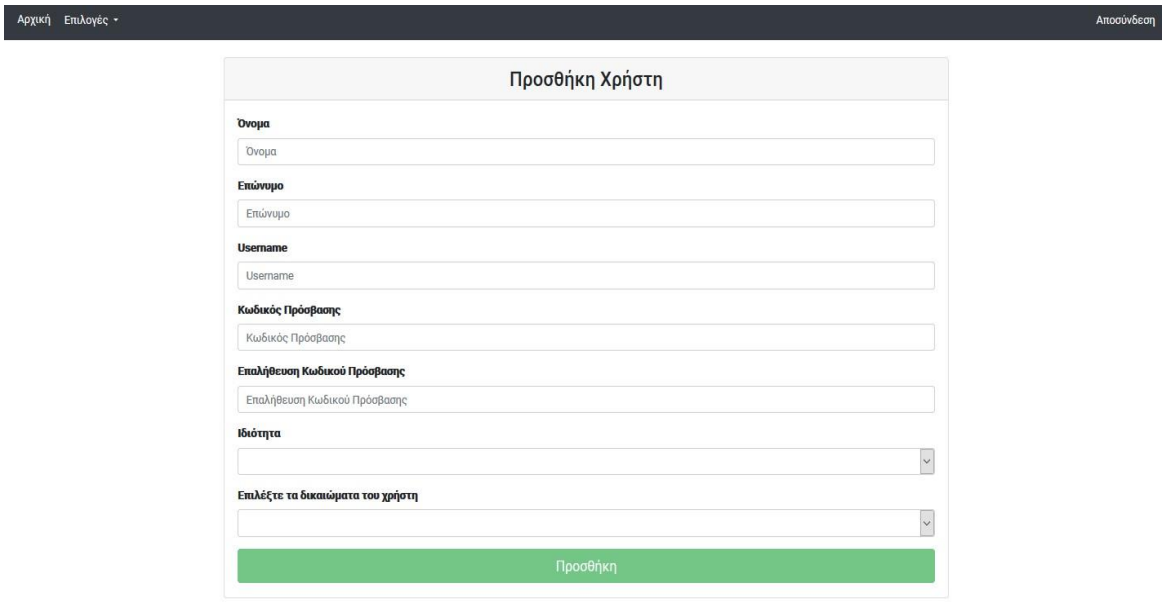
Φόρμα Προσθήκης Στοιχείων Συσκευασίας		
Όνομα Μεταβλητής	**Περιγραφή Πεδίου**	**Τύπος Component στο HTML**
dateOfPackaging	Ημερομηνία Συσκευασίας	<input matDatePicker>
checkOfHardnessOfKiwi	Έλεγχος Σκληρότητας Ακτινιδίου	<input type text>
checkOfBrix	Έλεγχος Σακχάρων	<input type text>
classOfKiwi	Κλάση Ακτινιδίου	<input type text>
numberOfKiwi	Νούμερο Ακτινιδίου	<select>
typeOfPackages	Είδος Συσκευασίας	<select>
packageCode	Κωδικός Παρτίδας Συσκευασίας	<input type text>

Πίνακας 16: Μοντέλο Φόρμας Προσθήκης Στοιχείων Συσκευασίας

8. Περιγραφή Λειτουργίας της Εφαρμογής

Υπάρχουν λοιπόν δύο είδη χρηστών, οι διαχειριστές (administrators) και οι απλοί χρήστες (users). Αρχικά, εμφανίζεται η φόρμα εισόδου ενός χρήστη. Είτε σαν διαχειριστής είτε σαν απλός χρήστης, πρέπει να συμπληρωθεί το username και τον κωδικό πρόσβασης, ώστε να επιτραπεί η είσοδος στο κύριο μενού. Αν λοιπόν ένας διαχειριστής κάνει επιτυχή είσοδο, μεταφέρεται στο dashboard που έχει σχεδιαστεί για αυτόν. Εκεί, υπάρχει ο πίνακας με τους διαθέσιμους απλούς χρήστες που έχουν δημιουργηθεί. Σε αυτόν τον πίνακα, ο διαχειριστής μπορεί να επιλέξει να επεξεργαστεί κάποια στοιχεία ενός χρήστη ή και να τον διαγράψει, πατώντας πάνω στο αντίστοιχο κουμπί. Αν επιλέξει την επεξεργασία, μεταφέρεται στη φόρμα που μπορεί να εισάγει τα νέα στοιχεία και να τα αποθηκεύσει. Επίσης, όπως αναφέρθηκε στην αρχή, ο διαχειριστής μπορεί να προσθέσει χρήστες. Μπαίνοντας στο μενού «Επιλογές», πατάει στην «Προσθήκη Χρήστη», όπου και μεταφέρεται στην αντίστοιχη φόρμα Προσθήκης Χρήστη.

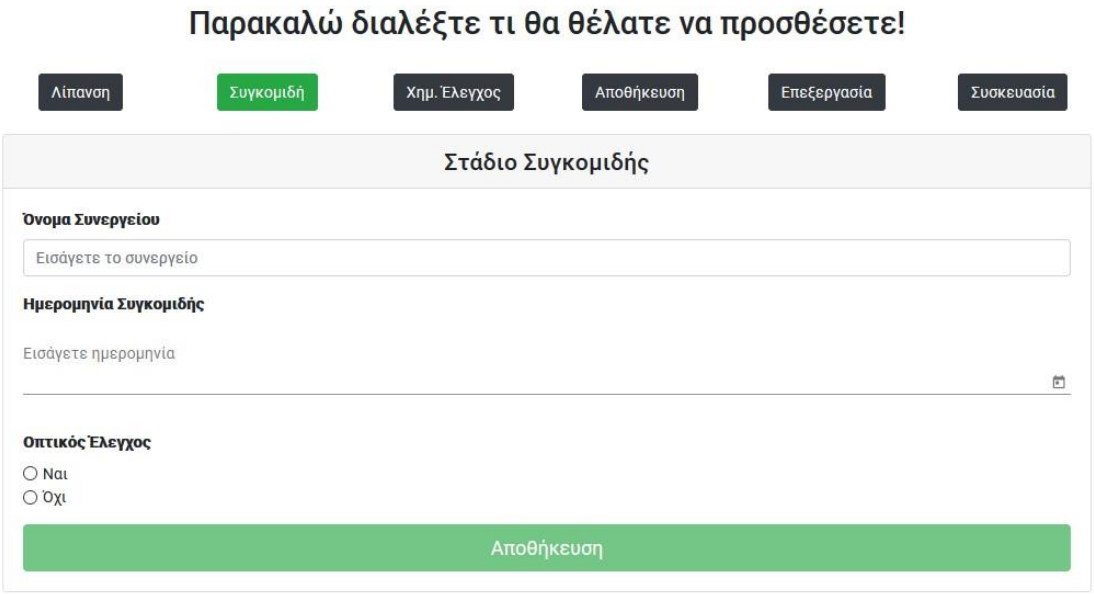
Εικόνα 17: Το dashboard των διαχειριστών



Εικόνα 18: Η φόρμα προσθήκης χρήστη

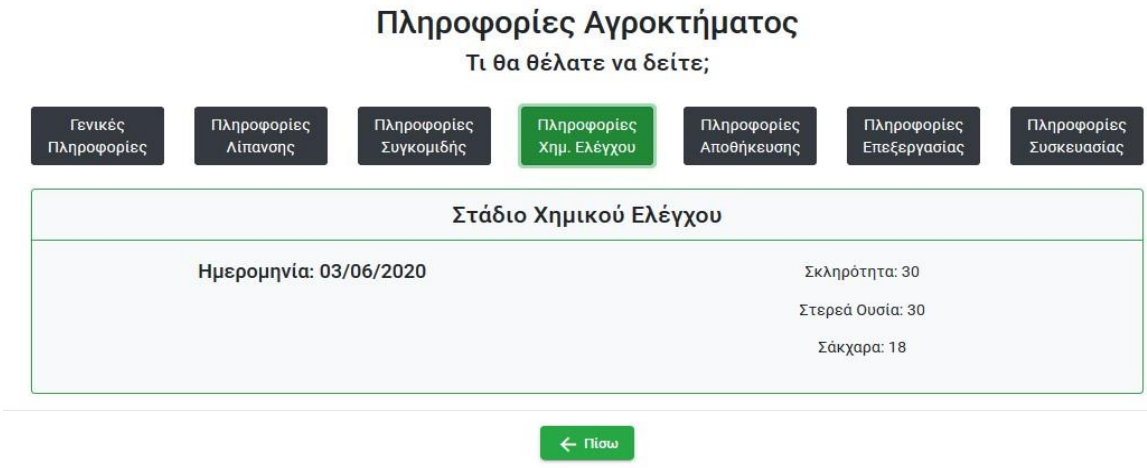
Από την άλλη πλευρά, αν ένας απλός χρήστης συνδεθεί επιτυχώς, μεταφέρεται στο κεντρικό dashboard που έχει δημιουργηθεί για αυτόν και έχει τελείως διαφορετικές δυνατότητες, αλλά και διαφορετικά δικαιώματα από αυτά του διαχειριστή. Στο κεντρικό dashboard, εμφανίζεται η λίστα με όλα τα αγροκτήματα ακτινιδίων που έχουν προστεθεί με κάποια από τα στοιχεία τους. Στο μενού «Επιλογές», ο χρήστης έχει τη δυνατότητα να αλλάξει τον κωδικό πρόσβασής του, αλλά και το πιο σημαντικό, να προσθέσει ένα αγρόκτημα. Οποιαδήποτε από τις δύο επιλογές επιλέξει, θα μεταφερθεί στην αντίστοιχη φόρμα, όπου θα του ζητηθεί να τη συμπληρώσει. Επίσης, ο χρήστης έχει δυνατότητα να διαγράψει πατώντας στο κεντρικό dashboard το αντίστοιχο κουμπί. Τα άλλα δύο κουμπιά δίπλα σε αυτό της διαγραφής αφορούν την επεξεργασία και την προβολή λεπτομερών δεδομένων του αγροκτήματος.

Αν επιλεγεί το κουμπί της επεξεργασίας, ο χρήστης μεταφέρεται σε μια σελίδα όπου έχει να διαλέξει από μια πληθώρα φορμών συμπλήρωσης στοιχείων για το αγρόκτημα, ανάλογα φυσικά με την ανάγκη. Για παράδειγμα, αν επιλέξει την καρτέλα «συγκομιδή», θα εμφανιστεί η αντίστοιχη φόρμα, ώστε να προσθέσει τα στοιχεία για κάθε φορά που θα κάνει συγκομιδή.



Εικόνα 19: Η φόρμα προσθήκης στοιχείων για το Στάδιο Συγκομιδής

Όλα τα στοιχεία που προστίθενται για το αγρόκτημα, αλλά και τα στοιχεία που συμπλήρωσε ο χρήστης κατά την προσθήκη του αγροκτήματος, μπορούν να προβληθούν στη σελίδα πληροφοριών αν ο χρήστης πατήσει στο αντίστοιχο κουμπί στο κεντρικό dashboard. Όπως και στην περίπτωση της επεξεργασίας του αγροκτήματος, ο χρήστης επιλέγει την καρτέλα με τις πληροφορίες που τον ενδιαφέρουν και αυτές εμφανίζονται σε μια κάρτα.



Εικόνα 20: Η κάρτα πληροφοριών για το Στάδιο Χημικού Ελέγχου

9. Υλοποίηση Εφαρμογής

9.1 Environment μεταβλητές

Έχουν δημιουργηθεί στην εφαρμογή αρκετές environment μεταβλητές. Οι environment μεταβλητές είναι μεταβλητές που περιέχουν κάποιο URL. Χρησιμοποιούνται για να γίνεται πιο εύκολη η χρήση κάποιου URL στα requests που γίνονται προς τον server. Όλα τα URLS συνηθίζεται να δηλώνονται σε ένα TypeScript αρχείο ονόματι environment.ts. Οι μεταβλητές που χρησιμοποιούνται στην εφαρμογή είναι τα παρακάτω.

