ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ELECTRONIFY: ΕΦΑΡΜΟΓΗ ΙΣΤΟΥ ΗΛΕΚΤΡΟΝΙΚΟΥ ΕΜΠΟΡΙΟΥ»

Του φοιτητή Μιχαήλ Άγγελος Γεωργίου Αρ. Μητρώου: 154415	Επιβλέπων ΣΙΔΗΡΟΠΟΥΛΟΣ ΑΝΤΩΝΙΟΣ Επίκουρος Καθηγητής

Ημερομηνία 20-07-2022

Τίτλος Δ.Ε.ΕΦΑΡΜΟΓΗ ΙΣΤΟΥ ΗΛΕΚΤΡΟΝΙΚΟΥ ΕΜΠΟΡΙΟΥ
Κωδικός Δ.Ε. 20207
Ονοματεπώνυμο φοιτητή ΜΙΧΑΗΛ ΑΓΓΕΛΟΣ ΓΕΩΡΓΙΟΥ
Ονοματεπώνυμο εισηγητή ΑΝΤΩΝΙΟΣ ΣΙΔΗΡΟΠΟΥΛΟΣ
Ημερομηνία ανάληψης Δ.Ε. 20-10-2020
Ημερομηνία περάτωσης Δ.Ε. 20-07-2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Η εργασία αυτή είναι αφιερωμένη στην γυναικά μου που μου έδινε δύναμη καθ' όλη την διάρκεια για να συνεχίσω μέχρι το τέλος και στους μέντορες μου που έδωσαν το κίνητρο και τις γνώσεις τους»

Πρόλογος

Η ραγδαία ανάπτυξη της τεχνολογίας και του διαδικτύου, μας έκανε μάρτυρες κοσμοϊστορικών αλλαγών στην ανθρωπότητα, από τα μέσα μαζικής επικοινωνίας μέχρι την ραγδαία ανάπτυξη του ηλεκτρονικού εμπορίου. Στις μέρες μας πλέον αν κάποιος δεν χρησιμοποιεί μια έξυπνη συσκευή η ζωή του φαντάζει πολύ πιο δύσκολη. Έτσι δημιούργησε την ανάγκη για τεχνολογική εξέλιξη σε μικρές και μεγάλες επιχειρήσεις να έχουν όλες την δική τους διαδικτυακή σελίδα ή εφαρμογή για να μπορέσουν να έχουν και αυτοί ένα μερίδιο σε αυτό τον τεράστιο αναπτυσσόμενο εμπόριο. Η σύγχρονη ζωή στην καθημερινότητα μας πλέον γίνεται όλο και πιο απαιτητική έτσι ο χρόνος λιγοστεύει. Μια απλή, εύχρηστη εφαρμογή όπου μπορεί να εξυπηρέτηση μικρομεσαίες επιχειρήσεις χωρίς μεγάλο κόστος, τους δίνει την δυνατότητα να μπουν και αυτοί στο παγκόσμιο ηλεκτρονικό εμπόριο.

Περίληψη

Η παρούσα πτυχιακή εργασία αποτέλεσε τον σχεδιασμό, ανάπτυξη και υλοποίηση μιας εφαρμογής ιστού για εύκολες αγορές ηλεκτρονικών συσκευών στο διαδίκτυο. Η εφαρμογή έχει στόχο την βελτιστοποίηση αγορών μέσω διαδικτύου με λίγα απλά βήματα να κάνεις τις αγορές σου, επίσης δίνει την ευκαιρία σε μικρομεσαίες επιχειρήσεις με λίγο κόστος να μπουν ανταγωνιστικά στον κόσμο του ηλεκτρονικού εμπορίου. Ταυτόχρονα ο σχεδιασμός έγινε με γνώμονα την εμπειρία του χρήστη και πως αυτός με λιγοστές γνώσεις σε εφαρμογές τέτοιου τύπου να μπορεί να κάνει και αυτός χρήση της χωρίς ιδιαίτερες δυσκολίες. Η εφαρμογή είναι ιστού δηλαδή χρειάζεσαι ένα πρόγραμμα περιήγησης για να έχει κάποιος πρόσβαση. Για την ανάπτυξη της εφαρμογής ως κύριο εργαλείο χρησιμοποιήθηκε το Play Framework είναι ανοικτού κώδικα πλαίσιο για την ανάπτυξη εφαρμογών ιστού που ακολουθεί το αρχιτεκτονικό μοντέλο MVC(model-view-controller). Το Play Framework είναι γραμμένο σε Scala και μπορεί να χρησιμοποιηθεί από άλλες γλώσσες προγραμματισμού που γίνονται compile σε JVM. Η εφαρμογή αναπτύχθηκε με την γλώσσα προγραμματισμού Scala με μικρά κομμάτια κώδικα JavaScript. Επιπρόσθετα στο front end κομμάτι έγινε χρήση του ανοικτού κώδικα πλαίσιο Bootstrap για να κάνει την εφαρμογή ανταποκρινόμενη. Για την ανάπτυξη της βάσης δεδομένων έγινε χρήση το ανοικτού κώδικα πλαίσιο Flyway το οποίο είναι εργαλείο μετανάστευσης βάσης δεδομένων με PostgreSQL για βάση δεδομένων. Η εφαρμογής έχει όλες τις απαραίτητες λειτουργίες και δυνατότητες που θα χρειαστεί ο ιδιοκτήτης του μαγαζιού για να διαχειρίζεται τα προϊόντα του αλλά και ο πελάτης για να κάνει εύκολα και ευχάριστα τις αγορές του. Στην εργασία θα παρουσιαστεί κάθε οπτική πλευρά της εφαρμογής με διαγράμματα υψηλού επιπέδου μέχρι κάθε τεχνική λεπτομέρεια και γιατί λήφθηκαν αποφάσεις σε συγκεκριμένα προβλήματα που προέκυψαν κατά την ανάπτυξη της εφαρμογής.

ELECTRONIFY: WEB E-COMMERCE APPLICATION

MICHAIL ANGELOS GEORGIOU

Abstract

This thesis was design, development and implementation of a web application for easy online purchases of electronic devices. The application aims to optimize online shopping with a few simple steps to make your purchases, it also gives the opportunity to small and medium-sizes businesses with little cost to enter competitively in the world of e-commerce. At the same time the design was made with the use's experience in mind and so that those with little knowledge in applications of this type can also use it without particular difficulties. The application is web-based, meaning you need a browser to access it. For the development of the application, the Play Framework was used as the main tool. It is an open-source framework for the developments of web applications that follows the MVC (model-view-controller) architectural model. The Play Framework is written in Scala and can be used by other programming languages that compile to JVM. The application was developed with Scala programming language with small pieces of JavaScript code. Additionally on the front-end part the open-source Bootstrap framework was used to make the application responsive. For the development of the database, the open-source Flyway framework was used which is a database migration tool with PostgreSQL for databases. The application has all the necessary functions and features that the shop owner will need to manage his products and also the customer to make his purchases easily and pleasantly. The paper will present every visual aspect of the application with high-level diagrams down to every technical detail and why decisions were made on specific problems that arose during the development of the application.

Ευχαριστίες

Σε αυτή την ενότητα και με το πέρας της πτυχιακής εργασίας θα ήθελα να ευχαριστήσω ολόκληρη μου την οικογένεια που μου στάθηκε σε όλη την διάρκεια αυτού του όμορφου ταξιδιού. Τους μέντορες μου που βρέθηκαν στο πέρασμα μου όταν έκανα την πρακτική μου και μοιράστηκαν μαζί μου ατέλειωτες από τον χρόνο τους ώρες να μοιράζονται τις γνώσεις τους μαζί μου δίνοντας μου κίνητρο και προετοιμάζοντας με στον να γίνω ένας σωστός επαγγελματίας. Τέλος θα ήθελα να ευχαριστήσω τον κύριο Αντώνη Σιδηρόπουλο που μου έδωσε την ευκαιρία να φέρω εις πέρας αυτή την εργασία και για τις ώρες που ο ίδιος προσωπικά αφιέρωσε εβδομαδιαίος σε όλους του φοιτητές καθοδηγώντας τους και δίνοντας μας συμβουλές.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Κώδικα

Συντομογραφίες

JVM	Java Virtual Machine
UX	User Experience
UI	User Interface
Π.Ε.	Πτυχιακή Εργασία
MVC	Model View Controller
PL	Play Framework
FE	Front end
BE	BE

Κεφάλαιο 1ο: Ηλεκτρονικό εμπόριο και ανάλυση λογισμικού

1.1 Εισαγωγή

Το ηλεκτρονικό εμπόριο είναι η αγορά και πώληση αγαθών ή υπηρεσιών μέσω ενός ηλεκτρονικού δικτύου, κυρίως του διαδικτύου. Αυτές οι επιχειρηματικές συναλλαγές πραγματοποιούνται είτε ως επιχείρηση προς επιχείρηση είτε από επιχείρηση προς καταναλωτή. Στις μέρες μας είναι ευρέος διαδεδομένο με συνεχή τάση να αυξάνεται λόγο τις απαιτητικής σύγχρονης ζωής όπου η έλλειψη χρόνου είναι ένα σημαντικός παράγοντας στην καθημερινότητα ενός ατόμου. Επιπρόσθετα δίνει την ευκαιρία στους αγοραστές να κάνουν την πιο σωστή και συμφέρουσα επιλογή από μια γκάμα πολύ μεγαλύτερη από ένα φυσικό μαγαζί από όπου και να βρίσκονται. Οι πωλητές από την άλλη να έχουν ένα πολύ μεγαλύτερο κοινό να προωθήσουν τα προϊόντα ή τις υπηρεσίες τους. Σε αυτή την πτυχιακής εργασία στόχος είναι η δημιουργία μιας σύγχρονης και εύχρηστης εφαρμογής διαδικτύου με πρωτοποριακές τεχνολογίες για ηλεκτρονικό εμπόριο μεταξύ επιχείρησης προς καταναλωτή για ηλεκτρονικές συσκευές.

1.2 Ανταγωνισμός

Το ηλεκτρονικό εμπόριο έχει αλλάξει όχι μόνο τον τρόπο με τον οποίο οι καταναλωτές ψωνίζουν, αλλά και το φάσμα των παροχών από τους οποίους οι καταναλωτές μπορούν να αγοράσουν προϊόντα και υπηρεσίες. Η ανάπτυξη του ηλεκτρονικού εμπορίου έχει τη δυνατότητα να αυξήσει τον ανταγωνισμό στις αγορές λιανικής, να ενισχύσει σημαντικά τις επιλογές των καταναλωτών και να προτρέψει και να διευκολύνει την καινοτομία στη διανομή προϊόντων. Ωστόσο, οι πρόσφατες εργασίες επιβολής του νόμου και υπεράσπισης ορισμένων αρχών ανταγωνισμού έχουν δείξει τη δυνατότητα εμφάνισης αθέμητων ανταγωνιστικών συμπεριφορών στο διαδικτυακό. Ως εκ τούτου, οι φόβοι σχετικά με την πιθανή κατάτμηση της αγοράς αποτελούν αξιοσημείωτη ανησυχία σε ορισμένους φορείς που έχουν εξετάσει το ζήτημα του ανταγωνισμού στον τομέα του ηλεκτρονικού εμπορίου. Συχνά εγείρονται και άλλα ερωτήματα σχετικά με τον ορισμό της αγοράς στη σφαίρα του ηλεκτρονικού εμπορίου, τους κάθετους και οριζόντιους περιορισμούς, τη μονομερή συμπεριφορά με το διαδίκτυο των πραγμάτων και τους ελέγχους συγχωνεύσεων.

Συνήθως, οι μικρές ή εκτός σύνδεσης επιχειρήσεις απλώς δεν είναι εξοπλισμένες για να ανταγωνιστούν και επομένως δεν έχουν το πλεονέκτημα έναντι των επωνυμιών που έχουν ήδη καθιερωθεί. Η ικανοποίηση των πελατών είναι ένας τεράστιος ενισχυτής αξιολόγησης ανταγωνισμού, και αν δεν υπάρχει αρκετό, μια επιχείρηση δεν μπορεί να διατηρήσει μια τεράστια βάση πελατών.

Το ηλεκτρονικό εμπόριο παραμένει το σημείο-στόχος, καθώς οι εταιρείες και οι επιχειρήσεις θα συνεχίσουν να δημιουργούν διαδικτυακά προφίλ. Μόνο στο Ηνωμένο Βασίλειο, ο κλάδος του ηλεκτρονικού εμπορίου αποτελούσε το 27,5% των πωλήσεων το 2020. Αυτός ο αριθμός αναμένεται να φτάσει το 32,1% το 2024, επομένως μπορείτε να δείτε ότι το ηλεκτρονικό εμπόριο θα γίνει πιο δημοφιλές [10]. Συνολικά, το ηλεκτρονικό εμπόριο ως ποσοστό των συνολικών παγκόσμιων πωλήσεων θα συνεχίσει να αυξάνεται τα επόμενα χρόνια και θα αρχίσει να τυλίγει την ψηφιακή επωνυμία ακόμη περισσότερο από πριν.

Επόμενος αν μια επιχείρηση θέλει να παραμείνει δυναμική στον ανταγωνισμό και να προσπαθήσει να διατήρηση μια βάση από πελάτες ή και ακόμη να την επεκτείνει θα πρέπει να προσθέσει στα πλάνα της μια ecommerce εφαρμογή και αν παρέχει την καλύτερη δυνατή εξυπηρέτηση στους πελάτες της.

