ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Μελέτη του Angular framework και σύγκριση διαφορετικών αρχιτεκτονικών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Λεμονόπουλος Πέτρος (ΑΜ: Μ013121017)

Επιβλέπουσα: ***Βασιλική Κουτσονικόλα, Ε.ΔΙ.Π.***

ΛΑΡΙΣΑ, Μάϊος 2023

UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Study of Angular framework and comparison of different architectures

MASTER THESIS

Lemonopoulos Petros (RN: M013121017)

Supervisor: ***Vasiliki Koutsonikola, Laboratory Teaching Staff***

LARISSA, May 2023

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

(Υπογραφή)

Λεμονόπουλος Πέτρος

Ημερομηνία

8/5/2023

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπουσα	**Βασιλική Κουτσονικόλα** Εργαστηριακό Διδακτικό Προσωπικό (Ε.ΔΙ.Π.), Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Γεώργιος Κακαρόντζας** Αναπληρωτής Καθηγητής Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Φώτιος Κόκκορας** Επίκουρος Καθηγητής Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 27-06-2023

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως θέμα τη μελέτη του Angular framework, το οποίο αποτελεί ένα από τα δημοφιλέστερα framework των προγραμματιστών τα τελευταία χρόνια, καθώς η εξέλιξη της ανάπτυξης εφαρμογών μεγάλης κλίμακας μεγαλώνει με ραγδαίους ρυθμούς. Είναι γεγονός ότι, με την πάροδο των χρόνων, οι εφαρμογές διαδικτύου γνωρίζουν μεγάλες αλλαγές τόσο ως προς τις νέες τεχνολογίες που συνεχώς εισέρχονται όσο και ως προς τις απαιτήσεις που υπάρχουν από τους χρήστες του διαδικτύου. Επομένως, η κοινότητα των προγραμματιστών καλείται να βρει τη βέλτιστη λύση για αυτήν, ώστε να εξυπηρετεί με το καλύτερο τρόπο το χρήστη. Σκοπός της συγκεκριμένης μεταπτυχιακής εργασίας είναι η μελέτη του Angular framework, η ανάλυση των δυνατοτήτων του, της αρχιτεκτονικής του, των πλεονεκτημάτων και των μειονεκτημάτων καθώς και η σύγκρισή του με το React framework. Επιπλέον, στην εργασία αυτή αναπτύχθηκε μία εφαρμογή για ηλεκτρονική καταγραφή συνταγών μαγειρικής με τη χρήση της Angular προκειμένου να μελετηθεί σε βάθος αυτό το framework χρησιμοποιώντας σχεδόν όλες τις δυνατότητες που μας δίνει το framework. Τέλος, η υλοποίηση της εφαρμογής πραγματοποιήθηκε με τρεις διαφορετικούς τρόπους προκειμένου να γίνει σύγκριση αυτών των διαφορετικών αρχιτεκτονικών και να καταλήξουμε σε σημαντικά συμπεράσματα ως προς την αποδοτικότητα του κάθε τρόπου.

Abstract

Περίληψη στα Αγγλικά.

The subject of this thesis is the study of the Angular framework, which is one of the most popular frameworks of programmers in recent years, as the development of large-scale application development is growing rapidly. It is a fact that, over the years, web applications have seen great changes both in terms of new technologies that are constantly entering and in terms of the demands that exist from internet users. Therefore, the community of developers is called upon to find the optimal solution for it to best serve the user. The purpose of this master's thesis is the study of the Angular framework, the analysis of its capabilities, its architecture, advantages and disadvantages as well as its comparison with the React framework. In addition, in this thesis an application was developed for electronic recording of cooking recipes using Angular in order to study this framework in depth using almost all the possibilities that the framework gives us. Finally, the implementation of the application was carried out in three different ways in order to compare these different architectures and reach important conclusions regarding the efficiency of each way.

Ευχαριστίες

Για τη διεκπεραίωση της παρούσας μεταπτυχιακής εργασίας, θα ήθελα να ευχαριστήσω την επιβλέπουσα, κυρία Βασιλική Κουτσονικόλα για την άψογη συνεργασία και την κατανόηση, την εμπιστοσύνη και τη στήριξη που μου έδειξε.

Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για την στήριξή τους όλα αυτά τα χρόνια.

Λεμονόπουλος Πέτρος

8/5/2023

Περιεχόμενα

1 Εισαγωγή

Στο κεφάλαιο 1 γίνεται η εισαγωγή σε βασικές έννοιες προκειμένου να κατανοηθεί το περιεχόμενο της εργασίας. Κατά συνέπεια σε αυτό το κεφάλαιο δίνονται οι βασικοί ορισμοί για την Angular και τις έννοιες που σχετίζονται με αυτή. Επιπλέον, παρατίθεται το ιστορικό πλαίσιο μέσα στο οποίο αναπτύχθηκε και εξελίχθηκε το framework. Τέλος, γίνεται αναφορά στη συνολική συνεισφορά της παρούσας εργασίας και στην δομή της.

1.1 Τι είναι η Angular

Η Angular (αναφέρεται επίσης ως "Angular 2+") [1] είναι ένα ελεύθερο και ανοιχτού κώδικα web application framework που βασίζεται σε TypeScript, το οποίο διευθύνεται από την Angular Team της Google και από μια κοινότητα ατόμων και εταιρειών. Η Angular είναι μια πλήρης επανεγγραφή από την ίδια ομάδα που δημιούργησε το AngularJS. Επίσης, η Angular είναι ένα Single Page Application (SPA) Framework που χρησιμοποιείται για τη δημιουργία γρήγορων εφαρμογών Ιστού. Συγκεκριμένα, χρησιμοποιεί έννοιες SPA στις οποίες η διεπαφή χρήστη παραδίδεται στην αρχή της αίτησης (request) εφαρμογής και αργότερα ζητούνται μόνο δεδομένα. Το γεγονός αυτό καθιστά τις εφαρμογές SPA γρήγορες. [2]. Συνοπτικά, συμπεραίνουμε ότι η Angular είναι ένα framework που επιτρέπει να δημιουργούμε reactive εφαρμογές μιας σελίδας.

1.2 Τι σημαίνει Single Page Application

Παραδοσιακά, οι εφαρμογές ήταν Multi-Page Applications (MPA) όπου με κάθε κλικ φορτωνόταν μια νέα σελίδα από τον διακομιστή(server). Αυτό δεν ήταν μόνο χρονοβόρο, αλλά αύξησε και τον φόρτο του διακομιστή και έκανε τον ιστότοπο πιο αργό. Το AngularJS είναι ένα web front-end framework που βασίζεται σε JavaScript, στην αμφίδρομη σύνδεση δεδομένων διεπαφής χρήστη και χρησιμοποιείται για το σχεδιασμό εφαρμογών SPA. Οι εφαρμογές SPA είναι εφαρμογές ιστού που φορτώνουν μια σελίδα HTML και μόνο ένα μέρος της σελίδας αντί ολόκληρης της σελίδας ενημερώνεται με κάθε κλικ του ποντικιού. Η σελίδα δεν φορτώνει ξανά. Κατά συνέπεια αυτό εξασφαλίζει υψηλή απόδοση και ταχύτερη φόρτωση σελίδων. Οι περισσότερες σύγχρονες εφαρμογές

χρησιμοποιούν την έννοια του SPA. Στο SPA, όλα τα δεδομένα αποστέλλονται στον πελάτη από τον διακομιστή στην αρχή. Καθώς ο πελάτης κάνει κλικ σε ορισμένα μέρη στην ιστοσελίδα, μόνο το απαιτούμενο μέρος των πληροφοριών λαμβάνεται από τον διακομιστή και η σελίδα ξαναγράφεται δυναμικά. Αυτό έχει ως αποτέλεσμα μικρότερο φόρτο στον διακομιστή και είναι πιο αποδοτικό. Τα SPA χρησιμοποιούν AJAX και HTML5 για τη δημιουργία responsive εφαρμογών ιστού(=web applications) και το μεγαλύτερο μέρος της εργασίας γίνεται από την πλευρά του πελάτη(=client). Δημοφιλείς εφαρμογές όπως το Facebook, το Gmail, το Twitter, το Google Drive, το Netflix και πολλές άλλες είναι παραδείγματα SPA εφαρμογών [3].

1.3 Ιστορία της Angular

Σε αυτή την ενότητα αναλύονται τα βασικά σημεία της ιστορίας της Angular, δηλαδή πως αυτή ξεκίνησε και πως εξελίχθηκε στη συνέχεια.

2010: Η δημιουργία του AngularJS

Το AngularJS δημιουργήθηκε από τον Miško Hevery, έναν υπάλληλο της Google, ο οποίος ανέπτυξε ένα δευτερεύον-εξωτερικό έργο. Αυτό το δευτερεύον έργο είχε ως στόχο να διευκολύνει τη δημιουργία διαδικτυακών εφαρμογών για μερικά εσωτερικά έργα στα οποία εργαζόταν ο ίδιος. Αυτό το δευτερεύον έργο έγινε αργότερα γνωστό ως AngularJS (Angular λόγω του < > σε HTML).

Στη συνέχεια, ο Misko και μερικοί άλλοι άρχισαν να δημιουργούν μερικές ακόμη εσωτερικές εφαρμογές με το AngularJS και τελικά το κυκλοφόρησαν ως έργο ανοιχτού κώδικα το 2010. Αυτό είχε ως αποτέλεσμα η κοινότητα της Google να το πάρει στα χέρια της και άρχισε να δημιουργεί καταπληκτικές εφαρμογές. Το Ionic Framework που χτίστηκε πάνω από το Apache Cordova από την Ionic (πρώην DriftyCo) συνέβαλε ώστε οι προγραμματιστές να χρησιμοποιούν το AngularJS για να τροφοδοτούν και τις εφαρμογές τους για κινητές συσκευές. Επίσης, ορισμένες από τις μεγαλύτερες εταιρείες, όπως η Deutsche Bank, άρχισαν να το ενσωματώνουν στο σύνολο εργαλείων τους τόσο για την ανάπτυξη εφαρμογών ιστού όσο και για κινητά.

2014 - 2015: Η μεγάλη αλλαγή της Angular

Αρκετά χρόνια μετά την αρχική του κυκλοφορία, το τοπίο της ανάπτυξης εφαρμογών ιστού άρχισε να αλλάζει και το AngularJS άρχισε να μην είναι το ίδιο αποδοτικό και χρήσιμο. Αυτό οφείλεται στο γεγονός ότι εμφανίστηκαν νέες αλλαγές και πρότυπα στην

JavaScript, με συνέπεια η Angular να αρχίσει να μένει πίσω από τις εξελίξεις. Το πιο σημαντικό είναι ότι η βασική ομάδα είχε φτάσει σε ένα όριο στο τι μπορούσε να κάνει για να βελτιώσει το framework μπροστά στις ολοένα αυξανόμενες απαιτήσεις.

Η ομάδα της Google και της κοινότητας της AngularJS πήρε αυτό που κάποτε ήταν ένα μικρό εσωτερικό έργο και το ώθησε σε νέα όρια, όπως οι εφαρμογές για κινητές συσκευές και μεγάλες επιχειρήσεις. Ωστόσο, ποτέ δεν προοριζόταν ούτε σχεδιάστηκε για αυτές τις καταστάσεις όταν το δημιούργησε για πρώτη φορά ο Miško. Συνεπώς, άρχισε η τροποποίηση της AngularJS προκειμένου να ανταποκρίνεται στις σύγχρονες ανάγκες.

Όταν η βασική ομάδα της Google προσπάθησε να παραδώσει το 2.0, δεν ήθελε να δεσμευτεί στον προηγούμενο σχεδιασμό του AngularJS. Ήθελαν να δημιουργήσουν ένα πλαίσιο από την αρχή για να λύσουν δύσκολα προβλήματα κατά την κατασκευή μεγάλων και πολλαπλών πλατφορμών εφαρμογών. Αυτό σήμαινε μια ολική επανεγγραφή.

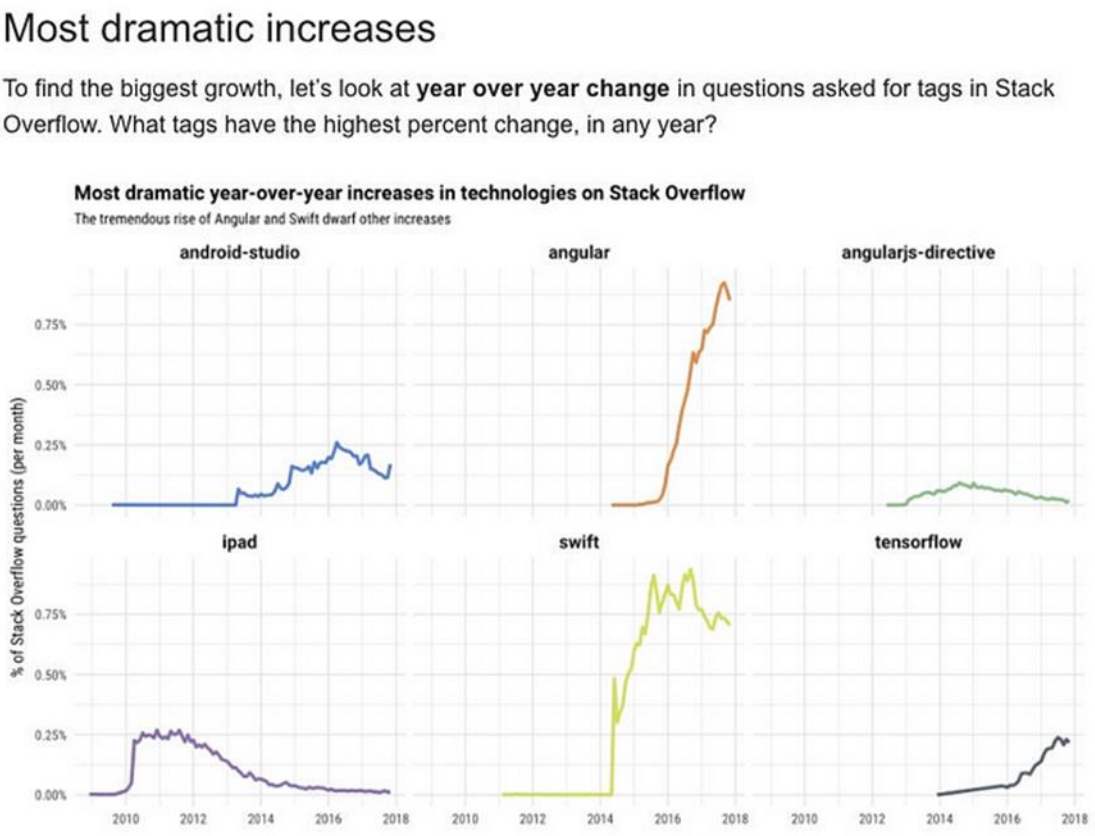
Συνεπώς, το framework της AngularJS θα έπρεπε πλέον να αλλάξει ριζικά παρ' όλο που πολλοί είχαν ασχοληθεί με αυτό και είχαν δημιουργήσει αρκετά έργα. Τελικά, όλα τα μεγάλα έργα τρίτων που δημιουργήθηκαν όπως το Angular Material (v1.x) και το Bootstrap καταργήθηκαν.

2016: Κυκλοφορεί η έκδοση Angular 2.0

Όλη αυτή τη περίοδος της μεγάλης μετάβασης, ήταν μία περίοδος πανικού για τους προγραμματιστές και τους διαχειριστές καθώς η πλήρης επανεγγραφή σήμαινε ένα σενάριο καταστροφής για τα τρέχοντα έργα τους AngularJS που τρέχουν στην παραγωγή. Επιπλεον, ακόμη χειρότερα, δεν ήταν σαφής η στρατηγική μετεγκατάστασης για τη μεταφορά μιας εφαρμογής AngularJS 1.X στο νέο Angular 2.0.

Ωστόσο, παρ'όλες τις δυσκολίες οι ομάδες προχώρησαν και δημιούργησαν νέες εφαρμογές Angular2.0 ενώ ήταν ακόμα σε beta. Σε αυτό το σημείο, οι ομάδες άρχισαν να συναντούν πολλές νέες έννοιες και δυσκολίες. Αυτό είχε ως αποτέλεσμα αρκετοί προγραμματιστές να γυρίσουν την πλάτη τους στην AngularJS για βιβλιοθήκες όπως η React.

Στο παρακάτω διάγραμμα απεικονίζεται η ανάπτυξη διάφορων τεχνολογιών χρόνο με τον χρόνο σε σχέση με τα ερωτήματα που τίθενται στο Stack Overflow.



Εικόνα 1: Ερωτήματα Stack Overflow για τεχνολογίες [4].

Το 2016, η Google κυκλοφόρησε την Angular 2. Σε αυτή την έκδοση έγινε μια σημαντική αναθεώρηση του framework, το οποίο ουσιαστικά δημιουργήθηκε από την αρχή με έμφαση στη βελτίωση της απόδοσης, της επεκτασιμότητας. Επίσης, η Angular 2 εισήγαγε πολλές νέες έννοιες, όπως components, directives, και modules και σχεδιάστηκε για να λειτουργεί καλά με τις σύγχρονες πρακτικές ανάπτυξης ιστού.

Οι επόμενες εκδόσεις

Angular 4-9: Η Google συνέχισε να βελτιώνει και να ανανεώνει την Angular με κάθε επόμενη κυκλοφορία, εισάγοντας νέες δυνατότητες και βελτιώσεις στο framework. Αυτές οι εκδόσεις εισήγαγαν βελτιώσεις στην απόδοση, τη μείωση μεγέθους και των εργαλείων επεξεργασίας.

Angular 10 και μετά: Η Angular 10 κυκλοφόρησε το 2020, με βελτιώσεις στην απόδοση και ορισμένες δυνατότητες όπως το Composition API. Η ομάδα ανάπτυξης συνεχίζει να εργάζεται για τη βελτίωση της αρχιτεκτονικής και του συνόλου εργαλείων της Angular για να κάνει τη δημιουργία εφαρμογών όσο γίνεται πιο εύκολη.

Συνολικά, η Angular γνώρισε σημαντική ανάπτυξη και αποδοχή από την κοινότητα προγραμματιστών από την αρχική της κυκλοφορία. Συνεπώς, παραμένει μια δημοφιλής επιλογή για την κατασκευή σύνθετων εφαρμογών μεγάλης κλίμακας.

1.4 Συνεισφορά της Μεταπτυχιακής Διπλωματικής Εργασίας

Η παρούσα μεταπτυχιακή διπλωματική εργασία έχει ως βασικό σκόπο την μελέτη και την ανάδειξη του framework της Angular για την ανάπτυξη εφαρμογών μεγάλης κλίμακας. Πιο συγκεκριμένα, σε αυτή την εργασία θα ερευνηθούν τα εξής:

i. Το framework της Angular με τις δυνατότητες που μπορεί να προσφέρει καθώς και η σύγκριση της Angular με το framework της React.

ii. Η δημιουργία εφαρμογής μέσω Angular με θέμα την αποθήκευση και την διαχείριση συνταγών μαγειρικής. Στην εφαρμογή αυτή παρουσιάζεται σχεδόν όλο το φάσμα των τεχνικών της Angular, ενώ παράλληλα γίνεται έρευνα και σύγκριση διαφορετικών τεχνικών υλοποίησης της εφαρμογής.

Συνοπτικά, ο βασικός στόχος της διπλωματικής είναι μέσα από την μελέτη της Angular:

1. Να γίνουν γνωστές οι δυνατότητες που έχει το framework, τα πλεονεκτήματα και τα μειονεκτήματά του.
2. Να παρουσιασθεί η υλοποίηση της εφαρμογής, να αναλυθεί ο κώδικας, η δομή και η αρχιτεκτονική της εφαρμογής.
3. Να γίνει σύγκριση διαφορετικών μεθόδων αρχιτεκτονικής και υλοποίησης της εφαρμογής.

1.5 Δομή της Μεταπτυχιακής Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία χωρίζεται σε έξι κεφάλαια:

Στο πρώτο κεφάλαιο, παρουσιάζεται το βασικό αντικείμενο μελέτης της διπλωματικής που είναι το framework της Angular, γίνεται η οριοθέτηση του ερευνητικού αντικειμένου, δίδονται οι βασικοί ορισμοί, το ιστορικό πλαίσιο εξέλιξης της Angular καθώς και ο σκοπός της μεταπτυχιακής εργασίας.

Στο δεύτερο κεφάλαιο, γίνεται μια σύγκριση της Angular με το framework της React όπου παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα του καθενός και ο προγραμματιστής μπορεί να αντιληφθεί καλύτερα τους λόγους για τους οποίους είναι καλύτερα να προτιμήσει τη χρήση του ενός ή του άλλου framework ανάλογα με την εφαρμογή που θέλει να δημιουργήσει.

Στο τρίτο κεφάλαιο, παρουσιάζεται αναλυτικότερα η Angular, οι βασικές αρχές της και η αρχιτεκτονική της.

Στο τέταρτο κεφάλαιο, γίνεται η παρουσίαση της δομής του συστήματος που υλοποιήθηκε στα πλαίσια της μεταπτυχιακής εργασίας καθώς και η σύγκριση και η μελέτη των διαφορετικών αρχιτεκτονικών που χρησιμοποιήθηκαν, καθώς η εργασία υλοποιήθηκε με τρεις διαφορετικές μεθόδους υλοποίησης..

Στο πέμπτο κεφάλαιο, γίνεται παρουσίαση του user interface (διεπαφή χρήστη) της εφαρμογής μέσα από εικόνες και περιγραφή των δυαντοτήτων.

Στο έκτο κεφάλαιο, παρουσιάζονται τα βασικά συμπεράσματα που προέκυψαν μέσα από τη διενέργεια της μεταπτυχιακής εργασίας και δίνονται προτάσεις και ιδέες για μελλοντικές βελτιώσεις.