Όνομα μεταβλητής	**Περιγραφή**
SIGNUP_URL	URL προσθήκης χρήστη
LOGIN_URL	URL εισόδου χρήστη
USERS_URL	URL χρηστών
USERS_DASHBOARD	URL κεντρικής σελίδας χρηστών
CHANGE_PASSWORD_URL	URL αλλαγής κωδικού πρόσβασης
ADD_FARM_URL	URL προσθήκης αγροκτήματος
GET_FARMS_URL	URL εμφάνισης αγροκτημάτων
ADD_FARM_FEATURES_URL	URL προσθήκης στοιχείων αγροκτήματος
ADD_HERBICIDES_EXISTENCE_URL	URL ύπαρξης ζιζανιοκτόνων
ADD_SITOFEX_EXISTENCE_URL	URL ύπαρξης sitofex

Πίνακας 17: Οι environment μεταβλητές της εφαρμογής

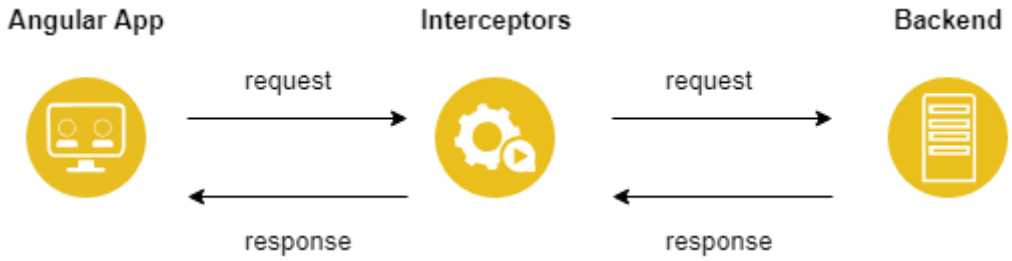
9.2 Κατασκευή ενός token interceptor (token-based authentication), αποθήκευση του token

Ο έλεγχος ταυτότητας με βάση το token (token-based authentication) είναι μόνο μία από τις πολλές μεθόδους ελέγχου ταυτότητας ιστού που χρησιμοποιούνται για τη δημιουργία μιας πιο ασφαλούς διαδικασίας επαλήθευσης ταυτότητας χρήστη.

Τα πιο συνηθισμένα συστήματα διακριτικών περιέχουν μια κεφαλίδα (header), ωφέλιμο φορτίο (payload) και υπογραφή (signature). Η κεφαλίδα αποτελείται από τον τύπο ωφέλιμου φορτίου, καθώς και τον αλγόριθμο υπογραφής που χρησιμοποιείται. Το ωφέλιμο φορτίο περιέχει τα claims, τα οποία είναι απλώς οποιαδήποτε στοιχεία που αφορούν τον χρήστη. Η υπογραφή είναι ακριβώς όπως ακούγεται - η υπογραφή που χρησιμοποιείται για να αποδείξει ότι το μήνυμα δεν έχει εκτεθεί κατά τη μεταφορά. Αυτά τα τρία στοιχεία

συνεργάζονται για τη δημιουργία ενός πολύ αποτελεσματικού και ασφαλούς συστήματος ελέγχου ταυτότητας.

Με απλά λόγια, δημιουργείται στο backend το token, το οποίο στέλνεται στο frontend και αποθηκεύεται συνήθως στο localStorage του browser. Για να μπορέσει να επαληθεύσει ο server το token, δημιουργείται ένα token interceptor, το οποίο υλοποιείται με βάση το HTTP Interceptor της Angular. Οπότε, τα HTTP requests που στέλνονται στο server ανακόπτονται για λίγο και μετατρέπονται έτσι ώστε να περιέχουν μέσα και το token και στη συνέχεια πηγαίνουν ολοκληρωμένα στο server. Όταν φτάσουν στο server, το token ελέγχεται αν είναι έγκυρο ή όχι και στέλνεται το ανάλογο response.



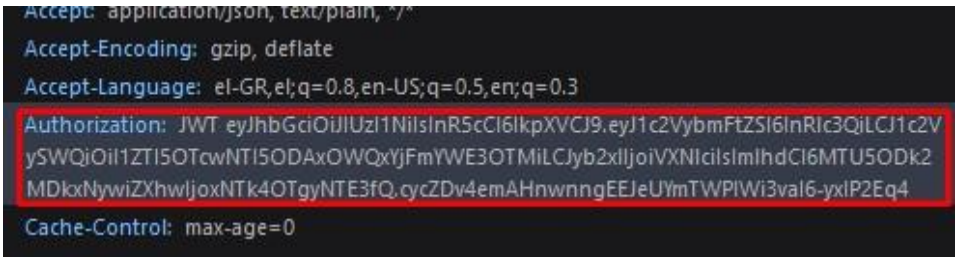
Εικόνα 21: Απεικόνιση της λειτουργίας των HTTP Interceptors

Πρωταρχικό μέλημα λοιπόν είναι να αποθηκευτεί το token που έρχεται σαν response από τον server. Έτσι, κατά τη στιγμή του login που έρχεται response, θα αποθηκεύεται το token στο localStorage. Δημιουργείται το token-interceptor, το οποίο είναι ένα service, άρα θα τοποθετηθεί στο φάκελο με τα services. Η επόμενη εικόνα περιέχει τον κώδικα στο token interceptor.

```
import {Injectable} from '@angular/core';
import {HttpEvent, HttpHandler, HttpInterceptor, HttpRequest} from '@angular/common/http';
import {AuthService} from './auth.service';
import {Observable} from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TokenInterceptorService implements HttpInterceptor {

  constructor(private auth: AuthService) {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

    const token = this.auth.getToken();
    req = req.clone( update: {
      headers: req.headers.set('Authorization', 'JWT ' + token)
    });
    return next.handle(req);
  }
}
```

Εικόνα 22: Ο κώδικας του token interceptor

Για να γίνει αντιληπτό όμως αν δουλεύει το token interceptor που δημιουργήθηκε, μπορούμε να ελέγξουμε στον browser μας την Εποπτεία Δικτύου και να δούμε τα headers (κεφαλίδες). Για παράδειγμα, βλέπουμε πως υπάρχει το παρακάτω token. Αν αντιγράψουμε το token και το κάνουμε επικόλληση σε έναν ιστότοπο που κάνει αποκρυπτογράφηση tokens, παρατηρούμε το token είναι επαληθευμένο μιας και περιέχει κρυπτογραφημένα τα στοιχεία του χρήστη test που υπάρχει στην εφαρμογή.



Εικόνα 23: Το token στην κεφαλίδα

Εικόνα 24: Το αποκρυπτογραφημένο token

9.3 Δημιουργία Route Guards και προσθήκη αυτών στις σελίδες – paths του app-routing.module.ts

Τα route guards είναι interfaces (διεπαφές) που μπορούν να πουν στο δρομολογητή εάν θα πρέπει ή όχι να επιτραπεί η πλοήγηση σε μια ζητούμενη σελίδα. Η απόφαση αυτή λαμβάνεται από την εύρεση μιας true ή false τιμής σε μια κλάση που εφαρμόζει ένα guard. Τα route guards είναι πολύ χρήσιμα λοιπόν αφού δεν πρέπει όλες οι σελίδες να έχουν πρόσβαση από όλους. Στην εφαρμογή υλοποιηθήκαν 3 route guards, τα **auth.guard.ts**, **admin.guard.ts** και **user.guard.ts**. Δημιουργήθηκε ένας φάκελος guards που θα περιέχει όλα τα guards.

9.3.1 auth.guard.ts

Το **auth.guard.ts** προέρχεται από το authentication guard (φρουρός αυθεντικοποίησης). Κύρια μέλημά του είναι να μην επιτρέπει σε κάποιον χρήστη να εισέλθει σε κάποια σελίδα χωρίς να έχει κάνει επιτυχή είσοδο στη φόρμα Εισόδου χρήστη. Για παράδειγμα, ακόμα και αν ο χρήστης πληκτρολογήσει το URL της φόρμας Προσθήκης Αγροκτήματος, χάρη στο **auth.guard.ts** δε θα μπορέσει να περιηγηθεί εκεί που ζήτησε και θα ανακατευθυνθεί στη φόρμα εισόδου χρήστη. Αν επιστραφεί τιμή true, ο guard επιτρέπει την πρόσβαση στη σελίδα. Αυτό υλοποιήθηκε δημιουργώντας μια μέθοδο ονόματι **isAuthenticated** στο **auth.service.ts**, που ελέγχει αν υπάρχει ή όχι το token του χρήστη. Οπότε αν δεν υπάρχει, δεν επιτρέπεται και η πρόσβαση στη σελίδα.

```
@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  private readonly notifier: NotifierService;

  constructor(private auth: AuthService,
              private router: Router,
              private notifierService: NotifierService) {
    this.notifier = notifierService;
  }

  canActivate(): boolean {
    if (this.auth.isAuthenticated()) {
      return true;
    } else {
      this.notifier.notify( type: 'error', message: 'Δεν επιτρέπεται η είσοδος στη διεύθυνση που ζητήσατε!');
      this.router.navigate( commands: ['login']);
      return false;
    }
  }

}
```

Εικόνα 25: Ο κώδικας του auth.guard.ts

9.3.2 admin.guard.ts

Ο admin guard εργάζεται σχεδόν όπως και ο authentication guard. Η μόνη διαφορά είναι πως αντί να ελέγχει αν υπάρχει το token αποθηκευμένο στο Local Storage του browser, ελέγχει αν η μεταβλητή role που είναι αποθηκευμένη είναι Admin. Με αυτό τον τρόπο, αποφεύγεται η πιθανότητα πρόσβασης ενός απλού χρήστη σε σελίδα που προορίζεται για διαχειριστές αν πληκτρολογήσει συγκεκριμένο URL.

```
@Injectable({
  providedIn: 'root'
})
export class AdminGuard implements CanActivate {
  private readonly notifier: NotifierService;

  constructor(private auth: AuthService,
              private notifierService: NotifierService,
              private router: Router) {
    this.notifier = notifierService;
  }

  canActivate(): boolean {
    this.auth.role.subscribe();
    if (localStorage.getItem( key: 'role') === 'Admin') {
      return true;
    } else {
      this.notifier.notify( type: 'error',  message: 'Μη εξουσιοδοτημένη πρόσβαση στη σελίδα που ζητήσατε!');
      this.router.navigate( commands: ['']);
      return false;
    }
  }
}
```

Εικόνα 26: Ο κώδικας του admin.guard.ts

9.3.3 user.guard.ts

Ο user guard δουλεύει ακριβώς όπως και ο admin guard απλά με τη διαφορά ότι δεν επιτρέπει σε κάποιον διαχειριστή να εισέλθει σε σελίδα που προορίζεται για απλό χρήστη.

9.3.4 Προσθήκη των guards στις σελίδες-paths

Η δημιουργία και η υλοποίηση των guards δεν έχουν κανένα απολύτως νόημα αν δεν τους ανατεθεί να ελέγχουν τις σελίδες.

Όλες οι σελίδες που έχουν δημιουργηθεί μέχρι στιγμής γράφονται στο Routes, μια σταθερά βρισκόμενη στο αρχείο **app-routing.module.ts** που δημιουργείται αυτόματα όταν δημιουργηθεί και το project. Η συγκεκριμένη σταθερά περιέχει το όνομα του path που χρειάζεται να έχει μια σελίδα, το component που ανήκει η σελίδα και τα guards αν φυσικά

υπάρχουν. Υπάρχουν βέβαια και πολλές άλλες επιλογές όπως να βάλεις να ανακατευθύνεται ένα path σε ένα άλλο κλπ.