Σχήμα 1.1: e-Commerce growth statistic [11].

1.3 Στόχοι της εφαρμογής

Οι κύριοι στόχοι που θέτει η εφαρμογή και η επίλυση προβλημάτων που μπορεί να προσφέρει σε μια επιχείρηση και στο απλό χρήστη είναι τα ακόλουθα:

1. Αύξηση των πωλήσεων της επιχείρησης και γενικότερα η αύξηση των εσόδων σε μια εταιρεία όπου με χαμηλό κόστος συντήρησης μπορεί να διαθέτει μια απλή και ταυτόχρονα ανταγωνιστική διαδικτυακή εφαρμογή
2. Προσέλκυση χρηστών χωρίς πολλές γνώσεις στο ηλεκτρονικό εμπόριο.
3. Εκσυγχρονισμός επιχείρησης.
4. Αποστολή και αγορά προϊόντων από οπουδήποτε σημείο στην γη.
5. Ευκολία αγοράς σε άτομα με κινητικές δυσκολίες που καθιστά την φυσική τους παρουσία σε ένα μαγαζί πολύ δύσκολη ή και αδύνατη.
6. Ανθρώπους που έχουν μειωμένο ελεύθερο χρόνο και η παραγγελία διαδικτυακά είναι η μόνη λύση.
7. Εύκολα μπορείς να συγκρίνει τιμές ο καταναλωτής και να επιλέξει την πιο συμφέρουσα για τον ίδιο επιλογή. Με λίγα λόγια εξαλείφετε το πρόβλημα της αισχροκέρδειας όπου προβαίνουν μερικές επιχειρήσεις εν αγνοία των καταναλωτών
8. Η ποικιλία αγοράς του καταναλωτή ακόμη και αν βρίσκεται σε μια μικρή επαρχία αυξάνεται δραματικά γιατί έχει την δυνατότητα αγοράς διαδικτυακά ότι χρειάζεται από όπου και αν βρίσκεται.

1.4 Απαιτήσεις λογισμικού

uΑπαιτήσεις λογισμικού είναι αυτές που καθορίζουν τις δυνατότητες που μπορεί να έχει το λογισμικό αλλά και τους περιορισμούς. Με άλλα λόγια θέλουμε να καθορίσουμε τον σκοπό του προϊόντος μας και να περιγράψουμε τι κατασκευάζουμε πληρώντας πάντα τις απαιτήσεις και τις κατάλληλες λειτουργίες που θέλουμε να κάνει το προϊόν μας [13].

1.4.1 Εμπειρία Χρήστη

Η εμπειρία χρήστη (UX) εστιάζει στη βαθιά κατανόηση των χρηστών, τι χρειάζονται, τι εκτιμούν, τις ικανότητές τους και επίσης τους περιορισμούς τους. Λαμβάνει επίσης υπόψη τους επιχειρηματικούς στόχους και στόχους του οργανισμού που διαχειρίζεται το έργο [12].

Στην εφαρμογή έπαιξε σημαντικό παράγοντα στην σχεδίαση της το UX και πως αυτό επηρεάζει την εμπειρία που θα έχει ένας χρήστης στην εφαρμογή μας. Πριν ξεκινήσει η υλοποίηση της έκανα μια μικρή έρευνα σχετικά με την εμπειρία των χρηστών στην εφαρμογή και ο κυριότερος μου στόχος ήταν οι ομάδες με χαμηλού επιπέδου εμπειρία σε ανάλογες εφαρμογές. Ο λόγος που επιδίωξα αυτές τις συγκεκριμένες ομάδες ατόμων είναι γιατί όπως ο κάθε ένας από εμάς έχει άτομα στον περίγυρο του που θα ήθελαν να μπουν στο ηλεκτρονικό εμπόριο είτε γενικότερα στο διαδίκτυο για τις αμέτρητες διευκολύνσεις που παρέχει όμως η έλλειψη εμπειρίας το καθιστά πολύ δύσκολο.

Φυσικά όμως μια έρευνα στην αρχή της υλοποιήσεις της εφαρμογής δεν ήταν αρκετή χρειάστηκα και ένα δεύτερο συμπέρασμα κάπου περίπου στα μισά. Τελικά τα αποτελέσματα ήταν απογοητευτικά για τις ομάδες χαμηλής εμπειρίας με ένα 50% (από 10 άτομα) περίπου να δυσκολεύεται στην ολοκλήρωση αγοράς. Ακολουθώντας προτάσεις από τους χρήστες έκανα ένα σχεδιασμό του καλαθιού αγορών και αποκόμισα τα παρακάτω σημεία για την συνέχεια του σχεδιασμού και υλοποίησης της εφαρμογής μου.

1. Το περιεχόμενό πρέπει να είναι πρωτότυπο και να καλύπτει μια ανάγκη.
2. Οι χρήστες πρέπει να εμπιστεύονται και να πιστεύουν αυτό που τους λέτε.
3. Η απλότητα είναι ένας από τους σημαντικότερους παράγοντες που έπαιξε ρόλο στην δυσκολία της εφαρμογής. Ο καταιγισμός πληροφοριών και αρκετών ενεργειών που έχει ο χρήστης να κάνει σε ένα βήμα.
4. Η καθοδήγηση του χρήστη χωρίς να του δίνεις πολλά περιθώρια λάθους καθώς αυτό προκαλεί εκνευρισμό όπου ισοδύναμή με κακή εμπειρία.
5. Η εικόνα, η ταυτότητα, η επωνυμία και άλλα σχεδιαστικά στοιχεία χρησιμοποιούνται για να προκαλέσουν συναισθήματα και εκτίμηση

1.4.2 Ασφάλεια

Καθώς η τεχνολογία αλλάζει, γίνεται όλο και πιο δύσκολο για τις επιχειρήσεις κάθε τύπου να διατηρούν ασφαλή τις προσωπικές τους πληροφορίες και τις πληροφορίες των πελατών τους στον ιστό. Η ασφάλεια του ιστού είναι σημαντική για να αποτρέψει τους χάκερ και τους κλέφτες του κυβερνοχώρου από την πρόσβαση σε ευαίσθητες πληροφορίες. Χωρίς μια προληπτική στρατηγική ασφαλείας, οι επιχειρήσεις διακινδυνεύουν την εξάπλωση και την κλιμάκωση κακόβουλου λογισμικού, επιθέσεις σε άλλους ιστότοπους, δίκτυα και άλλες υποδομές πληροφορικής. Εάν ένας χάκερ είναι επιτυχής, οι επιθέσεις μπορούν να εξαπλωθούν από υπολογιστή σε υπολογιστή, καθιστώντας δύσκολη την εύρεση της προέλευσης [22].

Η ασφάλεια στο ηλεκτρονικό εμπόριο παίζει ένα από τους σημαντικούς ρόλους για την αξιοπιστία της εφαρμογής μας αλλά και την ακεραιότητα μιας επιχείρησης. Οι εγκληματίες του κυβερνοχώρου έχουν ως στόχο πρώτα τις επιχειρήσεις ηλεκτρονικού εμπορίου [14]. Επομένως, μια επιχείρηση θα πρέπει να χρησιμοποιεί πρωτόκολλα και μέτρα ασφαλείας για το ηλεκτρονικό εμπόριο. Με αυτό τον τρόπο θα προσπαθήσει να κρατήσει την επιχείρηση και τους πελάτες απαλλαγμένους από επιθέσεις.

Στην εφαρμογή λοιπόν έχουν καθοριστεί μέτρα ασφάλειας για τους χρήστες και αποφευχθήκαν στρατηγικές μεγάλου ρίσκου όπου θα έθεταν σε μεγαλύτερο κίνδυνο τους πελάτες, την επιχείρηση αλλά και την ίδια την εφαρμογή σε περίπτωση μιας επιτυχούς επίθεσης. Καθώς κρίσιμα προσωπικά στοιχεία των χρηστών αποθηκεύονται στο σύστημα και σε περίπτωση μιας επιτυχημένης επίθεσης αυτό θα αποβεί μοιραίο για μας και για την επιχείρηση που θα έχει χάσει τα στοιχεία των πελατών της. Συγκριτικά όμως αυτό είναι το λιγότερο γιατί κανείς μετά δεν γνωρίζει για το πως τα δεδομένα αυτά θα χρησιμοποιηθούν που ίσως τίθεται η ακεραιότητα τους στον πραγματικό κόσμο σε κίνδυνο. Τα μέτρα που πάρθηκαν για την καλύτερη ασφάλεια της εφαρμογής είναι :

- Κρυπτογράφηση στην βάση δεδομένων σε όλα τα κρίσιμα απαραίτητα προσωπικά δεδομένα των χρηστών που χρειάζεται η εφαρμογή για να λειτουργήσει.
- Περιορισμό στις ενέργειες ενός χρήστης που δεν είναι διαχειριστής.
- Ανάθεση των πληρωμών σε τρίτη εταιρεία για ασφαλές συναλλαγές που παρέχουν αποκλειστικά τέτοιου είδους υπηρεσίες.
- Μη αποθήκευση δεδομένων σχετικά με τραπεζικούς λογαριασμούς των χρηστών.

1.4.3 Περιπτώσεις Χρήστη

Στις απαιτήσεις λοιπόν τις εφαρμογής κρίθηκε απαραίτητο η καταγραφή και ανάλυση κάποιον περιπτώσεων χρήστη όπου θα ξεκαθάριζαν συγκεκριμένες λειτουργίες της εφαρμογής και πως η εφαρμογή θα ανταποκρίνεται σε αυτές τις περιπτώσεις. Μια περίπτωση χρήσης είναι μια λίστα ενεργειών ή βημάτων συμβάντων που συνήθως ορίζουν τις αλληλεπιδράσεις μεταξύ ενός ρόλου και ενός συστήματος για την επίτευξη ενός στόχου. Ο ηθοποιός όπως είναι και γνωστός ο ρόλος στην ενοποιημένη γλώσσα μοντελοποίησης (UML) μπορεί να είναι άνθρωπος ή άλλο εξωτερικό σύστημα το όποιο αλληλοεπιδρά με ένα λογισμικό, στην προκυμμένη περίπτωση την εφαρμογή μας. Η περιπτώσεις χρήστη είναι ένα σενάριο με πρωταγωνιστή των ηθοποιό έχοντας κάποιες προϋποθέσεις για να μπορέσει να ολοκληρωθεί το σενάριο με μια κύρια ροή γεγονότων όπου αντιπροσωπεύει την περίπτωση που αναλύουμε. Επιπρόσθετα έχουμε και τις εναλλακτικές ροές γεγονότων όπου σχετίζονται με την κύρια ροή όμως λόγος κάποιας άλλης ενέργειας που έγινε κατά την διάρκεια του κύριου σεναρίου είχαμε διαφορετικό αποτέλεσμα.

Πίνακας 1.1 Περίπτωση χρήσης: Εγγραφή χρήστη στην εφαρμογή

Περίπτωση χρήσης	Εγγραφή χρήστη στην εφαρμογή
Χρήστης	Μη εγγεγραμμένος
Προϋπόθεση	Σύνδεση στο διαδίκτυο από πρόγραμμα περιήγησης
Ροή Γεγονότων	● Άγνωστος χρήστη επιλέγει εγγραφή από την κύρια σελίδα. ● Συμπληρώνει την φόρμα με τα κατάλληλα απαραίτητα στοιχεία. ● Πατάει το κουμπί εγγραφή ● Η εφαρμογή δημιουργεί και αποθηκεύει μια νέα οντότητα χρήστη στην βάση δεδομένων.

	● Ανακατευθύνει τον χρήστη στην κύρια σελίδα και τον συνδέει στην εφαρμογή με τα στοιχεία του.
Εναλλακτικές Ροές	● Ο χρήστης δεν συμπληρώνει απαραίτητα στοιχεία εμφάνιση μηνύματος για απαραίτητο στοιχείο και του τα δείχνει με κόκκινο πλαίσιο για να τα συμπληρώσει. ● Ο χρήστης συμπληρώνει λάθος στοιχεία. Εμφάνιση κατάλληλου μηνύματος και κόκκινο πλαίσιο στα κατάλληλα στοιχεία για διόρθωση. ● Το email που εισάγει υπάρχει ήδη και τον ανακατευθύνει στην κύρια σελίδα και με pop up δείχνει το κατάλληλο μήνυμα.

Πίνακας 1.2 Περίπτωση χρήσης: Σύνδεση εγγεγραμμένου χρήστη

Περίπτωση χρήσης	Σύνδεση εγγεγραμμένου χρήστη
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	● Εγγραφή Χρήστη ● Σύνδεση στο διαδίκτυο από πρόγραμμα περιήγησης
Ροή Γεγονότων	● Πατάει το κουμπί για σύνδεση στην κύρια σελίδα ● Ανοίγει η pop up φόρμα για εισαγωγή email και κωδικού ● Πατάει σύνδεση ● Συνδέεται στην εφαρμογή και σε pop up του εμφανίζεται μήνυμα καλωσορίσματος και πλέον βλέπει περισσότερες επιλογές στο κεντρικό μενού.
Εναλλακτικές Ροές	● Ο χρήστης δεν συμπληρώνει απαραίτητα στοιχεία εμφάνιση μηνύματος για απαραίτητο στοιχείο και του τα δείχνει με κόκκινο πλαίσιο για να τα συμπληρώσει. ● Ο χρήστης συμπληρώνει λάθος στοιχεία. Εμφάνιση κατάλληλου μηνύματος και κόκκινο πλαίσιο στα κατάλληλα στοιχεία για διόρθωση. ● Το email που εισάγει δεν υπάρχει pop up μήνυμα λάθους . ● Λάθος κωδικός pop up μήνυμα λάθους. ● Πάτημα κουμπιού για να κλείσει το pop up φόρμα σύνδεσης.