2 Σύγκριση Angular και React framework

Ένα κρίσιμο θέμα που καλείται να αντιμετωπίσει κάθε μηχανικός είναι η επιλογή και η χρήση σωστού framework της Javascript. Είναι αλήθεια ότι η επιλογή κατάλληλου framework για το ανάλογο έργο που πρόκειται να υλοποιηθεί απαιτεί γνώση πολλών κρίσιμων πτυχών. Σε αυτό το κεφάλαιο γίνεται μία σύγκριση σε δύο από τα πιο δημοφιλή frontend framework και συγκεκριμένα της Angular σε σχέση με τη React προκειμένου να κατανοήσουμε τις βασικές διαφορές, τα πλεονεκτήματα, τα μειονεκτήματα και την κατάλληλη χρήση του κάθε framework του frontend.

Είναι γεγονός ότι, στην πραγματικότητα το παρόν κεφάλαιο δεν έχει ως σκοπό να ορίσει το καλύτερο framework, αλλά να αναδείξη τα ιδιαίτερα χαρακτηριστικά που έχει το κάθε ένα από αυτά προκειμένου να φανερωθούν οι διαφορές μεταξύ της Angular και της React και επίσης να αναδειχθεί ποιο framework είναι καλύτερο ανάλογα με την κατάσταση που καλείται ο προγραμματιστής να αντιμετωπίσει.

2.1 Τι είναι το React framework

Η React είναι μια βιβλιοθήκη της JavaScript ανοιχτού κώδικα που χρησιμοποιείται για την ανάπτυξη του frontend των εφαρμογών. Αναλυτικότερα, χρησιμοποιείται για τη δημιουργία διεπαφής χρήστη ή στοιχείων διεπαφής χρήστη. Επίσης, αρχιτεκτονική της React βασίζεται σε component και επιτρέπει στους προγραμματιστές να δημιουργούν εύκολα διαδραστικές και σύνθετες διεπαφές χρήστη. Οι προγραμματιστές μπορούν να δημιουργήσουν γρήγορες και επεκτάσιμες εφαρμογές για όλες τις πλατφόρμες. Η React διαχειρίζεται μεγάλες πλατφόρμες όπως το Facebook και μια τεράστια κοινότητα προγραμματιστών [5].

2.2 Σύγκριση Angular και React

Η Angular είναι ένα framework της Javascript που δημιουργήθηκε με τη χρήση της Typescript, ενώ το Reactjs είναι μια βιβλιοθήκη της Javascript και χτίστηκε με χρήση Javascript. Η Angular χρησιμοποιείται κυρίως για τη δημιουργία σύνθετων εφαρμογών

εταιρικής ποιότητας, όπως εφαρμογές μιας σελίδας και progressive εφαρμογές ιστού, ενώ η React χρησιμοποιείται για τη δημιουργία σύγχρονων και μεγάλων εφαρμογων ιστού με συχνά μεταβλητά δεδομένα καθώς και υβριδικές εφαρμογές για συσκευές Android και iOS. Τέλος, η εκμάθηση της Angular είναι πιο απότομη λόγω των πάρα πολλών ενσωματωμένων λειτουργιών της, ενώ η React είναι πιο εύκολη στην εκμάθηση ακόμη και από κάποιον αρχάριο.

Σε ένα ευρύτερο πλαίσιο αυτά και μερικά άλλα διακριτά χαρακτηριστικά κάνουν την Angular και την React να διαφέρουν θεμελιωδώς, και ως εκ τούτου, πρέπει να θέσουμε το κοινό έδαφος πάνω στο οποίο μπορούμε να συγκρίνουμε αυτά τα δύο framework.

Κύρια χαρακτηρηστικά της React:

- Επιτρέπει τη χρήση άλλων βιβλιοθηκών (3rd party libraries)
- Εξοικονόμηση χρόνου
- Απλότητα και σύνθεση
- Υποστηρίζεται πλήρως από το Facebook.
- Καλύτερη εμπειρία χρήστη και πολύ γρήγορη απόδοση.
- Ταχύτερη Ανάπτυξη
- Σταθερότητα κώδικα με δέσμευση δεδομένων μίας κατεύθυνσης
- React Components

Κύρια χαρακτηρηστικά της Angular:

- Ενσωματωμένη υποστήριξη για AJAX, HTTP και Observables
- Υποστήριξη μεγάλης κοινότητας
- Συνεπής με την τεχνολογία και τις ανανεώσεις των εκδόσεών της
- Η Typescript προσφέρει αποτελεσματικότητα
- Πιο καθαρή και ευκρινή κωδικοποίηση
- Βελτιωμένη υποστήριξη για τη διαχείριση σφαλμάτων
- Απρόσκοπτες ενημερώσεις χρησιμοποιώντας το Angular CLI
- Φόρμες και έλεγχος-επικύρωση των πεδίων της φόρμας
- Shadow DOM / τοπικό CSS

Πλεονεκτήματα της React

- Εύκολη στην εκμάθηση λόγω του απλού σχεδιασμού της.
- Η σύνταξη μοιάζει με HTML για την οποία επιτρέπει τη δημιουργία προτύπων(templates) και την εξαιρετικά λεπτομερή τεκμηρίωση.
- Οι προγραμματιστές μπορούν να αφιερώσουν περισσότερο χρόνο γράφοντας σύγχρονη JavaScript και λιγότερο χρόνο ανησυχώντας για τον κώδικα που αφορά το framework.
- Καλή υποστήριξη και απόδοση από την πλευρά του διακομιστή, καθιστώντας το ένα ισχυρό framework για εφαρμογές εστιασμένες στο περιεχόμενο.
- Το Facebook προσφέρει τη δυνατότητα "codemod" για την αυτοματοποίηση μεγάλου μέρους της διαδικασίας.
- Οι δεξιότητες που αποκτήθηκαν στην React μπορούν να εφαρμοστούν και στην ανάπτυξη εφαρμογών React Native.
- Όταν συνδυάζετε με το ES6/7, η ReactJS είναι τέλεια για τη διαχείριση μεγάλων δεδομένων με σχετική ευκολία.

Πλεονεκτήματα του Angular

- Προσφέρει ανάπτυξη καθαρού κώδικα.
- Υψηλότερες Επιδόσεις.
- Διασύνδεση τύπου Material Design, Bootstrap, Ionic.
- Το angular framework μπορεί να φροντίσει τη δρομολόγηση(routing), πράγμα που σημαίνει ότι η μετακίνηση από τη μια όψη στην άλλη είναι εύκολη.
- Εύκολες ενημερώσεις με χρήση Angular CLI.

Μειονεκτήματα του React

- Η ενσωμάτωση των Reacts στο παραδοσιακό πλαίσιο MVC, όπως το Rail, απαιτεί περίπλοκη διαμόρφωση.
- Το ReactJS θα απαιτούσε από τους χρήστες να έχουν εις βάθος γνώση σχετικά με την ενσωμάτωση της διεπαφής χρήστη στο πλαίσιο MVC.

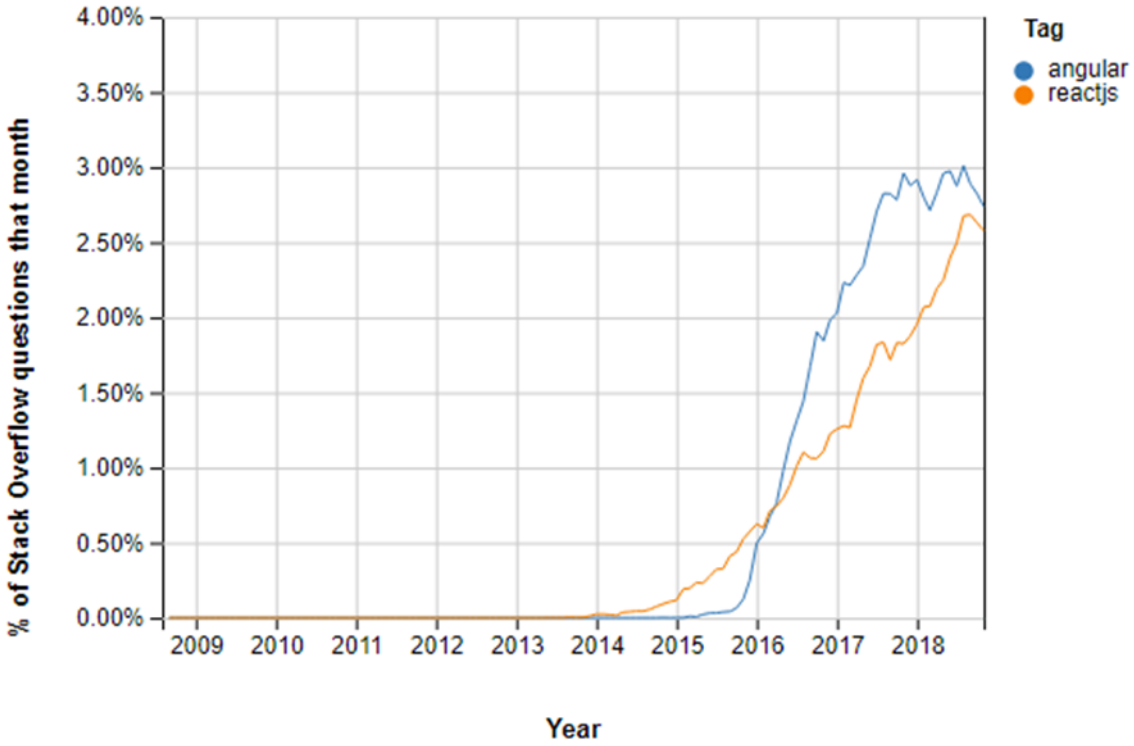
Μειονεκτήματα του Angular

- Είναι δύσκολο στην εκμάθηση, ιδιαίτερα για αρχάριους και απαιτεί χρόνο.

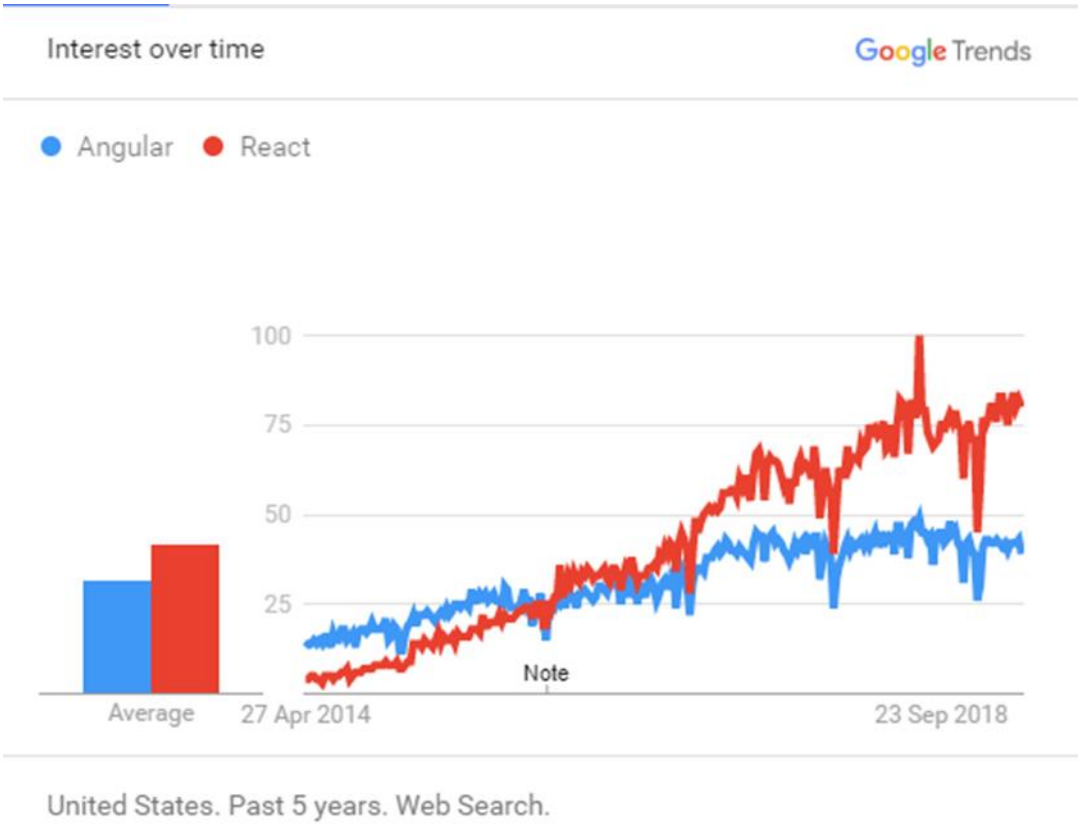
- Υπάρχουν πολλές εκδόσεις της Angular, που μερικές φορές δημιουργεί σύγχυση.
- Οι δυνατότητες Βελτιστοποίηση Ιστοσελίδων για τις Μηχανές Αναζήτησης (SEO) είναι περιορισμένες.

Διαγράμματα σύγκρισης των δύο framework

Στα παρακάτω διαγράμματα φαίνονται κάποια στατιστικά για τη σύγκριση της δημοφιλίας των δύο framework.



Εικόνα 2: Ερωτήσεις στο Stack Overflow για τη React και την Angular [6].



Εικόνα 3: Δημοτικότητα της React έναντι της Angular στο Google Trends [6].

Περιπτώσεις χρήσης React

Το React native framework αποτελεί μία καλή επιλογή για τις παρακάτω περιπτώσεις:

- Εφαρμογές με πολλά συμβάντα(events).
- Όταν η ομάδα ανάπτυξης εφαρμογών έχει εξειδίκευση σε HTML, CSS και JavaScript.
- Όταν οι απαιτήσεις είναι για ιδιαίτερα προσαρμοσμένη και συγκεκριμένη εφαρμογή.

Περιπτώσεις χρήσης Angular

Το Angular είναι ένα πλαίσιο που αποτελεί μία ιδανική επιλογή για τις παρακάτω περιπτώσεις:

- Έτοιμες προς χρήση λύσεις και υψηλότερη παραγωγικότητα.
- Όταν πρόκειται για μια μεγάλης κλίμακας εφαρμογή πλούσια σε χαρακτηριστικά.

- Όταν η ομάδα ανάπτυξης έχει εμπειρία με Java, C# και προηγούμενες εκδόσεις του Angular
- Η πολυπλοκότητα της εφαρμογής παραμένει χαμηλό έως μεσαίο επίπεδο [6].

Συνοπτικά στην παρακάτω εικόνα απεικονίζονται μερικά παραδείγματα με γνωστές εφαρμογές που χρησιμοποιούν είτε το framework της React είτε της Angular.

Εικόνα 4: Εφαρμογές που χρησιμοποιούν Angular ή React [5].

2.3 Συμπεράσματα

Συμπερασματικά θα μπορούσαμε να πούμε τα ότι το Angular είναι ένα πλήρες framework ανάπτυξης για κινητά και web. Από την άλλη πλευρά, το React είναι ένα framework μόνο για την ανάπτυξη διεπαφής χρήστη, το οποίο μπορεί να μετατραπεί σε μια ολοκληρωμένη λύση με τη βοήθεια πρόσθετων βιβλιοθηκών. Επίσης, η React φαίνεται πιο απλή με την πρώτη ματιά και χρειάζεται λιγότερος χρόνος για να ξεκινήσει κανείς να εργάζεται σε ένα έργο με React. Ωστόσο, αυτή η απλότητα ως κύριο πλεονέκτημα της React εξουδετερώνεται επειδή πρέπει κανείς να μάθει να εργάζεται με πρόσθετα πλαίσια και εργαλεία JavaScript. Αντίθετα, η Angular είναι πιο περίπλοκη και χρειάζεται αρκετό χρόνο για εκμάθηση. Ωστόσο, είναι ένα ισχυρό εργαλείο που προσφέρει μια ολιστική εμπειρία ανάπτυξης ιστού και μόλις κάποιος μάθει πώς να εργάζεται με αυτή, αποκομίζει τους καρπούς και τα πλεονεκτήματα της Angular.

Είναι αλήθεια ότι δεν υπάρχει καλύτερο ή χειρότερο framework, καθώς και τα δύο ενημερώνονται συνεχώς για να συμβαδίζουν με τους ανταγωνιστές τους. Για παράδειγμα, ενώ η React πιστεύεται ότι κέρδιζε λόγω του εικονικού DOM του, το Angular ισοφάρισε τη βαθμολογία εφαρμόζοντας τον εντοπισμό αλλαγών. Στη συνέχεια, ενώ η Angular θεωρήθηκε ότι κερδίζει επειδή αναπτύχθηκε από μια τόσο έγκυρη εταιρεία όπως η Google, η τεράστια αφοσιωμένη κοινότητα της React αντιστάθμισε πλήρως τη φήμη της Google και έκανε τη React παρόμοια με την Angular. Τελικά, η σύγκριση React vs Angular είναι θέμα προσωπικής προτίμησης, θέμα δεξιοτήτων και συνηθειών. Λόγου χάρη, ένας αρχάριος στον προγραμματισμό, πιθανότατα θα ωφεληθεί περισσότερο αν ξεκινήσει με την React, ενώ ένας έμπειρος προγραμματιστής, το καλύτερο είναι να συνεχίσει να εργάζεται με ό,τι γνωρίζει καλύτερα [7].

3 Ανάλυση του Angular framework

Στο κεφάλαιο 3 γίνεται ανάλυση και εξήγηση της αρχιτεκτονικής και των βασικών δομών του framework της Angular. Συγκεκριμένα, η Angular αναλύονται βασικές έννοιες όπως η TypeScript, το Component, τα Templates, τα Modules, τα Services και άλλες βασικές αρχές του framework.

3.1 Τι είναι η typescript

Η TypeScript είναι ένα υπερσύνολο της JavaScript. Ειδικότερα, η TypeScript δημιουργείται πάνω από την JavaScript. Στην πραγματικότητα έχουμε την εξής ακολουθία: α) πρώτα, γράφεται ο κώδικας της TypeScript, β) στη συνέχεια, μεταγλωττίζεται ο κώδικα της TypeScript σε απλό κώδικα JavaScript χρησιμοποιώντας έναν μεταγλωττιστή TypeScript και γ) αφού δημιουργηθεί ο απλός κώδικας JavaScript, τότε μπορεί να αναπτυχθεί σε οποιοδήποτε περιβάλλον που εκτελεί JavaScript. Η παρακάτω εικόνα απεικονίζει αυτή την ακολουθία.

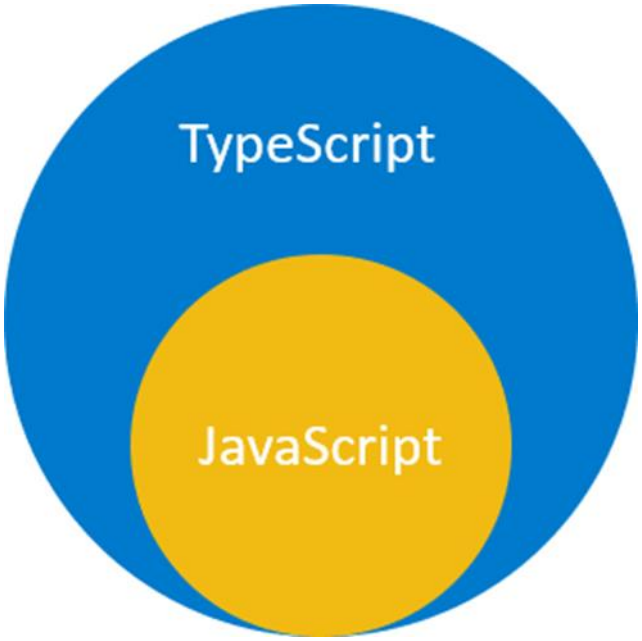
Εικόνα 5: Περιγραφή της TypeScript [8].

Χαρακτηριστικά της TypeScript:

- Τα αρχεία TypeScript χρησιμοποιούν την επέκταση .ts αντί για την επέκταση .js των αρχείων JavaScript.
- Η TypeScript χρησιμοποιεί τη σύνταξη της JavaScript και προσθέτει επιπλέον στοιχεία για την υποστήριξη των τύπων.
- Ένα πρόγραμμα της JavaScript που δεν έχει συντακτικά σφάλματα, είναι επίσης πρόγραμμα της TypeScript, αυτό σημαίνει ότι όλα τα προγράμματα

JavaScript είναι προγράμματα TypeScript. Συνεπώς, αυτό είναι πολύ χρήσιμο στην περίπτωση μετεγκατάστασης μιας υπάρχουσας βάσης κώδικα JavaScript στο TypeScript.

Το παρακάτω διάγραμμα δείχνει τη σχέση μεταξύ TypeScript και JavaScript :



Εικόνα 6: Η TypeScript ως υπερσύνολο της JavaScript [8].

3.2 Component

Τα Components είναι τα δομικά στοιχεία που συνθέτουν μια εφαρμογή. Ειδικότερα, ένα Component περιλαμβάνει μια κλάση TypeScript με το @Component(), ένα HTML template και ένα αρχείο CSS. Το @Component() καθορίζει τις ακόλουθες πληροφορίες ειδικά για το Angular:

- Έναν επιλογέα CSS που ορίζει πώς χρησιμοποιείται το component σε ένα template.
- Ένα HTML template που καθοδηγεί την Angular πώς να αποδώσει το component.
- Ένα προαιρετικό σύνολο CSS που καθορίζει την εμφάνιση των στοιχείων του HTML template.

Το παρακάτω είναι ένα απλό παράδειγμα Angular component.

```
import { Component } from '@angular/core';
@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Για να χρησιμοποιήσετε αυτό το component, γράφετε τα ακόλουθα σε ένα template:

```
<hello-world></hello-world>
```

Όταν το Angular αποδίδει αυτό το component, το DOM της HTML που προκύπτει μοιάζει με αυτό:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Το μοντέλο των component της Angular προσφέρει ισχυρή ενθυλάκωση και μια διαισθητική δομή εφαρμογής. Τα components καθιστούν επίσης την εφαρμογή σας ανώδυνη στο unit test και μπορούν να βελτιώσουν τη γενική αναγνωσιμότητα του κώδικά σας καθώς διατηρούν τον κώδικα αρκετά καθαρό [9]..

3.3 Templates

Κάθε component έχει ένα HTML template που δηλώνει πώς αποδίδεται αυτό το component. Αυτό το template ορίζεται είτε ενσωματωμένο στο .ts αρχείο του component είτε ξεχωριστά στο HTML αρχείο.

Το Angular προσθέτει στοιχεία σύνταξης που επεκτείνουν το HTML, ώστε να μπορεί κάποιος να εισαγάγει δυναμικές τιμές από το component. Το Angular ενημερώνει αυτόματα το αποδοθέν DOM όταν αλλάζει η κατάσταση του component. Μια εφαρμογή αυτής της δυνατότητας είναι η εισαγωγή δυναμικού κειμένου, όπως φαίνεται στο παρακάτω παράδειγμα.