Αυτό που πρέπει να γίνει είναι να προστεθούν τα guards σε κάθε σελίδα με τη χρήση του **canActivate**, ενός interface της Angular που επιτρέπει τη πλοήγηση αν και μόνο αν οι guards επιστρέψουν την τιμή true.

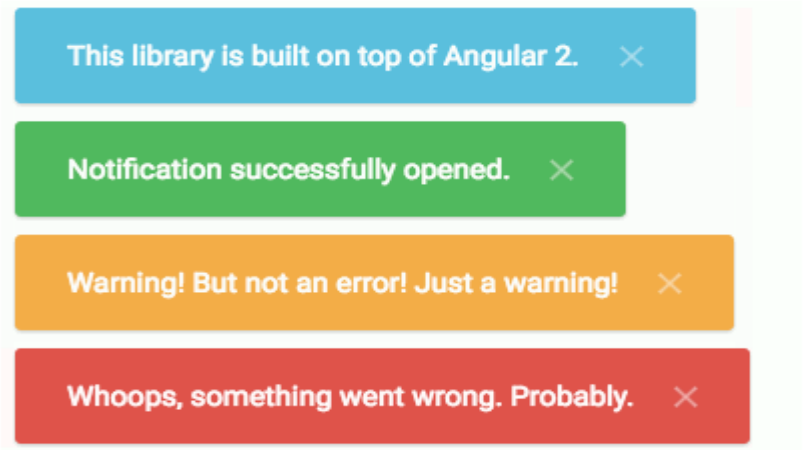
```
const routes: Routes = [
  {path: '', redirectTo: '/login', pathMatch: 'full'},
  {path: 'login', component: LoginComponent},
  {path: 'usersDashboard', component: UsersDashboardComponent, canActivate: [AuthGuard, UserGuard]},
  {path: 'farmDescription', component: FarmDescriptionComponent, canActivate: [AuthGuard, UserGuard]},
  {path: 'admin', component: AdminDashboardComponent, canActivate: [AuthGuard, AdminGuard]},
  {path: 'update/:id', component: UpdateUserComponent, canActivate: [AuthGuard, AdminGuard]},
  {path: 'registration', component: RegistrationComponent, canActivate: [AuthGuard, AdminGuard]},
  {path: 'changePassword', component: ChangePasswordComponent, canActivate: [AuthGuard, UserGuard]},
  {path: 'viewFarm/:id', component: ViewFarmComponent, canActivate: [AuthGuard, UserGuard]},
  {path: 'farmEdit/:id', component: FarmEditComponent, canActivate: [AuthGuard, UserGuard]}
];
```

Εικόνα 27: Τα routes της εφαρμογής

9.4 Ειδοποιήσεις και error interceptor

Πώς θα ήταν μια εφαρμογή χωρίς να έχει ειδοποιήσεις; Πώς θα ήταν σε αυτή την εφαρμογή να μην πάρει ειδοποίηση ο χρήστης αν για παράδειγμα άλλαξε τον κωδικό πρόσβασης; Η εμπειρία ενός χρήστη θα ήταν πολύ δύσκολη και είναι αδιανόητο να μην υπάρχουν ειδοποιήσεις σε μια εφαρμογή.

Οπότε υπάρχουν ειδοποιήσεις για οτιδήποτε μπορεί να συμβεί κατά τη διάρκεια χρήσης της εφαρμογής από κάποιον χρήστη. Υπάρχουν μηνύματα για επιτυχία, προειδοποιήσεις, κίνδυνο κλπ.

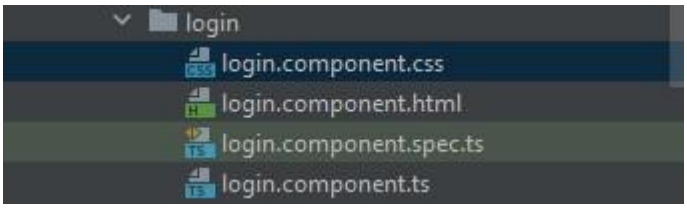


Εικόνα 28: Διάφορα είδη ειδοποιήσεων με τη βιβλιοθήκη angular-notifier

Ο error interceptor είναι ένα αρχείο που διαχειρίζεται τα σφάλματα που μπορούν να συμβούν και ενημερώνει το χρήστη για το τι έχει συμβεί. Δεν δημιουργήθηκε σαν service όπως και ο token interceptor, αλλά σαν component**.** Περιέχει σφάλματα όπως ότι ο παλιός κωδικός πρόσβασης δε πληκτρολογήθηκε σωστά ή ότι το username που πληκτρολόγησε ο διαχειριστής δεν είναι έγκυρο, αλλά και άλλα πολλά.

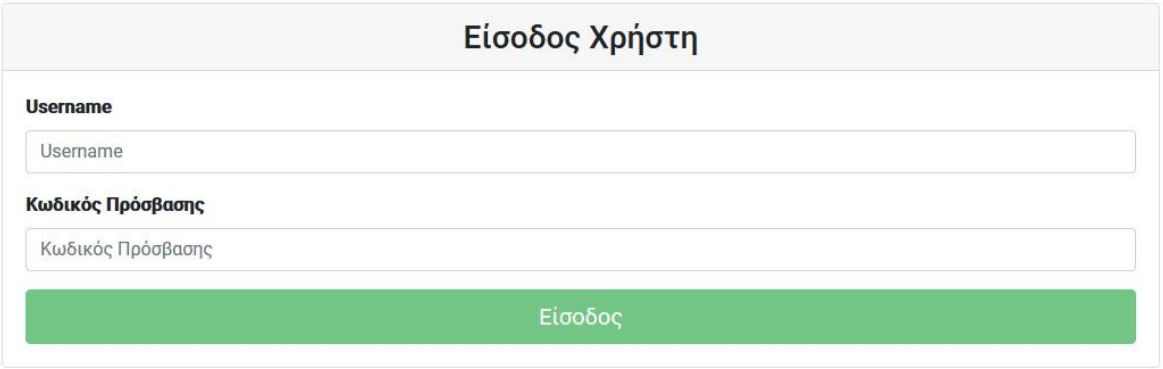
9.5 Φόρμα εισόδου χρήστη (login component)

Επόμενο βασικό βήμα είναι η κατασκευή της φόρμας εισόδου χρήστη. Σύμφωνα με τις απαιτήσεις που είχε η εφαρμογή, η συγκεκριμένη φόρμα θα πρέπει να είναι και η αρχική/πρώτη σελίδα που θα εμφανίζεται όταν κάποιος θέλει να περιηγηθεί σε αυτή.
Όλα ξεκινούν με τη δημιουργία του component. Το component περιέχει ένα TypeScript, ένα HTML, ένα CSS και ένα TypeScript spec αρχείο με την ονομασία που του έχει δοθεί. Στην περίπτωση αυτή ονομάζεται login.



Εικόνα 29: Η δομή του login component, όπως φαίνεται στο WebStorm

Αρχικά, δηλώνεται και σχεδιάζεται τη φόρμα στο TypeScript αρχείο γράφοντας τα ονόματα των πεδίων που πρέπει να περιέχει. Επίσης, μπορούν στα πεδία να προστεθούν διάφορα Validators, πχ το πεδίο να είναι μόνο αριθμός, ώστε σε περίπτωση που ο χρήστης πληκτρολογήσει κάτι διαφορετικό, να του εμφανιστεί το σφάλμα. Ύστερα, ξεκινάει ο σχεδιασμός στο HTML αρχείο και με τη βοήθεια και του CSS έρχεται το επιθυμητό αποτέλεσμα στην εμφάνιση.



Εικόνα 30: Η Φόρμα Εισόδου Χρήστη

9.6 Δημιουργία Authentication Service και σύνδεση της φόρμας εισόδου χρήστη με τη βάση δεδομένων

Τα services στην Angular είναι αντικείμενα που δημιουργούνται μόνο μια φορά κατά τη διάρκεια ζωής μιας εφαρμογής. Περιέχουν μεθόδους που διατηρούν δεδομένα καθ' όλη τη διάρκεια ζωής της εφαρμογής. Σαν κύριος σκοπός ενός service, είναι να οργανώνει και να μοιράζεται με τα άλλα components δεδομένα. Για παράδειγμα, μπορεί να γραφεί ένα service που θα περιέχει κώδικα για τη λήψη δεδομένων από τον server ή για την αποστολή κάποιων δεδομένων αντίστοιχα. Τα δεδομένα που θα ληφθούν, υπάρχει η δυνατότητα να χρησιμοποιηθούν με διάφορους τρόπους στην εφαρμογή.

Έχοντας σχεδόν ολοκληρωθεί η φόρμα εισόδου χρήστη, πρέπει να υλοποιηθεί τη σύνδεση με τον server, δηλαδή να σταλθούν τα στοιχεία που πληκτρολογήθηκαν στα πεδία στο server. Στην αυτή την περίπτωση πρέπει να σταλθούν δεδομένα, επομένως θα χρησιμοποιηθεί ένα post request που θα περιέχει τα στοιχεία της φόρμας. Αφότου υλοποιηθεί η απαραίτητη μέθοδος στο service, στο **login.component.ts** υλοποιείται μια μέθοδος, η οποία θα καλεί την μέθοδο του service όταν θα πατηθεί το κουμπί της υποβολής της φόρμας. Αν και εφόσον είναι σωστά τα στοιχεία που πληκτρολόγησε ο χρήστης, θα συνδεθεί επιτυχώς, αλλιώς θα εμφανιστεί στη σελίδα ένα μήνυμα σφάλματος.

```
24    loginUser(credentials: UsersModel) {
25      return this.http.post<any>(environment.LOGIN_URL, credentials);
26    }
```

Εικόνα 31: Η μέθοδος για την αποστολή των δεδομένων της φόρμας

9.7 Φόρμα εγγραφής χρηστών (registration component)

Βασική επόμενη κίνηση είναι η δημιουργία της φόρμας εγγραφής χρηστών. Πρόκειται για ένα από τα δικαιώματα των διαχειριστών, όπου θα εγγράφουν-προσθέτουν νέους χρήστες με τη χρήση της συγκεκριμένης φόρμας. Αξίζει να σημειωθεί πως η εγγραφή των διαχειριστών στο σύστημα δε γίνεται με κάποια φόρμα στην εφαρμογή. Η εγγραφή τους έγινε χειροκίνητα στη βάση δεδομένων στο backend.

Αρχικά, δημιουργείται το component**.** Ύστερα δημιουργείται η φόρμα στο TypeScript αρχείο, η οποία σχεδιάζεται και μορφοποιείται με τη βοήθεια του HTML και του CSS. Όπως και στην περίπτωση της φόρμας εισόδου χρήστη, πρέπει να σταλθούν τα δεδομένα της φόρμας στο server ώστε να εγγραφεί ο χρήστης. Κατασκευάζεται λοιπόν η μέθοδος με

το post request στο **auth.service.ts** αρχείο που έχει δημιουργηθεί βάζοντας το σωστό URL και ύστερα μια μέθοδο στο αρχείο **registration.component.ts** που θα καλεί αυτή του service. Εφόσον δεν υπάρξει κάποιο πρόβλημα, όπως για παράδειγμα να υπάρχει ήδη το username στη βάση δεδομένων, ο διαχειριστής έχει εγγράψει επιτυχώς το νέο χρήστη.