Πίνακας 1.3 Περίπτωση χρήσης: Πρόσθεση προϊόντος στο καλάθι

Περίπτωση χρήσης	Πρόσθεση προϊόντος στο καλάθι
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	● Εγγραφή Χρήστη ● Να είναι συνδεδεμένος στην εφαρμογή
Ροή Γεγονότων	● Ανοίγει την σελίδα κατάστημα από το μενού στο πάνω μέρος της σελίδας. ● Σελίδα ανοίγει με τον κατάλογο του καταστήματος με τα προϊόντα, την διαθεσιμότητα και την τιμή. ● Ο χρήστης επιλέγει αριθμό που θέλει για ένα προϊόν. ● Επιλέγει πρόσθεση ● Η εφαρμογή βάζει στο καλάθι του χρήστη το προϊόν με την κατάλληλη ποσότητα και αφαιρείται από διαθεσιμότητα.

Εναλλακτικές Ροές	● Προϊόν δεν είναι πλέον διαθέσιμο και δεν μπορεί ο χρήστης πλέον τον προσθέσει.

Πίνακας 1.4 Περίπτωση χρήσης: Περιήγηση μη εγγεγραμμένου χρήστη

Περίπτωση χρήσης	Περιήγηση μη εγγεγραμμένου χρήστη
Χρήστης	Μη εγγεγραμμένος
Προϋποθέσεις	● Σύνδεση στο διαδίκτυο από πρόγραμμα περιήγησης
Ροή Γεγονότων	● Ανοίγει κύρια σελίδα ● Επιλέγει κατάστημα από το κυρίως μενού ● Βλέπει κατάλογο με όλα τα προϊόντα αλλά δεν μπορεί να προσθέσει τίποτα στο καλάθι.
Εναλλακτικές Ροές	● Περιήγηση στις υπόλοιπες σελίδες της εφαρμογής χωρίς καμία λειτουργικότητα πέρα από το να διαβάζει πληροφορίες σχετικά με την εφαρμογή.

Πίνακας 1.5 Περίπτωση χρήσης: Κατάλογος ενεργών χρηστών για διαχειριστή

Περίπτωση χρήσης	Κατάλογος ενεργών χρηστών για διαχειριστή
Χρήστης	Διαχειριστής
Προϋποθέσεις	● Εγγραφή Χρήστη ● Να είναι διαχειριστής χρήστης ● Να είναι συνδεδεμένος
Ροή Γεγονότων	● Πατάει το κουμπί users από το δευτερεύον μενού ● Ανοίγει σελίδα με όλους του ενεργούς χρήστες ● Πατάει το κουμπί disable , διαγράφει ένα χρήστη και το κουμπί γίνεται enable. ● Ο χρήστης αλλάζει κατάσταση.
Εναλλακτικές Ροές	● Ο διαχειριστής αυτή την φορά πατάει το κουμπί enable σε ένα disable χρήστη. ● Αλλάζει η κατάσταση του χρήστη.

Πίνακας 1.6 Περίπτωση χρήσης: Προβολή προφίλ

Περίπτωση χρήσης	Προβολή προφίλ
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	● Εγγραφή Χρήστη ● Να είναι συνδεδεμένος στην εφαρμογή
Ροή Γεγονότων	● Πατάει το κουμπί από το δευτερεύον μενού με το username του ● Ανοίγει σελίδα με τα στοιχεία του και επίπεδο χρήστη. ● Πατάει κουμπί αλλαγή στοιχείων ● Ανακατευθύνεται σε μια φόρμα εγγραφής για να αλλάξει τα στοιχεία του
Εναλλακτικές Ροές	● Ο χρήστης δεν συμπληρώνει απαραίτητα στοιχεία εμφάνιση μηνύματος για απαραίτητο στοιχείο και του τα δείχνει με κόκκινο πλαίσιο για να τα συμπληρώσει.

	• Ο χρήστης συμπληρώνει λάθος στοιχεία. Εμφάνιση κατάλληλου μηνύματος και κόκκινο πλαίσιο στα κατάλληλα στοιχεία για διόρθωση. • Το email που εισάγει υπάρχει ήδη και τον ανακατευθύνει στην κύρια σελίδα και με pop up δείχνει το κατάλληλο μήνυμα.

Πίνακας 1.7 Περίπτωση χρήσης: Καλάθι αγορών και τοποθέτηση παραγγελίας

Περίπτωση χρήσης	Καλάθι αγορών και τοποθέτηση παραγγελίας
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	• Εγγραφή Χρήστη • Να είναι συνδεδεμένος στην εφαρμογή
Ροή Γεγονότων	• Πατάει το κουμπί cart από το κυρίως μενού ή το δευτερεύον • Ανοίγει η σελίδα με το καλάθι του χρήστη. • Πατάει το κουμπί Proceed Checkout • Τον ανακατευθύνει σε φόρμα για εισαγωγή στοιχείων διεύθυνσης • Συμπληρώνει τα απαραίτητα στοιχεία. • Πατάει το κουμπί Submit • Ανακατευθύνει τον χρήστη σε σελίδα με καρτέλα με τα στοιχεία αποστολής • Πατάει το κουμπί πληρωμή • Ανακατευθύνει τον χρήστη στην σελίδα τρίτης εταιρείας για ασφαλείς πληρωμές. • Κάνει την πληρωμή. • Γίνεται η παραγγελία του χρήστη και ανακατευθύνετε ο χρήστης στην κύρια σελίδα.
Εναλλακτικές Ροές	• Ο χρήστης δεν συμπληρώνει απαραίτητα στοιχεία εμφάνιση μηνύματος για απαραίτητο στοιχείο και του τα δείχνει με κόκκινο πλαίσιο για να τα συμπληρώσει. • Ο χρήστης έχει ήδη στοιχεία αποστολής ανακατευθύνετε αμέσως στην σελίδα πληρωμής και κάρτα αποστολής. • Ο χρήστης έχει ήδη στοιχεία αποστολής ανακατευθύνετε αμέσως στην σελίδα πληρωμής και κάρτα αποστολής. Πατάει το κουμπί αλλαγής διεύθυνσης και ανακατευθύνετε στην φόρμα εισαγωγής αποστολής. • Ο χρήστης δεν κάνει επιτυχημένη πληρωμή ακυρώνεται η παραγγελία παραμένουν τα αντικείμενα στο καλάθι.

Πίνακας 1.8 Περίπτωση χρήσης: Επεξεργασία στο καλάθι αγορών

Περίπτωση χρήσης	Επεξεργασία στο καλάθι αγορών
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	• Εγγραφή Χρήστη • Να είναι συνδεδεμένος στην εφαρμογή • Προϊόντα στο καλάθι αγορών
Ροή Γεγονότων	• Πατάει το κουμπί cart από το κυρίως μενού ή το δευτερεύον • Ανοίγει η σελίδα με το καλάθι του χρήστη.

	• Πατάει το κουμπί X • Διαγράφη το προϊόν από το καλάθι του • Υπολογίζεται εκ νέου η αξία του.
Εναλλακτικές Ροές	• Ο χρήστης πατάει το κουμπί Remove All και αδειάζει το καλάθι του

Πίνακας 1.9 Περίπτωση χρήσης: Περιγραφή προϊόντος

Περίπτωση χρήσης	Περιγραφή προϊόντος
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	• Εγγραφή Χρήστη • Να είναι συνδεδεμένος στην εφαρμογή
Ροή Γεγονότων	• Πατάει το κουμπί store • Ανοίγει ο κατάλογος με όλα τα προϊόντα • Πατάει το σύνδεσμο που είναι το όνομα ενός προϊόντος. • Ανοίγει η σελίδα με την κάρτα του προϊόντος με την αναλυτική του περιγραφή.
Εναλλακτικές Ροές	• Επιλέγει ένα επιθυμητό αριθμό από το μετρητή αποθέματος • Πατάει το κουμπί add to cart. • Το προϊόν προστίθεται στο καλάθι αγορών του.

Πίνακας 1.10 Περίπτωση χρήσης: Έκπτωση μεταφορικών

Περίπτωση χρήσης	Έκπτωση μεταφορικών
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	• Εγγραφή Χρήστη • Να είναι συνδεδεμένος στην εφαρμογή • Να έχει προϊόντα στο καλάθι αγορών
Ροή Γεγονότων	• Πατάει το κουμπί proceed to checkout. • Συμπληρώνει την φόρμα διεύθυνσης. • Χώρα αποστολής είναι στην Ευρωπαϊκή Ένωση. • Τότε στα μεταφορικά γίνεται μια έκπτωση. • Διακρίνεται όταν γίνεται ολοκλήρωση αγοράς ή επιστροφής στο καλάθι αγορών

Πίνακας 1.11 Περίπτωση χρήσης: Αποσύνδεση

Περίπτωση χρήσης	Εγγραφή χρήστη στην εφαρμογή
Χρήστης	Εγγεγραμμένος
Προϋποθέσεις	• Εγγραφή Χρήστη • Να είναι συνδεδεμένος στην εφαρμογή
Ροή Γεγονότων	• Πατάει το κουμπί log out για αποσύνδεση από το δευτερεύον μενού. • Ανακατευθύνει τον χρήστη στην αρχική σελίδα με λιγότερες επιλογές στο μενού και καμιά λειτουργεία πέρα από read only σε κάθε σελίδα της εφαρμογής.
Εναλλακτικές Ροές	• Κλείνει την εφαρμογή. • Ξανά ανοίγει την εφαρμογή παραμένει συνδεδεμένος.

1.5 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε και αναλύθηκε τι είναι το ηλεκτρονικό εμπόριο, πως αυτό επηρεάζει την καθημερινότητα μας, τον ανταγωνισμό στο τομέα του εμπορίου, τους κυριότερους στόχους της εφαρμογής και σε ποιες ομάδες απευθύνεται. Επιπρόσθετα έγινε μια ανάλυση για της απαιτήσεις λογισμικού με περισσότερη έμφαση και ανάλυση στην εμπειρία χρήστη με μια μικρή έρευνα, την ασφάλεια της εφαρμογής και ποια μέτρα λήφθηκαν και τέλος αναλύθηκαν περιπτώσεις χρήστη.

Κεφάλαιο 2ο: Περιγραφή τεχνολογιών

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο της Π.Ε. θα γίνει ανάλυση και αναφορά σε όλες τις τεχνολογίες οι οποίες χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Το Play Framework αποτελεί το κύριο εργαλείο για την υλοποίηση της εφαρμογής όπου δίνει την δυνατότητα δημιουργίας μια διαδικτυακής εφαρμογής χρησιμοποιώντας JVM γλώσσες προγραμματισμού. Σε συνδυασμό με τα κατάλληλα εργαλεία και βιβλιοθήκες συνέδραμαν όλα εξίσου για να ολοκληρωθεί η εφαρμογή και να ικανοποιήσει όλες τις απαιτήσεις λογισμικού που τέθηκαν στην αρχή του προτζεκτ .

2.2 Play Framework

Κάθε ένα από τα πλαίσια ιστού κάνει ουσιαστικά την ίδια δουλειά, κάνει επεξεργασία ενός αιτήματος HTTP. Το πλαίσιο ιστού λαμβάνει τα byte που του παραδίδονται από το λειτουργικό σύστημα, το μετατρέπει στη λογική αναπαράσταση HTTP (GET, POST, κ.λπ.), πραγματοποιεί κάποιες δικές του κλήσεις (σε μια βάση δεδομένων, σε δίσκο ή σε ένα REST API) και επιστρέφει το αποτέλεσμα πίσω στο πρόγραμμα περιήγησης, τυλιγμένο σε μια απόκριση HTTP. Για να γίνει αυτό, το πλαίσιο ιστού πρέπει να χρησιμοποιήσει ένα νήμα, το οποίο περιέχει τις πληροφορίες που απαιτούνται για την εκτέλεση εντολών από έναν πυρήνα CPU[24].

Το Play Framework είναι ένα πλαίσιο για ανάπτυξη εφαρμογών ιστού ανοικτού κώδικα που ακολουθεί το αρχιτεκτονικό σχέδιο μοντέλου-προβολής-ελεγκτή (MVC). Είναι γραμμένο σε Scala και μπορεί να χρησιμοποιηθεί από άλλες γλώσσες προγραμματισμού που έχουν μεταγλωττιστεί στο JVM Bytecode, π.χ. Java και Scala [15]. Στόχος του είναι να βελτιστοποιήσει την παραγωγικότητά του προγραμματιστή χρησιμοποιώντας την γρήγορη επανάληψη φόρτωσης κώδικα και την εμφάνιση σφαλμάτων στο πρόγραμμα περιήγησης. Η υποστήριξη για τη γλώσσα προγραμματισμού Scala έγινε διαθέσιμη από την έκδοση 1.1 του πλαισίου. Ο λόγος που χρησιμοποίησα PF για την ανάπτυξη της εφαρμογής, καθιστά εύκολη τη δημιουργία εφαρμογών ιστού με Scala. Το PF βασίζεται σε μια ελαφριά και φιλική προς ιστό αρχιτεκτονική. Χτισμένο στο Akka platform, το PF παρέχει προβλέψιμη και ελάχιστη κατανάλωση πόρων (CPU, μνήμη, νήματα) για εφαρμογές μεγάλης κλίμακας. Το Akka είναι μια δωρεάν και ανοιχτού κώδικα εργαλειοθήκη που απλοποιεί την κατασκευή παράλληλων και κατανεμημένων εφαρμογών στο JVM [16].