```
<p>{{ message }}</p>
```

Η τιμή για το μήνυμα προέρχεται από το component:

```
import { Component } from '@angular/core';

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
   message = 'Hello, World!';
}
```

Όταν η εφαρμογή φορτώνει το component και το template, ο χρήστης βλέπει τα εξής:

```
<p>Hello, World!</p>
```

Η χρήση διπλών αγγύλων δίνουν εντολή στην Angular να παρεμβάλλει τα περιεχόμενα μέσα τους, δηλαδή στο παράδειγμα παραπάνω δίνει την εντολή να δείξει

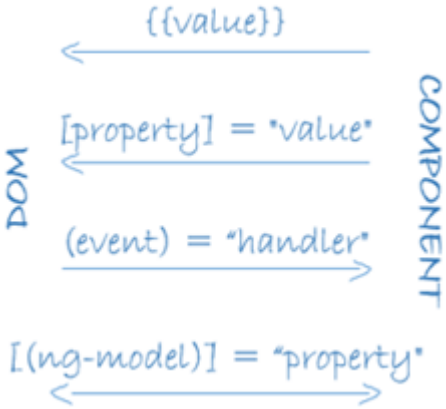
την τιμή της μεταβλητής message , η οποία στο παράδειγμα έχει την τιμή 'Hello, World!' [9].

3.4 Data binding – Σύνδεση Δεδομένων

Χωρίς τη χρήση του framework, θα έπρεπε να είναι στην ευθύνη του προγραμματιστή η προώθηση τιμών δεδομένων στα στοιχεία ελέγχου της HTML και η μετατροπή των απαντήσεων των χρηστών σε ενέργειες(actions) και ενημερώσεις τιμών. Η σύνταξη μιας τέτοιας λογικής push and pull με το χέρι είναι κουραστική, επιρρεπής σε σφάλματα και εφιάλτης για ανάγνωση, όπως μπορεί να επιβεβαιώσει κάθε έμπειρος frontend προγραμματιστής σε JavaScript.

Η Angular υποστηρίζει αμφίδρομη σύνδεση δεδομένων(two-way data binding), έναν μηχανισμό για τον συντονισμό των τμημάτων ενός template με τα μέρη ενός component. Το γεγονός αυτό επιτυγχάνεται αν προσθέσει κάποιος δεσμευτική σήμανση(binding markup) στο HTML template το οποίο δείχνει στην Angular πώς να συνδέσει τις δύο πλευρές.

Το παρακάτω διάγραμμα δείχνει τις τέσσερις περιπτώσεις σήμανσης σύνδεσης δεδομένων(data binding markup). Κάθε περίπτωση έχει μια κατεύθυνση: προς το DOM, από το DOM ή και τα δύο [10].



Εικόνα 7: Data binding [10].

3.5 Pipes – Σωλήνες

Τα Angular pipes επιτρέπουν την δήλωση μετασχηματισμών τιμών-εμφάνισης(display-values) στο HTML template. Μια κλάση με το @Pipe ορίζει μια

συνάρτηση που μετατρέπει τις τιμές εισόδου σε τιμές εξόδου για εμφάνιση σε μια προβολή.

Επίσης, η Angular ορίζει διάφορα pipes, όπως το pipe ημερομηνίας και το pipe νομίσματος και πολλά ακόμη που βρίσκονται στη λίστα του Pipes API της Angular. Επίσης, υπάρχει η δυνατότητα να ορίσει ο προγραμματιστής νέους σωλήνες.

Για να καθοριστεί ένας μετασχηματισμός τιμής σε ένα HTML template, χρησιμοποιείται ο τελεστή σωλήνα (|) όπως φαίνεται στο παράδειγμα παρακάτω.

{{interpolated_value | pipe_name}}

Επιπλέον, υπάρχει η δυνατότητα να αλληλοσυνδεθούν pipes, στέλνοντας την έξοδο της μίας συνάρτησης pipe για να μετατραπεί από μια άλλη συνάρτηση pipe. Ένα pipe μπορεί επίσης να λάβει ορίσματα που ελέγχουν τον τρόπο με τον οποίο εκτελεί τον μετασχηματισμό του. Στο παράδειγμα που ακολουθεί, φαίνεται πως μπορεί κάποιος να περάσει την επιθυμητή μορφή στη γραμμή ημερομηνίας.

```
<!-- Default format: output 'Jun 15, 2015'-->
<p>Today is {{today | date}}</p>

<!-- fullDate format: output 'Monday, June 15, 2015'-->
<p>The date is {{today | date:'fullDate'}}</p>

<!-- shortTime format: output '9:43 AM'-->
<p>The time is {{today | date:'shortTime'}}</p>
```

3.6 Directives

Είναι γεγονός ότι, τα Angular templates είναι δυναμικά. Όταν η Angular αποδίδει τα templates, μετατρέπει το DOM σύμφωνα με τις οδηγίες που δίνονται από τα directives. Ένα directive είναι μια κλάση με το διακριτικό @Directive().

Επιπρόσθετα, ένα component είναι τεχνικά ένα directive. Ωστόσο, τα components είναι τόσο διακριτά και κεντρικά στις εφαρμογές της Angular που η Angular ορίζει το

@Component(), το οποίο ουσιαστικά επεκτείνει το @Directive() με χαρακτηριστικά προσανατολισμένα σε template.

Επιπλέον, εκτός από τα components, υπάρχουν και δύο άλλα είδη των directives: τα structural και τα attribute. Η Angular ορίζει έναν αριθμό από directives και των δύο ειδών και δίνει την δυνατότητα να ορίσει κάποιος τα δικά του χρησιμοποιώντας το διακριτικό @Directive().

Τέλος, όπως και για τα components, τα μεταδεδομένα για ένα directive συσχετίζουν την κλάση με έναν selector που χρησιμοποιείτε για να εισαχθεί στην HTML. Στα templates, τα directives εμφανίζονται συνήθως μέσα σε μια ετικέτα στοιχείου(element tag) ως χαρακτηριστικά, είτε ονομαστικά είτε ως στόχος μιας ανάθεσης ή μιας δέσμευσης.

Υπάρχουν δύο βασικές κατηγορίες από directives: α) τα Structural directives και β) τα Attribute directives.

Structural directives

Τα Structural directives αλλάζουν τη διάταξη προσθέτοντας, αφαιρώντας και αντικαθιστώντας στοιχεία στο DOM. Το παράδειγμα με το παρακάτω template χρησιμοποιεί δύο ενσωματωμένες structural directives για να προσθέσει λογική στην εφαρμογή στον τρόπο απόδοσης της προβολής(view).

```
<li *ngFor="let hero of heroes"></li>
<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

Πίνακας 1: Directives

DIRECTIVES	**ΛΕΠΤΟΜΕΡΕΙΕΣ**
*ngFor	Λειτουργεί επαναληπτικά, που λέει στην Angular να δημιουργήσει ένα ανά hero στη λίστα με heroes.
*ngIf	Ένας έλεγχος (conditional), ο οποίος περιλαμβάνει το στοιχείο HeroDetail μόνο εάν υπάρχει ένας επιλεγμένος hero.

Attribute directives

Τα Attribute directives αλλάζουν την εμφάνιση ή τη συμπεριφορά ενός υπάρχοντος στοιχείου. Στα templates μοιάζουν με κανονικά HTML attributes, εκεί άλλωστε οφείλεται και το όνομά τους attributes.

Η οδηγία ngModel, η οποία εφαρμόζει αμφίδρομη σύνδεση δεδομένων, είναι ένός attribute directive. Το ngModel τροποποιεί τη συμπεριφορά ενός υπάρχοντος στοιχείου (συνήθως <input>) ορίζοντας την ιδιότητα τιμής εμφάνισης του και ανταποκρινόμενος σε συμβάντα αλλαγής.

<input type="text" id="hero-name" [(ngModel)]="hero.name">

Το Angular περιλαμβάνει προκαθορισμένα directives που αλλάζουν:

- Τη δομή διάταξης, όπως το ngSwitch και
- Πτυχές των components και στοιχείων DOM, όπως το ngStyle και το ngClass [11].

3.7 Dependency injection

Το Dependency injection επιτρέπει τη δήλωση των εξαρτήσεων των κλάσεων TypeScript χωρίς να πρέπει κάποιος να φροντίσει για τη δημιουργία τους. Αντίθετα, η Angular χειρίζεται το instantiation χωρίς να χρειάζεται κάτι επιπλέον από τον προγραμματιστή. Αυτό το μοτίβο σχεδίασης επιτρέπει τη δημιουργία πιο ελεγχόμενου και ευέλικτου κώδικα. Η κατανόηση του dependency injection δεν είναι απαραίτητη για τα πρώτα στάδια χρήσης της Angular, αλλά συνιστάται ανεπιφύλακτα ως βέλτιστη πρακτική. Πολλές πτυχές της Angular το εκμεταλλεύονται σε κάποιο βαθμό.

Το ακόλουθο κομμάτι κώδικα αποτελεί ένα ενδεικτικό παράδειγμα για το πώς λειτουργεί το dependency injection. Το πρώτο αρχείο, logger.service.ts, ορίζει μια κλάση Logger. Αυτή η κλάση περιέχει μια συνάρτηση writeCount που καταγράφει έναν αριθμό στην κονσόλα.

import { Injectable } from '@angular/core';

@Injectable({providedIn: 'root'})

```
export class Logger {
  writeCount(count: number) {
    console.warn(count);
  }
}
```

Στη συνέχεια, το αρχείο hello-world-di.component.ts ορίζει ένα Angular component. Αυτό το στοιχείο περιέχει ένα κουμπί που χρησιμοποιεί τη συνάρτηση writeCount της κλάσης Logger. Για πρόσβαση σε αυτήν τη συνάρτηση, η υπηρεσία Logger εισάγεται στην κλάση HelloWorldDI προσθέτοντας private logger: Logger στον κατασκευαστή.

```
import { Component } from '@angular/core';
import { Logger } from '../logger.service';

@Component({
  selector: 'hello-world-di',
  templateUrl: './hello-world-di.component.html'
})
export class HelloWorldDependencyInjectionComponent  {
  count = 0;
  constructor(private logger: Logger) { }

  onLogMe() {
    this.logger.writeCount(this.count);
    this.count++;
  }
}
```

3.8 Angular CLI

Το Angular CLI είναι ο πιο γρήγορος, απλός και συνιστώμενος τρόπος ανάπτυξης εφαρμογών Angular. Το Angular CLI κάνει ορισμένες εργασίες χωρίς προβλήματα και αποτελεί ένα πολύτιμο εργαλείο για τη δημιουργία των εφαρμογών.

Στον παρακάτω πίνακα παρουσιάζονται βασικές εντολές του Angular CLI.

Πίνακας 2: εντολές Angular CLI

ΕΝΤΟΛΗ	**ΛΕΠΤΟΜΕΡΕΙΕΣ**
ng build	Μεταγλωττίζει μια εφαρμογή Angular.
ng serve	Δημιουργεί και εξυπηρετεί την εφαρμογή σας, αναδημιουργώντας τις αλλαγές αρχείων.
ng generate	Δημιουργεί ή τροποποιεί αρχεία με βάση ένα σχηματικό.
ng test	Εκτελεί unit tests σε ένα έργο.
ng e2e	Δημιουργεί και εξυπηρετεί μια εφαρμογή Angular και, στη συνέχεια, εκτελεί δοκιμές από άκρο σε άκρο.

3.9 First-party libraries

Στον παρακάτω πίνακα παρατίθεται μια σύντομη επισκόπηση με μερικά από τα βασικά αρχιτεκτονικά στοιχεία που χρησιμοποιούνται κατά την κατασκευή εφαρμογών Angular. Η χρήση αυτών των βιβλιοθηκών δίνει πολλά οφέλη στον προγραμματιστή ιδιαίτερα όταν πρόκειται για μεγάλες εφαρμογές στις οποίες μπορεί κάποιος να προσθέσει επιπλέον λειτουργίες όπως η πλοήγηση στον ιστότοπο ή η εισαγωγή χρήστη.

Μερικές από τις βιβλιοθήκες που υπάρχουν στη διάθεσή του προγραμματιστή παρουσιάζονται παρακάτω.

Πίνακας 3: First-party libraries της Angular

ΒΙΒΛΙΟΘΗΚΗ	**ΛΕΠΤΟΜΕΡΕΙΕΣ**
Angular Router	Προηγμένη πλοήγηση και δρομολόγηση από την πλευρά του πελάτη με βάση τα Angular components. Υποστηρίζει lazy-loading, nested routes, προσαρμοσμένη αντιστοίχιση διαδρομής και πολλά άλλα.
Angular Forms	Ενιαίο σύστημα για φόρμες και επικύρωση των πεδίων της φόρμας.
Angular HttpClient	HTTP client που μπορεί να τροφοδοτήσει πιο προηγμένη επικοινωνία πελάτη-διακομιστή.
Angular Animations	Πλούσιο σύστημα για animations με βάση την κατάσταση εφαρμογής.
Angular PWA	Εργαλεία για τη δημιουργία Progressive Web Applications (PWA)
Angular Schematics	Αυτοματοποιημένα εργαλεία ανακατασκευής και ενημέρωσης που απλοποιούν την ανάπτυξη σε μεγάλη κλίμακα.

Αυτές οι βιβλιοθήκες επεκτείνουν τις δυνατότητες της εφαρμογής, ενώ επιτρέπουν στον προγραμματιστή να εστιάσει περισσότερο στις δυνατότητες που κάνουν την εφαρμογή πιο αποδοτική. Τέλος, είναι αλήθεια ότι αυτές οι βιβλιοθήκες μπορούν να βοηθήσουν τον προγραμματιστή να προσθέσει δυνατότητες στις εφαρμογές και να λύσει πολλά προβλήματα [9].

3.10 Angular Routing

Σε μια εφαρμογή μιας σελίδας (SPA), αλλάζει αυτό που βλέπει ο χρήστης εμφανίζοντας ή αποκρύπτοντας τμήματα της οθόνης που αντιστοιχούν σε συγκεκριμένα components, αντί να ζητηθεί από τον διακομιστή να φέρει μια νέα σελίδα. Καθώς οι χρήστες εκτελούν εργασίες εφαρμογών, πρέπει να μετακινούνται μεταξύ των

διαφορετικών προβολών που έχουν οριστεί από τον προγραμματιστή. Με άλλα λόγια το routing σχετίζεται με τον τρόπο διαχείρισης των paths που αντιστοιχούν οι «σελίδες» και ο τρόπος με τον οποίο αυτές αλλάζουν. Πιο συγκεκριμένα, για να χειριστεί κάποιος την πλοήγηση από τη μια προβολή στην άλλη, χρησιμοποιεί το Angular Router. Το Angular Router ενεργοποιεί την πλοήγηση ερμηνεύοντας μια διεύθυνση URL του προγράμματος περιήγησης ως οδηγία για την αλλαγή της προβολής [12].

4 Μεθοδολογία Ανάλυσης, σχεδίασης και ανάπτυξης της εφαρμογής

Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική της εφαρμογής που αναπτύχθηκε στα πλαίσια της εργασίας. Αναλυτικότερα, η εφαρμογή είναι ένα ηλεκτρονικό βιβλίο συνταγών. Με τη χρήση αυτής της εφαρμογής ο χρήστης έχει την δυνατότητα να δημιουργήσει προσωπικό λογαριασμό στην εφαμογή, να συνδεθεί σε αυτή και στη συνέχεια να δημιουργήσει και να αποθηκεύσει τις συνταγές που επιθυμεί. Επίσης, ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει τα δεδομένα των συνταγών του. Ακόμη, η εφαρμογή δίνει την δυνατότητα στο χρήστη να δημιουργήσει μία λίστα αγορών είτε συμπληρώνοντας ο ίδιος τα υλικά με τις αντίστοιχες ποσότητες που θέλει να αγοράσει είτε περνώντας αυτόματα τα συστατικά των συνταγών στη λίστα αγορών.

Αξίζει να σημειωθεί ότι η εφαρμογή έχει υλοποιηθεί με τρεις διαφορετικούς τρόπους-αρχιτεκτονικές, οι οποίες συγκρίνονται μεταξύ τους προκειμένου να καταλήξουμε σε ασφαλή συμπεράσματα. Η πρώτη υλοποίηση έγινε με απλό τρόπο χρησιμοποιώντας μόνο ένα κεντρικό module το «app.module.ts». Στη δεύτερη υλοποίηση έγιναν βελτιστοποιήσεις και δημιουργήθηκαν και άλλα modules, ώστε να εφαρμοστεί η τεχνική lazy loading στην εφαρμογή και στην τρίτη υλοποίηση έγινε χρήση του NgRx στην εφαρμογή.

Επιπλέον, πρέπει ακόμη να σημειωθεί πως η ανάπτυξη της εφαρμογής είναι βασισμένη σε ήδη υπάρχουσες υλοποιήσεις που βρίσκονται στον ιστό και σε open-source πλατφόρμες. Ειδικότερα, βασική πηγή για τις υλοποιήσεις αποτελούν η σειρά από τα βίντεο για την εκμάθηση της Angular που βρίσκεται εδώ [13] καθώς επίσης και δύο repositories από το GitHub [14] [15].

Συνοψίζοντας, το κεφάλαιο αυτό έχει τους εξής στόχους:

A. Την ανάλυση της αρχιτεκτονικής της εφαρμογής.
B. Τη σύγκριση τριών διαφορετικών αρχιτεκτονικών που χρησιμοποιήθηκαν για την εφαρμογή.

4.1 NgRx

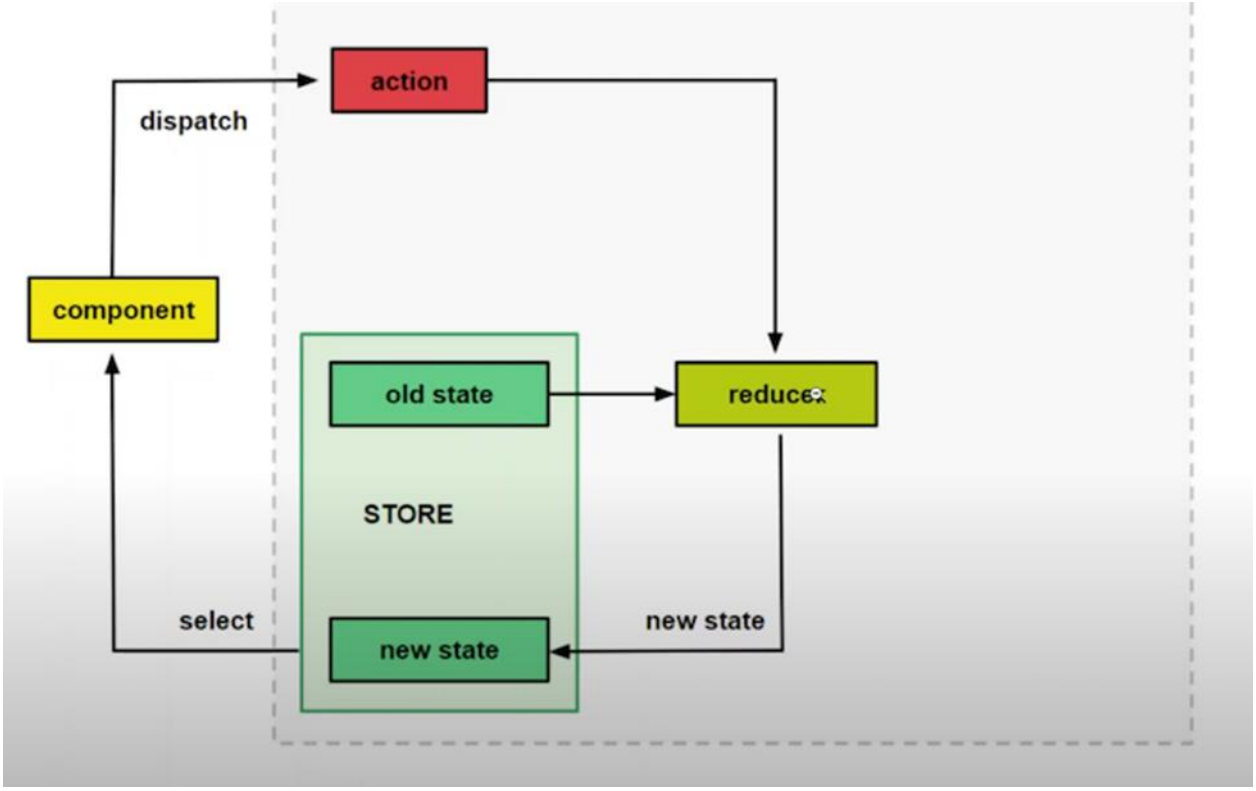
Το NgRx είναι ένα framework που μας επιτρέπει να δημιουργήσουμε reactive εφαρμογές στην Angular. Επίσης, το NgRx βασίζεται στο μοτίβο Redux - ενοποιεί τα συμβάντα(events) στην εφαρμογή και εξάγει την κατάσταση(state) χρησιμοποιώντας το RxJS (Reactive Extensions Library for Javascript). Η βασική ιδέα είναι ότι το NgRx αποθηκεύει ένα μεμονωμένο state και χρησιμοποιεί actions για να εκφράσει τις αλλαγές στο state. Το NgRx υπερέχει στη διαχείριση πολύπλοκων states, καθιστώντας το ιδανικό για εφαρμογές με πολλές αλληλεπιδράσεις με τους χρήστες και πολλαπλές πηγές δεδομένων [16]. Επιπλέον, το NgRx κάνει την ανάπτυξη Angular ευκολότερη απλοποιώντας την κατάσταση της εφαρμογής σε αντικείμενα και επιβάλλοντας μονοκατευθυντική ροή δεδομένων [17].