Στη συγκεκριμένη φόρμα, πρέπει να συμπληρωθεί Κωδικός Πρόσβασης και Επαλήθευση Κωδικού Πρόσβασης. Μιας και τα πεδία αυτά είναι τύπου password, δε φαίνεται τι ακριβώς πληκτρολογεί εκείνη τη στιγμή, με πιθανό αποτέλεσμα ο κωδικός και η επαλήθευση κωδικού να μην είναι ίδια. Σε αυτή την περίπτωση, χρειάστηκε να κατασκευαστεί ένα Angular Directive, όπου θα περιέχει έναν custom validator που θα ενεργοποιείται on-the-fly στο HTML όταν καταλαβαίνει πως δεν ταιριάζουν οι χαρακτήρες του πεδίου Κωδικός Πρόσβασης με το πεδίο Επαλήθευση Κωδικού Πρόσβασης και θα ειδοποιεί το χρήστη.

Γίνεται η δημιουργία του directive και τοποθετείται στο φάκελο registration. Αυτό βέβαια δε σημαίνει πως δε θα μπορεί να χρησιμοποιηθεί ξανά αν χρειαστεί σε κάποιο άλλο component, μιας και μπορεί να χρησιμοποιηθεί κυριολεκτικά σχεδόν παντού. Ο συγκεκριμένος custom validator συγκρίνει δύο πεδία και αν δεν είναι ίδια, επιστρέφει true, αλλιώς επιστρέφει null.

```
import { Directive, Input } from '@angular/core';
import {AbstractControl, NG_VALIDATORS, Validator} from '@angular/forms';

@Directive({
  selector: '[appConfirmEqualValidator]',
  providers: [{
    provide: NG_VALIDATORS,
    useExisting: ConfirmEqualValidatorDirective,
    multi: true
  }]
})

// Εδώ ο κώδικας για τη σύγκριση των passwords. Δηλαδή: password, confirmPass -
export class ConfirmEqualValidatorDirective implements Validator {
  @Input() appConfirmEqualValidator: string;
  validate(control: AbstractControl): { [key: string]: any} | null {
    const controlToCompare = control.parent.get(this.appConfirmEqualValidator);
    if (controlToCompare && controlToCompare.value !== control.value) {
      return { notEqual: true};
    }
    return null;
  }

  constructor() { }

}
```

Εικόνα 32: Ο κώδικας του custom validator

Εικόνα 33: Το μήνυμα σφάλματος ότι οι 2 κωδικοί δεν είναι ίδιοι

9.8 Δημιουργία του dashboard του διαχειριστή και της φόρμας επεξεργασίας χρήστη (admin-dashboard component και update-user component)

Αφού ολοκληρώθηκαν δύο από τις βασικότερες φόρμες της εφαρμογής, πρέπει να δημιουργηθεί το dashboard των χρηστών που θα είναι και η σελίδα που θα μεταβαίνουν οι χρήστες αφού συνδεθούν επιτυχώς.

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, το dashboard του διαχειριστή θα περιέχει έναν πίνακα που με τη σειρά του θα περιέχει κάποια στοιχεία των εγγεγραμμένων χρηστών (απλών χρηστών). Αρχικά, δηλώνουμε έναν πίνακα στο TypeScript αρχείο που ουσιαστικά θα περιέχει δεδομένα τύπου **UsersModel**. Έπειτα δημιουργούμε στο HTML τον πίνακα που θα περιλαμβάνει τα **Όνομα, Επώνυμο, Username** και **Ιδιότητα** των χρηστών, αλλά και τα κουμπιά **Επεξεργασίας** και **Διαγραφής**.

Αφού τελειώσει ο σχεδιασμός του πίνακα, θα δημιουργηθεί ένα καινούριο service, όπου θα περιλαμβάνονται μέθοδοι που θα αφορούν τους χρήστες. Έτσι, στο φάκελο services θα δημιουργηθεί το **users.service.ts**. Η κύρια μέθοδος που πρέπει να υλοποιηθεί στο καινούριο service είναι μια getUsers. Η **getUsers** θα επιστρέφει όλους τους χρήστες που υπάρχουν εγγεγραμμένοι στη βάση δεδομένων. Άρα θα χρησιμοποιηθεί ένα get request, μιας και χρειάζεται να ληφθούν δεδομένα από τον server. Φυσικά, αφού ολοκληρωθεί η **getUsers**, καλείται στο **admin-dashboard.component.ts**.

```
getUsers(): Observable<UsersModel[]> {
  return this.http.get<UsersModel[]>( url: environment.USERS_URL + '/getAll');
}
```

Εικόνα 34: Η getUsers στο users.service.ts

Επόμενο βήμα είναι η σύνδεση του κουμπιού της διαγραφής των χρηστών. Όταν ένας διαχειριστής αποφασίσει να διαγράψει κάποιον χρήστη, πατάει στο κουμπί της διαγραφής.

Οπότε θα δημιουργηθεί μια μέθοδος **deleteUser** στο **users.service.ts** που θα στέλνει ένα delete request στον server με το id του χρήστη που έχει ζητηθεί η διαγραφή του. Θα κατασκευαστεί επίσης και μια μέθοδος στο **admin-dashboard.component.ts** που θα καλεί την deleteUser όταν πατηθεί το κουμπί Διαγραφή. Αφού γίνει επιτυχώς η διαγραφή, οι αλλαγές πρέπει να γίνουν άμεσα διακριτές στον πίνακα. Γι' αυτό το λόγο, κατασκευάζεται στο **admin-dashboard.component.ts** μια μέθοδος ονόματι refresh, η οποία όταν αντιληφθεί πως υπάρχει κάποια αλλαγή, θα κάνει ανανέωση τον πίνακα με τους χρήστες. Για να συμβεί αυτό, θα χρησιμοποιηθεί τη μέθοδο **detectChanges** που περιλαμβάνεται στην κλάση **ChangeDetectorRef** της Angular. Ουσιαστικά καλούμε τη **getUsers** ξανά αν υπάρξει κάποια αλλαγή στον πίνακα.

```
deleteUser(id: string): Observable<UsersModel> {
  const url = `${environment.USERS_URL}/${id}`;
  return this.http.delete<UsersModel>(url, httpOptions);
}
```

Εικόνα 35: Η μέθοδος deleteUser στο users.service.ts

Αν ο διαχειριστής πατήσει στο κουμπί **Επεξεργασία**, μεταφέρεται σε μια νέα σελίδα που περιέχει τη φόρμα Επεξεργασίας Χρήστη. Όπως και στις προηγούμενες περιπτώσεις, αφού πρόκειται για νέα σελίδα, θα πρέπει να κατασκευαστεί ένα component. Στη φόρμα αυτή, ο διαχειριστής μπορεί να αλλάξει αν επιθυμεί τα πεδία **Όνομα, Επώνυμο, Username** και **Ιδιότητα** του επιλεγμένου χρήστη. Πάνω στα πεδία της φόρμας, αναγράφονται τα τωρινά στοιχεία του χρήστη π.χ. Username=>test. Για να εμφανίζονται τα στοιχεία αυτά στη σελίδα, αλλά και για να μπορεί να γίνει η επεξεργασία του συγκεκριμένου χρήστη, χρειάζεται μια μέθοδος στο **users.service.ts**. Όπως και στην περίπτωση της Διαγραφής ενός χρήστη, χρησιμοποιείται το id του, αλλά αυτή τη φορά με ένα get request προς τον server. Η μέθοδος ονομάζεται **getUserDetails**.

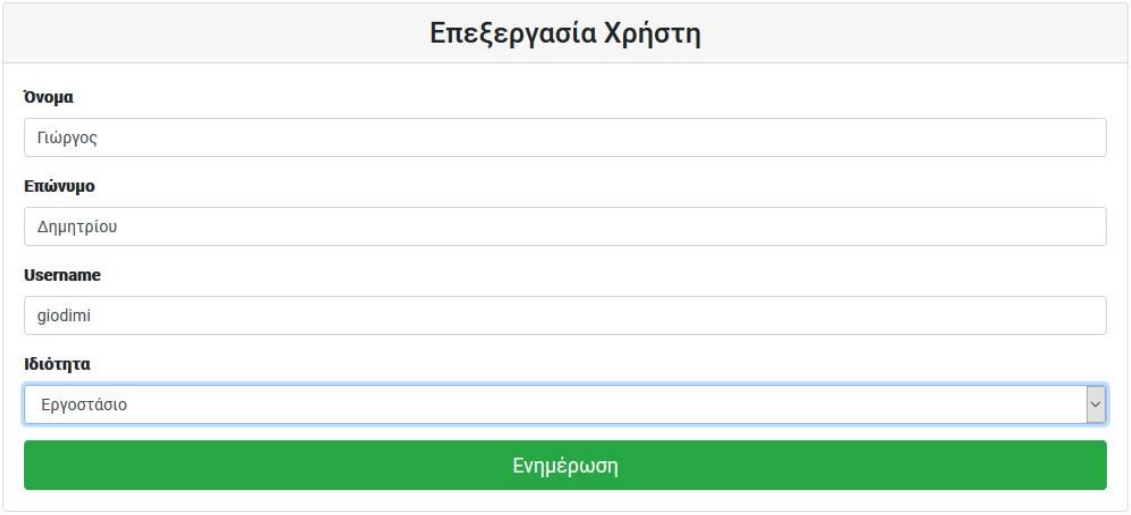
```
getUserDetails(id: string): Observable<UsersModel> {
  const url = `${environment.USERS_URL}/${id}`;
  return this.http.get<UsersModel>(url);
}
```

Εικόνα 36: Η getUserDetails στο users.service.ts

Ο διαχειριστής μπορεί να επεξεργαστεί όποιο από τα τέσσερα πεδία που προαναφέρθηκαν πιο πάνω ή ακόμα και όλα. Όταν πατηθεί το κουμπί **Ενημέρωση**, καλείται μέσα στη μέθοδο **onUpdate** του **update-user.component.ts** η μέθοδος **updateUser** που βρίσκεται στο **users.service.ts** και στέλνει ένα patch request στο server για να αντικαταστήσει τα ήδη υπάρχοντα δεδομένα.

```
updateUser(credentials: UsersModel, id: string) {
  const url = `${environment.USERS_URL}/${id}`;
  return this.http.patch<UsersModel>(url, credentials);
}
```

Εικόνα 37: Η μέθοδος updateUser στο users.service.ts



Εικόνα 38: Η φόρμα Επεξεργασίας Χρήστη

9.9 Υλοποίηση μεθόδου logout (Αποσύνδεσης)

Αφού ολοκληρωθεί το κομμάτι του authentication που επιτρέπει σε κάποιον συνδεδεμένο χρήστη ανεξαρτήτως δικαιωμάτων (είτε διαχειριστής είτε απλός χρήστης) να έχει συνεδρία (session), πρέπει να του δοθεί η δυνατότητα να αποσυνδέεται. Για αυτό το λόγο δημιουργείται στο **auth.service.ts** μια μέθοδος **logoutUser**. Στη συγκεκριμένη μέθοδο το βασικό που πρέπει να γίνει, είναι αφαιρεθεί το token από το localStorage. Η μέθοδος καλείται όταν ο εκάστοτε χρήστης επιθυμεί να αποσυνδεθεί και πατά το αντίστοιχο κουμπί.

```
logoutUser() {
  localStorage.removeItem( key: 'token');
  localStorage.removeItem( key: 'status');
  localStorage.removeItem( key: 'role');
  this.router.navigate( commands: ['login']);
}
```

Εικόνα 39: Η μέθοδος logoutUser στο auth.service.ts

8.10 Κατασκευή φόρμας Προσθήκης Αγροκτήματος, service αγροκτημάτων και δημιουργία του dashboard του απλού χρήστη(farm-description component, farms service και users-dashboard component)