Το Play είναι πλαίσιο πλήρους στοίβας, περιλαμβάνει όλα τα στοιχεία που χρειάζεται κάποιος για τη δημιουργία εφαρμογών ιστού και υπηρεσιών REST. Διαθέτει έναν ενσωματωμένο διακομιστή HTTP, διαχείριση φόρμας, προστασία από πλαστογράφηση αιτημάτων μεταξύ ιστότοπων (CSRF) και έναν ισχυρό μηχανισμό δρομολόγησης. Το PF εξοικονομεί πολύτιμο χρόνο ανάπτυξης υποστηρίζοντας

άμεσα τις καθημερινές εργασίες και τη γρήγορη επαναφόρτιση, ώστε να μπορείς να βλέπεις αμέσως τα αποτελέσματα της εργασίας σου. Μια εφαρμογή Play χρειάζεται μόνο να περιλαμβάνει τα αρχεία Play JAR για να εκτελεστεί σωστά. Αυτά τα αρχεία JAR δημοσιεύονται στο Maven Repository, επομένως μπορείς να χρησιμοποιήσεις οποιοδήποτε εργαλείο δημιουργίας Java ή Scala για να δημιουργήσεις ένα νέο έργο PF [23] .

Εικόνα 2.1: Play Framework

2.2.1 Scala

Η Scala είναι σχεδιασμένη από τον Γερμανό καθηγητή πληροφορικής Martin Odersky, είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται κυρίως για λειτουργικό προγραμματισμό για ισχυρά στατικά συστήματα και εφαρμογές ιστού [17].Είναι αντικειμενοστραφής και τρέχει σε JVM. Έχει τη δυνατότητα να λειτουργεί με ήδη υπάρχοντα κώδικα Java και να κάνει χρήση της βιβλιοθήκες του. Συνοπτικά πολλές από τις αποφάσεις σχεδιασμού της Scala στοχεύουν στην αντιμετώπιση κριτικών της Java και προβλημάτων της. Ένα από αυτά τα προβλήματα και από τα πιο σημαντικά είναι ο πολύ μεγάλος όγκος κώδικα που χρειάζεται για να υλοποιηθεί μια λειτουργία με Java. Έτσι με ένα από τα μεγάλα προτερήματα της για την ανάπτυξη οποιασδήποτε εφαρμογής είναι ‘’does more with less code’’ [18] δηλαδή γράψε πιο μικρό κώδικα κάνοντας περισσότερα πράγματα. Ο λόγος που είναι σημαντικό είναι γιατί όσο πιο μικρός είναι ο κώδικας τότε είναι και πιο εύκολος ο εντοπισμός λαθών και η πραγματοποιήσει δοκιμών. Όλα αυτά με την ίδια δυναμική που παρέχει η γλώσσα Java και με πιο γρήγορη ανάπτυξη κώδικα. Μερικοί προγραμματιστές ισχυρίζονται ακόμη ότι μπορεί να αναπτύξεις μια εφαρμογή σε Scala 20% πιο γρήγορα από ότι σε Java [19].

Για το κομμάτι λοιπών του Back-end για την ανάπτυξη κώδικα χρησιμοποιήθηκε εξολοκλήρου Scala, συμπεριλαμβανομένου και της περισσότερης λειτουργικότητας του Front-end. Το PL σου δίνει την δυνατότητα να αναπτύξεις κώδικα με όλη την δυναμική που η γλώσσα διαθέτει. Περισσότερη ανάλυση επί του θέματος θα γίνει σε παρακάτω κεφάλαιο όπου με παραδείγματα κώδικα και εκτενή ανάλυση θα μελετήσουμε πώς αυτό έγινε εφικτό και πόσο διαφέρει η ανάπτυξη στο Front-end και στο Back-end με ακριβώς την ίδια γλώσσα προγραμματισμού.

Ο λόγος της επιλογής να γίνει ανάπτυξη σε Scala ήταν ο ενθουσιασμός της εξερεύνησης δοκιμάζοντας τα όρια μιας σύγχρονης γλώσσας όπου συνδυάζει αντικειμενοστραφή και λειτουργικό προγραμματισμό παίρνοντάς θετικά στοιχεία και από τις δύο τεχνικές και που παραδοσιακά δεν χρησιμοποιείται για να ανάπτυξη εφαρμογών ιστού για το Front-end κομμάτι. Επίσης τι πιο συναρπαστικό να δημιουργήσεις μια εφαρμογή ιστού με θεωρητικά όχι και τόσο παραδοσιακή γλώσσα ιστού που συνδυάζει όλα αυτά τα στοιχεία .

2.2.2 JavaScript και Front-end

Η JavaScript συχνά αναφέρεται με την συντομογραφία JS, είναι μια γλώσσα προγραμματισμού που αποτελεί μια από τις βασικές τεχνολογίες του παγκόσμιου ιστού, μαζί με την HTML και το CSS. Από το 2022, το 98% των ιστότοπων χρησιμοποιούν JavaScript στην πλευρά του πελάτη για το πώς μια

ιστοσελίδα λειτουργεί, συχνά ενσωματώνοντας τρίτες βιβλιόθήκες για καλύτερη εμπειρία ανάπτυξης ή και απλά γιατί παρέχονται έτοιμα πακέτα με μεγάλη δυναμική. Όλα τα μεγάλα προγράμματα περιήγησης ιστού διαθέτουν ειδική μηχανή JavaScript για την εκτέλεση του κώδικα στις συσκευές των χρηστών [25].

Η JavaScript είναι μια γλώσσα υψηλού επιπέδου, συχνά μεταγλωττισμένη έγκαιρα, η οποία συμμορφώνεται με το πρότυπο ECMAScript. Διαθέτει δυναμική πληκτρολόγηση, αντικειμενικό προσανατολισμό που βασίζεται σε πρωτότυπα και λειτουργίες πρώτης κατηγορίας. Είναι πολλαπλών παραδειγμάτων, υποστηρίζοντας στυλ προγραμματισμού που βασίζονται σε γεγονότα, λειτουργικά και επιτακτικά. Διαθέτει διεπαφές προγραμματισμού εφαρμογών (API) για εργασία με κείμενο, ημερομηνίες, κανονικές εκφράσεις, τυπικές δομές δεδομένων και το document object model (DOM).

Οι μηχανές JavaScript χρησιμοποιούνταν αρχικά μόνο σε προγράμματα περιήγησης ιστού, αλλά τώρα αποτελούν βασικά στοιχεία ορισμένων διακομιστών και μιας ποικιλίας εφαρμογών. Το πιο δημοφιλές σύστημα χρόνου εκτέλεσης για αυτήν τη χρήση είναι το Node.js [25].

Στην εφαρμογή δεν έγινε σε μεγάλο βαθμό χρήση JavaScript πράγμα ιδιαίτερα παράξενο για μια εφαρμογή ιστού αλλά λόγο του πλαισίου PF που σου επιτρέπει να γράφεις κώδικα Scala και στο FE τότε θεώρησα πιο πρέπον και καινοτόμο να κάνω χρήση της εν λόγο γλώσσας που δεν είναι τόσο σύνηθες η ανάπτυξη τέτοιου είδους εφαρμογές. Φυσικά δεν απέκλεισα εντελώς την JavaScript η οποία χρησιμοποιήθηκε σε σημεία κλειδιά της εφαρμογής και γι' αυτό άλλωστε γίνεται αναφορά και ανάλυση της. Σημεία κλειδιά λοιπόν στην εφαρμογή για το Front-end κομμάτι είναι αυτά όπου με JavaScript κάνουν πιο εύκολή η χειραγώγηση του DOM. Για παράδειγμα pop-up φόρμες, animation στο κυρίως μενού και άλλα. Η JavaScript αποτελεί το 4.2% της εφαρμογής και το 50% Scala για Back-end και Front-end μαζί. Η υπόλοιπη εφαρμογή στο Front-end κομμάτι είναι ένας συνδυασμός HTML-CSS.

2.2.3 Responsive εφαρμογή

Στις μέρες μας μια εφαρμογή δεν είναι εφικτό να δημιουργηθεί για ένα μέγεθος οθόνης. Οι απαιτήσεις αυξήθηκαν σημαντικά καθώς η διαφόρων μεγεθών εφαρμογές ολοένα και πληθαίνουν. Έτσι λοιπόν για να αντιμετωπιστεί αυτή η πρόκληση έκανα χρήση της τεχνολογίας Bootstrap.

Το Bootstrap είναι ένα δωρεάν και ανοιχτού κώδικα πλαίσιο. Έχει σχεδιαστεί για να διευκολύνει τη διαδικασία ανάπτυξης ιστότοπων και εφαρμογών ιστού που ανταποκρίνονται, παρέχοντας μια μεγάλη συλλογή σύνταξης για σχέδια προτύπων.

Με άλλα λόγια, το Bootstrap βοηθά τους προγραμματιστές ιστού να αναπτύσσουν ιστοσελίδες πιο γρήγορα, καθώς δεν χρειάζεται να ανησυχούν για βασικές εντολές και λειτουργίες. Αποτελείται από σενάρια που βασίζονται σε HTML, CSS και JS για διάφορε λειτουργίες και στοιχεία που σχετίζονται με το σχεδιασμό εφαρμογών ιστού. Ο πρωταρχικός στόχος του Bootstrap είναι να δημιουργήσει ιστοτόπους που να ανταποκρίνονται σε κινητά. Διασφαλίζει ότι όλα τα στοιχεία διεπαφής ενός ιστότοπου λειτουργούν βέλτιστα σε όλα τα μεγέθη οθόνης χωρίς να χάνεται η λειτουργικότητα της [26].

Το CSS μαζί με HTML αποτελούν το 45.8% της εφαρμογής με χρήση των βιβλιοθηκών του Bootstrap σε μεγάλο βαθμό. Φυσικά χρειάστηκαν και αρκετά προσαρμοσμένα CSS να αναπτυχθούν για να βγει ένα ολοκληρωμένο και όμορφο αποτέλεσμα. Σε κύρια σημεία όμως των Views της εφαρμογής, όπως το body έγινε χρήση έτοιμών κλάσεων από το Bootstrap.

2.3 Flyway

Το Flyway είναι ένα εργαλείο ανοιχτού κώδικα, με άδεια χρήσης Apache License 2.0, που βοηθά να υλοποιηθούν αυτοματοποιημένες και βασισμένες σε έκδοση μετεγκαταστάσεις βάσεων δεδομένων. Επιτρέπει να ορίσει ο προγραμματιστής τις απαιτούμενες λειτουργίες ενημέρωσης σε μια δέσμη ενεργειών SQL ή ως κώδικα Java [21].

Το καλό με αυτή τη διαδικασία είναι ότι το Flyway εντοπίζει τις απαιτούμενες λειτουργίες ενημέρωσης και τις εκτελεί. Επομένως, δεν χρειάζεται να γνωρίζετε ποιες δηλώσεις ενημέρωσης SQL πρέπει να εκτελεστούν για να ενημερώσετε την τρέχουσα βάση δεδομένων σας. Εσείς και οι συνάδελφοί σας απλώς ορίζετε τις λειτουργίες ενημέρωσης για τη μετεγκατάσταση της βάσης δεδομένων από τη μια έκδοση στην άλλη. Και το Flyway εντοπίζει την τρέχουσα έκδοση και εκτελεί τις απαραίτητες λειτουργίες ενημέρωσης για να μεταφέρει τη βάση δεδομένων στην πιο πρόσφατη έκδοση.

Ο κυριότερος στόχος που έκανα χρήση του εργαλείου Flyway είναι η ασφάλεια της βάσης μου καθώς κάθε φορά που έκανα μια αλλαγή έβαζα εκδόσεις και την ανέβαζα στο GitHub γνωρίζοντας έτσι πως όλες μου οι αλλαγές μαζί με την βάση δεδομένων παρέμεναν ασφαλείς. Επιπρόσθετα μου παρέχει την δυνατότητα από οποιοδήποτε υπολογιστή που έχει πρόσβαση στο διαδίκτυο να δημιουργήσω ξανά την βάση μου κρατώντας μαζί και όλες τις εκδόσεις της χωρίς να έχει χαθεί το οτιδήποτε, μπορώντας έτσι να διακρίνουμε και την εξέλιξη της.

2.3.1 PostgreSQL

Στη σύγχρονη τεχνολογική κοινωνία, η ακεραιότητα και η ασφάλεια των δεδομένων αποτελούν κορυφαία προτεραιότητα. Ειδικά, αν μιλάμε για μεγάλα έργα, τα οποία απαιτούν τη διατήρηση εκατοντάδων ή χιλιάδων μπλοκ πληροφοριών σε ένα ασφαλές μέρος [20].