Επιπρόσθετα, το NgRx χρησιμοποιεί την έννοια Redux της μονοκατευθυντικής ροής δεδομένων, όπου όλα τα δεδομένα εφαρμογής περνούν από τον ίδιο κύκλο ζωής. Αυτή η μονοκατευθυντική ροή δεδομένων καθιστά την κατάσταση της εφαρμογής πιο προβλέψιμη και επομένως πιο κατανοητή [16]. Το NgRx αποτελείται από πέντε κύριες έννοιες:

I. Store, ένα κεντρικό store που διατηρεί όλη το state της εφαρμογής
II. Action, το οποίο περιγράφει όλες τις αλλαγές στο state της εφαρμογής
III. Reducers, που συνδέουν το store και τα actions μεταξύ τους χρησιμοποιώντας την καθορισμένο action για να πραγματοποιήσουν μια μετάβαση από ένα παλιό state σε άλλο νεότερο, ανάλογα με το action.
IV. Selectors, είναι συναρτήσεις που χρησιμοποιούνται για τη λήψη νέων τμημάτων state από το store.
V. Effects, χειρίζονται τις επιπτώσεις του κάθε action. Αυτές οι επιπτώσεις αφορούν την επικοινωνία με ένα εξωτερικό API μέσω HTTP κατά την αποστολή ενός συγκεκριμένου action έως την αποστολή ενός άλλου action για την ενημέρωση ενός άλλου μέρους του state. Πιο συγκεκριμένα, σε μια εφαρμογή Angular που βασίζεται σε υπηρεσίες, τα components είναι υπεύθυνα για την αλληλεπίδραση με εξωτερικούς πόρους απευθείας μέσω υπηρεσιών. Αντίθετα, τα effects παρέχουν έναν τρόπο αλληλεπίδρασης με αυτές τις υπηρεσίες και τις απομονώνουν από τα components. Τα effects είναι χειρίζονται εργασίες όπως η

λήψη δεδομένων(fetching data) και άλλες εξωτερικές αλληλεπιδράσεις όπου τα components δεν χρειάζονται ρητή γνώση για αυτές τις αλληλεπιδράσεις [18].

Το παρακάτω διάγραμμα δείχνει τον κύκλο ζωής της διαχείρισης κατάστασης σε NgRx.



Εικόνα 8: Κύκλος ζωής του NgRx [17].

Το παραπάνω διάγραμμα απεικονίζει ξεκάθαρα την έννοια του Redux και δείχνει πώς διαχειρίζεται το state. Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα κουμπί σε ένα συγκεκριμένο component που όταν πατηθεί, αλλάζει την τιμή της κεφαλίδας της σελίδας. Θα χρησιμοποιήσουμε το NgRx για να το χειριστούμε. Πρώτον, το component αποστέλλει ένα action. Στη συνέχεια, το action πηγαίνει στον reducer. Ο reducer είναι μια απλή μέθοδος που παίρνει το τρέχων state και το action ως παράμετρο και στη συνέχεια επιστρέφει ένα νέο state. Όταν ο reducer επιστρέφει το νέο state, το component στη συνέχεια εγγράφεται σε έναν επιλογέα(selector) για να πάρει τη νέα τιμή. Ακόμη, το state δεν αλλάζει ποτέ άμεσα. Αντίθετα, ο reducer δημιουργεί πάντα ένα νέο state. Αυτό είναι γνωστό ως αμετάβλητο(immutability) [17].

4.2 Firebase

Το Firebase είναι ένα σύνολο υπηρεσιών υποστήριξης cloud computing και πλατφορμών ανάπτυξης εφαρμογών που παρέχονται από την Google. Φιλοξενεί βάσεις δεδομένων, υπηρεσίες, έλεγχο ταυτότητας(authentication) και ενοποίηση για μια ποικιλία εφαρμογών, συμπεριλαμβανομένων των Android, iOS, JavaScript, Node.js, Java, Unity, PHP και C++ [19].

Το Firebase χρησιμοποιήθηκε στην εργασία για δύο σκοπούς: α) για βάση δεδομένων με τη χρήση του Realtime Database και β) για έλεγχο ταυτότητας – authentication κατά την είσοδο του χρήστη.

Firebase Real-time Database

Το Firebase Real-time Database είναι μια βάση δεδομένων NoSQL που φιλοξενείται στο cloud που επιτρέπει την αποθήκευση και τον συγχρονισμό μεταξύ των χρηστών σε πραγματικό χρόνο. Η βάση δεδομένων σε πραγματικό χρόνο είναι πραγματικά μόνο ένα μεγάλο αντικείμενο JSON που οι προγραμματιστές μπορούν να διαχειριστούν σε πραγματικό χρόνο. Με το API, η βάση δεδομένων Firebase παρέχει στην εφαρμογή τόσο την τρέχουσα τιμή των δεδομένων όσο και τυχόν ενημερώσεις σε αυτά τα δεδομένα.

Επιπλέον, ο συγχρονισμός σε πραγματικό χρόνο διευκολύνει τους χρήστες να έχουν πρόσβαση στα δεδομένα τους από οποιαδήποτε συσκευή, είτε είναι ιστός είτε κινητή. Η βάση δεδομένων σε πραγματικό χρόνο βοηθά επίσης τους χρήστες να συνεργάζονται μεταξύ τους [20].

Τέλος, στην εφαρμογή η βάση δεδομένων χρησιμοποιείται για την αποθήκευση και τη λήψη δεδομένων των συνταγών, αυτό επιτυγχάνεται μέσω ενός http request στη βάση δεδομένων του firebase με τη χρήση ενός μοναδικού API key.

Firebase Authentication

Το Firebase Authentication παρέχει υπηρεσίες υποστήριξης, εύχρηστα SDK και έτοιμες βιβλιοθήκες διεπαφής χρήστη για τον έλεγχο ταυτότητας των χρηστών στην εφαρμογή. Επίσης, υποστηρίζει έλεγχο ταυτότητας χρησιμοποιώντας κωδικούς πρόσβασης, αριθμούς τηλεφώνου, δημοφιλείς ομοσπονδιακούς παρόχους ταυτότητας όπως το Google, το Facebook και το Twitter και πολλά άλλα [21].

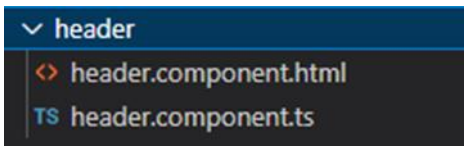
Στην εργασία αυτή για τον έλεγχο ταυτότητας χρησιμοποιούνται δύο στοιχεία του χρήστη: α) email και β) κωδικός. Ειδικότερα, τα στοιχεία αυτά εισάγονται από τον χρήστη και στέλνονται στον server για έλεγχο. Εάν όλα αυτά τα δεδομένα είναι έγκυρα, ο server θα στείλει στον client ένα token, ένα JSON Web Token. Αυτή είναι μια κωδικοποιημένη συμβολοσειρά που περιέχει πολλά μεταδεδομένα. Πιο αναλυτικά, αυτή η συμβολοσειρά είναι κωδικοποιημένη και δεν είναι κρυπτογραφημένη, αυτό είναι σημαντικό, διότι σημαίνει ότι αυτή η συμβολοσειρά θα μπορούσε να αναλυθεί και να διαβαστεί από τον πελάτη.

Επιπρόσθετα, το πιο σημαντικό είναι ότι αυτή η συμβολοσειρά, αυτό το token δημιουργείται στον server με έναν συγκεκριμένο αλγόριθμο και ένα συγκεκριμένο μυστικό που μόνο ο server γνωρίζει και μόνο ο server μπορεί να επικυρώσει τα εισερχόμενα token για την εγκυρότητά τους, επειδή η ιδέα είναι ότι ο client, που είναι το πρόγραμμα περιήγησής μας , το Angular App μας, αποθηκεύει αυτό το token σε κάποιο χώρο αποθήκευσης, όπως έναν τοπικό χώρο αποθήκευσης του προγράμματος περιήγησης και επισυνάπτει το token σε κάθε αίτημα που στη συνέχεια αποστέλλεται στον server που πρέπει να πιστοποιηθεί.

4.3 Δομή του κώδικα

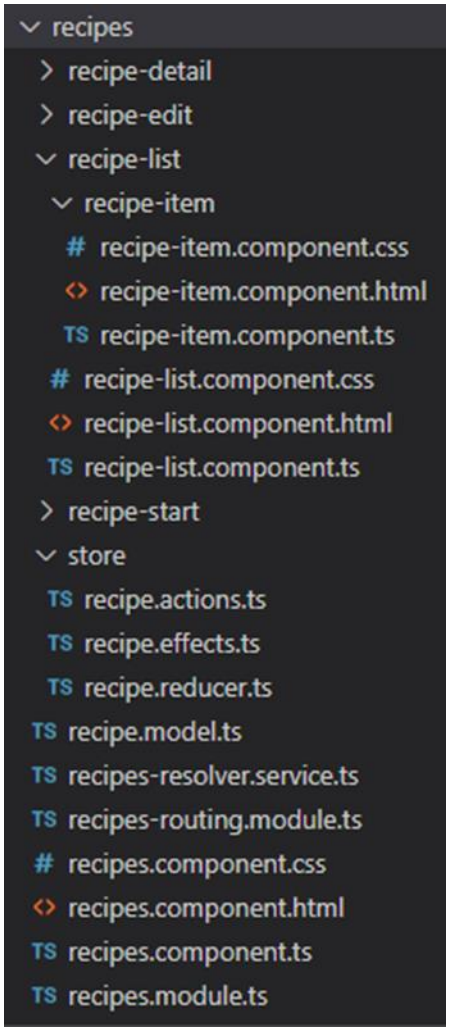
Σε αυτή την ενότητα γίνεται παρουσίαση της βασικής δομής του κώδικα. Σε αυτή η ενότητα γίνεται η παρουσίαση της δομής των αρχείων σύμφωνα με την τελευταία υλοποίηση της εφαρμογής, δηλαδή με τη χρήση του NgRx και τις βελτιστοποιήσεις για lazy loading.

Header Component



Περιλαμβάνει τον κώδικα που χρησιμοποιείται για την υλοποίηση της γραμμής πλοήγησης που υπάρχει στην εφαρμογή (navigation bar). Δηλαδή την γραμμή που περιέχει τις επιλογές «Βιβλίο Συνταγών», «Συνταγές», «Λίστα Αγορών», «Έξοδος», «Είσοδος», «Διαχείριση».

Φάκελος «recipes»



recipes-detail

Αυτό το component χρησιμοποιείται για την παρουσίαση της συνταγή με όλες τις λεπτομέρειές της. Συγκεκριμένα, παρουσιάζεται το όνομα της συνταγής, η φωτογραφία, η περιγραφή για την εκτέλεση, τα υλικά και οι αντίστοιχες ποσότητες που χρειάζονται για την εκτέλεση. Τέλος, υπάρχει κουμπί για την «Διαχείριση Συνταγής» με επιλογές «Προσθήκη στη Λίστα Αγορών», «Επεξεργασία συνταγής» και «Διαγραφή συνταγής».

recipes-edit

Χρησιμοποείται για την επιλογή «Επεξεργασία συνταγής». Το παραπάνω component μας δίνει την δυνατότητα να επεξεργαστούμε και να αλλάξουμε τα στοιχεία μιας συνταγής, στη συνέχεια έχουμε την επιλογή να κάνουμε αποθήκευση των αλλαγών ή

ακύρωση. Η επεξεργασία της συνταγής γίνεται πάνω στα πεδία της φόρμας που χρησιμοποιούμε για την δημιουργία των συνταγών.

recipe-list

Το παραπάνω component περιέχει εμφωλευμένο ακόμα ένα component το «recipe-item». Το recipe-list χρησιμοποιείται για να μας δώσει την επιλογή για δημιουργία νέας συνταγής με το κουμπί «Νέα Συνταγή» και για να παρουσιάσει την λίστα με τις αποθηκευμένες συνταγές, δηλαδή όλα τα recipe items.

recipes-start

Component για την αρχίκη εμφάνιση της σελίδας με τις συνταγές. Δηλαδή πριν επιλέξουμε προκειμένου να δούμε κάποια συνταγή, εμφανίζεται το μήνυμα «Παρακαλώ επιλέξτε μία συνταγή!».

Recipe model

Το αρχείο «recipe.model.ts» περιέχει την κλάση για την δημιουργία μιας συνταγής. Με άλλα λόγια περιέχει τα βασικά δομικά στοιχεία μιας συνταγής. Αυτά τα στοιχεία είναι το όνομα, η περιγραφή, η φωτογραφία και τα συστατικά.

Recipe service

Το «recipes-resolver.service.ts» είναι η υπηρεσία συνταγών, που είναι υπεύθυνη για τη διαχείριση των δεδομένων των συνταγών.

Recipe routing

Το «recipes-routing.module.ts» εκεί γίνεται η διαχείριση των paths για τα recipes.

Recipe module

Από αυτό το module φορτώνουμε και διαχειριζόμαστε τα components που αφορούν τα recipes.

Store

Recipe actions

Περιέχει όλα τα actions που συμβαίνουν για τη διαχείριση των συνταγών (προσθήκη, επεξεργασία, διαγραφή).

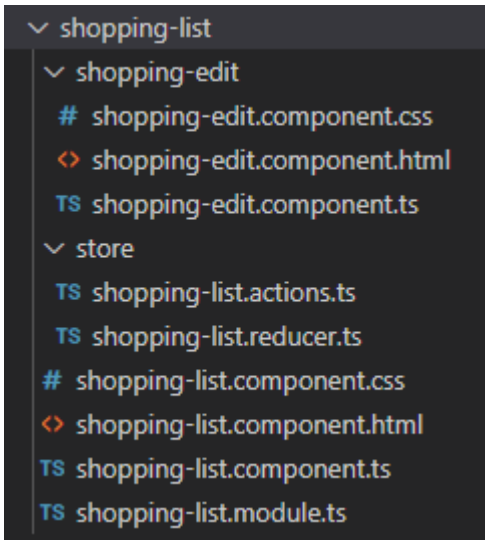
Recipe effects

Στο «recipe.effects. ts» γίνεται η διαχείριση των δεδομένων σε σχέση με το backend (firebase). Σε αυτό το αρχείο γίνεται λήψη δεδομένων από τη βάση δεδομένων στο firebase και αποθήκευση των δεδομένων στη βάση δεδομένων.

Recipe reducer

Είναι υπεύθυνο για την διαχείριση του state, παίρνει το τρέχων state και το action ως παράμετρο και στη συνέχεια επιστρέφει ένα νέο state.

Shopping List



shopping-list component

Αυτό το component παρουσιάζει την λίστα με τα υλικά που είναι για αγορά(όνομα υλικού και ποσοτητα) και μας δίνει τις επιλογές για προσθήκη νέου υλικού και εκκαθάριση των δεδομένων.

shopping-edit component

Αυτό το component χρησιμοποιείται όταν επιλέγουμε κάποιο από τα υλικά της λίστας προκειμένουν να μας δώσει την δυνατότητα για επεξεργασία κάνοντας ενημέρωση των στοιχείων του υλικού.

shopping-list module

Από αυτό το module φορτώνουμε και διαχειριζόμαστε τα components που αφορούν το shopping list.

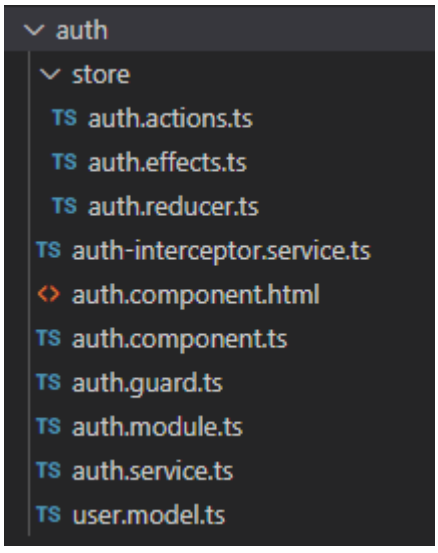
shopping-list actions

Περιέχει όλα τα actions που συμβαίνουν για τη διαχείριση της λίστας αγορών (προσθήκη, επεξεργασία, διαγραφή).

shopping-list reducer

Είναι υπεύθυνο για την διαχείριση του state, παίρνει το τρέχων state και το action ως παράμετρο και στη συνέχεια επιστρέφει ένα νέο state.

Auth



Ο έλεγχος ταυτότητας (authentication) είναι η διαδικασία αντιστοίχισης του επισκέπτη μιας διαδικτυακής εφαρμογής με το προκαθορισμένο σύνολο ταυτότητας

χρήστη στο σύστημα. Με άλλα λόγια, είναι η διαδικασία αναγνώρισης της ταυτότητας του χρήστη. Ο έλεγχος ταυτότητας είναι πολύ σημαντική διαδικασία στο σύστημα όσον αφορά την ασφάλεια [22].

Ο χρήστης της εφαρμογής προκειμένου να αποκτήσει πρόσβαση στο περιεχόμενό της θα πρέπει πρώτα να κάνει εγγραφή συμπληρώνοντας e-mail και κωδικό και στη συνέχεια να πραγματοποιήση είσοδο με αυτά τα στοιχεία.

Auth service

Στο αρχείο «auth.service.ts», είναι μια υπηρεσία που θα είναι υπεύθυνη για την εγγραφή χρηστών, τη σύνδεση χρηστών και τη διαχείριση του token του χρήστη.

User model

Στο αρχείο «user.model.ts» αποθηκεύονται όλα τα βασικά δεδομένα που συνθέτουν έναν χρήστη (email, id, token, tokenExpirationDate) και μας βοηθά να επιβεβαιώσουμε εάν token εξακολουθεί να ισχύει επειδή, το token θα λήξει μετά από ορισμένο χρονικό διάστημα , επομένως ελέγχουμε όχι μόνο αν το token υπάρχει αλλά και το εάν συνεχίζει να βρίσκεται σε ισχύ ή έχει λήξει.

Auth interceptor

Το αρχείο «auth-interceptor.service.ts» χρησιμοποιείται για την αποθήκευση και τη λήψη των δεδομένων των συνταγών, διότι σε αυτές τις διαδικασίες όπου στέλνονται τα http requests θα πρέπει να ενημερώνεται το firebase με το αντίστοιχο token.

Auth guard

Το αρχείο «auth.guard.ts» χρησιμοποιείται για να αποτρέπει τον χρήστη από την είσοδό του σε άλλα route paths (πχ. http://localhost:4200/recipes) χωρίς να έχει κάνει πρώτα είσοδο με τα απαραίτητα στοιχεία (email, κωδικός). Επομένως, με αυτό τον τρόπο αποτρέπεται η μη εξουσιοδοτημένη πρόσβαση σε συγκεκριμένο τμήμα της εφαρμογής μέσω δρομολόγησης με τη χρήση της μεθόδου CanActivate. Η μέθοδος αυτή χρησιμοποιείται για τη διακοπή της πρόσβασης σε μια διαδρομή.

Auth module

Από αυτό το module φορτώνουμε και διαχειριζόμαστε τα components που αφορούν το authentication.

Auth actions

Περιέχει όλα τα actions που συμβαίνουν για τη διαχείριση του authentication (login, logout, authentication fail/success).

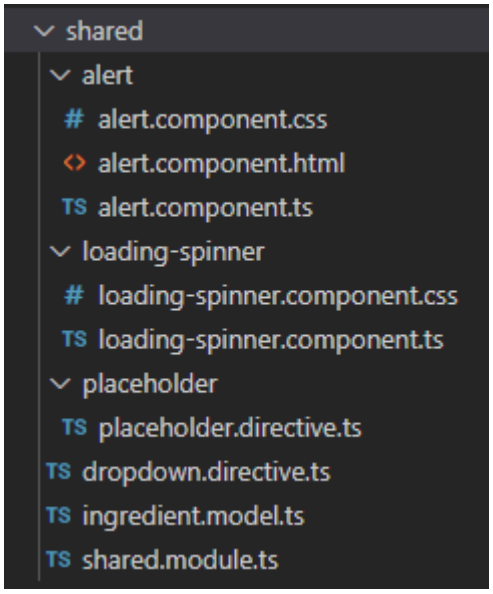
Auth effects

Στο «auth.effects.ts» γίνεται η διαχείριση των δεδομένων σε σχέση με το backend (firebase). Σε αυτό το αρχείο γίνεται το ερώτημα στη βάση δεδομένων για το εάν ο χρήστης είναι ήδη εγγεγραμμένος ή όχι, επίσης γίνεται η καταγραφή ενός νέου χρήστη στη βάση.

Auth reducer

Είναι υπεύθυνο για την διαχείριση του state, παίρνει το τρέχων state και το action ως παράμετρο και στη συνέχεια επιστρέφει ένα νέο state.

Shared



Περιέχει συνήθως κοινά χρησιμοποιούμενα directives, pipes και components που επαναχρησιμοποιούνται και αναφέρονται από άλλα components των feature modules [23].

Alert

Το component αυτό χρησιμοποιείται για την προβολή μηνυμάτων ειδοποίησης στον χρήστη σε περίπτωση λάθους κατά την διαδικασία εγγραφής και σύνδεσης. Αυτό το component είναι ένα δυναμικό component. Τα Dynamic Components μας επιτρέπουν να δημιουργήσουμε και να αποδώσουμε components κατά το χρόνο εκτέλεσης. Αυτό επιτυγχάνεται με την αναφορά σε ένα container που χρησιμοποιεί μια μεταβλητή προτύπου (template variable) και εισάγει components που δημιουργήθηκαν πρόσφατα σε αυτό ως αποτέλεσμα κάποιας αλληλεπίδρασης με τον χρήστη [24].

Placeholder

Το αρχείο «placeholder.directive.ts» χρησιμοποιείται βοηθητικά για να έχουμε πρόσβαση στο αρχείο «auth.component.html» προκειμένου να εισάγουμε το μήνυμα ειδοποίησης του alert component δυναμικά. Αυτό επιτυγχάνεται με τη χρήση του ng-template στο «auth.component.html» για να εναλλάσεται ο κώδικας προγραμματιστικά.