Όπως έχει αναφερθεί και πιο πριν, στο dashboard του απλού χρήστη εμφανίζεται η λίστα όλων των εγγεγραμμένων αγροκτημάτων ακτινιδίων. Οπότε σειρά παίρνει η κατασκευή της φόρμας για την Προσθήκη Αγροκτήματος. Επειδή η φόρμα αφορά περισσότερο τα αγροκτήματα και όχι τόσο το χρήστη και επειδή θα υπάρξουν και άλλες φόρμες που αφορούν το αγρόκτημα, δημιουργείται ένας φάκελος farms. Να σημειωθεί πως το component δεν ονομάστηκε για παράδειγμα add-farm ή κάτι παρόμοιο, αλλά farm-description επειδή ουσιαστικά περιγράφεται το αγρόκτημα από τα πεδία που περιλαμβάνονται στη φόρμα.

```
this.farmDescriptionForm = this.fb.group( controlsConfig: {
  farmerName: ['', [Validators.required, Validators.minLength( minLength: 2)]],
  farmCode: ['', [Validators.required, Validators.pattern( pattern: '^[0-9]*$')]],
  farmerContactPhone: ['', [Validators.minLength( minLength: 10), Validators.maxLength( maxLength: 10), Validators.pattern( pattern: '^[0-9]*$')]],
  farmPlantingYear: ['', [Validators.required]],
  farmCoordinates: this.fb.group( controlsConfig: {
    latitude: ['', [Validators.required, Validators.pattern( pattern: '^[+-]?\\d+(\\.\\d+)?$')]],
    longitude: ['', [Validators.required, Validators.pattern( pattern: '^[+-]?\\d+(\\.\\d+)?$')]]
  }),
  farmTown: ['', [Validators.pattern( pattern: '^[a-zA-Zα-ωA-Oά-ώΑ-Ω]*$')]],
  farmArea: [''],
  farmAreaToponyms: [''],
  farmAcres: ['', [Validators.pattern( pattern: '^[0-9]*$')]],
  ph: ['', [Validators.min( min: 0), Validators.max( max: 14), Validators.pattern( pattern: '^[0-9]*$')]],
  electricalConductivity: ['', [Validators.pattern( pattern: '^[0-9]*$')]],
  farmSoil: ['', [Validators.required]],
  farmIrrigationWater: this.fb.group( controlsConfig: {
    ph: ['', [Validators.required, Validators.min( min: 0), Validators.max( max: 14), Validators.pattern( pattern: '^[0-9]*$')]],
    ec: ['', [Validators.required, Validators.pattern( pattern: '^[0-9]*$')]],
  }),
  varietyOfKiwi: ['', [Validators.required]],
  barCode: ['', [Validators.required, Validators.pattern( pattern: '^[0-9]*$')]]
});
```

Εικόνα 40: Δημιουργία της φόρμας στο farm-description.component.ts

Δημιουργείται η φόρμα στο TypeScript αρχείο του component, βάζοντας και τους Validators που έχουν ζητηθεί από τις απαιτήσεις και τη σχεδιάζεται στο HTML αρχείο. Φυσικά, για να ολοκληρωθεί η προσθήκη του αγροκτήματος θα πρέπει να σταλθούν τα δεδομένα της φόρμας στο server. Σε αυτή την περίπτωση λοιπόν, κατασκευάζεται ένα service που θα περιέχει μεθόδους που θα αφορούν requests για χαρακτηριστικά και δεδομένα των αγροκτημάτων. Στο **farms.service.ts** λοιπόν που δημιουργήθηκε, θα υλοποιηθεί μια μέθοδος ονόματι **addFarm** που στέλνει με ένα post request τα δεδομένα στο

server και η οποία καλείται στο **farm-description.component.ts** όταν πατηθεί το κουμπί Προσθήκη της φόρμας.

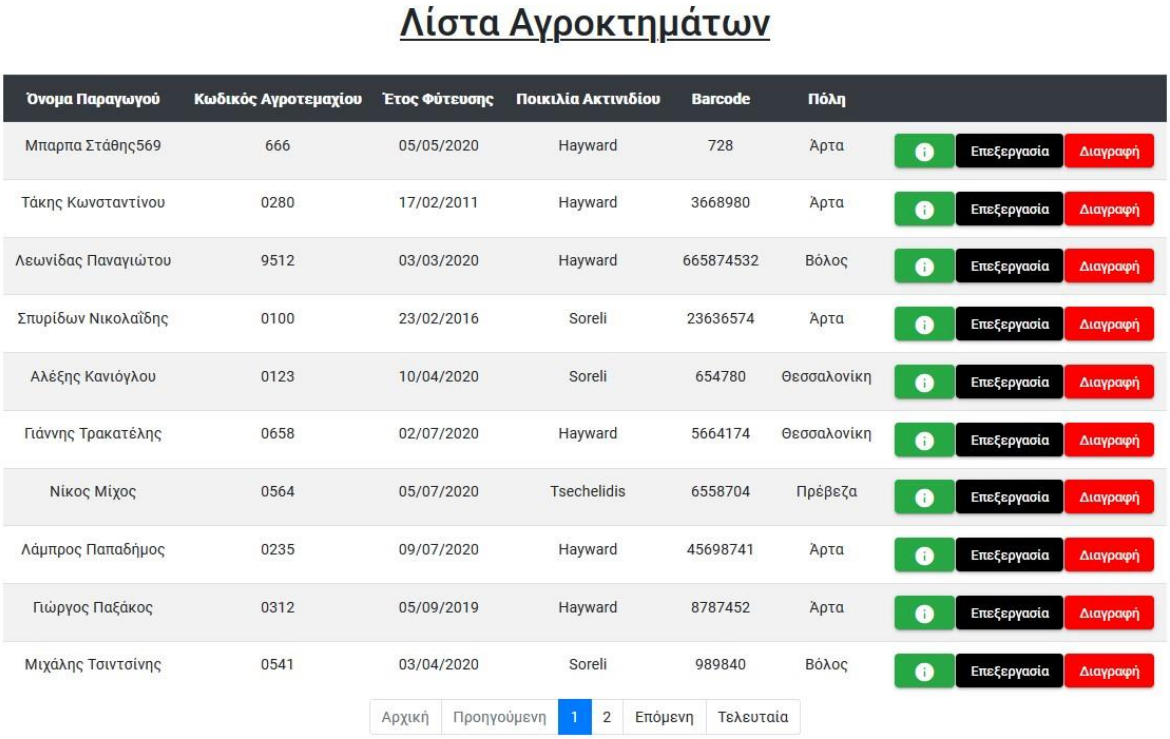
```
addFarm(credentials: FarmModel) {
  return this.http.post(environment.ADD_FARM_URL, credentials);
}
```

Εικόνα 41: Η μέθοδος addFarm στο farms.service.ts

Αφού ολοκληρώθηκε και ο σχεδιασμός της φόρμας, ήρθε η στιγμή να σχεδιαστεί και το dashboard των χρηστών, ώστε να εμφανίζονται όσα προσθέτει κάποιος χρήστης. Δημιουργείται ένας φάκελος users και μέσα σε αυτόν το users-dashboard component.

Αφού δημιουργηθεί το component, δηλώνεται ο πίνακας στο **users-dashboard.component.ts** και στη συνέχεια σχεδιάζεται στο **users-dashboard.component.html**. Ο πίνακας δε θα περιέχει όλα τα δεδομένα ενός αγροκτήματος, αλλά μόνο τα **Όνομα Παραγωγού, Κωδικός Αγροτεμαχίου, Έτος Φύτευσης, Ποικιλία Ακτινιδίου, Barcode** και **Πόλη**. Φυσικά θα υπάρχουν και τα κουμπιά για τις **Περισσότερες Πληροφορίες**, **Επεξεργασία** και **Διαγραφή**.

Όπως και στην περίπτωση του dashboard του διαχειριστή, για να εμφανιστούν τα δεδομένα στον πίνακα θα πρέπει να υλοποιηθεί στο **farms.service.ts** μια μέθοδος που θα στέλνει ένα get request στο server κάθε φορά που χρησιμοποιείται το συγκεκριμένο component. Να σημειωθεί πως μέθοδος ονομάζεται **getProducts** και όχι getFarms για λόγους σχεδιασμού στο backend.



Εικόνα 42: Ο πίνακας με τα αγροκτήματα

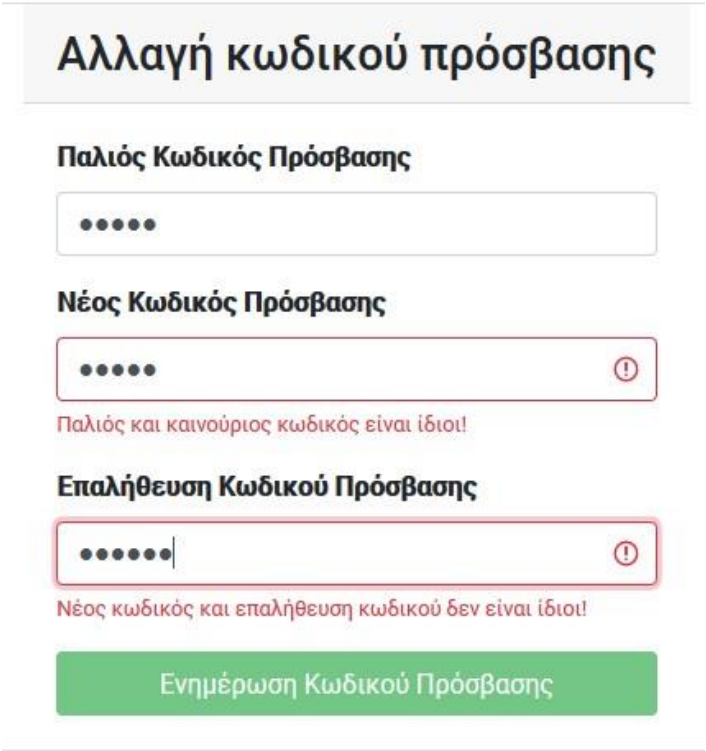
Όπως φαίνεται στην Εικόνα 42, κάτω από τον πίνακα ο χρήστης μπορεί να επιλέξει σελίδα για να περιηγηθεί ανάμεσα στα αγροκτήματα. Προστέθηκε στον πίνακα η δυνατότητα σελιδοποίησης, καθώς υπάρχει περίπτωση να υπάρχει μεγάλος αριθμός δεδομένων, το οποίο σημαίνει και πως δυσκολεύει την περιήγηση του χρήστη αλλά και κάνει το server να δουλεύει πιο αργά. Η υλοποίηση που έγινε με τη λογική σελιδοποίησης από πλευράς server (server-side pagination). Σχεδόν όλη η δουλειά γίνεται στο server, όπου κανονίζει πόσα αποτελέσματα ουσιαστικά θα εμφανίζονται ανά σελίδα, αλλά και πόσες θα είναι οι σελίδες. Από πλευράς frontend, χρειάζεται η δήλωση του ενός αντικειμένου ονόματι **pager** στο **users-dashboard.component.ts**, το οποίο χρειάζεται στην ανάπτυξη κώδικα για τον τρόπο λειτουργίας και εμφάνισης του pagination στο **users-dashboard.component.html**. Για παράδειγμα, αν βρίσκεται στην τελευταία σελίδα αγροκτημάτων, το κουμπί **Τελευταία** του pagination θα είναι απενεργοποιημένο.

```
<div class="container">
  <ul *ngIf="pager.pages && pager.pages.length" class="pagination">
    <li [ngClass]="{disabled:pager.currentPage === 1}" class="page-item first-item">
      <a (click)="loadPage( page: 1 )" class="page-link">Αρχική</a>
    </li>
    <li [ngClass]="{disabled:pager.currentPage === 1}" class="page-item previous-item">
      <a (click)="loadPage( page: pager.currentPage - 1 )" class="page-link">Προηγούμενη</a>
    </li>
    <li *ngFor="let page of pager.pages" [ngClass]="{active:pager.currentPage === page}"
        class="page-item number-item">
      <a (click)="loadPage( page )" class="page-link">{{page}}</a>
    </li>
    <li [ngClass]="{disabled:pager.currentPage === pager.totalPages}" class="page-item next-item">
      <a (click)="loadPage( page: pager.currentPage + 1 )" class="page-link">Επόμενη</a>
    </li>
    <li [ngClass]="{disabled:pager.currentPage === pager.totalPages}" class="page-item last-item">
      <a (click)="loadPage( pager.totalPages )" class="page-link">Τελευταία</a>
    </li>
  </ul>
</div>
```