Έτσι, η επιλογή μιας σωστής βάσης δεδομένων για το επόμενο έργο σας δεν είναι κάτι περιθωριακό. Απαιτεί να γνωρίζετε όλες τις επιλογές, τα πλεονεκτήματα και τα μειονεκτήματά τους και να κατανοήσετε ποιες από αυτές είναι οι καταλληλότερες για εσάς.

Είναι ένα εξαιρετικά σταθερό σύστημα διαχείρισης βάσεων δεδομένων, το οποίο υποστηρίζεται από περισσότερα από 20 χρόνια κοινοτικής ανάπτυξης που έχει συμβάλει στα υψηλά επίπεδα ανθεκτικότητας, ακεραιότητας και ορθότητας. Η PostgreSQL χρησιμοποιείται ως η κύρια αποθήκευση δεδομένων ή η αποθήκη δεδομένων για πολλές εφαρμογές ιστού, κινητών και αναλυτικών στοιχείων.

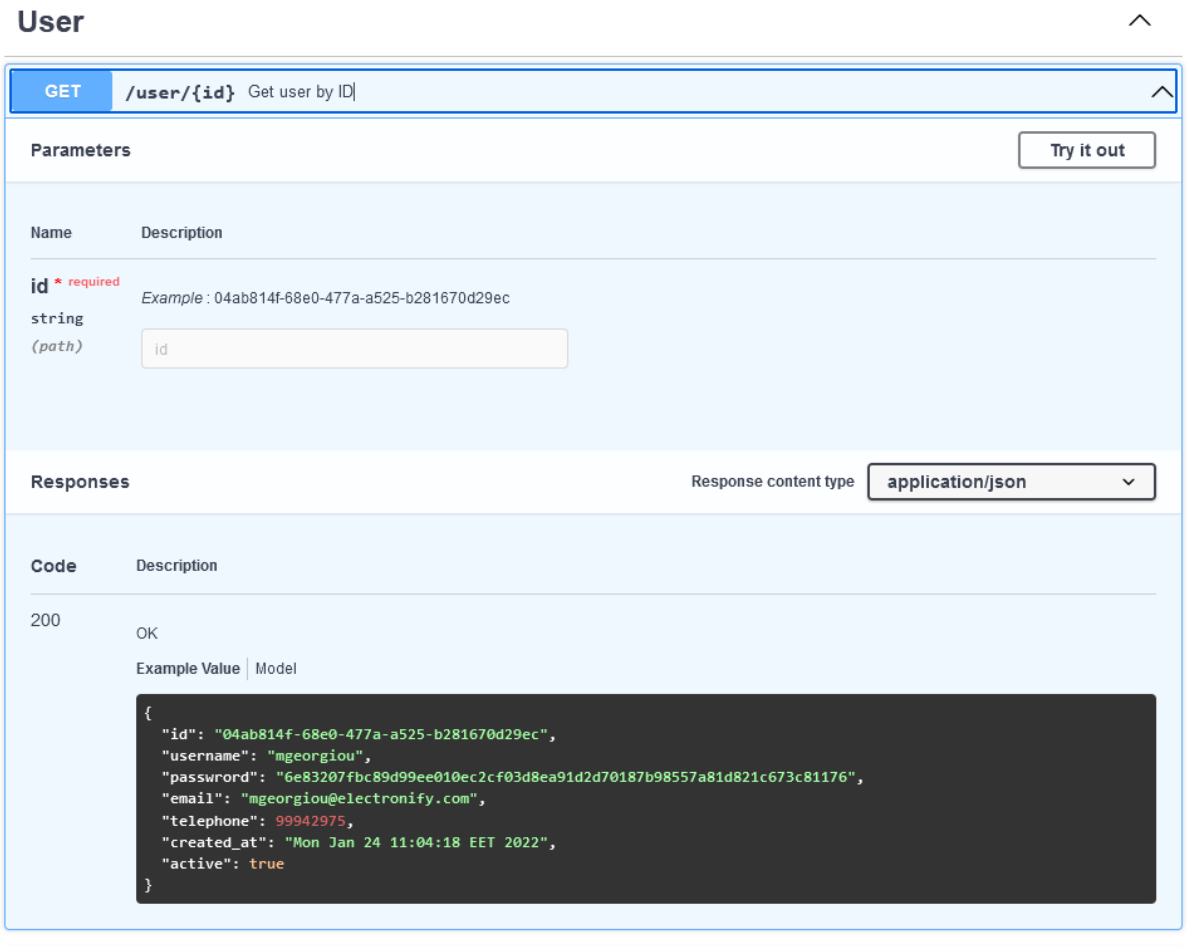
2.4 Swagger

Το Swagger επιτρέπει την περιγραφή της δομής των API , ώστε τα μηχανήματα να μπορούν να τα διαβάσουν. Η ικανότητα των API να περιγράφουν τη δική τους δομή είναι η ρίζα κάθε δυναμικής στο Swagger. Διαβάζοντας τη δομή του API, δίνει την δυνατότητα να δημιουργηθεί αυτόματα όμορφη και δια δραστική τεκμηρίωση API. Μπορεί επίσης να δημιουργήσει ο προγραμματιστής αυτόματες βιβλιοθήκες πελατών για το API σε πολλές γλώσσες και να εξερευνήσει άλλες δυνατότητες όπως η αυτοματοποιημένη δοκιμή. Το Swagger το κάνει αυτό ζητώντας από το API να επιστρέψει ένα YAML ή JSON που περιέχει μια λεπτομερή περιγραφή ολόκληρου του API. Αυτό το αρχείο είναι ουσιαστικά μια λίστα πόρων του API που συμμορφώνεται με την προδιαγραφή OpenAPI. Η προδιαγραφή ζητά να συμπεριλαμβάνονται οι πληροφορίες όπως:

1. Ποιες είναι όλες οι λειτουργίες που υποστηρίζει το API ;
2. Ποιες είναι οι παράμετροι του API και τι επιστρέφει;

3. Το API χρειάζεται κάποια εξουσιοδότηση; (authorization)
4. Και ακόμη όροι, στοιχεία επικοινωνίας και άδεια χρήσης του API.

Μπορείς να γράψεις μια προδιαγραφή Swagger για το API με μη αυτόματο τρόπο ή να δημιουργήσεις αυτόματα από σχολιασμούς στον πηγαίο σου κώδικα. Πλέον υπάρχουν και βιβλιοθήκες που σου επιτρέπει να αναπτύξεις κώδικα οποίος μεταγλωττίζεται σε Yaml για πιο γρήγορη ανάπτυξη και στον εντοπισμό πιο γρήγορα και εύκολα λαθών.



Εικόνα 2.2: Get user by id swagger

2.5 Επίλογος

Στο παραπάνω κεφάλαιο έγινε αναφορά και ανάλυση στις κυριότερες τεχνολογίες και γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την ανάπτυξη και υλοποίηση της εφαρμογής, καθώς και γιατί επιλέχθηκαν. Στο επόμενο κεφάλαιο θα αναλυθεί περαιτέρω που ακριβώς έγινε η χρήση τους και ποια προβλήματα επίλυσαν.

Κεφάλαιο 3ο: Υλοποίηση της εφαρμογής

3.1 Εισαγωγή

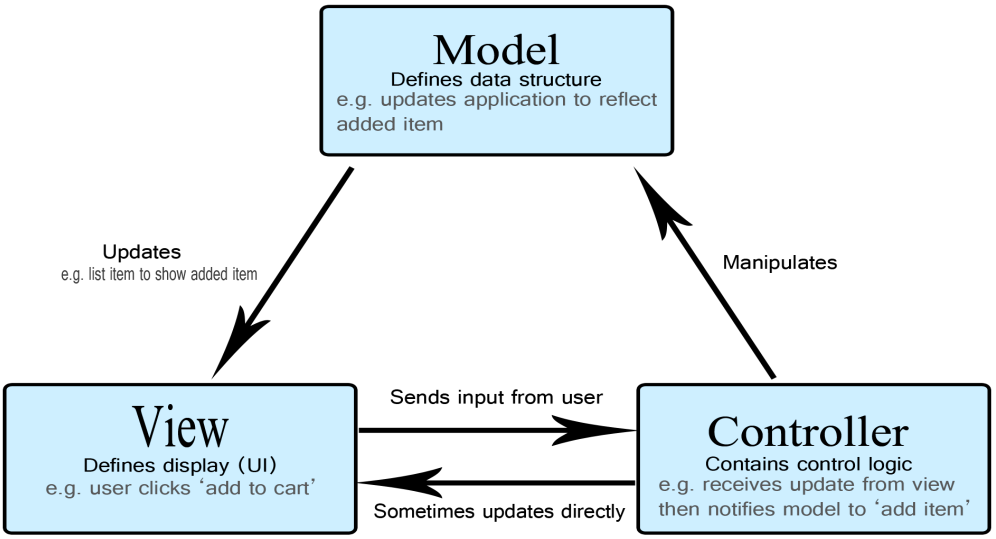
Σε αυτό το κεφάλαιο θα γίνει εκτενείς ανάλυση για το πως η εφαρμογή υλοποιήθηκε με παραδείγματα κώδικα και ποιες τεχνικές χρησιμοποιήθηκαν . Επιπρόσθετα θα γίνει αναφορά σε προκλήσεις οι

οποίες παρουσιάστηκαν κατά την διάρκεια της ανάπτυξης της εφαρμογής και πως αυτό επηρέασε στο να προστεθούν επιπλέον εργαλεία για την ολοκλήρωση της εφαρμογής.

3.2 Διαμορφώσεις και αρχιτεκτονική

Στην ενότητα αυτή θα να γίνει αναλύσει για τις κυριότερες διαμορφώσεις που χρειάστηκαν να οριστούν για να ξεκινήσει η ανάπτυξη της εφαρμογής αλλά και την αρχιτεκτονική η οποία ακολουθήθηκε σύμφωνα με το πλαίσιο PF (MVC). Το MVC είναι ένα αρχιτεκτονικό μοτίβο που αποτελείται από τρία μέρη: Μοντέλο, Προβολή, Ελεγκτής.

1. Μοντέλο: Χειρίζεται τη λογική δεδομένων.
2. Προβολή: Εμφανίζει τις πληροφορίες από το μοντέλο στον χρήστη.
3. Ελεγκτής: Ελέγχει τη ροή δεδομένων σε ένα αντικείμενο μοντέλου και ενημερώνει την προβολή κάθε φορά που αλλάζουν τα δεδομένα



Εικόνα 3.1: MVC

3.2.1 Configurations

Για αρχή για την δημιουργία της εφαρμογής μας πρέπει να υλοποιήσουμε το sbt αρχείο. Το sbt είναι ένα εργαλείο δημιουργίας ανοιχτού κώδικα για έργα Scala και Java. Τα κύρια χαρακτηριστικά του είναι: Εγγενής υποστήριξη για τη σύνταξη κώδικα Scala και την ενοποίηση της με πολλά πλαίσια. Σε αυτό το αρχείο sbt γίνεται το ‘’χτίσιμο’’ της εφαρμογής μας , εκεί καθορίζονται όλες η βιβλιοθήκες όποιες χρειάστηκαν για την ανάπτυξη της εφαρμογής μαζί με τις πηγές τους και ποια έκδοση θα χρησιμοποιηθεί για κάθε τεχνολογία .

Στο built.sbt αρχείο της εφαρμογής δηλώθηκαν οι παρακάτω τεχνολογίες :

1. Play Framework 2.6.10
2. Scala 2.13.3
3. PostgreSQL 42.3.3
4. Play Slick 5.0.0
5. Scala Test Plus Play 5.0.0

```
name := """Electronify"""
organization := "com.electronics"

version := "1.0-SNAPSHOT"

lazy val root = (project in file(".")).enablePlugins(PlayScala)

scalaVersion := "2.13.3"
libraryDependencies += guice
libraryDependencies += "org.scalatestplus.play" %% "scalatestplus-play" %
"5.0.0" % Test
libraryDependencies += jdbc
libraryDependencies ++= Seq(
  "org.postgresql" % "postgresql" % "42.3.3",
  "com.typesafe.play" %% "play-slick" % "5.0.0",
  "com.github.tototoshi" %% "slick-joda-mapper" % "2.5.0"
)
libraryDependencies += ws
libraryDependencies ++= Seq(
  "org.playframework.anorm" %% "anorm-postgres" % "2.6.10"
)
libraryDependencies += ehcache
```

Κώδικας 3.1: built.sbt

Μετά την ολοκλήρωση του built αρχείου σειρά είχε το application.conf. Σε αυτό το αρχείο λοιπόν έγινε δήλωση των διαπιστευτηρίων της βάσης δεδομένων για σύνδεση της εφαρμογής με αυτήν. Επιπρόσθετα απενεργοποιήθηκαν default ρυθμίσεις του PF που δεν χρειαζόμασταν και έγινε δήλωση του χειριστή λαθών της εφαρμογής όπου χρησιμοποιήθηκε σαν logger factory για την εφαρμογή για πιο εύκολο debugging.

```
slick.dbs.default {
  connectionPool = "HikariCP"
  profile = "slick.jdbc.PostgresProfile$"
  db {
    driver = "org.postgresql.Driver"
    url="jdbc:postgresql://localhost:5432/postgres"
    serverName = "localhost"
    databaseName = "postgres"
    user="postgres"
    port="5432"
    password="postgres"
  }
}

play.http.errorHandler = "AppErrorHandler"
play.filters {
 # Disabled filters remove elements from the enabled list.
  disabled += play.filters.csrf.CSRFFilter
}

db.default.logSql=true
play.ws.cache.enabled=true
play.ws.cache.heuristics.enabled=true
```

Κώδικας 3.2: application.conf

3.2.2 Routes

Το αρχείο routes είναι το αρχείο διαμόρφωσης που χρησιμοποιείται από το Router. Αυτό το αρχείο παραθέτει όλες τις διαδρομές που χρειάζεται η εφαρμογή. Κάθε διαδρομή ξεκινά με τη μέθοδο HTTP, ακολουθούμενη από το μοτίβο URI που σχετίζεται με μια οντότητα της εφαρμογής. Μπορεί να προσθέσεις ένα σχόλιο στο αρχείο διαδρομής, με τον χαρακτήρα "#".