Loading spinner

Το component αυτό χρησιμοποιείται κατά τη φόρτωση της σελίδας, προβάλλοντας ένα loading spinner.

Dropdown directive

Το αρχείο «dropdown.directive.ts» χρησιμοποιείται για την λειτουργία των κουμπιών που είναι πτυσσόμενα (dropdown) όπως η «Διαχείριση» για αποθήκευση και λήψη δεδομένων και το κουμπί «Διαχείριση Συνταγής».

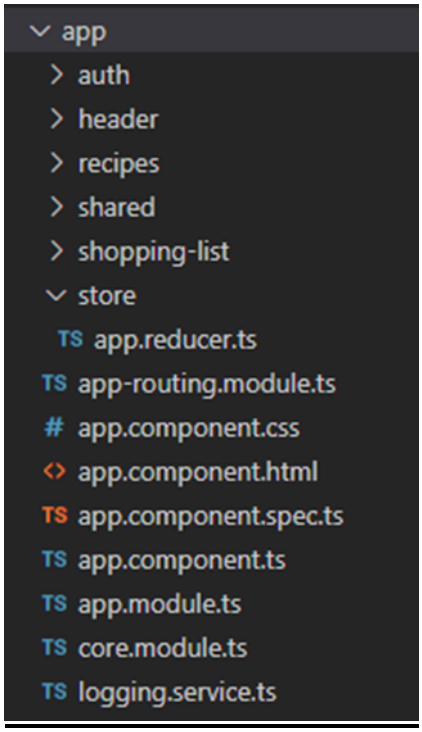
Ingredient model

Στο αρχείο αυτό βρίσκονται τα χαρακτηριστικά που διαμορφώνουν ένα υλικό της συνταγής (όνομα συστατικού, ποσότητα).

Shared module

Αποφυγη χρησης ιδιου κωδικα δυο φορες. Τα features που είναι κοινα τα μαζευουμε εκει.

App



Στο app βρίσκεται το αρχικό component της εφαρμογής.

App module

Το App Module μπορεί επίσης να ονομάζεται root module. Κάθε εφαρμογή πρέπει να έχει ένα κεντρικό App Module, από εκεί εκκινούμε τις εφαρμογές μας [25].

App-routing module

Το αρχείο «app-routing.module.ts» είναι υπεύθυνο για τη διαχείριση των routing paths της εφαρμογής.

Core module

Το Core Module είναι το μέρος όπου θέλουμε να τοποθετήσουμε τις κοινές μας υπηρεσίες singleton (shared singleton services). Έτσι, οι υπηρεσίες που θέλουμε μόνο μία παρουσία, ενώ τις έχουμε κοινές μεταξύ πολλών modules, θα πρέπει να υπάρχουν εδώ. Στη συγκεκριμένη εργασία χρησιμοποιούμε το core module για το AuthInterceptorService.

Αυτό το module φορτώνεται με την εφαρμογή και περιέχει στοιχεία που θα δημιουργηθούν μόνο μία φορά σε όλη την εφαρμογή ή θα χρησιμοποιηθούν μόνο σε ένα μέρος. Επίσης, εισάγεται μόνο μία φορά από το AppModule, καμία άλλη λειτουργική μονάδα δεν εισάγει αυτό το module [23].

Logging service

Η υπηρεσία «logging.service.ts» είναι υπεύθυνη για την εκτύπωση των logs μηνυμάτων.

App reducer

Είναι υπεύθυνο για την διαχείριση του state, παίρνει το τρέχων state και το action ως παράμετρο και στη συνέχεια επιστρέφει ένα νέο state.

4.4 Βελτιστοποιήσεις και συμπεράσματα

Η παρούσα εργασία έχει υλοποιηθεί με τρεις διαφορετικούς τρόπους:

i. Με μόνο ένα κεντρικό module, το app module το οποίο είναι υπεύθυνο για όλη την εφαρμογή και χωρίς άλλες βελτιστοποιήσεις.
ii. Με βελτιστοποίηση lazy loading και δημιουργία άλλων επιμέρους modules τα οποία φορτώνουν ξεχωριστά ανάλογα με την σελίδα που επισκεπτόμαστε.
iii. Με όλες τις παραπάνω βελτιστοποιήσεις και την χρήση του NgRx.

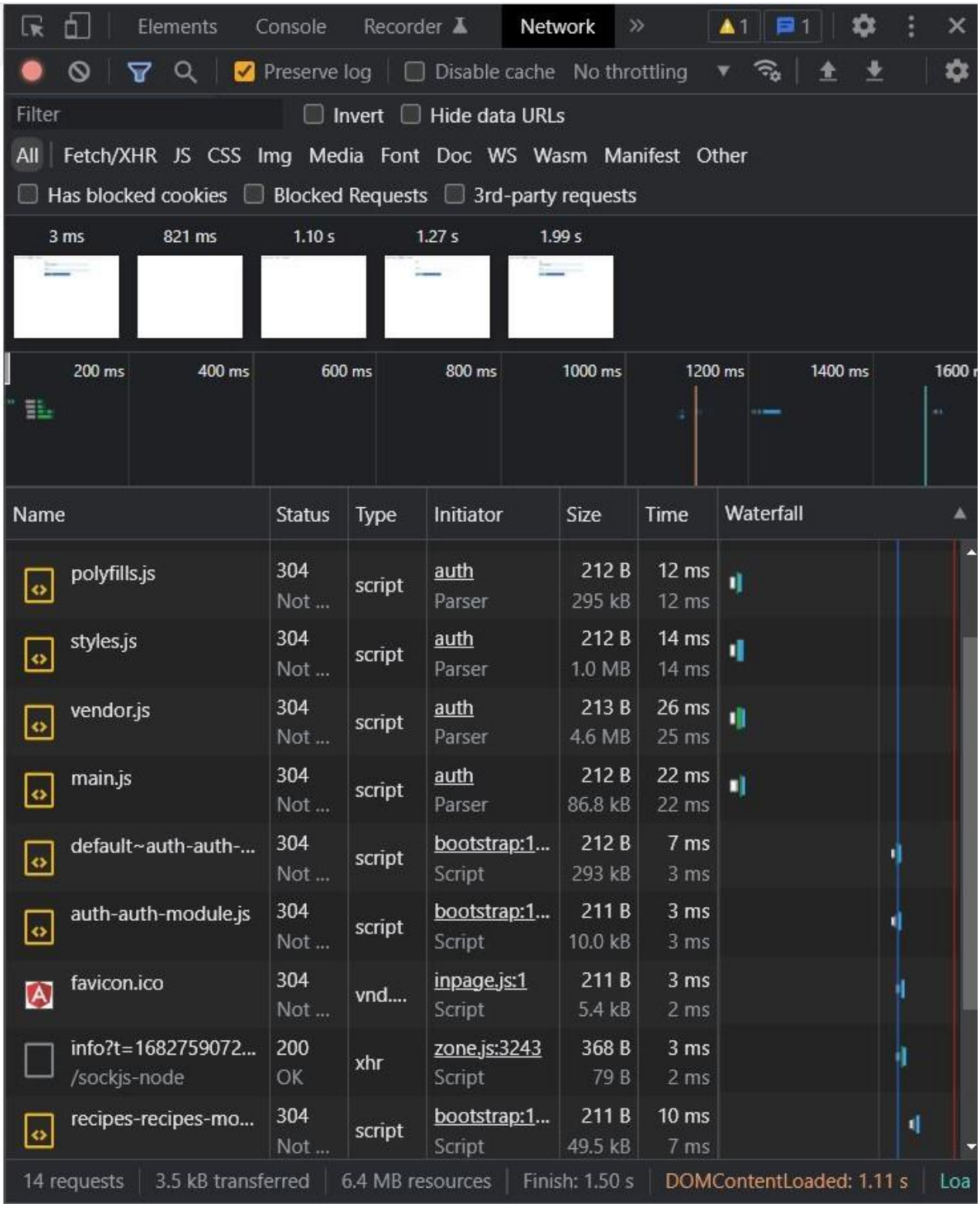
Τι είναι το Lazy Loading

Όταν δεν χρησιμοποιούμε lazy loading, κάθε φορά που επισκεπτόμαστε οποιαδήποτε σελίδα, φορτώνουμε τα πάντα, δηλαδή όλο τον κώδικα της εφαρμογής. Επομένως, ένας τρόπος να βελτιώσουμε την απόδοση της εφαρμογής είναι το lazy loading, δηλαδή να

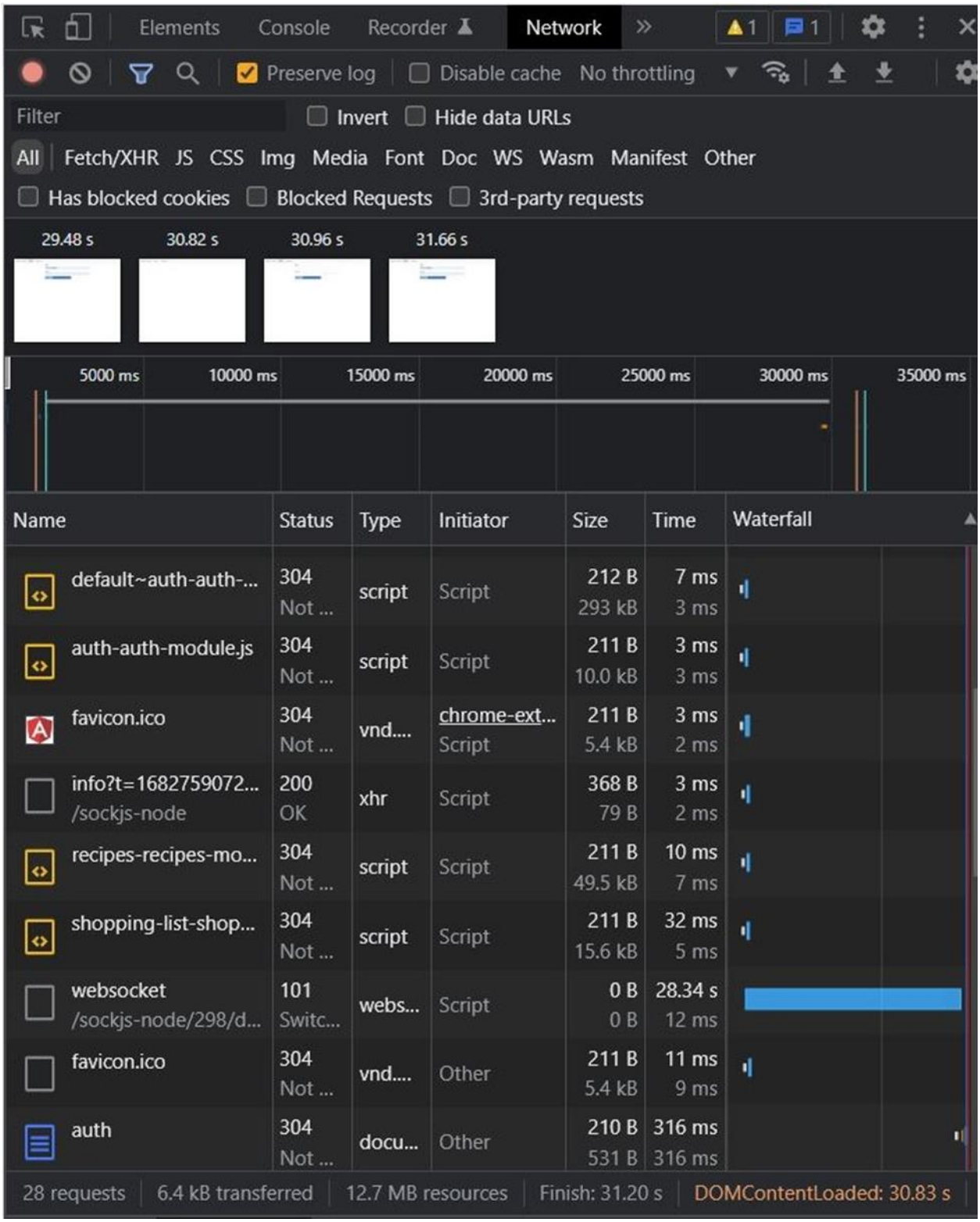
φορτώνουμε κάθε φορά μόνο τον κώδικα της σελίδας που χρειαζόμαστε. Πιο συγκεκριμένα, με το lazy loading, αρχικά φορτώνουμε μόνο το περιεχόμενο του root route, με άλλα λόγια μόνο τον κώδικα του app module και όχι τα άλλα modules, παρα μόνο όταν επισκεπτόμαστε την αντίστοιχη σελίδα(πχ. recipes).

Στην εφαρμογή μας έχουν δημιουργηθεί modules τα οποία φορτώνουν ξεχωριστά ανάλογα με την σελίδα που επισκεπτόμαστε. Όπως είδαμε και παραπάνω στην ανάλυση της δομής των αρχείων της εφαρμογής υπάρχουν ξεχωριστά modules για τη σελίδα του authentication, τα recipes, το shared και το shopping list.

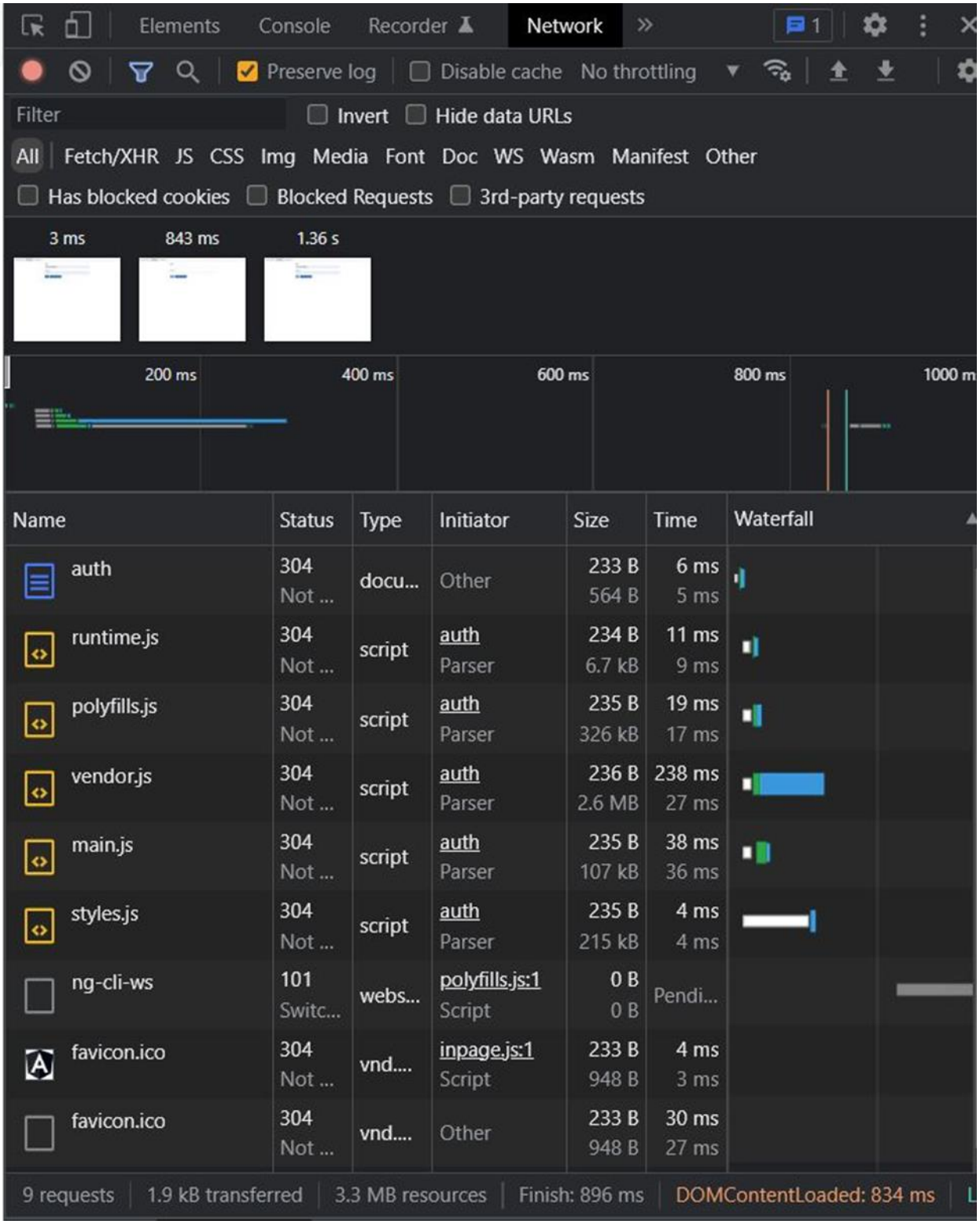
Προκείμένου να διαπιστώσω την επίδραδη του lazy loading στην δική μου εργασία έκανα ένα πείραμα μετρώντας την απόδοση της εφαρμογής με δύο διαφορετικούς τρόπους υλοποίησης: α) με lazy loading και β) χωρίς lazy loading. Στις παρακάτω εικόνες μπορούμε να δούμε την επίδρασή του lazy loading στην εφαρμογή για μια δεδομένη χρονική στιγμή. Η δύο πρώτες εικόνες παρουσιάζουν την απόδοση της εφαρμογής με τη χρήση του lazy loading, ενώ οι δύο τελευταίες χωρίς το lazy loading.



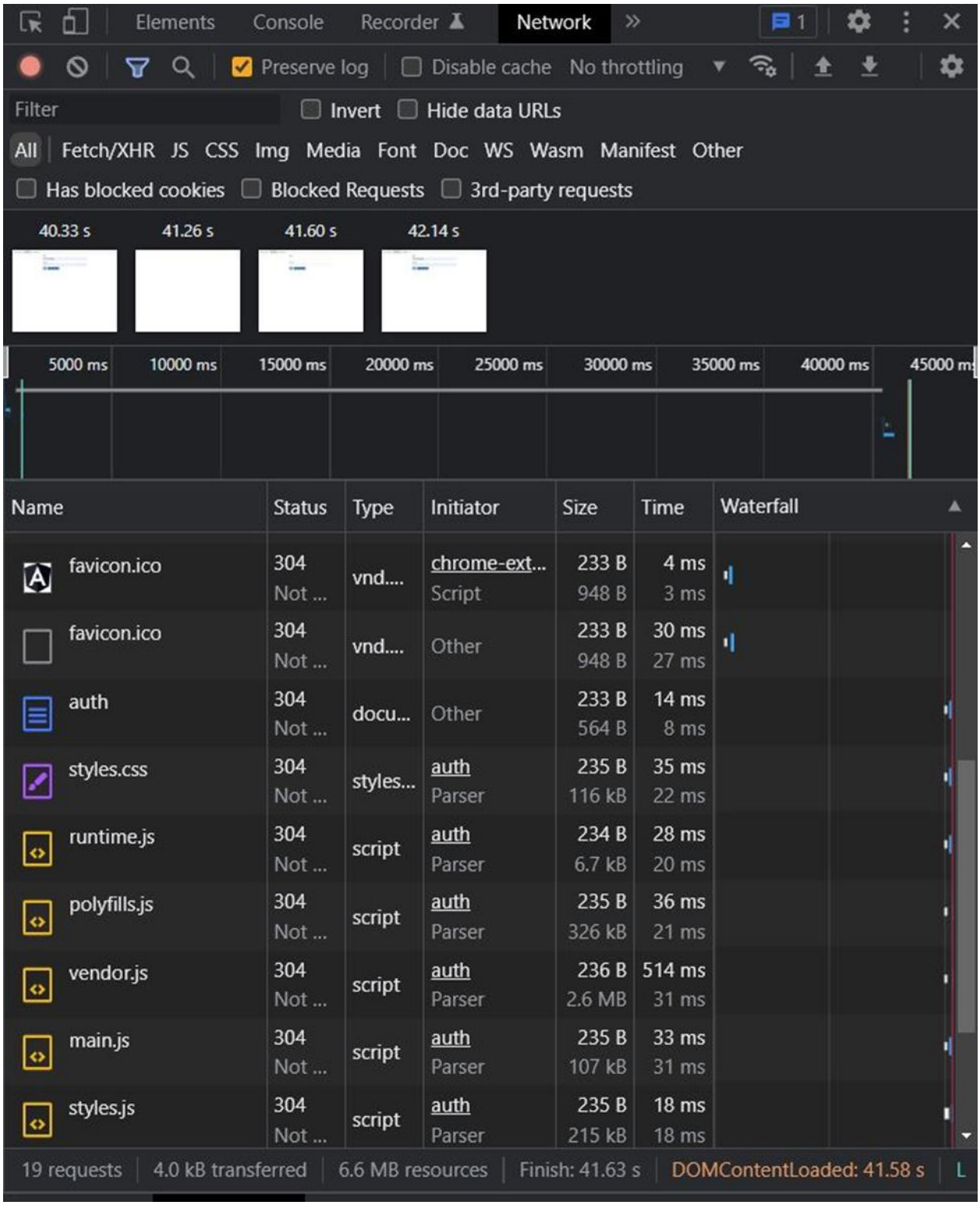
Εικόνα 9: Με lazy loading ο χρόνος φόρτωσης του DOM είναι 1.11s



Εικόνα 10: Με lazy loading ο χρόνος φόρτωσης του DOM είναι 30.83s



Εικόνα 11: Χωρίς lazy loading ο χρόνος φόρτωσης του DOM είναι 834ms



Εικόνα 12: Χωρίς lazy loading ο χρόνος φόρτωσης του DOM είναι 41.58s

Συμπεράσματα από τις μετρήσεις των δύο διαφορετικών υλοποιήσεων(με ή χωρίς lazy loading):

Όπως μπορούμε να παρατηρήσουμε από τα παραπάνω παραδείγματα η εφαρμογή είναι πιο γρήγορη και αποδοτική με τη χρήση του lazy loading όσο φορτώνονται περισσότερα δεδομένα. Ωστόσο, η διαφορά δεν είναι αρκετά σημαντική στην δική μας περίπτωση.