Εικόνα 43: Ο κώδικας στο HTML αρχείο για τη λειτουργία σελιδοποίησης

9.11 Φόρμα Αλλαγής Κωδικού Πρόσβασης (change-password component)

Ο χρήστης έχει τη δυνατότητα να αλλάξει τον αρχικό κωδικό πρόσβασης που του όρισε ο διαχειριστής. Οπότε επόμενο βήμα είναι η κατασκευή μιας φόρμας που σαν πεδία θα έχει τον παλιό κωδικό πρόσβασης, τον καινούριο και την επαλήθευση του καινούριου. Δημιουργείται στον υπάρχον φάκελο users το change-password component και αφού δηλωθεί και υλοποιηθεί η φόρμα με τους απαραίτητους Validators στο **change-password-component.ts**, σχεδιάζεται και στο HTML αρχείο. Έπειτα, θα κατασκευαστεί η μέθοδος **changePassword** στο **auth.service.ts**, ώστε να στέλνει να πεδία της φόρμας με ένα post request στον server και η οποία θα καλείται όταν πατηθεί το κουμπί της φόρμας για αποστολή των στοιχείων.



Εικόνα 44: Η φόρμα Αλλαγής Κωδικού Πρόσβασης

Όπως φαίνεται και στην Εικόνα 44, στο πεδίο Επαλήθευση Κωδικού Πρόσβασης χρησιμοποιείται ο custom Validator που είχε κατασκευαστεί για να ελέγχονται αν δύο πεδία δεν είναι ίδια. Πέραν όμως του συγκεκριμένου Validator, χρειάζεται και η κατασκευή ενός άλλου που αυτή τη φορά θα συγκρίνει τα πεδία και θα ειδοποιεί αν είναι ίδια. Αυτό συμβαίνει γιατί δεν πρέπει ο χρήστης να πληκτρολογήσει σαν καινούριο κωδικό τον παλιό του. Η μόνη διαφορά σε σχέση με τον κώδικα του προηγούμενου custom Validator πέραν

της ονομασίας εντοπίζεται σε έναν λογικό τελεστή. Στην περίπτωση σύγκρισης Παλιού με νέο κωδικό, ο τελεστής είναι το ίσο (===) αντί για το διάφορο (!==).

```
validate(control: AbstractControl): { [key: string]: any } | null {
  const controlToCompare = control.parent.get(this.appConfirmNotEqualValidator);
  if (controlToCompare && controlToCompare.value === control.value) {
    return {Equal: true};
  }
  return null;
```

Εικόνα 45: Ο κώδικας του confirm-not-equal-validator.directive.ts

9.12 Δημιουργία σελίδας για την επεξεργασία αγροκτήματος και πολλές φόρμες προσθήκης διάφορων στοιχείων για το αγρόκτημα (farm-edit component)

Όπως έχει ήδη αναφερθεί, ο απλός χρήστης έχει τη δυνατότητα να επιλέξει και να προσθέσει–επεξεργαστεί διάφορα δεδομένα που αφορούν τα διάφορα στάδια των αγροκτημάτων και κατά συνέπεια των ακτινιδίων. Η κύρια ιδέα είναι αφού διαλέξει ένα αγρόκτημα και πατήσει το κουμπί Επεξεργασία, να μεταφέρεται σε μια σελίδα που θα περιέχει όλα τα διαθέσιμα δεδομένα που αφορούν τα ακτινίδια.

Στον φάκελο farms λοιπόν, δημιουργείται το farm-edit component. Όταν επιλεγεί λοιπόν το κουμπί Επεξεργασία, το id του αγροκτήματος χρησιμοποιείται για να σταλεί ένα get request στο server που θα επιστρέφει δεδομένα, αν και δε θα χρειαστεί στη συγκεκριμένη σελίδα εμφάνιση δεδομένων παρά μόνο να προσθήκη. Αυτό γίνεται με την κατασκευή της μεθόδου **viewFarmDetails** στο **farms.service.ts**.

Φόρμες στο farm-edit component	
Όνομα Φόρμας	**Κατηγορία**
Προσθήκη Αζώτου	Στάδιο Λίπανσης
Προσθήκη Φωσφόρου	Στάδιο Λίπανσης
Προσθήκη Καλίου	Στάδιο Λίπανσης
Προσθήκη Άλλων Στοιχείων	Στάδιο Λίπανσης
Ύπαρξη Ζιζανιοκτόνων	Στάδιο Λίπανσης
Ύπαρξη Sitofex	Στάδιο Λίπανσης
Λίπανση	Στάδιο Λίπανσης
Συγκομιδή	Στάδιο Συγκομιδής
Χημικός Έλεγχος	Στάδιο Χημ. Ελέγχου
Αποθήκευση	Στάδιο Αποθήκευσης
Επεξεργασία	Στάδιο Επεξεργασίας
Συσκευασία	Στάδιο Συσκευασίας

Πίνακας 18: Οι φόρμες που περιλαμβάνονται στο farm-edit component

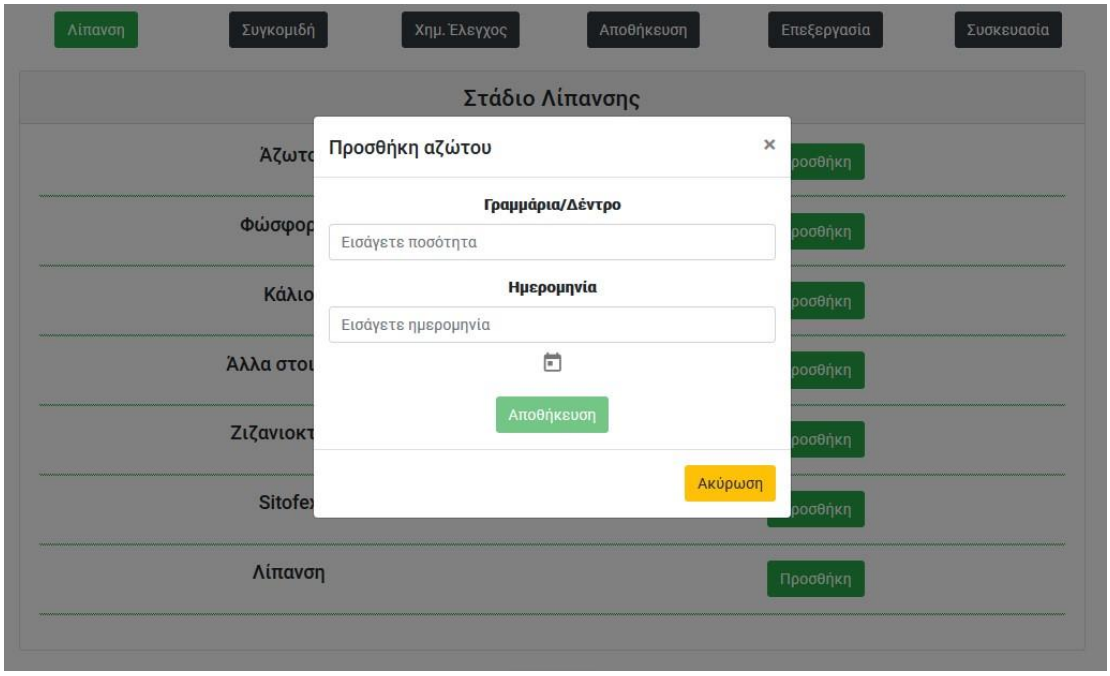
Στο farm-edit component θα υπάρχουν συνολικά 12 φόρμες. Στο **farm-edit.component.ts** δηλώνονται και δημιουργούνται οι φόρμες και δηλώνονται 6 boolean μεταβλητές που αντιστοιχούν στα 6 διαφορετικά στάδια των ακτινιδίων. Κατασκευάζονται 6 μέθοδοι που η καθεμιά αντιστοιχεί σε ποιο στάδιο έχει επιλεχτεί. Για παράδειγμα, αν επιλεγεί το Στάδιο Λίπανσης, η αντίστοιχη boolean μεταβλητή γίνεται true και όλες οι υπόλοιπες false. Με αυτή την υλοποίηση, στο **farm-edit.component.html** κατασκευάζονται τα 6 κουμπιά και συνδέονται το καθένα με την αντίστοιχη μέθοδο. Έτσι όταν επιλέγεται κάποιο κουμπί θα εμφανίζεται η αντίστοιχη φόρμα. Στην περίπτωση του Σταδίου Λίπανσης που περιέχει πιο πολλές φόρμες, οι φόρμες εμφανίζονται σαν μια λίστα

και ο χρήστης επιλέγει ποια θέλει να χρησιμοποιήσει. Η φόρμα θα εμφανίζεται σαν ένα παράθυρο διαλόγου, το οποίο ονομάζεται modal και ανήκει στο Bootstrap.

Προφανώς για να σταλούν δεδομένα από όλες αυτές τις φόρμες, θα χρειαστούν αντίστοιχα τόσες μέθοδοι στο **farms.service.ts**. Όμως δεν προτείνεται να χρησιμοποιούνται πάρα πολλά URLs. Γι' αυτό το λόγο, θα αλλάξει λίγο ο τρόπος που στέλνεται ένα post request, χρησιμοποιώντας ένα κοινό URL. Πέραν των φορμών **Ύπαρξη Ζιζανιοκτόνων** και **Ύπαρξη Sitofex**, όλες οι άλλες φόρμες στέλνουν στοιχεία σαν πίνακας και αποθηκεύονται στη βάση δεδομένων σαν πίνακας. Οπότε όλες οι φόρμες στέλνουν post request στο **ADD_FARM_FEATURES_URL**, με τη διαφορά να είναι πως στέλνει ένα body με το όνομα του ανάλογου Σταδίου. Άρα η βάση δεδομένων βλέπει το body και τοποθετεί στο σωστό πίνακα τα απεσταλμένα δεδομένα.

```
addHarvest(credentials: FarmModel, id: string) {
  const url = `${environment.ADD_FARM_FEATURES_URL}/${id}`;
  return this.http.post(url,  body: {harvest: credentials});
}
```

Εικόνα 46: Η μέθοδος addHarvest στο farms.service.ts



Εικόνα 47: Η φόρμα Προσθήκης Αζώτου στην καρτέλα του Σταδίου Λίπανσης

Μέθοδος	Φόρμα	URL που χρησιμοποιεί
addNitrogen	Προσθήκη Αζώτου	ADD_FARM_FEATURES_URL
addPhosphorus	Προσθήκη Φωσφόρου	ADD_FARM_FEATURES_URL
addPotassium	Προσθήκη Καλίου	ADD_FARM_FEATURES_URL
addOtherElements	Προσθήκη Άλλων Στοιχείων	ADD_FARM_FEATURES_URL
addHerbicidesExistence	Ύπαρξη Ζιζανιοκτόνων	ADD_HERBICIDES_EXISTENCE_URL
addSitofexExistence	Ύπαρξη Sitofex	ADD_SITOFEX_EXISTENCE_URL
addFertilize	Λίπανση	ADD_FARM_FEATURES_URL
addHarvest	Συγκομιδή	ADD_FARM_FEATURES_URL
addChemicalCheck	Χημικός Έλεγχος	ADD_FARM_FEATURES_URL
addStorage	Αποθήκευση	ADD_FARM_FEATURES_URL
addProcessing	Επεξεργασία	ADD_FARM_FEATURES_URL
addPackaging	Συσκευασία	ADD_FARM_FEATURES_URL

Πίνακας 19: Οι μέθοδοι των φορμών του farm-edit component στο farms.service.ts

9.13 Δημιουργία σελίδας για την προβολή λεπτομερών δεδομένων των αγροκτημάτων – ακτινιδίων (view-farm component) και διαγραφή αγροκτήματος

Στη σελίδα προβολής λεπτομερών δεδομένων ισχύει η ίδια λογική με τη σελίδα επεξεργασίας όσον αφορά την εμφάνιση. Θα επιλέγει ένα από τα υπάρχοντα κουμπιά και θα εμφανίζονται οι αντίστοιχες πληροφορίες για το κάθε Στάδιο. Στο φάκελο farms θα δημιουργηθεί το view-farm component.