Στην εφαρμογή για κάθε ενέργεια που κάνει ο χρήστης, γίνεται η κλήση μιας διαδρομής. Κάθε φορά λοιπόν που θα γίνεται μια ενέργεια από τον χρήστη στην εφαρμογή τότε το ανάλογο API θα καλείται και θα εκτελεί τις ενέργειες οποίες έχουν υλοποιηθεί στο BE κομμάτι. Το κάθε κάλεσμα είναι μοναδικό και κάνει μια ενέργεια για την εφαρμογή. Φυσικά εξυπακούεται πως για διαφορετικές ενέργειες μπορεί να καλείται το ίδιο API εφόσον οι συγκεκριμένες ενέργειες θέλουμε να έχουν το ίδιο αποτέλεσμα. Παραδείγματος χάρει πολύ καλό παράδειγμα αυτής της χρήσης είναι όταν ο χρήστης πατήσει στο κυρίως μενού το κουμπί Home ή το λογότυπο της ε0 ταιρείας να το παίρνει στην αρχική σελίδα της εφαρμογής, ένα άλλο παράδειγμα είναι όταν ο χρηστής θέλει να προσθέσει ένα προϊόν στο καλάθι του τότε από οποιοδήποτε σημείο της εφαρμογής μας έχει κουμπί add to cart τότε να καλείται το ίδιο API με διαφορετικές παραμέτρους αναλόγως από που έχει γίνει η κλήση.

```
#User
GET         /user
controllers.UserController.getUserActions(id: String, action: String)
POST        /user
controllers.UserController.userPostAction(id: String ,action: String)
```

Κώδικας 3.3: Api παραδείγματα

Για κάθε οντότητα στην εφαρμογή υλοποιήθηκαν δύο διαδρομές GET και POST. Ανάλογα με τις παραμέτρους που στέλνονται από το FE γίνονται και οι κατάλληλες ενέργειες. Σε κάθε διαδρομή της εφαρμογής στέλνεται και μια παράμετρος action τύπου String όπου με το κατάλληλο validation το BE γνωρίζει τι ενέργεια θα κάνει κάθε φορά για την συγκεκριμένη οντότητα. Το actions τα οποία στέλνει το FE είναι συγκεκριμένα και δημιουργήθηκαν υπό την μορφή enumeration παρέχοντας id τύπου Int και value τύπου String, παρέχοντας στις σταθερές ένα type safe τρόπο να στέλνει το FE στα api call και στο BE εύκολο και ασφαλή τρόπο ελέγχου για τα actions .

```
object ActionsUser extends Enumeration {
  type ActionsUser = Value
  val UserById = "UserById"
  val ActiveUsers = "ActiveUsers"
  val LogOut = "LogOut"
  val SignIn = "SignIn"
  val SignUp = "SignUp"
  val Disable = "Disable"
  val Checkout = "CheckOut"
  val Update = "Update"
  val CreateAddress = "CreateAddress"
  val Payment = "Payment"
  val ChangeAddress = "ChangeAddress"
}
```

Κώδικας 3.4: User enumeration

3.2.3 Controllers

Τα controllers είναι το πιο σημαντικό μέρος οποιασδήποτε εφαρμογής Play. Μια εφαρμογή Play αναπτυγμένη σε Scala μοιράζεται τις ίδιες έννοιες με μια κλασική εφαρμογή Play, αλλά χρησιμοποιεί έναν πιο λειτουργικό τρόπο για να περιγράφει ενέργειες. Ο controller είναι ένα αντικείμενο Scala singleton, το όποιο αναπτύσσετε στον φάκελο εκλεκτών. Μπορούμε να δηλώσουμε όσους ελεκτές θέλουμε σε ένα φάκελο. Το singleton αντικείμενο στην Scala είναι αντικείμενο που δηλώνεται χρησιμοποιώντας λέξη-κλειδί αντικειμένου αντί από μια κλάση. Δεν απαιτείται αντικείμενο για την κλήση μεθόδων που δηλώνονται μέσα σε ένα αντικείμενο singleton. Στην Scala, δεν υπάρχει στατική έννοια. Έτσι, η Scala δημιουργεί ένα αντικείμενο singleton για να παρέχει σημείο εισόδου για την εκτέλεση του προγράμματός σας.

Στον φάκελο controllers είναι όλα τα αρχεία που έχουν τον ρόλο του controller οποίος είναι να καλείται όταν ένα API καλεί μια από τις μεθόδους τους. Ο ρόλος του είναι να μετατρέψει το κάλεσμα σε κώδικα και να επιστρέψει την ανταπόκριση που χρειάζεται πίσω το API. Με άλλα λόγια χειρίζεται το request που έγινε από το API και θα επιστρέψει ένα response στο FE για να το διαχειριστεί αναλόγως. Αυτό μπορεί είτε να είναι να επιστρέψει δεδομένα , να κάνει επεξεργασία δεδομένων ή ακόμη και να εμφανίζει τις σελίδες της εφαρμογής μας. Ανεξαιρέτως από το ποια θα είναι η ενέργεια η όποια θα πραγματοποιηθεί πάντα θα επιστρέφεται ένα μια ανταπόκριση είτε επιτυχίας είτε αποτυχίας.

Στην εφαρμογή κάθε οντότητα είχε τον δικό της εκλεκτή και ο λόγος είναι για να γίνει πιο καλή διαχείριση των καλεσμάτων, έτσι κάθε εκλεκτής να χειρίζεται μια κατηγορία καλεσμάτων για κάθε οντότητα. Αυτό μπορεί να βοηθήσει σε μελλοντικό εμπλουτισμό της εφαρμογής και ανάπτυξη της γιατί τώρα βλέπουμε ένα πολύ μικρό κομμάτι της δυναμικής που μπορεί να αποκτήσει η συγκεκριμένη εφαρμογή. Όσο περισσότερο αναπτύσσεται η εφαρμογή και εμπλουτίζεται με περισσότερες λειτουργίες τότε θα την καθιστά και πιο δύσκολη στην συντήρηση και αναβάθμιση έτσι μια καλή στρατηγική ακόμη και στον διαχωρισμό των εκλεκτών μπορεί να παίξει σημαντικό ρόλο στην βιωσιμότητα της εφαρμογής μακροπρόθεσμα.

```
@Singleton
class UserController @Inject() (cc: ControllerComponents, service:
UserService)(
    implicit ec: ExecutionContext
) extends AbstractController(cc) {

  def userPostAction(id: String = "", action: String): Action[AnyContent] =
    Action { implicit request: Request[AnyContent] =>
      action match {
        case SignIn  => signIn(logInForm.bindFromRequest().value, request)
        case SignUp  => signUp(userForm.bindFromRequest().value, request)
        case Disable => disableUser(extractUUID(request.body.asJson))
        case CreateAddress =>
          createAddress(id, checkoutForm.bindFromRequest().value, request)
        case ChangeAddress =>
          changeAddress(id, deleteAddress.bindFromRequest().value, request)
        case Payment =>
          createPaymentMethod(id)
        case Update =>
          // change user details
          Ok(views.html.index())
            .withSession(Global.SESSION_USERNAME_KEY -> "")
        case _ =>
          BadRequest(views.html.index())
            .withSession(Global.SESSION_USERNAME_KEY -> "")
      }
    }
```

Κώδικας 3.5: User controller και μέθοδος χειρισμού για POST ενέργειες

3.2.4 Views

Μια προβολή, είναι μια κλάση λογισμικού που περιέχει ένα πρότυπο και μια φόρμα δεδομένων και παράγει μια απόκριση για το πρόγραμμα περιήγησης. Λαμβάνει δεδομένα από τον ελεγκτή του MVC και τα συσκευάζει και τα παρουσιάζει στο πρόγραμμα περιήγησης για εμφάνιση.

Στην εφαρμογή όλα τα views αναπτύχθηκαν κάτω από τον φάκελο views και είναι όλα τύπου scala.html, αυτά λοιπόν είναι πρότυπα play scala τα οποία είναι αρχεία απλού κειμένου που περιέχουν μικρά μπλοκ κώδικα scala. Τα πρότυπα μπορούν να δημιουργήσουν οποιαδήποτε μορφή που βασίζεται σε κείμενο, όπως HTML, XML ή CSV. Το σύστημα προτύπων έχει σχεδιαστεί για να αισθάνονται άνετα σε όσους έχουν συνηθίσει να εργάζονται με HTML, επιτρέποντας στους προγραμματιστές FE να εργάζονται εύκολα με τα πρότυπα. Τα πρότυπα μεταγλωττίζονται ως τυπικές συναρτήσεις Scala, ακολουθώντας μια απλή σύμβαση ονομασίας. Ένα πρότυπο είναι σαν μια συνάρτηση, οπότε χρειάζεται παραμέτρους, οι οποίες πρέπει να δηλωθούν στην αρχή του αρχείου προτύπου.

```
@(address: UserAddress)(implicit session: Session)
```

Κώδικας 3.6: Παράμετροι του payment.scala.html

Εξ ορισμού, ένα πρότυπο δημιουργείται ως στατική συνάρτηση που μπορεί να κληθεί σε οποιοδήποτε πλαίσιο. Αν το πρότυπό σας έχει εξαρτήσεις από συστατικά, όπως τα messages του API, μπορεί να σας είναι πιο εύκολο να το ενσωματώσετε με τα στοιχεία που απαιτούνται (και άλλα πρότυπα) και στη συνέχεια να μπορείτε να ενσωματώσετε το πρότυπο αυτό στους controllers που το χρησιμοποιούν.

Το πρότυπο Scala χρησιμοποιεί το @ ως τον μοναδικό ειδικό χαρακτήρα. Κάθε φορά που συναντάται αυτός ο χαρακτήρας, υποδεικνύει την αρχή μιας δυναμικής δήλωσης. Δεν απαιτείται να κλείσει ρητά το μπλοκ κώδικα εάν είναι μια απλή έκφραση, το τέλος της δυναμικής δήλωσης θα συνάγεται από τον κώδικά.

```
<span class="input-text qty text">Max: @product.stock</span>
```

Κώδικας 3.7 HTML tag με απλή δήλωση Scala

Επειδή η μηχανή προτύπων εντοπίζει αυτόματα το τέλος του μπλοκ κώδικα αναλύοντας τον κώδικά σας, αυτή η σύνταξη υποστηρίζει μόνο απλές δηλώσεις. Αν θέλετε να εισάγετε μια δήλωση με πολλά στοιχεία, γίνεται το @ και ανοίγοντας αγκύλες και κλείνοντας αγκύλες.

```
<ul>
@for(p <- products) {

 <li>@p.name ($@p.price)</li>
}
</ul>
```

Κώδικας 3.8: HTML με πιο σύνθετη δήλωση Scala

```
<form
action="@routes.CartController.postCart(session.get(SESSION_ID).getOrElse("
00000000-0000-0000-0000-000000000000"), product.id, ActionsCart.AddCart)"
method="post" class="form-container">
    <div class="product-carousel-price">
        <ins class="price-tag">&euro;
            @scaleDouble(product.price)</ins>
    </div>
    @if(product.inStock) {
        <div class="quantity buttons_added">
            <input type="number" size="3" class="input-text qty text"
title="Qty" value="1" min="0" max="@product.stock" step="1"
name="quantity">
            <span class="input-text qty text">Max: @product.stock</span>
        </div>
        <div class="product-option-shop">
        @if(session.get(SESSION_ID).isEmpty) {
            <button class="add_to_cart_button" title="Log In" type="submit"
disabled >
                <del>Add to cart</del></button>
        } else {
            <button class="add_to_cart_button" title="Add To Cart"
type="submit">
                Add to cart</button>
        }
        </div>
    } else {
        <div class="quantity buttons_added">
            <input disabled type="number" size="3" class="input-text qty
text" title="Qty" value="0" min="0" max="@product.stock" step="1"
name="quantity">
        </div>
        <div class="product-option-shop">
            <button class="add_to_cart_button out-of-stock" disabled
type="submit" >
                Out of Stock</button>
        </div>
    }
</form>
```

Κώδικας 3.9: Form από shop.scala.html με κώδικα Scala

3.2.5 Model

Το μοντέλο κατέχει κεντρική θέση σε μια εφαρμογή Play. Είναι η αντιπροσώπευση της πληροφορίας που σχετίζεται με τον τομέα πάνω στον οποίο λειτουργεί η εφαρμογή.
Ο Martin Fowler το ορίζει ως εξής:
‘’Υπεύθυνο για την αναπαράσταση των εννοιών της επιχείρησης, των πληροφοριών σχετικά με την κατάσταση της επιχείρησης και των επιχειρηματικών κανόνων. Η κατάσταση που αντικατοπτρίζει την επιχειρηματική κατάσταση ελέγχεται και χρησιμοποιείται εκεί, παρόλο που οι τεχνικές λεπτομέρειες της αποθήκευσης ανατίθενται στην υποδομή. Αυτό το επίπεδο είναι η καρδιά του επιχειρηματικού λογισμικού’’ [23].
Στον φάκελο models λοιπόν δημιουργήθηκαν δύο άλλοι φάκελοι οι οποίοι αντικατοπτρίζουν μια κατηγορία μοντέλου στην εφαρμογή.