Συνεπώς, οδηγούμαστε στο συμπέρασμα ότι για εφαρμογές μικρού μεγέθους, το lazy loading δεν μας προσφέρει σημαντική διαφορά. Αλλά εάν η εφαρμογή είναι αρκετά μεγάλη, δηλαδή με μεγάλο όγκο δεδμένων και κώδικα τότε η βελτιστοποίηση που μας προσφέρει το lazy loading είναι πολύ σημαντική.

Τελικά, αυτό που προτείνω είναι ότ για μεγάλες εφαρμογές με πολλά routes, μπορείτε να εξετάσετε το ενδεχόμενο της χρήσης του lazy loading. Το lazy loading θα βοηθήσει να διατηρήσετε το αρχικό μέγεθος του πακέτου (bundle) μικρότερο, που σημαίνει ότι μόνο η βασική μονάδα θα φορτωθεί στην αρχή όταν φορτωθεί η εφαρμογή και θα μειώσει τον αρχικό χρόνο φόρτωσης της εφαρμογής σας [23].

Συμπεράσματα από τη χρήση του NgRx

Η χρήση του NgRx για την υλοποίηση της εφαρμογής οδηγεί σε κάποια σημαντικά συμπεράσματα σχετικά με τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης του NgRx.

Συμπερασματικά τα πιο σημαντικά οφέλη είναι τα εξής:

- Η ροή δεδομένων στην εφαρμογή είναι σαφής.
- Τα δεδομένα είναι σε ένα οργανωμένο σχήμα, εύκολα προσβάσιμα μέσω ολόκληρης της εφαρμογής.
- Τα components είναι απλούστερα λόγω της ανάθεσης λογικής στους reducers και στα effects.
- Έχουμε την δυνατότητα να χωρίσουμε τα actions και τα effects σε πολλά αρχεία και modules.
- Το αμετάβλητο(Immutability) του state σε επίπεδο component εξαλείφει τα φορτία πολλά σφάλματα(bugs).

- Χάρη στις προηγμένες επεκτάσεις εντοπισμού σφαλμάτων για το NgRx, χρησιμοποιώντας @ngrx/store-dev-tools, ο εντοπισμός σφαλμάτων είναι εύκολος.
- Υποστήριξη για lazy loading μέσω των StoreModule.forFeature() και EffectsModule.forFeature() που συμβάλλει στη βελτίωση της απόδοσης της εφαρμογής καθώς κάθε module φορτώνει μόνο τα μέρη του store που απαιτούνται.

Από την άλλη πλευρά έχουμε τα παρακάτω μειονεκτήματα:

- Το NgRx είναι αρκετά δύσκολο στην εκμάθηση, απαιτεί χρόνο μέχρι να κατανοηθούν όλες οι ένοιες. Στην αρχή, πολλές έννοιες είναι δυσνόητες. Αλλά μόλις κατανοήσει κάποιος τις κύριες αρχές, πιθανότατα θα μπορέσει να προχωρήσει γρήγορα και να κατανοήσει ακόμη και στις πιο προηγμένες λειτουργίες.
- Η καλή γνώση του RxJS είναι απαραίτητη για να προχωρήσει κάποιος στις υλοποιήσεις. Το RxJS είναι μια βιβλιοθήκη για τη σύνθεση ασύγχρονων προγραμμάτων και προγραμμάτων που βασίζονται σε συμβάντα χρησιμοποιώντας observable ακολουθίες. Παρέχει έναν τύπο πυρήνα(=core), το Observable, Observer, Schedulers, Subjects και τελεστές εμπνευσμένους από μεθόδους Array (map, filter, reduce, every, κ.λπ.) για να επιτρέπεται ο χειρισμός ασύγχρονων συμβάντων ως συλλογών [26].
- Μπορεί να είναι δύσκολο να βρεθούν σωστά ονόματα για τα actions και τα effects,
- Η χρήση NgRx παντού είναι υπερβολική. Εάν έχετε να κάνετε με μικρά και απλά δεδομένα, απλώς ανακτήστε τα απευθείας.
- Απαιτεί αρκετό χρόνο για να γίνει η υλοποίηση και να αλλάξει η αρχιτεκτονική και η δομή της εφαρμογής στην αρχή, αλλά στη συνέχεια είναι πολύ εύκολο να κάνεις επιπλέον επεκτάσεις και να διαχειρίζεσαι τον κώδικα [27].

<u>Σε ποιες περιπτώσεις είναι ωφέλιμο να χρήσιμοποιηθεί NgRx;</u>

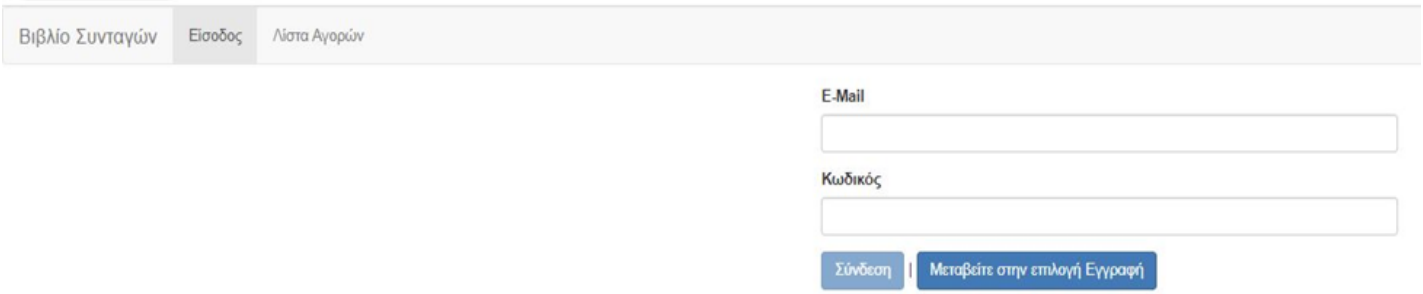
Από τα πειράματα που πραγματοποιήθηκαν στην εργασία παραπάνω συμπεραίνω ότι γενικά, είναι καλή πρακτική να χρησιμοποιούμε NgRx όταν έχουμε μία εφαρμογή μεγάλης κλίμακας με δεδομένα που φορτώνονται ασύγχρονα και χρησιμοποιούνται σε πολλά διαφορετικά μέρη. Επιπλέον, η εκμάθηση του NgRx μας διδάσκει μια νέα

νοοτροπία ανάπτυξης και καθιστά την εφαρμογή μας λιγότερο επιρρεπή σε αλλαγές απαιτήσεων. Τέλος, μας βοηθά να αναπτύξουμε εφαρμογές Angular με δυνατότητα συντήρησης, κλιμάκωσης και ευκολία στον εντοπισμό σφαλμάτων [28].

5 Παρουσίαση υλοποίησης της εφαρμογής

Στο κεφάλαιο 5 γίνεται η παρουσίαση του front-end της εφαρμογής μέσα από εικόνες καθώς επίσης και σύντομες επεξηγήσεις του τρόπου λειτουργίας της εφαρμογής. Στην παρούσα εργασία για τον σχεδιασμό του front-end χρησιμοποιήθηκε το Bootstrap. Το Bootstrap είναι ένα δωρεάν front-end framework για ταχύτερο και ευκολότερο web development. Πιο συγκεκριμένα, το Bootstrap περιλαμβάνει πρότυπα σχεδίασης βασισμένα σε HTML και CSS για τυπογραφία, φόρμες, κουμπιά, πίνακες, πλοήγηση, modals, καρουζέλ εικόνων και πολλά άλλα, καθώς και προαιρετικά πρόσθετα JavaScript. Τέλος, το Bootstrap δίνει επίσης τη δυνατότητα στον προγραμματιστή να δημιουργεί με ευκολία σχέδια με responsive designs.

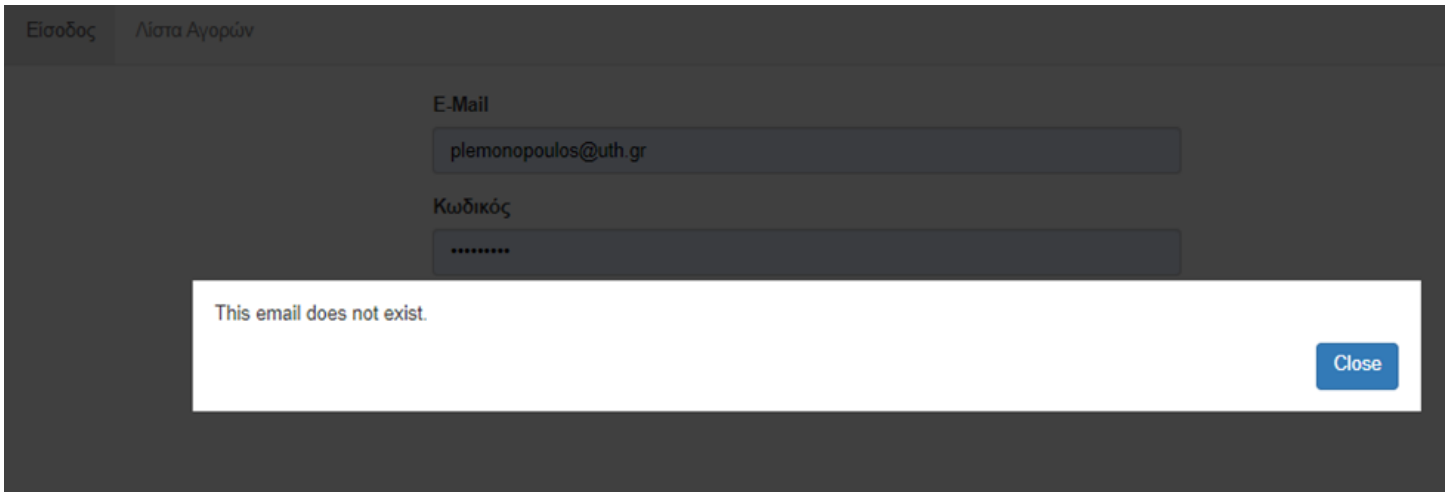
Είσοδος



Route path: http://localhost:4200/auth

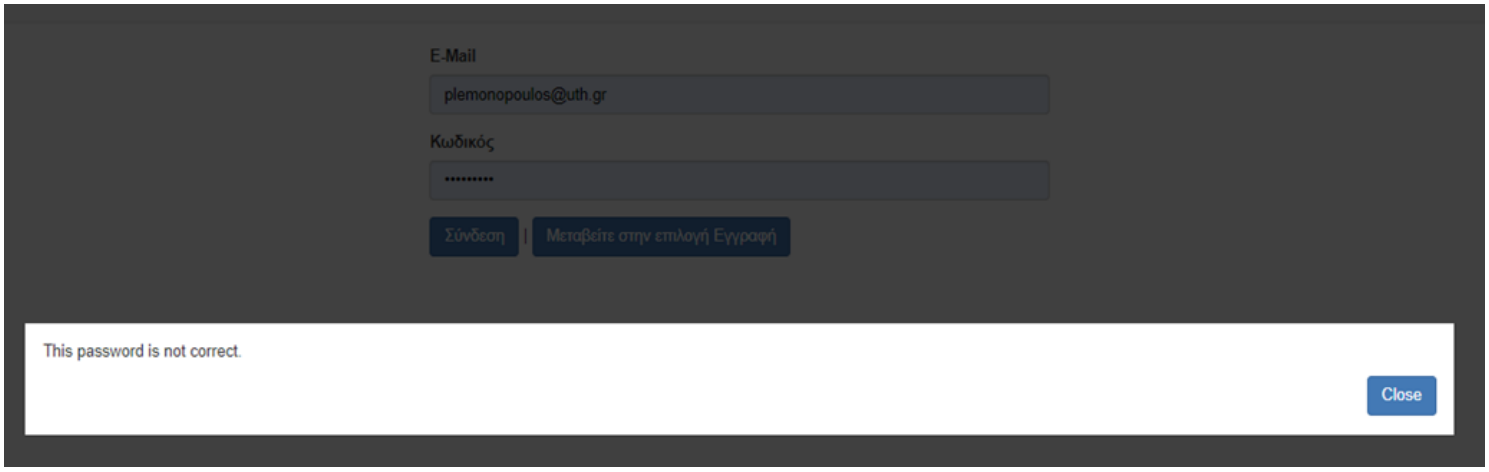
Στην παραπάνω εικόνα παρουσιάζεται η σελίδα όπου ο χρήστης μπορεί να κάνει είσοδο στην εφαρμογή χρησιμοποιώντας «E-mail» και «Κωδικό». Ο χρήστης έχει την επιλογή να κάνει είτε «Εγγραφή» νέου χρήστη είτε «Σύνδεση» με τα στοιχεία του ήδη εγγεγραμμένου χρήστη.

Σύνδεση με χρήστη που δεν υπάρχει



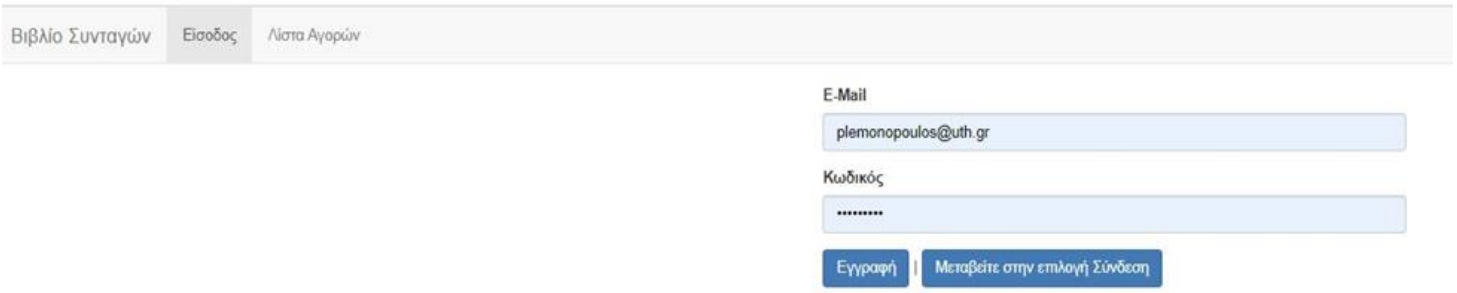
Έλεγχος σε περίπτωση που ο χρήστης δεν υπάρχει, δηλαδή στην περίπτωση που γίνεται είσοδος με στοιχεία ενός χρήστη που δεν είναι ήδη εγγεγραμμένος. Στην περίπτωση αυτή η είσοδος απορρίπτεται και η εφαρμογή προβάλλει ένα μήνυμα ειδοποίησης στον χρήστη.

Σύνδεση με λάθος κωδικό

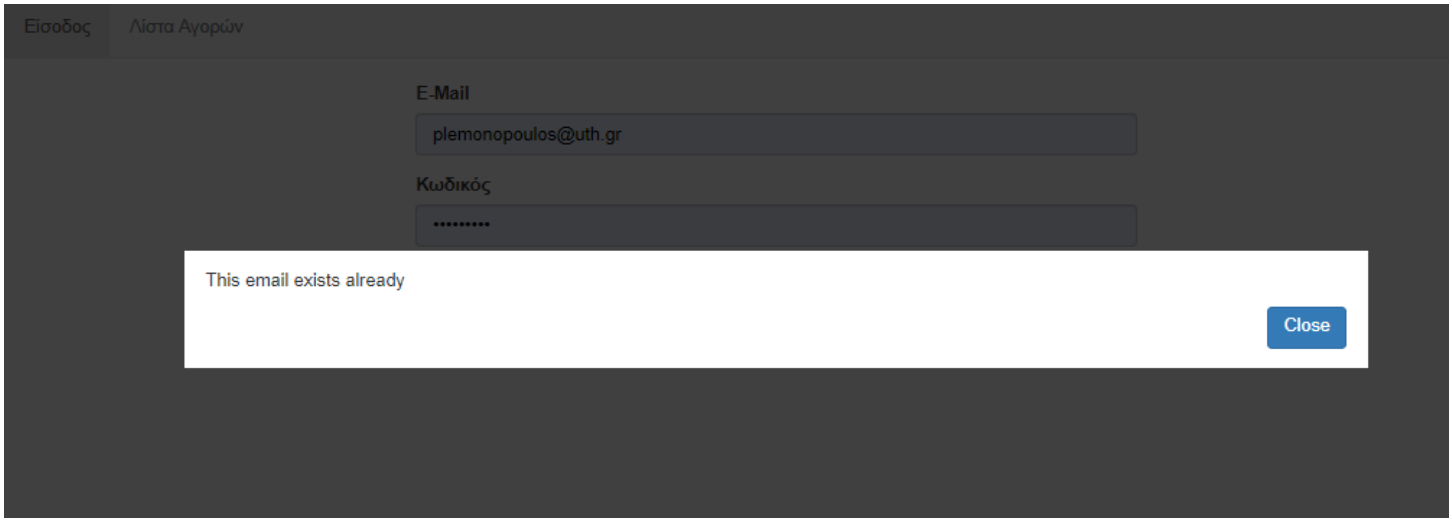


Στην περίπτωση που ο κωδικός είναι λάθος κατά τη σύνδεση του χρήστη, προβάλλεται το αντίστοιχο μήνυμα λάθους.

Στοιχεία εγγραφής



Εγγραφή με ήδη εγγεγραμμένο χρήστη



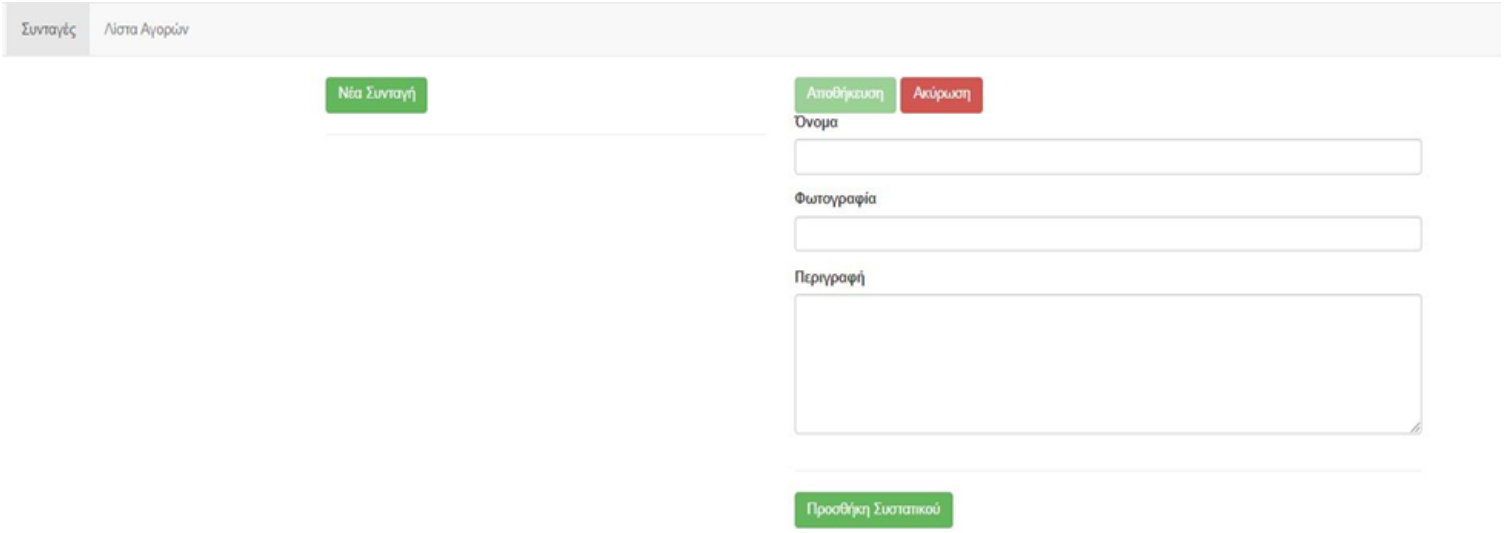
Στην περίπτωση που γίνεται εγγραφή ενός ήδη εγγεγραμμένου χρήστη, προβάλλεται το αντίστοιχο μήνυμα ειδοποίησης στον χρήστη.

Αρχική Οθόνη

Route path: http://localhost:4200/recipes

Είσοδος στην εφαρμογή και αρχική οθόνη.

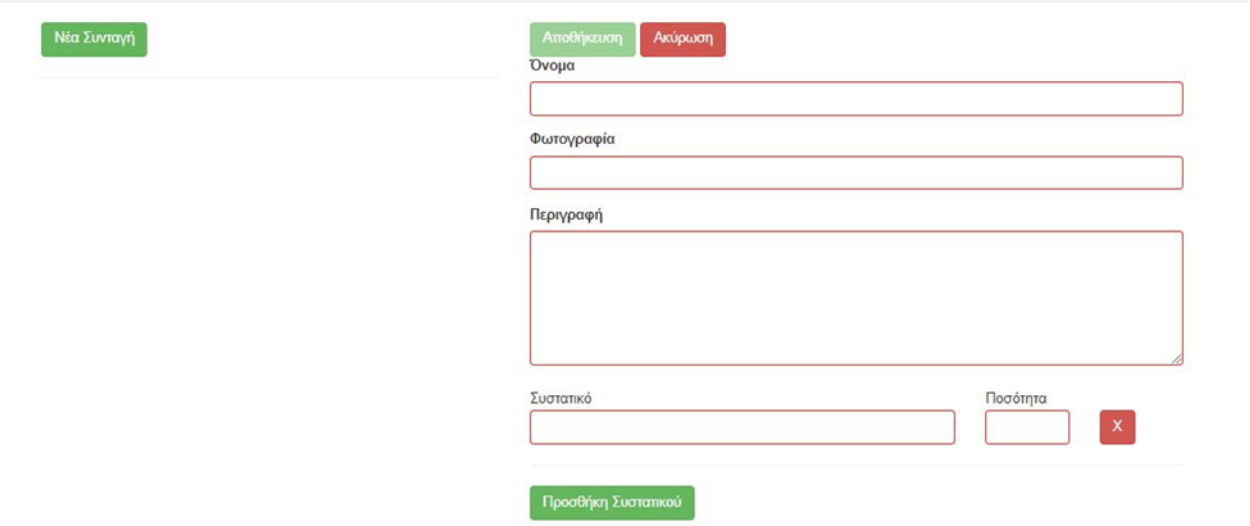
Προσθήκη Νέας Συνταγής



Route path: http://localhost:4200/recipes/new

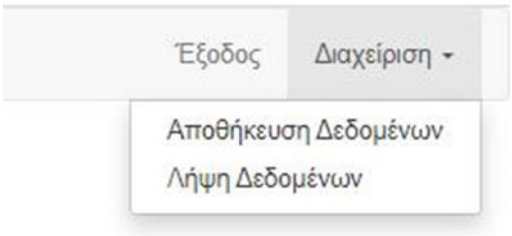
Επιλογή «Νέα Συνταγή» όπου ο χρήστης μπορεί να δημιουργήσει μία νέα συνταγή συμπληρώνοντας τα πεδία της φόρμας (Όνομα, Φωτογραφία(url φωτογραφίας), Περιγραφή, Προσθήκη Συστατικού).