Αρχικά θα δηλωθούν 7 μεταβλητές τύπου boolean στο **view-farm.component.ts**. Μπορεί τα Στάδια να είναι 6, αλλά χρειάζεται να προβληθούν και οι γενικές πληροφορίες του αγροκτήματος που προστέθηκαν από τη φόρμα Προσθήκη Αγροκτήματος. Στη συνέχεια θα κατασκευαστούν και οι αντίστοιχες 7 μέθοδοι που θα λειτουργούν όπως και στην περίπτωση της σελίδας επεξεργασίας. Στο **view-farm.component.html** θα κατασκευαστούν τα 7

κουμπιά και θα συνδεθούν με την αντίστοιχη μέθοδο. Τα δεδομένα θα παρασταθούν με τη βοήθεια των καρτών (cards) του Bootstrap.

Εικόνα 48: Η κάρτα με τις πληροφορίες του Σταδίου Λίπανσης

Όσον αφορά το πώς θα εμφανιστούν τα δεδομένα, χρειάζεται ένα get request προς το server με το id του αγροκτήματος. Θα χρησιμοποιηθεί η **viewFarmDetails** του **farms.service.ts** που χρησιμοποιήθηκε και στο farm-edit component.

```
viewFarmDetails(id: string): Observable<FarmModel> {
  const url = `${environment.FARMS_URL}/${id}`;
  return this.http.get<FarmModel>(url);
}
```

Εικόνα 49: Η μέθοδος viewFarmDetails στο farms.service.ts

Τέλος, για να ολοκληρωθεί και η διαγραφή ενός αγροκτήματος, κατασκευάζεται στο **farms.service.ts** η μέθοδος **deleteFarm** που στέλνει στο server ένα delete request με το id του αγροκτήματος που επιλέγεται για διαγραφή.

```
deleteFarm(id: string): Observable<FarmModel> {
  const url = `${environment.FARMS_URL}/${id}`;
  return this.http.delete<FarmModel>(url, httpOptions);
}
```

Εικόνα 50: Η μέθοδος deleteFarm στο farms.service.ts

9.14 Δημιουργία navigation bar

Μια navigation bar είναι ένα τμήμα μιας γραφικής διεπαφής χρήστη που βοηθάει τους επισκέπτες – χρήστες μιας ιστοσελίδας να έχουν πρόσβαση σε άλλες σελίδες με πληροφορίες. Πρόκειται για τη μπάρα που συνηθίζεται να είναι στην κορυφή μιας ιστοσελίδας και που συνήθως περιέχει συνδέσμους (links) για άλλες σελίδες.

Αρχικά δημιουργείται το component. Ονομάζεται navbar μιας και θα χρησιμοποιηθεί η navbar του Bootstrap σαν navigation bar. Ανάλογα με το ρόλο του χρήστη θα εμφανίζονται διαφορετικοί σύνδεσμοι, αλλά θα υπάρχουν και κάποιοι κοινοί. Αν για παράδειγμα είναι συνδεδεμένος κάποιος διαχειριστής, τότε στη μπάρα υπάρχει σύνδεσμος ώστε να μεταβαίνει στην προσθήκη ενός χρήστη.

Σύνδεσμος	**Ρόλος που απαιτείται**
Αρχική σελίδα	Οποιοσδήποτε
Προσθήκη Αγροκτήματος	Απλός Χρήστης
Αλλαγή Κωδικού Πρόσβασης	Απλός Χρήστης
Προσθήκη Χρήστη	Διαχειριστής
Αποσύνδεση	Οποιοσδήποτε

Πίνακας 20: Οι σύνδεσμοι που χρησιμοποιούνται στη navigation bar

Στο **navbar.component.ts** υλοποιούνται όλες οι μέθοδοι που θα μεταφέρουν το χρήστη στις σελίδες που φαίνονται στον παραπάνω πίνακα. Έτσι, δημιουργούνται οι **goHome**, **goToAddFarm**, **goToChangePassword**. Για την αποσύνδεση, έχει ήδη υλοποιηθεί η **logout** στο **auth.service.ts**. Επίσης, δημιουργείται μια μέθοδος ονόματι **getCurrentRole** που επιστρέφει το ρόλο του συνδεδεμένου χρήστη. Στο **navbar.component.html** γίνεται ο σχεδιασμός της navbar σύμφωνα με το documentation του Bootstrap. Με τη συνεργασία της **getCurrentRole** και της ***ngIf** που περιλαμβάνεται στην Angular, επιτυγχάνεται η εμφάνιση των σωστών συνδέσμων για κάθε χρήστη. Τέλος, για να εμφανίζεται σε κάθε σελίδα η navbar, πρέπει να την εισαχθεί σε όλα τα HTML αρχεία από όλα τα υπάρχοντα components. Η εισαγωγή γίνεται γράφοντας σε κάθε HTML αρχείο στην πρώτη γραμμή τον selector του component της navbar, **<app-navbar></app-navbar>.**

9.15 Μετατροπή της εφαρμογής σε desktop application με τη χρήση του Electron

Η μετατροπή της web εφαρμογής σε desktop έγινε με τη χρήση του Electron. Η εφαρμογή αναπτύχθηκε συγκεκριμένα για το λειτουργικό Windows, αλλά πολύ εύκολα γίνεται να αναπτυχθεί και για MACOS ή Linux. Η διαδικασία μετατροπής της εφαρμογής και χρήσης του Electron είναι σχεδόν στάνταρ και χρειάζεται να ακολουθηθούν συγκεκριμένες οδηγίες ώστε να ολοκληρωθεί.

Αφού έχει εγκατασταθεί το Electron, δημιουργείται ένα JavaScript αρχείο με το όνομα **main.js** στο φάκελο του project και προστίθεται ο ακόλουθος κώδικας:

```
const { app, BrowserWindow } = require('electron')

function createWindow () {
  // Create the browser window.
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true
    }
  })

  // and load the index.html of the app.
  win.loadFile('index.html')

  // Open the DevTools.
  win.webContents.openDevTools()
}

app.whenReady().then(createWindow)

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit()
  }
})

app.on('activate', () => {
  // On macOS it is common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow()
  }
})
```

Προφανώς και το αρχείο αυτό δε θα μείνει ως έχειν, καθώς χρειάζεται τροποποίηση ώστε να το προσαρμοστεί στα μέτρα της εφαρμογής. Βασικό στοιχείο αυτού του κώδικα είναι η γραμμή **win.loadFile('index.html')**, όπου εκεί γράφεται το **index.html** αρχείο που υπάρχει στο project, το οποίο είναι ό,τι έχει υλοποιηθεί ως τώρα σαν web application και αυτό πρέπει να φορτώνει-βλέπει το αρχείο **main.js**.

Στη συνέχεια τροποποιείται το αρχείο **package.json**, προσθέτοντας κάποια στοιχεία που αφορούν την εφαρμογή και κάποια scripts που θα χρειαστεί να «τρέξουν».

```
"version": "0.0.0",
"productName": "Kiwi Application",
"license": "MIT",
"description": "Kiwi Electron Application",
"author": "George Dimitriou",
"main": "main.js",
"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e",
  "electron": "electron .",
  "electron-build": "ng build && electron .'
```

Εικόνα 51: Τα στοιχεία που προστέθηκαν στο αρχείο package.json

Το version αφορά την έκδοση της εφαρμογής, το productName είναι το όνομα που θα έχει η εφαρμογή όταν θα εγκατασταθεί, το description είναι μια περιγραφή της εφαρμογής, το author είναι το όνομα του developer και το main αναφέρει ποιο αρχείο πρέπει να εκτελέσει.

Επόμενο βήμα είναι το build της εφαρμογής. Αυτό γίνεται τρέχοντας το script που προστέθηκε και φαίνεται στην παραπάνω εικόνα. Μόλις ολοκληρωθεί το build, τρέχει το δεύτερο κομμάτι της εντολής “electron .”, όπου η εφαρμογή τρέχει σαν ξεχωριστό application.

Όλες οι εφαρμογές σήμερα έχουν κάποιο εικονίδιο. Έτσι και σε αυτή την εφαρμογή θα προστεθεί ένα. Για να γίνει αυτό, δημιουργείται ένας φάκελος με το όνομα assets μέσα στο project. Μέσα σε αυτόν, υπάρχει άλλος ένας υποφάκελος icons και μέσα στον icons άλλοι τρεις υποφακέλοι με τα ονόματα mac, win και png, τα οποία αντιστοιχούν στα 3 διαφορετικά λειτουργικά συστήματα. Το εικονίδιο είναι ένα ακτινίδιο και υπάρχει σε 3 διαφορετικές

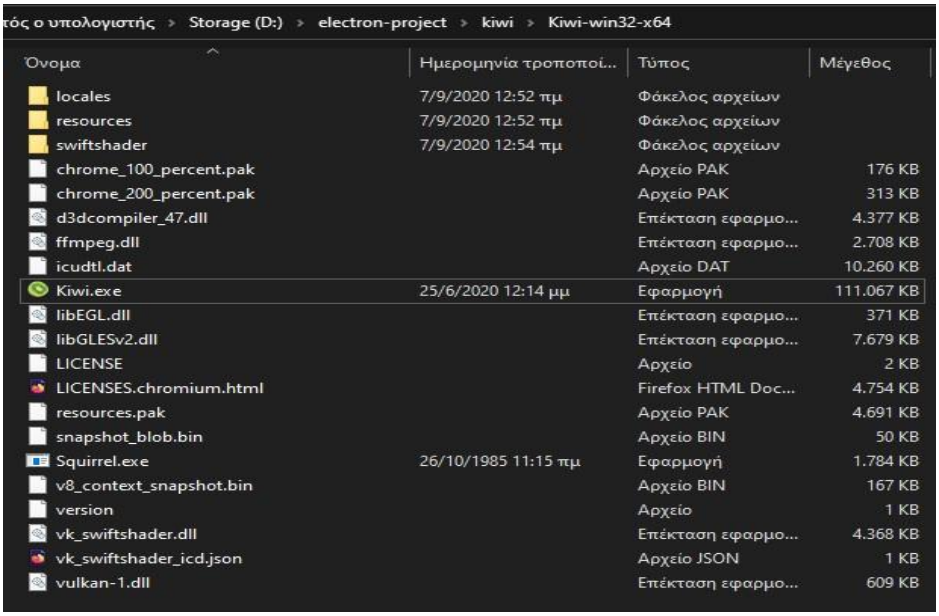
επεκτάσεις, μιας και κάθε λειτουργικό χρειάζεται διαφορετική επέκταση στην εικόνα. Οι επεκτάσεις είναι icns για το MACOS, ico για τα Windows και png για το Linux. Τα εικονίδια ελήφθησαν από το https://iconarchive.com, μια ιστοσελίδα με μεγάλη γκάμα εικονιδίων για προσωπική και για εμπορική χρήση. Αν δεν επιλεγεί να χρησιμοποιηθεί κάποιο εικονίδιο, η εφαρμογή έχει σαν προεπιλογή το εικονίδιο του Electron.