1. Raw: είναι τα μοντέλα από την εισαγωγή του χρήστη δηλαδή η μη επεξεργασμένες πληροφορίες που έρχονται από το FE από κλήση ενός API. Τα μοντέλα με την χρήση χαρτών (Map) που δημιουργήθηκαν σύμφωνα με το JSON που περιμένουμε από το FE θα εξάγουν τις

πληροφορίες και θα τις μετατρέψουν σε Scala case class έτσι ώστε να μπορέσουμε να επεξεργαστούμε τις πληροφορίες.

```
case class RawUser(
    username: String,
    password: String,
    passwordR: String,
    email: String,
    telephone: Int
)

object RawUser {
  implicit val format : Format[RawUser] = Json.format[RawUser]
}
```

Κώδικας 3.10: RawUser case class για πληροφορίες εγγραφής και companion object

Το companion object στη Scala είναι ένα αντικείμενο που δηλώνεται στο ίδιο αρχείο με μια κλάση και έχει το ίδιο όνομα με την κλάση. Τα οφέλη αυτού του γεγονότος είναι πολλά. Πρώτον, ένα συνοδευτικό αντικείμενο και η κλάση του μπορούν να έχουν πρόσβαση στα ιδιωτικά μέλη (πεδία και μεθόδους) του άλλου. Σε ορισμένα παραδείγματα αυτής της Π.Ε. θα παρατηρήσατε ότι μπορείτε να δημιουργήσετε νέες περιπτώσεις ορισμένων κλάσεων χωρίς να χρειάζεται να χρησιμοποιήσετε τη λέξη κλειδί new πριν από το όνομα της κλάσης, όπως σε αυτό το παράδειγμα.

```
case class UserCart(
    id: String,
    products: Products,
    total: Double,
    shipping: Double = 19.99
)

object UserCart extends Logger {
  implicit val format: OFormat[UserCart] = Json.format[UserCart]
}


val userCart: UserCart =
UserCart(
  userId,
  Products(userProducts),
  total,
  shipping = total * 0.05
)
```

Κώδικας 3.11: User class και δημιουργία instance της κλάσης

Η λειτουργικότητα αυτή προέρχεται από τη χρήση συνοδευτικών αντικειμένων. Συμβαίνει ότι όταν ορίζετε μια μέθοδο apply σε ένα companion object, έχει μια ειδική σημασία για τον μεταγλωττιστή της Scala. Υπάρχει μια μικρή συντακτική ζάχαρη ενσωματωμένη στη Scala που σας επιτρέπει να πληκτρολογήσετε αυτόν τον κώδικα. Μπορείτε να δημιουργήσετε ακόμη και διαφορετικές apply μεθόδους που να επιστρέφουν αυτό το αντικείμενο αλλάζοντας τις παραμέτρους ικανοποιώντας όμως όλα τα πεδία της κλάσης. Η μέθοδος apply στο αντικείμενο companion ενεργεί ως μέθοδος Factory

και η συντακτική ζάχαρη της Scala σας επιτρέπει να χρησιμοποιήσετε τη σύνταξη που φαίνεται παρακάτω, δημιουργώντας νέες περιπτώσεις κλάσεων χωρίς να χρησιμοποιήσετε τη λέξη κλειδί new.

```
case class Products(products: Seq[Product])

object Products extends Logger {
  implicit val format: Format[Products] = Json.format

def apply(products: Vector[Seq[Product]]): Products = {
  Products(products.flatten.sortWith((a, b) => a.stock >= b.stock))
 }
}


val products: Vector[Seq[Product]] = getFutureValue(db.run(query))

Products(products) // new Instance of Products
```

Κώδικας 3.12: Products apply customize method

```
val userForm: Form[RawUser] = Form {
  mapping(
    "username" -> text,
    "password" -> text,
    "passwordR" -> text,
    "email" -> email,
    "telephone" -> number
  )(RawUser.apply)(RawUser.unapply)
}
```

Κώδικας 3.13: Mapping για την εξαγωγή των δεδομένων από το JSON

2. DB: σε αυτό τον φάκελο ανήκουν όλα τα μοντέλα τα οποία είτε δημιουργήθηκαν χρησιμοποιεί η εφαρμογή για επεξεργασία, λογική επικύρωση (conditions, validation), να διαβάσει και να γράψει στην βάση δεδομένων.

```
implicit val getUserResult: AnyRef with GetResult[User] =
  GetResult(result =>
    User(
      Some(uuidMapping(result).nextUUID),
      result.nextString(),
      result.nextString(),
      result.nextString(),
      result.nextInt(),
      result.nextString(),
      result.nextBoolean(),
      result.nextInt(),
      result.nextString()
    )
  )
```

Κώδικας 3.14: Implicit μεταβλητή στο User object για τα δεδομένα που διάβασε από την βάση

Μια συνάρτηση μπορεί να έχει έμμεσες παραμέτρους (implicit) στη Scala, οι οποίες επισημαίνονται από τη λέξη κλειδί implicit στην αρχή της λίστας παραμέτρων. Αν η παράμετρος δεν έχει περάσει, ως συνήθως, ο μεταγλωττιστής αναζητά μια σιωπηρή τιμή του σωστού τύπου και την περνάει ως όρισμα της συνάρτησης. Το έμμεσο σύστημα της Scala επιτρέπει στον μεταγλωττιστή να προσαρμόζει τον κώδικα χρησιμοποιώντας έναν καλά καθορισμένο μηχανισμό αναζήτησης. Ένας προγραμματιστής στη Scala μπορεί να παραλείψει πληροφορίες τις οποίες ο μεταγλωττιστής θα προσπαθήσει να συμπεράνει κατά τη μεταγλώττιση. Ο μεταγλωττιστής της Scala μπορεί να συμπεράνει μία από τις εξής δύο καταστάσεις: Μια κλήση μεθόδου ή ενός κατασκευαστή με μια παράμετρο που λείπει [1] .

Επίσης στον φάκελο models έχουμε δύο ακόμα φακέλους όπου ο ένας περιέχει όλες τις απαριθμήσεις που δημιουργήθηκαν για τις ανάγκες της εφαρμογής και ο άλλος περιέχει όλες τις μεταβλητές αντιστοίχισης για την εξαγωγή δεδομένων από το JSON που αποστέλλει ο FE στον BE.

Τέλος, έχουμε ένα trait Logger factory, το οποίο είναι αυτό που χρησιμοποιούμε για τα αρχεία καταγραφής (logs) σε όλη την εφαρμογή για πιο εύκολο και πρακτικό τρόπο εντοπισμού σφαλμάτων ή για να δείξουμε κάποιες πληροφορίες στην κονσόλα μας.

3.3 Services

Τα services είναι ένας από τους πιο σημαντικούς καταλόγους της εφαρμογής καθώς περιέχει όλους τους client οι οποίοι συνδέονται στην βάση μας και για να διαβάσουν ή να γράψουν. Για κάθε controller που διαθέτει η εφαρμογή έχουμε και των αντίστοιχο client οποίος γίνεται @Inject στον controller ορίζοντας έτσι την εξάρτηση του στον συγκεκριμένο client, παρέχοντας έτσι την δυνατότητα στον controller να έχει πρόσβαση σε όλες της κατάλληλες μεθόδους του client.

```
class CartController @Inject() (cc: ControllerComponents, service:
CartService)(
    implicit ec: ExecutionContext
) extends AbstractController(cc) { … }
```

Κώδικας 3.15: CartController injecting CartService (database client)

Τα services με την σειρά τους κάνουν inject τα κατάλληλα component από libraries για να μπορέσουν όχι μόνο να συνδεθούν στην βάση αλλά και να για να μπορούν να αλληλοεπιδρούν με αυτή εκτελώντας τα κατάλληλα queries.

```
class CartService @Inject() (protected val dbConfigProvider:
DatabaseConfigProvider)(implicit ec: ExecutionContext)
    extends HasDatabaseConfigProvider[JdbcProfile]
    with Logger { … }
```

Κώδικας 3.16: Δήλωση CartService (database client)

Για τα SQL queries των services έγιναν με την χρήση της βιβλιοθήκης Slick όπου επιτρέπει την δημιουργία τους δυναμικά χρησιμοποιώντας μεταβλητές Scala μέσα σε αυτά. Χρησιμοποιώντας στην αρχή της έκφρασης μια από τις μεθόδους που σου παρέχει το Slick μπορείς να δημιουργήσετε εύκολα των κατάλληλο query για είτε να διαβάζει είτε να γράψει στην βάση. Η sql μέθοδος χρησιμοποιήθηκε για τα SELECT queries όπου χτίζει από String, η sqlu για τα UPDATES και INSERT επίσης από String και η τρίτη μέθοδος είναι το tsql όπου δεν έγινε χρήση της στην εφαρμογή και κτίζει τα queries από τα type που θα ορίσεις.

```
def resetStock(productId: String, quantity: Int): Int = {
  val productO = getProductById(productId)
  productO match {
    case Some(product) =>
      val newStock = product.stock + quantity
      val inStock = Utils.inStock(product.stock)
      val updateProduct =
        sqlu"update electronify.product set stock = $newStock , in_stock =
$inStock where id = ${product.id};"
      updateQueries(updateProduct)
    case None =>
      logger.info("Failed to get product")
      -1
  }
}

def getProductById(id: String): Option[Product] = {
  val getProduct =
    sql"select * from electronify.product where id = $id;"
      .as[Product]
  getFutureValue(db.run(getProduct)).headOption
}
```

Κώδικας 3.17: Cart's service methods getProductById and resetStock

Όλα τα services της εφαρμογής κάνουν χρήση abstract μεθόδων για το response το οποίο πήραν από την βάση δεδομένων. Εάν το response που πήραν πίσω δεν είναι κάποιο αποτέλεσμα τότε υπάρχει πρόβλημα στο query που εκτελέσαμε ή κάποιο σφάλμα στην βάση μας. Οι λογικοί έλεγχοι της εφαρμογής είτε γίνονται πριν την εκτέλεση ενός query είτε μετά αναλόγως της εργασίας που θέλουμε να κάνουμε και αν υπάρχει κάποιο σφάλμα εκεί τότε το κάνουμε handle στο κώδικα με το κατάλληλο δικό μας response και συνήθως με ένα κατάλληλο μήνυμα στα logs για να μπορέσουμε να εντοπίσουμε και να επιλύσουμε το πρόβλημα πιο γρήγορα και αποτελεσματικά.

```
def getFutureValue[T](future: Future[T]): T = {
  futureValidation(Some(Await.result(future, 10.seconds)))
}

def futureValidation[T](result: Option[T]): T = {
  result match {
    case Some(value) => value
    case None        => throw new Exception("Failed to execute query")
  }
}
```

Κώδικας 3.18: Utils object abstract methods

3.4 Common Utility

Το common utility είναι μια από τις καλύτερες τεχνικές ανάπτυξης λογισμικού μπορεί να εφαρμόσει κάποιος. Συνήθως είναι βοηθητικές λειτουργίες οποίες μπορούν να χρησιμοποιηθούν μέσα στην εφαρμογή περισσότερο από μια φορές και μπορείς εύκολες να δημιουργήσεις εύκολα δοκιμές καλύπτοντας έτσι πιθανά σφάλματα στην λογική τους. Δεν θέλετε να αναμειγνύετε όλες αυτές τις βοηθητικές λειτουργίες με την υπόλοιπη επιχειρησιακή λογική της εφαρμογής σας έτσι μερικά αρχεία

με αυτές τις λειτουργίες είναι απαραίτητα. Ωστόσο, καθώς τα αρχεία βοηθητικών προγραμμάτων σας αυξάνονται, συνιστάται να τα ομαδοποιείτε ανάλογα με τη λειτουργία που παρέχουν. Παραδείγματος χάρει στην εφαρμογή έχουν δημιουργηθεί δύο αρχεία utility. Το ένα αφορά ότι σχετίζεται με ημερομηνίες DateUtils και το άλλο κοινές βοηθητικές λειτουργίες Utils.

```
  def currentDate: String = {
    new Date().toString
  }
```

Κώδικας 3.19: DateUtils currenctDate method

```
def extractUUID(jsValue: Option[JsValue]): String = {
  jsValue match {
    case Some(js) =>
      js.validate[Id] match {
        case JsSuccess(value, _) =>
          value.id.getOrElse(UUID.fromString("")).toString
        case JsError(errors) =>
          logger.info("Invalid id" + errors)
          ""
      }
    case None =>
      logger.info("No ID was given")
      ""
  }
}
```

Κώδικας 3.20: Utils extractUUID method

3.5 Σχήμα βάσης δεδομένων

Το σχήμα της βάσης διαφοροποιήθηκε από τον αρχικό σχεδιασμό καθώς αναπτυσσόταν η εφαρμογή χρειάστηκαν να προστεθούν ή να αφαιρεθούν νέα πεδία, να δημιουργηθούν νέες συσχετίσεις μεταξύ τους αλλά και να αφαιρεθούν εν τέλει πίνακες που δεν χρειάστηκαν.