Φόρμα προσθήκης συνταγής



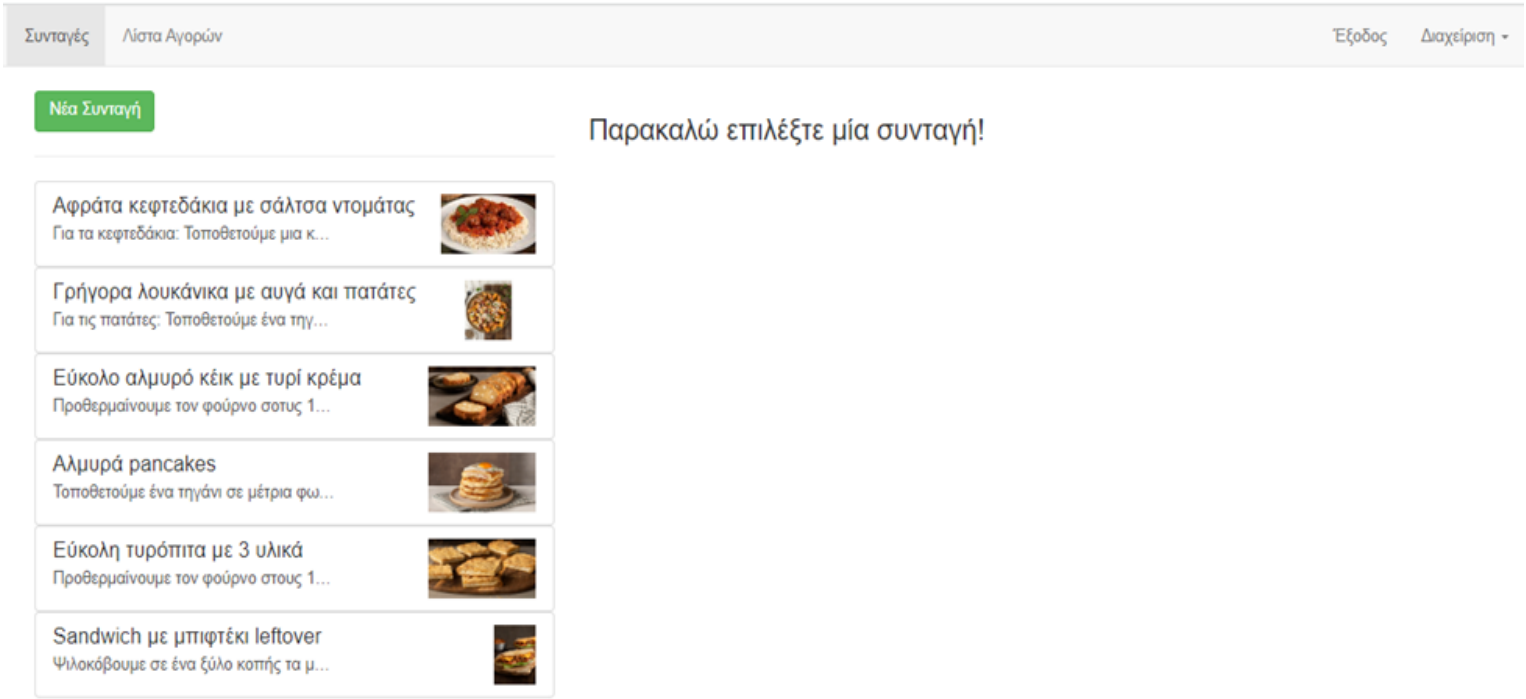
Στην παραπάνω εικόνα βλέπουμε τα πεδία της φόρμας για τη δημιουργία μίας νέας συνταγής με ένα κόκκινο περίγραμμα. Η φόρμα είναι «reactive». Πιο συγκεκριμένα, όπως μπορούμε να παρατηρήσουμε στην φόρμα υπάρχουν έλεγχοι για τη συμπλήρωση των πεδίων. Στην περίπτωση που ο χρήστης δεν συμπληρώσει τα πεδία το κουμπί «Αποθήκευση» απενεργοποιείται και είναι αδύνατον να κάνει αποθήκευση νέας συνταγής ο χρήστης. Τέλος, ο χρήστης έχει την δυνατότητα να βλέπει σε πραγματικό χρόνοα τα πεδία που απαιτούν συμπλήρωση.

Διαχείριση Δεδομένων



Στην παραπάνω εικόνα βλέπουμε την επιλογή «Διαχείριση» που βρίσκεται δεξιά στον header navigation. Αναλυτικότερα, ο χρήστης έχει την δυνατότητα να επιλέξει «Αποθήκευση Δεδομένων» για να αποθηκεύσει τα δεδομένα των συνταγών που έχει στην εφαρμογή ή «Λήψη Δεδομένων» για «φέρει» τα δεδομένα, που έχει ήδη αποθηκεύσει, από την βάση δεδομένων μέσω firebase.

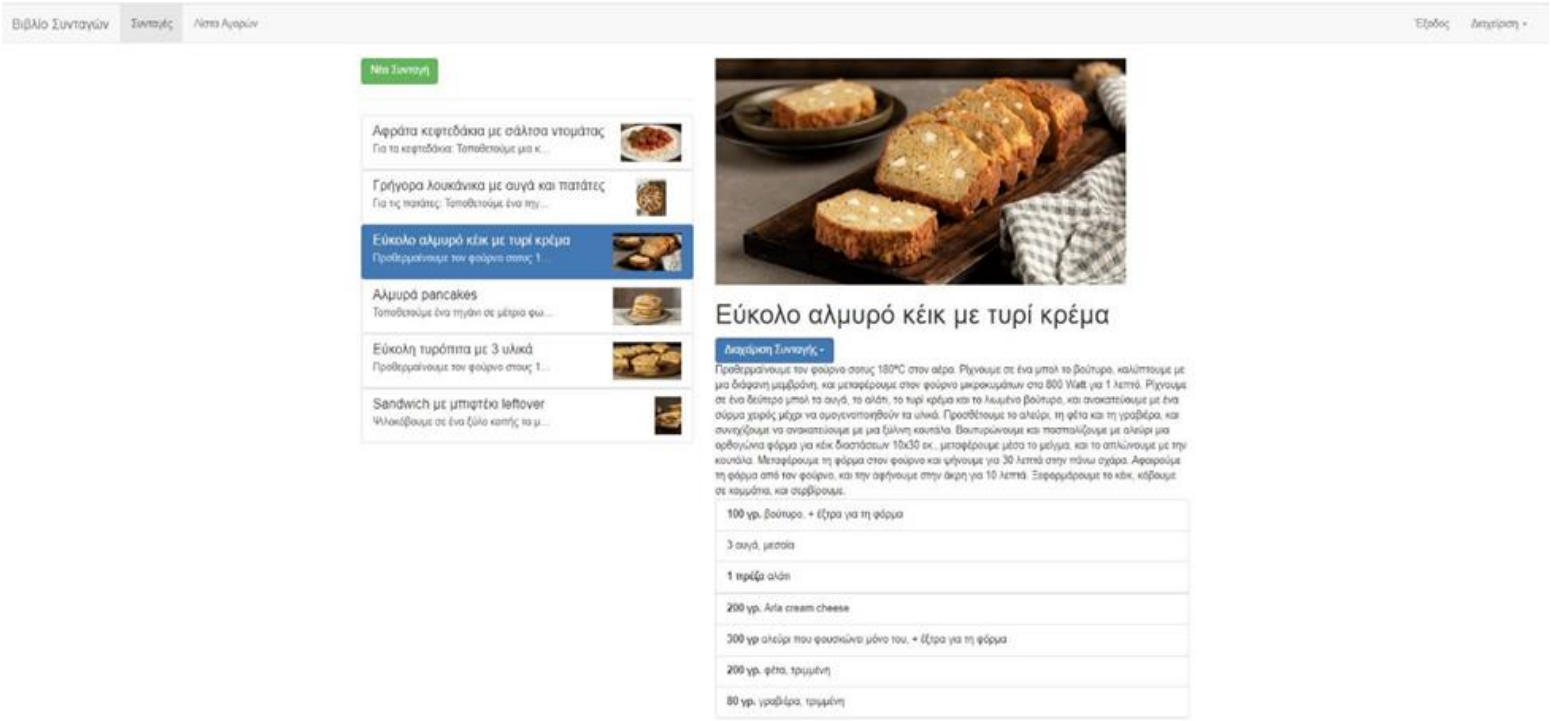
Λίστα Συνταγών



Route path: http://localhost:4200/recipes

Αφού γίνει η «Λήψη Δεδομένων» ο χρήστης βλέπει στην αρχική οθόνη την λίστα με τις συνταγές του που βρίσκονται αποθηκευμένες.

Λήψη Συνταγών και προβολή λεπτομερειών μια συνταγής



Route path: http://localhost:4200/recipes/2

Ο χρήστης μπορεί να επιλέξει κάποια από τις συνταγές και να δει τις λεπτομέρειες. Στις λεπτομέρειες της συνταγής φαίνονται η φωτογραφία, η περιγραφή, τα υλικά και οι ποσότητες.

Διαχείριση Συνταγής

Εύκολο αλμυρό κέικ με τυρί κρέμα

Διαχείριση Συνταγής

- Προσθήκη στη Λίστα Αγορών
- Επεξεργασία συνταγής
- Διαγραφή συνταγής

180°C στον αέρα. Ρίχνουμε σε ένα μπολ το βούτυρο, καλύπτουμε με
υμε στον φούρνο μικροκυμάτων στα 800 Watt για 1 λεπτό. Ρίχνουμε
άτι, το τυρί κρέμα και το λιωμένο βούτυρο, και ανακατεύουμε με ένα
θούν τα υλικά. Προσθέτουμε το αλεύρι, τη φέτα και τη γραβιέρα, και
ξύλινη κουτάλα. Βουτυρώνουμε και πασπαλίζουμε με αλεύρι μια
ορθογώνια φόρμα για κέικ διαστάσεων 10x30 εκ., μεταφέρουμε μέσα το μείγμα, και το απλώνουμε με την κουτάλα. Μεταφέρουμε τη φόρμα στον φούρνο και ψήνουμε για 30 λεπτά στην πάνω σχάρα. Αφαιρούμε τη φόρμα από τον φούρνο, και την αφήνουμε στην άκρη για 10 λεπτά. Ξεφορμάρουμε το κέικ, κόβουμε σε κομμάτια, και σερβίρουμε.

100 γρ. βούτυρο, + έξτρα για τη φόρμα
3 αυγά, μεσαία
1 πρέζα αλάτι
200 γρ. Arla cream cheese
300 γρ αλεύρι που φουσκώνει μόνο του, + έξτρα για τη φόρμα
200 γρ. φέτα, τριμμένη
80 γρ. γραβιέρα, τριμμένη

Ο χρήστης έχει την δυνατότητα να διαχειριστεί την συνταγή. Πιο αναλυτικά έχει τις εξής επιλογές:

i. «Προσθήκη στη Λίστα Αγορών»

Προσθέτει αυτόματα όλα τα υλικά της συνταγής μαζί με τις απαιτούμενες ποσότητες στη λίστα αγορών όπως φαίνεται και παρακάτω (εικόνα).

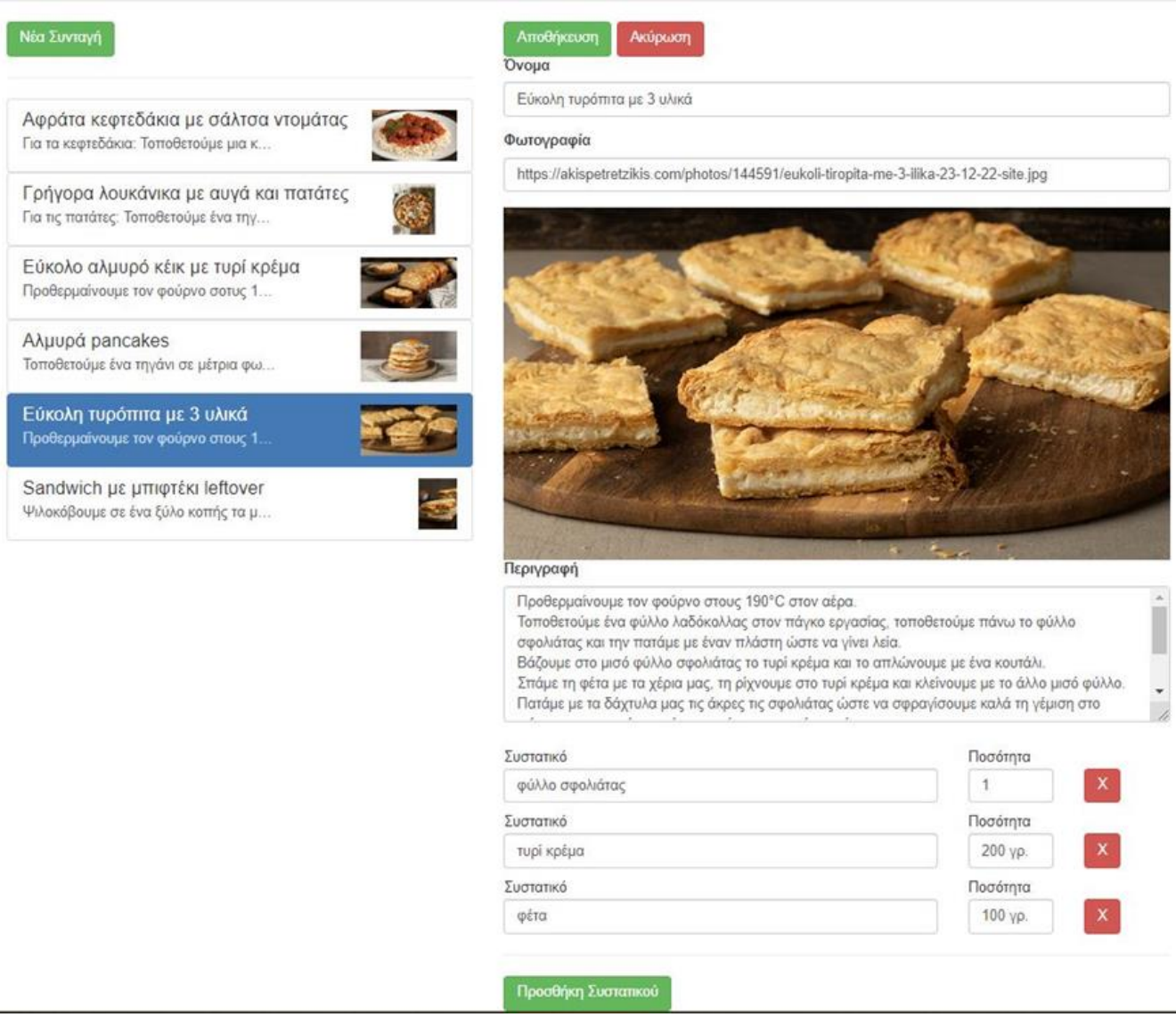
ii. «Επεξεργασία συνταγής»

Δίνει την δυνατότητα για αλλαγή των στοιχείων της συνταγής.

iii. «Διαγραφή συνταγής»

Δίνει την δυνατότητα στον χρήστη να διαγράψει την συνταγή.

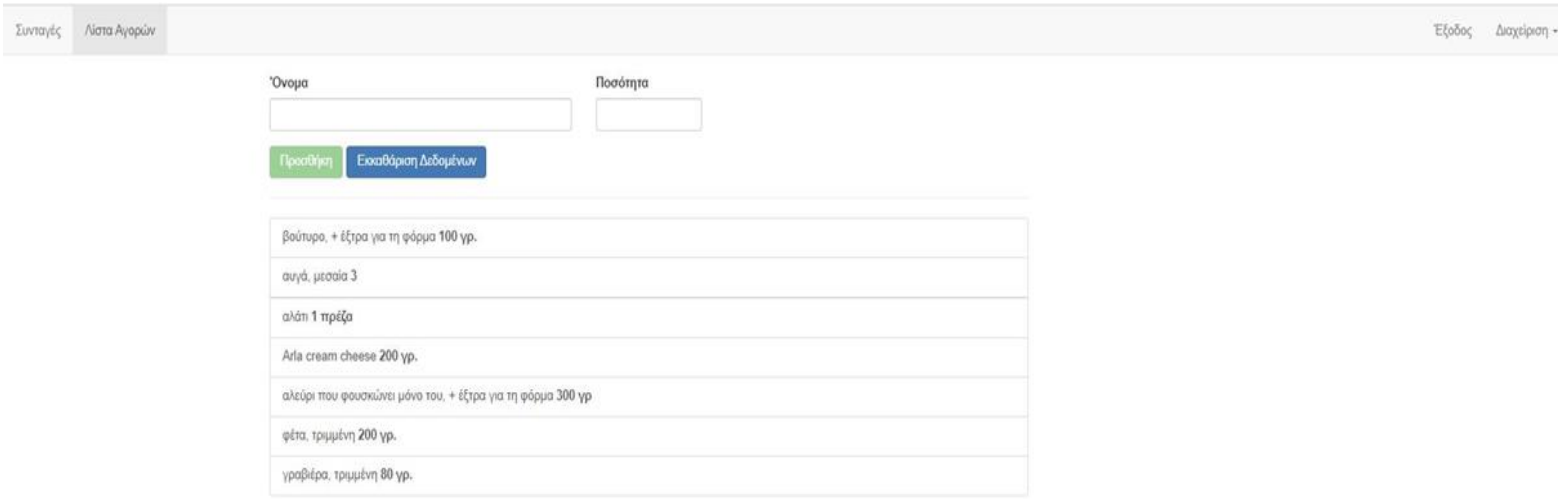
Επεξεργασία Συνταγής



Route path: http://localhost:4200/recipes/4/edit

Στο παραπάνω παράδειγμα φαίνεται πως γίνεται η επεξεργασία και η αλλαγή των στοιχείων μιας συνταγής, μέσα από την «reactive» φόρμα που χρησιμοποιούμε.

Λίστα αγορών

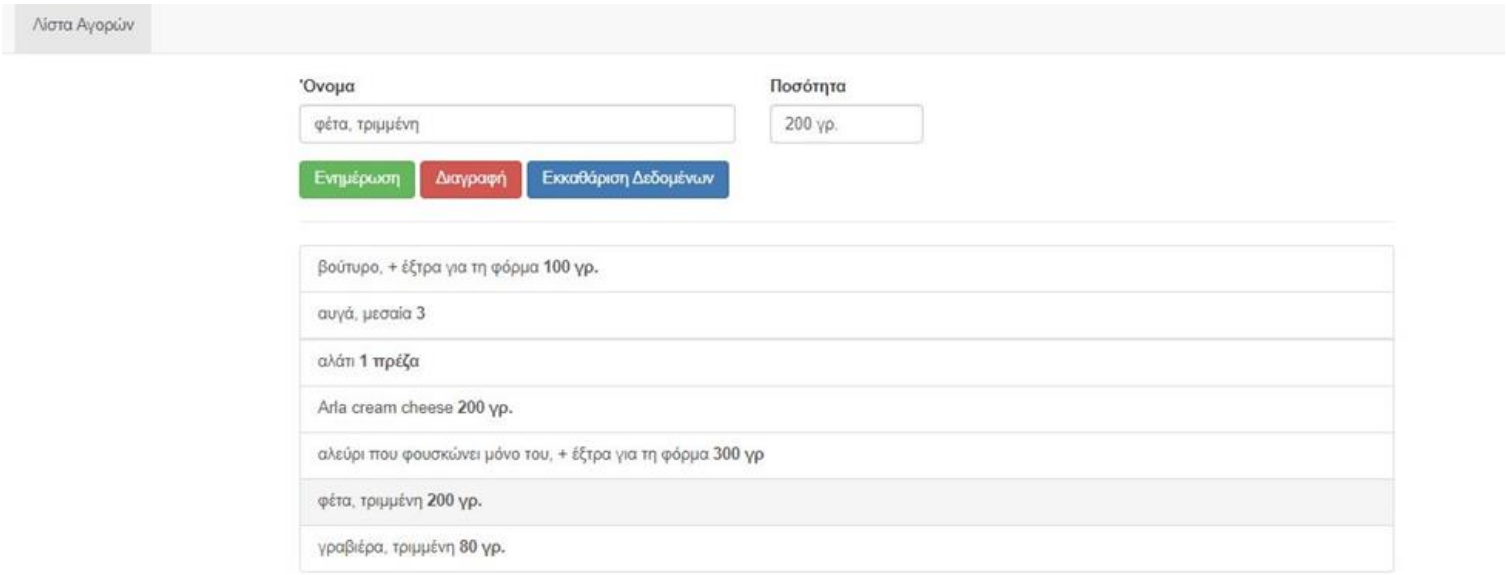


Route path: http://localhost:4200/shopping-list

Στη «Λίστα Αγορών» υπάρχουν τα υλικά που περνάμε από τις συνταγές. Ο χρήστης επιλέγει από την διαχείριση της συνταγής την προσθήκη στη λίστα αγορών. Στη συνέχεια όλα τα υλικά της συνταγής μαζί με τις ποσότητες έρχονται στη λίστα αγορών όπως φαίνεται και στην παραπάνω εικόνα.