Αφού είναι έτοιμα και τα εικονίδια, πρέπει η εφαρμογή να «πακεταριστεί». Με τον όρο αυτό, εννοείται η ετοιμασία της εφαρμογής ώστε να εκτελείται συγκεκριμένα σε κάποιο λειτουργικό ή ακόμα και σε όλα. Αφού εγκατασταθεί το πακέτο electron-packager, προστίθενται τρία νέα scripts στο **package.json**, που αφορούν το «πακετάρισμα» για κάθε λειτουργικό σύστημα Έτσι δημιουργείται το εκτελέσιμο αρχείο της εφαρμογής για την επιθυμητή πλατφόρμα, όποια και αν είναι.

```
"electron": "electron .",
"electron-build": "ng build && electron .",
"pack": "electron-packager .",
"package-win": "electron-packager . --overwrite --platform=win32 --arch=x64 --icon=assets/icons/win/icon.ico Kiwi",
"package-mac": "electron-packager . --overwrite --platform=darwin --arch=x64 --icon=assets/icons/mac/icon.icns --prune=true --out=release-builds",
"package-linux": "electron-packager . --overwrite --asar=true --platform=linux --arch=x64 --icon=assets/icons/png/icon.png --prune=true --out=release-builds"
```

Εικόνα 52: Τα scripts για το packaging για όλα τα λειτουργικά συστήματα

Το packaging ολοκληρώνεται τρέχοντας το script που αφορά το λειτουργικό Windows. Όταν ολοκληρωθεί αυτή η διαδικασία, θα έχει δημιουργηθεί στο φάκελο του project ο φάκελος Kiwi-win32-x64 με την παρακάτω δομή:



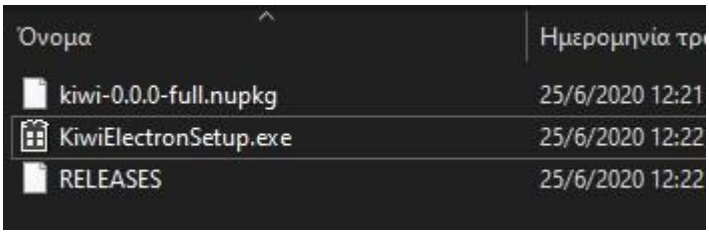
Εικόνα 53: Περιεχόμενα του φακέλου Kiwi-win32-x64 που δημιουργήθηκε

Τελευταίο βήμα είναι να κατασκευή του installer, ώστε να μένει εγκατεστημένη η εφαρμογή στον υπολογιστή. Αρχικά τοποθετείται ο φάκελος του project σε έναν άλλο φάκελο π.χ. electron-project. Μέσα σε αυτόν δημιουργείται άλλος ένας φάκελος installers όπου θα δημιουργηθεί ο installer της εφαρμογής. Επίσης, στον «πατέρα» φάκελο δημιουργείται ένα JavaScript αρχείο κατά προτίμηση με όνομα **build-win.js**. Στο συγκεκριμένο προστίθεται ο κώδικας που είναι υπεύθυνος για τη δημιουργία του installer. Ο κώδικας είναι περίπου στάνταρ σύμφωνα με το documentation του πακέτου, αλλά φυσικά πρέπει να προσαρμοστεί στα μέτρα της εφαρμογής.

```
var electronInstaller = require('electron-winstaller');

// In this case, we can use relative paths
var settings = {
    // Specify the folder where the built app is located
    appDirectory: 'C:/electron-project/kiwi/Kiwi-win32-x64',
    // Specify the existing folder where
    outputDirectory: 'C:/electron-project/installers',
    // The name of the Author of the app (the name of your company)
    authors: 'George Dimitriou',
    // The name of the executable of your built
    exe: './Kiwi.exe',
    // The name of the setup.exe
    setupExe: 'KiwiElectronSetup.exe',
    // We just need the exe installer, not the msi
    noMsi: true
};

resultPromise = electronInstaller.createWindowsInstaller(settings);

resultPromise.then(() => {
    console.log("The installers of your application were succesfully created !");
}, (e) => {
    console.log(`Well, sometimes you are not so lucky: ${e.message}`)
```

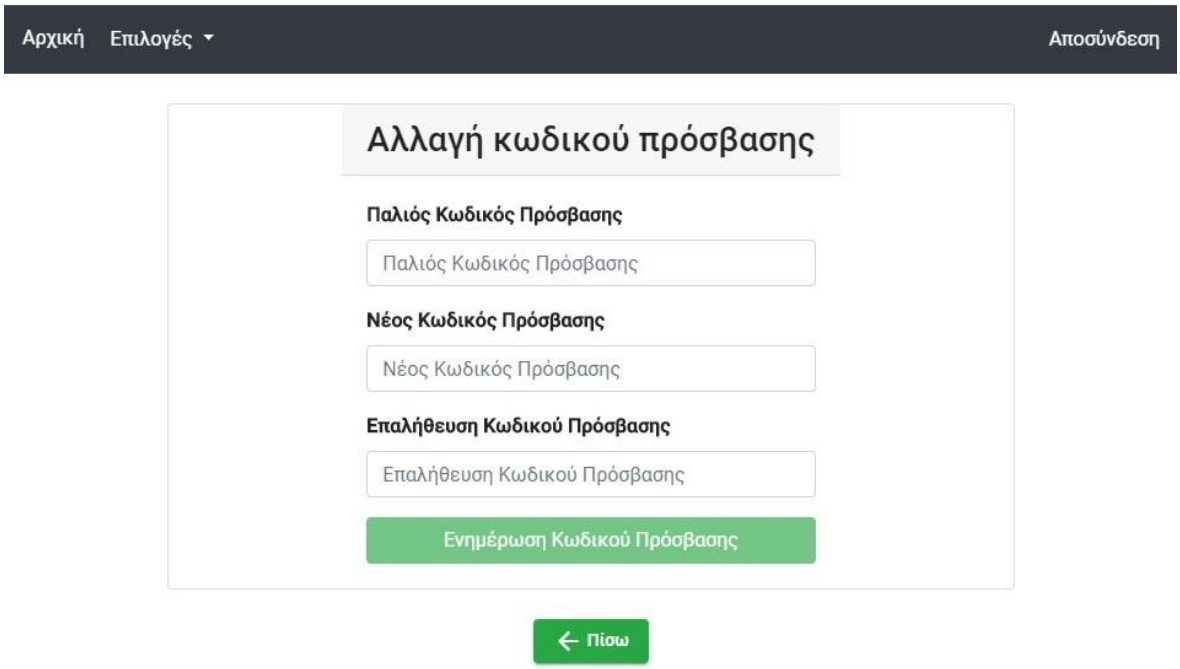
Εικόνα 54: Ο κώδικας του αρχείου build-win.js

Για να ολοκληρωθεί η διαδικασία και να δημιουργηθεί ο installer, πρέπει να εκτελεστεί το **build.js**. Όταν τελειώσει η διαδικασία, ο installer έχει δημιουργηθεί.



Εικόνα 55: Ο installer που μόλις έχει δημιουργηθεί

Τέλος, τρέχοντας το KiwiElectronSetup.exe, η εφαρμογή εγκαθίσταται στον υπολογιστή και έχει επίσημα μετατραπεί σε desktop application.



Εικόνα 56: Η εφαρμογή Kiwi

10. Μελλοντικές επεκτάσεις

Η εφαρμογή αφορά την ιχνηλασιμότητα των ακτινιδίων από τη στιγμή που κάποιος φυτεύει σε ένα αγρόκτημα μέχρι και τη στιγμή που συσκευάζεται το ακτινίδιο. Υπάρχουν κάποιοι τρόποι επέκτασης μελλοντικά όπως:

- Συνέχεια ιχνηλασιμότητας σε επίπεδο προώθησης των ακτινιδίων στην αγορά (σουπερμάρκετ, λαχαναγορές κλπ.) μέχρι και το σημείο όπου ο καταναλωτής θα μπορεί να έχει αν όχι όλες, πολλές από αυτές τις πληροφορίες.
- Άλλος τρόπος επέκτασης της εφαρμογής είναι να αφορά και άλλα προϊόντα εκτός από ακτινίδια. Προφανώς θα χρειαστεί τροποποιήσεις, καθώς κάθε προϊόν έχει διαφορετικά χαρακτηριστικά.
- Μία ακόμα επέκταση της εφαρμογής αλλά μικρότερου βεληνεκούς είναι η προσθήκη διάφορων τρόπων ταξινόμησης στον πίνακα των αγροκτημάτων ή και η προσθήκη μπάρας αναζήτησης στη navbar ώστε να αναζητά κάποιος κάποιο συγκεκριμένο αγρόκτημα.

Βιβλιογραφία

[1] «Mobcoder,» [Ηλεκτρονικό]. Available: https://mobcoder.com/blog/cross-platform-apps-need-of-the-hour-for-budding-businesses-2/.

[2] «Medium,» [Ηλεκτρονικό]. Available: https://medium.com/flutterdevs/native-vs-hybrid-vs-cross-platform-what-to-choose-3221130f7cc5.

[3] «DEV Community - Native Vs Hybrid Vs Cross-Platform,» [Ηλεκτρονικό]. Available: https://dev.to/techticsolutions/native-vs-cross-platform-vs-hybrid-app-development-1olp?fbclid=IwAR1Y9nGNUraTervg_kbMsDf_xo1BvNHO2ZvrEfE42C1h6jPElH0LgOAM4lI.

[4] «Ami Web Solutions - Native VS Hybrid,» [Ηλεκτρονικό]. Available: https://www.amiwebsolutions.com/mobile-app-development/native-hybrid-web-apps/.

[5] «Electron | Build cross-platform desktop apps with JavaScript, HTML and CSS.,» [Ηλεκτρονικό]. Available: https://www.electronjs.org/.

[6] «Ionic - Cross-Platform Mobile App Development,» [Ηλεκτρονικό]. Available: https://ionicframework.com.

[7] «11 Popular Cross-Platform Tools for App Development in 2020 | Hacker Noon,» [Ηλεκτρονικό]. Available: https://hackernoon.com/9-popular-cross-platform-tools-for-app-development-in-2019-53765004761b.

[8] «wikimedia html,» [Ηλεκτρονικό]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/b/b7/Html-source-code.png/521px-Html-source-code.png.

[9] «MVC Architecture | Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm.

[10] «5 things that you'll learn when you start to use React Native | Medium,» [Ηλεκτρονικό]. Available: https://medium.com/magnetis-backstage/5-things-that-youll-learn-when-you-start-to-use-react-native-1ed601e6eac.

[11] «Getting Started With Rails - Ruby on Rails Guides | Hackr Blog,» [Ηλεκτρονικό]. Available: https://hackr.io/blog/getting-started-with-rails.

[12] «The Pros and Cons of Vue.js | AltexSoft,» [Ηλεκτρονικό]. Available: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/.

[13] «JavaScript Tutorial - Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/javascript/index.htm.

[14] «TypeScript Tutorial - Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/typescript/typescript_overview.htm.

[15] «Ruby Tutorial - Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/ruby/ruby_overview.htm.

[16] «XML Tutorial - Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/xml/xml_overview.htm.

[17] «JSON,» [Ηλεκτρονικό]. Available: https://www.json.org/json-en.html.

[18] «Features - WebStorm,» [Ηλεκτρονικό]. Available: https://www.jetbrains.com/webstorm/features/.