Επιπροσθέτως, ο χρήστης έχει την δυνατότητα να προσθέσει μόνος του κάποια υλικά που δεν βρίσκονται στις συνταγές, επιλέγοντας «Προσθήκη». Ενώ με την εκκαθάριση των δεδομένων διαγράφονται τα δεδομένα που βρίσκονται προς επεξεργασία.

Επεξεργασία Λίστας αγορών



Τέλος, όπως μπορούμε να παρατηρήσουμε στην παραπάνω εικόνα ο χρήστης έχει την δυνατότητα να επιλέγει κάποιο από τα ήδη αποθηκευμένα υλικά της λίστας και να τα επεξεργάζεται κάνοντας «Ενημέρωση», «Διαγραφή» ή «Εκκαθάριση Δεδομένων».

6 Συμπεράσματα

6.1 Συμπεράσματα

Φτάνοντας στο τέλος της μεταπτυχιακής εργασίας αυτής προκύπτουν κάποια συμπεράσματα. Ειδικότερα, η εργασία αυτή είχε ως στόχο να παρουσιάσει το framework της Angular καθώς και τις δυνατότητες που αυτό μπορεί να δώσει στον προγραμματιστή και στον χρήστη. Επίσης, στην εργασία αυτή έγινε παρουσίαση και σύγκριση της Angular με το framework της React που αποτελεί ένα από τα δημοφιλέστερα frontend framewokrs. Μέσα από αυτή τη σύγκριση συνειδητοποιήσαμε ότι κάθε framework έχει τα δικά του πλεονεκτήματα και μειονεκτήματα καθώς και ιδιαίτερα χαρακτηριστικά που το κάνουν μοναδικό και ξεχωριστό. Επομένως, ο μηχανικός που θα πρέπει να επιλέξει την κατάλληλη τεχνολογία ανάπτυξης της εφαρμογής οφείλει να λάβει υπόψιν αυτά τα χαρακτηριστικά προκειμένου να επιλέξει την καλύτερη τεχνολογία ανάλογα με το έργο που πρόκειται να αναπτύξει.

Επιπλέον, στην πλαίσια της εργασίας αναπτύχθηκε μία εφαρμογή για ηλεκτρονική καταγραφή συνταγών με τη χρήση της Angular. Πιο αναλυτικά, μέσω αυτής της εφαρμογής είχα την δυαντότητα να μελετήσω σε βάθος τις δυνατότητες της Angular και να χρησιμοποιήσω σχεδόν όλες τις τεχνικές που μπορεί να μας προσφέρει αυτό το framework. Η εφαρμογή δίνει την δυνατότητα στον χρήστη να εγγραφεί και να συνδεθεί μέσω του authentication που έχει υλοποιηθεί. Επίσης, ο χρήστης μπορεί να καταγράψει τις αγαπημένες του συνταγές αποθηκεύοντας τον τίτλο, την φωτογραφία, την περιγραφή και τα συστατικά της συνταγής. Η αποθήκευση γίνεται μέσω φόρμας η οποία είναι reactive form και ελέγχει κάθε φορά τα πεδία αν είναι συμπληρωμένα. Όλα αυτά αποθηκεύονται μέσω της βάσης δεδομένων που υπάρχει στο firebase και ο χρήστης μπορεί ανα πάσα ώρα να κάνει λήψη των δεδομένων του. Ακόμη, ο χρήστης μπορεί να δημιουργεί στην εφαρμογή μία λίστα αγορών συμπληρώνοντας το συστατικό και την ποσότητα, ενώ υπάρχει και η δυνατότητα να θέτει αυτόματα τα συστατικά των συνταγών στην λίστα για αγορά. Τέλος, για την ανάπτυξη αυτής της εφαρμογής ανέπτυξα μια εφαρμογή με τρεις διαφορετικές υλοποιήσεις. Στις υλοποιήσεις αυτές δοκιμάστηκαν διαφορετικές αρχιτεκτονικές ανάπτυξης της εφαρμογής και έγινε σύγκριση ως προς την αποδοτικότητα της κάθε μιας.

6.2 Μελλοντικές κατευθύνσεις

Εν κατακλείδι, αξιολογώντας τα παραπάνω δεδομένα γίνεται, λοιπόν, αντιληπτό ότι στην παρούσα μεταπτυχιακή εργασία έγινε μια προσπάθεια ερευνας και μελέτης του Angular framework βιβλιογραφικά, αλλά κυρίως μέσω της ανάπτυξης εφαρμογής και πειραμάτων που έγιναν σε αυτή την εφαρμογή. Ωστόσο, είναι αλήθεια ότι τα θέματα της έρευνας στο πεδίο της τεχνολογίας παραμένουν πάντοτε ανεξάντλητα. Συνεπώς, οι προτάσεις μου για τη μελλοντική ανάπτυξη αυτής της εργασίας είναι οι εξής δύο: α) μετατροπή της εφαρμογής σε Progressive Web App για να είναι πιο εύχρηστη στα κινητά τηλέφωνα, β) υλοποίηση της ίδιας εφαρμογής με άλλα frameworks όπως React και Vuejs και σύγκριση της αποδοτικότητας του κάθε ένα από αυτά.

7 Βιβλιογραφία

[1] «Sitepoint AngularJS vs Angular,» [Ηλεκτρονικό]. Available: https://www.sitepoint.com/angularjs-vs-angular/.

[2] «Angular wiki,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Angular_(web_framework).

[3] «What is SPA,» [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/what-is-spa-single-page-application-in-angularjs/.

[4] «History of Angular,» [Ηλεκτρονικό]. Available: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7.

[5] «Angular vs React,» [Ηλεκτρονικό]. Available: https://www.simform.com/blog/angular-vs-react/.

[6] «React vs Angular diff,» [Ηλεκτρονικό]. Available: https://www.guru99.com/react-vs-angular-key-difference.html.

[7] «angular vs react what to choose for your app,» [Ηλεκτρονικό]. Available: https://www.freecodecamp.org/news/angular-vs-react-what-to-choose-for-your-app-2/.

[8] «what is typescript,» [Ηλεκτρονικό]. Available: https://www.typescripttutorial.net/typescript-tutorial/what-is-typescript/.

[9] «what is angular - guide,» [Ηλεκτρονικό]. Available: https://angular.io/guide/what-is-angular.

[10] «architecture of components,» [Ηλεκτρονικό]. Available: https://angulardoc.com.br/guide/architecture-components.

[11] «components architecture from angular.io,» [Ηλεκτρονικό]. Available: https://angular.io/guide/architecture-components.

[12] «routing overview,» [Ηλεκτρονικό]. Available: https://angular.io/guide/routing-overview.

[13] «learning, webmozart - music and learning,» [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=dWOVbzqHAHg&list=PLXpeitMSFzt4O41rSzT4-5bDOu0o59H8V. [Πρόσβαση 20 2 2023].

[14] «GitHub RecipeBook,» [Ηλεκτρονικό]. Available: https://github.com/bhavikvashi1804/RecipeBook. [Πρόσβαση 20 2 2023].

[15] «NgRx RecipeBook,» [Ηλεκτρονικό]. Available: https://github.com/ccodecheff/angular-recipe-book-optimized-ngrx. [Πρόσβαση 10 3 2023].

[16] «state management in angular with ngrx,» [Ηλεκτρονικό]. Available: https://auth0.com/blog/state-management-in-angular-with-ngrx-1/.

[17] «angular state management made simple with ngrx,» [Ηλεκτρονικό]. Available: https://blog.logrocket.com/angular-state-management-made-simple-with-ngrx/.

[18] «effects guide,» [Ηλεκτρονικό]. Available: https://ngrx.io/guide/effects.

[19] «Firebase wiki,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Firebase.

[20] «firebase for web firebase realtime database,» [Ηλεκτρονικό]. Available: https://medium.com/codingurukul/firebase-for-web-firebase-realtime-database-9280a52ced83.

[21] «firebase for web authentication auth with email and password,» [Ηλεκτρονικό]. Available: https://medium.com/codingurukul/firebase-for-web-authentication-auth-with-email-and-password-cc4f7b4efc1b.

[22] «angular8 authentication and authorization,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/angular8/angular8_authentication_and_authorization.htm.

[23] «what should be included in core shared and feature modules,» [Ηλεκτρονικό]. Available: https://stackoverflow.com/questions/64040341/what-should-be-included-in-core-shared-and-feature-modules.

[24] «when to use angulars dynamic components,» [Ηλεκτρονικό]. Available: https://medium.com/@damodara.puddu/when-to-use-angulars-dynamic-components-ce40db069359.

[25] «app core shared feature modules,» [Ηλεκτρονικό]. Available: https://thetombomb.com/posts/app-core-shared-feature-modules.

[26] «RxJS Introduction,» [Ηλεκτρονικό]. Available: https://rxjs.dev/guide/overview.

[27] «angular state managemen using NgRx,» [Ηλεκτρονικό]. Available: https://massivepixel.io/blog/angular-state-management/.

[28] «ngrx is it worth it,» [Ηλεκτρονικό]. Available: https://medium.com/web-factory-llc/ngrx-is-it-worth-it-6ad9585dcbaa.

Παράρτημα Α

Στο παράρτημα αυτό παρατίθενται, ενδεικτικά, κάποια σημαντικά τμήματα από τον κώδικα τις εφαρμογής. Στην αρχή κάθε φωτογραφίας αναγράφεται το όνομα του αρχείου που περιέχει τον κώδικα.

auth.component.html

```
<ng-template appPlaceholder></ng-template>
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <div *ngIf="isLoading" style="text-align: center;">
      <app-loading-spinner></app-loading-spinner>
    </div>
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input
          type="email"
          id="email"
          class="form-control"
          ngModel
          name="email"
          required
          email
        />
      </div>
      <div class="form-group">
        <label for="password">Κωδικός</label>
        <input
          type="password"
          id="password"
          class="form-control"
          ngModel
          name="password"
          required
          minlength="6"
        />
      </div>
      <div>
        <button
          class="btn btn-primary"
          type="submit"
          [disabled]="!authForm.valid"
        >
          {{ isLoginMode ? 'Σύνδεση' : 'Εγγραφή' }}
        </button>
        |
        <button class="btn btn-primary" (click)="onSwitchMode()" type="button">
          Μεταβείτε στην επιλογή {{ isLoginMode ? 'Εγγραφή' : 'Σύνδεση' }}
        </button>
      </div>
    </form>
  </div>
</div>
```

auth.component.ts

```ts
import {
  Component,
  ComponentFactoryResolver,
  ViewChild,
  OnDestroy,
  OnInit
} from '@angular/core';
import { NgForm } from '@angular/forms';
import { Subscription } from 'rxjs';
import { Store } from '@ngrx/store';

import { AlertComponent } from '../shared/alert/alert.component';
import { PlaceholderDirective } from '../shared/placeholder/placeholder.directive';
import * as fromApp from '../store/app.reducer';
import * as AuthActions from './store/auth.actions';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent implements OnInit, OnDestroy {
  isLoginMode = true;
  isLoading = false;
  error: string = null;
  @ViewChild(PlaceholderDirective, { static: false }) alertHost: PlaceholderDirective;

  private closeSub: Subscription;
  private storeSub: Subscription;

  constructor(
    private componentFactoryResolver: ComponentFactoryResolver,
    private store: Store<fromApp.AppState>
  ) {}

  ngOnInit() {
    this.storeSub = this.store.select('auth').subscribe(authState => {
      this.isLoading = authState.loading;
      this.error = authState.authError;
      if (this.error) {
        this.showErrorAlert(this.error);
      }
    });
  }

  onSwitchMode() {
    this.isLoginMode = !this.isLoginMode;
  }
```

```
  onSubmit(form: NgForm) {
    if (!form.valid) {
      return;
    }
    const email = form.value.email;
    const password = form.value.password;

    if (this.isLoginMode) {
      // authObs = this.authService.login(email, password);
      this.store.dispatch(
        new AuthActions.LoginStart({ email: email, password: password })
      );
    } else {
      this.store.dispatch(
        new AuthActions.SignupStart({ email: email, password: password })
      );
    }

    form.reset();
  }

  onHandleError() {
    this.store.dispatch(new AuthActions.ClearError());
  }

  ngOnDestroy() {
    if (this.closeSub) {
      this.closeSub.unsubscribe();
    }
    if (this.storeSub) {
      this.storeSub.unsubscribe();
    }
  }

  private showErrorAlert(message: string) {
    // const alertCmp = new AlertComponent();
    const alertCmpFactory = this.componentFactoryResolver.resolveComponentFactory(
      AlertComponent
    );
    const hostViewContainerRef = this.alertHost.viewContainerRef;
    hostViewContainerRef.clear();

    const componentRef = hostViewContainerRef.createComponent(alertCmpFactory);

    componentRef.instance.message = message;
    this.closeSub = componentRef.instance.close.subscribe(() => {
      this.closeSub.unsubscribe();
      hostViewContainerRef.clear();
    });
  }
}
```

auth.effects.ts

```
const handleError = (errorRes: any) => {
  let errorMessage = 'An unknown error occurred!';
  if (!errorRes.error || !errorRes.error.error) {
    return of(new AuthActions.AuthenticateFail(errorMessage));
  }
  switch (errorRes.error.error.message) {
    case 'EMAIL_EXISTS':
      errorMessage = 'This email exists already';
      break;
    case 'EMAIL_NOT_FOUND':
      errorMessage = 'This email does not exist.';
      break;
    case 'INVALID_PASSWORD':
      errorMessage = 'This password is not correct.';
      break;
  }
  return of(new AuthActions.AuthenticateFail(errorMessage));
};
```

header.component.html

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a routerLink="/" class="navbar-brand">Βιβλίο Συνταγών</a>
    </div>

    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li routerLinkActive="active" *ngIf="isAuthenticated">
          <a routerLink="/recipes">Συνταγές</a>
        </li>
        <li routerLinkActive="active" *ngIf="!isAuthenticated">
          <a routerLink="/auth">Είσοδος</a>
        </li>
        <li routerLinkActive="active">
          <a routerLink="/shopping-list">Λίστα Αγορών</a>
        </li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        <li *ngIf="isAuthenticated">
          <a style="cursor: pointer;" (click)="onLogout()">Εξοδος</a>
        </li>
        <li class="dropdown" appDropdown *ngIf="isAuthenticated">
          <a style="cursor: pointer;" class="dropdown-toggle" role="button"
            >Διαχείριση <span class="caret"></span
          ></a>
          <ul class="dropdown-menu">
            <li>
              <a style="cursor: pointer;" (click)="onSaveData()">Αποθήκευση Δεδομένων</a>
            </li>
            <li>
              <a style="cursor: pointer;" (click)="onFetchData()">Λήψη Δεδομένων</a>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

header.component.ts

```
@Component({
  selector: 'app-header',
  templateUrl: './header.component.html'
})
export class HeaderComponent implements OnInit, OnDestroy {
  isAuthenticated = false;
  private userSub: Subscription;

  constructor(
    private store: Store<fromApp.AppState>
  ) {}

  ngOnInit() {
    this.userSub = this.store
      .select('auth')
      .pipe(map(authState => authState.user))
      .subscribe(user => {
        this.isAuthenticated = !!user;
        console.log(!user);
        console.log(!!user);
      });
  }

  onSaveData() {
    // this.dataStorageService.storeRecipes();
    this.store.dispatch(new RecipeActions.StoreRecipes());
  }

  onFetchData() {
    // this.dataStorageService.fetchRecipes().subscribe();
    this.store.dispatch(new RecipeActions.FetchRecipes());
  }

  onLogout() {
    this.store.dispatch(new AuthActions.Logout());
  }

  ngOnDestroy() {
    this.userSub.unsubscribe();
  }
```

recipe-detail.component.html

```
<div class="row">
  <div class="col-xs-12">
    <img
      [src]="recipe.imagePath"
      alt="{{ recipe.name }}"
      class="img-responsive"
      style="max-height: 300px;">
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    <h1>{{ recipe.name }}</h1>
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    <div
      class="btn-group"
      appDropdown>
      <button
        type="button"
        class="btn btn-primary dropdown-toggle">
        Διαχείριση Συνταγής <span class="caret"></span>
      </button>
      <ul class="dropdown-menu">
        <li><a (click)="onAddToShoppingList()" style="cursor: pointer;">Προσθήκη στη Λίστα Αγορών</a></li>
        <li><a style="cursor: pointer;" (click)="onEditRecipe()">Επεξεργασία συνταγής</a></li>
        <li><a style="cursor: pointer;" (click)="onDeleteRecipe()">Διαγραφή συνταγής</a></li>
      </ul>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    {{ recipe.description }}
  </div>
</div>
<div class="row">
  <div class="col-xs-12">
    <ul class="list-group">
      <li
        class="list-group-item"
        *ngFor="let ingredient of recipe.ingredients">
        <b>{{ ingredient.amount }}</b> {{ ingredient.name }}
      </li>
    </ul>
  </div>
</div>
```

recipe-edit.component.html

```
<div class="row">
  <div class="col-xs-12">
    <form [formGroup]="recipeForm" (ngSubmit)="onSubmit()">
      <div class="row">
        <div class="col-xs-12">
          <button
            type="submit"
            class="btn btn-success"
            [disabled]="!recipeForm.valid"
          >
            Αποθήκευση
          </button>
          <button type="button" class="btn btn-danger" style="margin-left: 5px;" (click)="onCancel()">
            Ακύρωση
          </button>
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <div class="form-group">
            <label for="name">Όνομα</label>
            <input
              type="text"
              id="name"
              formControlName="name"
              class="form-control"
            />
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <div class="form-group">
            <label for="imagePath">Φωτογραφία</label>
            <input
              type="text"
              id="imagePath"
              formControlName="imagePath"
              class="form-control"
              #imagePath
            />
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <img [src]="imagePath.value" class="img-responsive" />
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <div class="form-group">
            <label for="description">Περιγραφή</label>
            <textarea
```

```
            type="text"
            id="description"
            class="form-control"
            formControlName="description"
            rows="6"
          ></textarea>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-xs-12" formArrayName="ingredients">
        <div
          class="row"
          *ngFor="let ingredientCtrl of ingredientsControls; let i = index"
          [formGroupName]="i"
          style="margin-top: 10px;"
        >
          <div class="col-xs-8"> Συστατικό
            <input type="text" class="form-control" formControlName="name" />
          </div>
          <!-- <div class="col-xs-2">
            <input
              type="number"
              class="form-control"
              formControlName="amount"
            />
          </div> -->
          <div class="col-xs-2"> Ποσότητα
            <input
              class="form-control"
              formControlName="amount"
            />
          </div>
          <div class="col-xs-2"> <br>
            <button
              type="button"
              class="btn btn-danger"
              (click)="onDeleteIngredient(i)"
            >
              X
            </button>
          </div>
        </div>
        <hr />
        <div class="row">
          <div class="col-xs-12">
            <button
              type="button"
              class="btn btn-success"
              (click)="onAddIngredient()"
            >
              Προσθήκη Συστατικού
            </button>
          </div>
        </div>
```

ingredient.model.ts

```
export class Ingredient {
  constructor(public name: string, public amount: number) {}
}
```

shopping-edit.component.html

```
<div class="row">
  <div class="col-xs-12">
    <form (ngSubmit)="onSubmit(f)" #f="ngForm">
      <div class="row">
        <div class="col-sm-5 form-group">
          <label for="name">'Ονομα</label>
          <input
            type="text"
            id="name"
            class="form-control"
            name="name"
            ngModel
            required
          >
        </div>
        <div class="col-sm-2 form-group">
          <label for="amount">Ποσότητα</label>
          <input
            type="text"
            id="amount"
            class="form-control"
            name="amount"
            ngModel
            required
          >
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <button
            class="btn btn-success"
            style="margin-right: 5px;"
            type="submit"
            [disabled]="!f.valid">{{ editMode ? 'Ενημέρωση' : 'Προσθήκη' }}</button>
          <button
            class="btn btn-danger"
            style="margin-right: 5px;"
            type="button"
            (click)="onDelete()"
            *ngIf="editMode">Διαγραφή</button>
          <button class="btn btn-primary" type="button" (click)="onClear()">Εκκαθάριση Δεδομένων</button>
        </div>
      </div>
    </form>
  </div>
</div>
```

shopping-list.component.html

```
<div class="row">
  <div class="col-xs-10">
    <app-shopping-edit></app-shopping-edit>
    <hr>
    <ul class="list-group">
      <a
        class="list-group-item"
        style="cursor: pointer"
        *ngFor="let ingredient of (ingredients | async).ingredients; let i = index"
        (click)="onEditItem(i)"
      >
      {{ ingredient.name }} <b>{{ ingredient.amount }}</b>
      </a>
    </ul>
  </div>
</div>
```

app.module.ts (με τις βελτιστοποιήσεις για lazy loading και NgRx)

```
@NgModule({
  declarations: [AppComponent, HeaderComponent],
  imports: [
    BrowserModule,
    HttpClientModule,
    AppRoutingModule,
    StoreModule.forRoot(fromApp.appReducer),
    EffectsModule.forRoot([AuthEffects, RecipeEffects]),
    StoreDevtoolsModule.instrument({ logOnly: environment.production }),
    StoreRouterConnectingModule.forRoot(),
    SharedModule,
    CoreModule
  ],
  bootstrap: [AppComponent]
  // providers: [LoggingService]
})
export class AppModule {}
```

app.module.ts (χωρίς καμία βελτιστοποίηση, έχοντας ένα κεντρικό app module από το οποίο φορτώνονται κάθε φορά όλα τα άλλα)

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    RecipesComponent,
    RecipeListComponent,
    RecipeDetailComponent,
    RecipeItemComponent,
    ShoppingListComponent,
    ShoppingEditComponent,
    DropdownDirective,
    RecipeStartComponent,
    RecipeEditComponent,
    AuthComponent,
    LoadingSpinnerComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    AppRoutingModule
  ],
  providers: [
    ShoppingListService,
    RecipeService,
    {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthInterceptorService,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```