Users	
Id	Varchar (50) Primary Key
Username	Varchar (50)
Password	Varchar (200)
Email	Varchar (50)
Telephone	Varchar (50)
Created_at	Timestamp
Active	Boolean
Auth	Integer
Profile_image	Varchar (50)

Σχήμα 3.1: Users

Users_address	
Id	Varchar (50) Primary Key
Address_1	Varchar (50)
Address_2	Varchar (50)
City	Varchar (50)
Postal_code	Varchar (50)
Country	Varchar (50)

Telephone	Varchar (50)
Name	Varchar (50)
Comments	Varchar (50)
User_id	Varchar (50) Foreign Key

Σχήμα 3.2: User address

Shopping_session	
Id	Varchar (50) Primary Key
User_id	Varchar (50) Foreign Key
Login	Timestamp
Logout	Timestamp

Σχήμα 3.3: Shopping session

Product_category	
Id	Varchar (50) Primary Key
Name	Varchar (50)
Created_at	Timestamp

Σχήμα 3.4: Product category

Product	
Id	Varchar (50) Primary Key
Name	Varchar (50)
Brand	Varchar (50)
Price	Numeric
On_sale	Boolean
In_stock	Boolean
Created_at	Timestamp
Product_category	Varchar (50) Foreign Key
Sale	Integer
Stock	Integer
Image	Varchar (100)
Description	Varchar (200)

Σχήμα 3.5: Product

Payment	
Id	Varchar (50) Primary Key
Status	Varchar (50)
Amount	Numeric
User_id	Varchar (50) Foreign Key
Order_id	Varchar (50) Foreign Key

Σχήμα 3.6: Payment

Orders	
Id	Varchar (50) Primary Key
Total	Numeric
Status	Varchar (50)
Items	Varchar [] []
Created_at	Timestamp
User_id	Varchar (50) Foreign Key

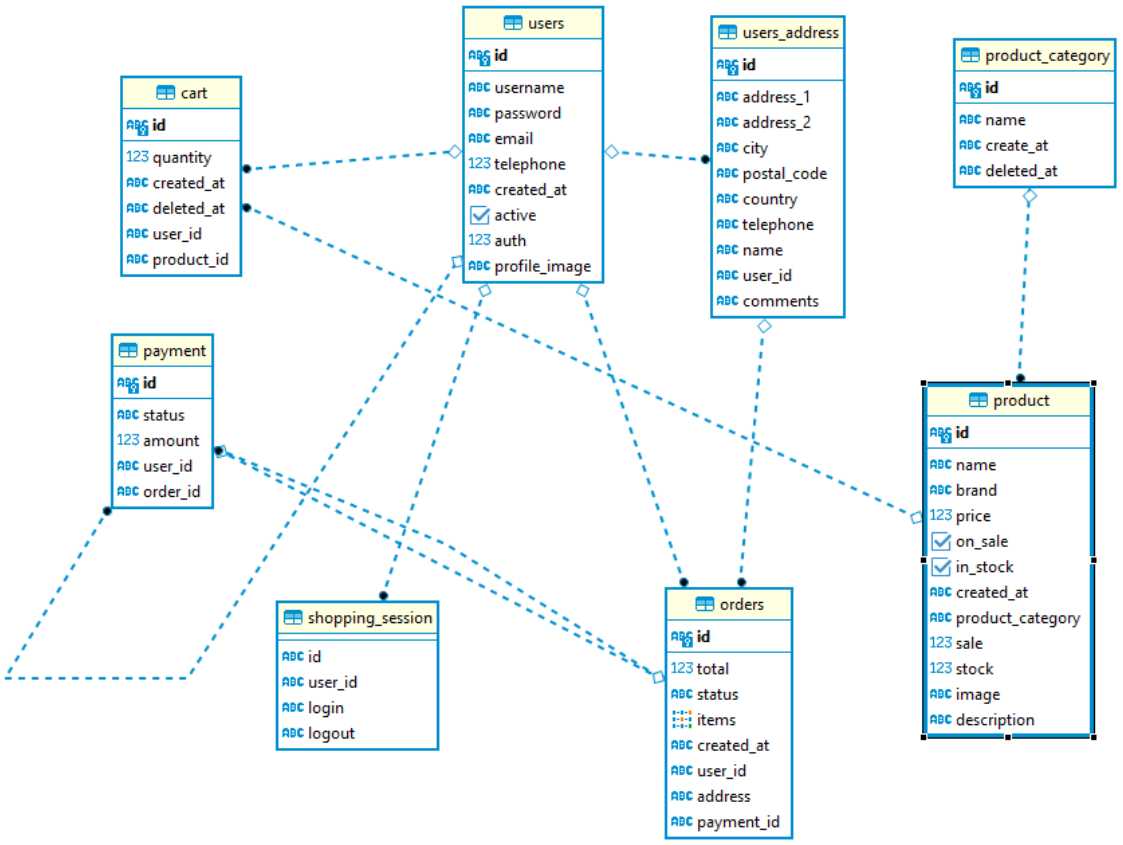
Address	Varchar (50) Foreign Key
Payment_id	Varchar (50) Foreign Key

Σχήμα 3.7: Orders

Cart	
Id	Varchar (50) Primary Key
Quantity	Numeric
Created_at	Timestamp
Delete_at	Timestamp
User_id	Varchar (50) Foreign Key
Product_id	Varchar (50) Foreign Key

Σχήμα 3.8: Cart

Παρακάτω παρατίθεται το ER διάγραμμα της βάσης από το εργαλείο DBeaver :



Εικόνα 3.2: ER diagram database from DBeaver

3.6 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε και αναλύθηκαν όλα τα κύρια χαρακτηρίστηκα κομμάτια της εφαρμογής τα οποία χρειάστηκαν για να ξεκινήσει, να αναπτυχθεί και να ολοκληρωθεί η εφαρμογή. Παρουσιάστηκαν μπλοκ κώδικα σαν παραδείγματα για το πως λυθήκαν προβλήματα και το πως αναπτύχθηκαν λειτουργίες μέσα στην εφαρμογή. Επιπλέον έγιναν μικρές επεξηγήσεις σε κομμάτια του κώδικα σε πραγματικά της Scala και των βιβλιοθηκών που χρησιμοποιήθηκαν έτσι ώστε να κατανοηθεί ο λόγος χρήσης τους.

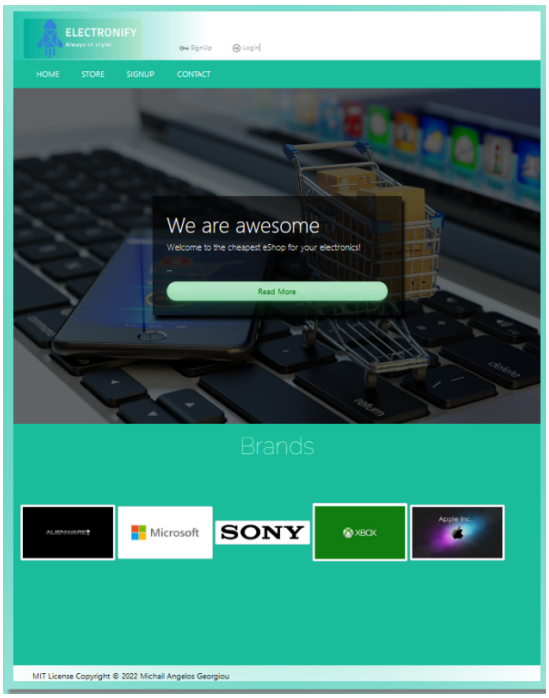
Κεφάλαιο 4ο: Παρουσίαση εφαρμογής

4.1 Εισαγωγή

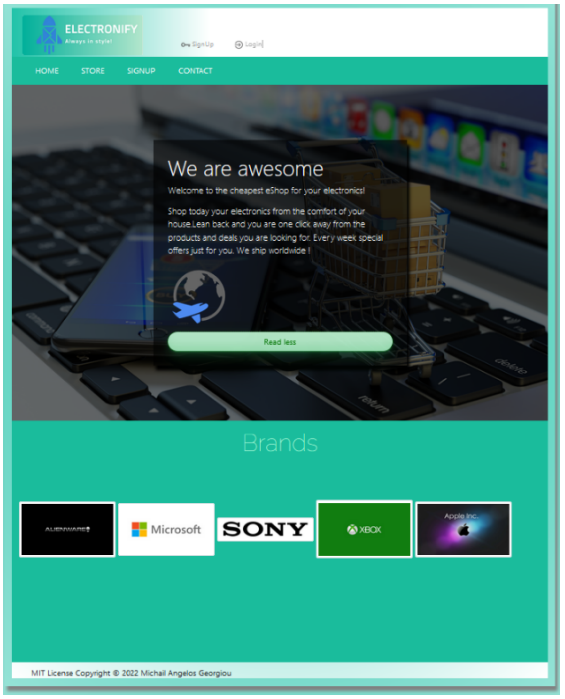
Σε αυτό το κεφάλαιο παρουσιάζονται και περιγράφονται όλες οι οθόνες της εφαρμογής. Με τις οθόνες θα γίνει παρουσίαση των λειτουργειών κάθε οθόνης και με τα κατάλληλα στιγμιότυπα τις περιπτώσεις χρήσης του 1ου κεφαλαίου και αν εκπληρώθηκαν όλες οι απαιτήσεις λογισμικού που τέθηκαν στην αρχή του σχεδιασμού της εφαρμογής.

4.2 Αρχική Οθόνη

Όταν χρήστης ανοίγει την εφαρμογή από ένα browser. Στην αρχική οθόνη της εφαρμογής ο χρήστης μπορεί να διακρίνει τα δύο μενού της εφαρμογής, μερικές πληροφορίες ανοίγοντας την κεντρική κάρτα της σελίδας, να εγγραφή στην εφαρμογή αν δεν είναι ήδη εγγεγραμμένος, να συνδεθεί ή και ακόμη να περιηγηθεί στις υπόλοιπες σελίδες της εφαρμογής χωρίς όμως περεταίρω δυνατότητες καθώς δεν είναι συνδεδεμένος.



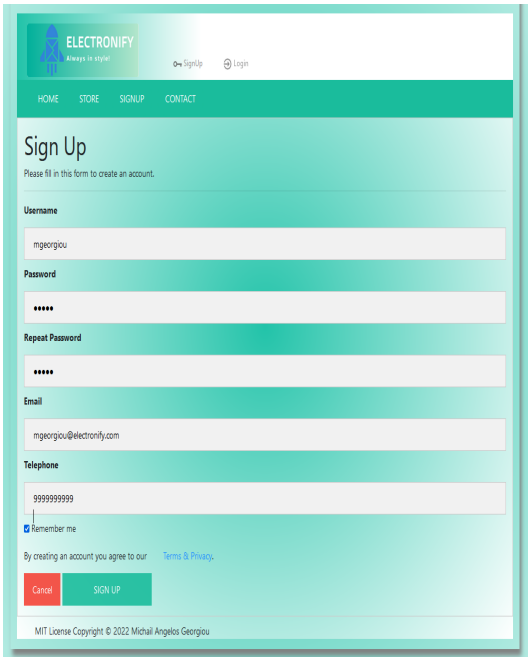
Εικόνα 4.1: Αρχική σελίδα



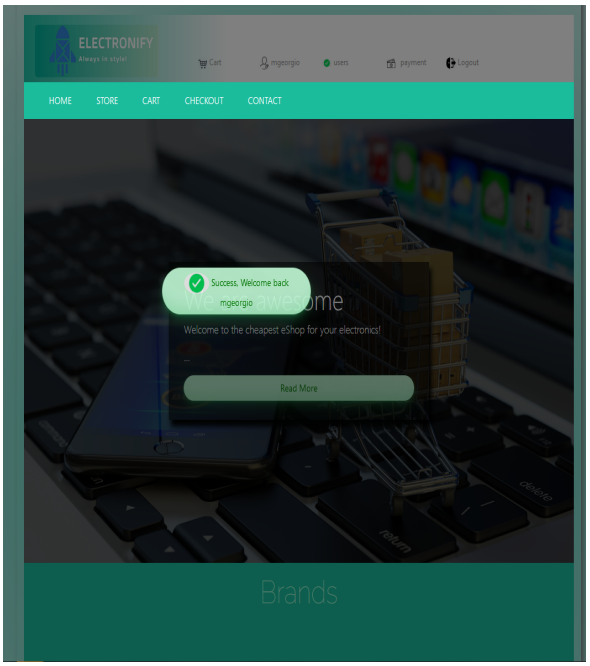
Εικόνα 4.2: Αρχική σελίδα με read more επιλεγμένο

4.3 Οθόνη εγγραφής

Καθώς ο χρήστης βρίσκεται σε οποιαδήποτε σελίδα και επιθυμεί να εγγραφή τότε επιλέγοντας από το κυρίως μενού το sign up μπορεί του εμφανίζει την φόρμα για να συμπληρώσει τα στοιχεία του και πατώντας το κουμπί sign up εάν όλα πήγαν καλά τότε συνδέεται στην εφαρμογή και του εμφανίζει μήνυμα επιτυχίας.



Εικόνα 4.3: Φόρμα εγγραφής



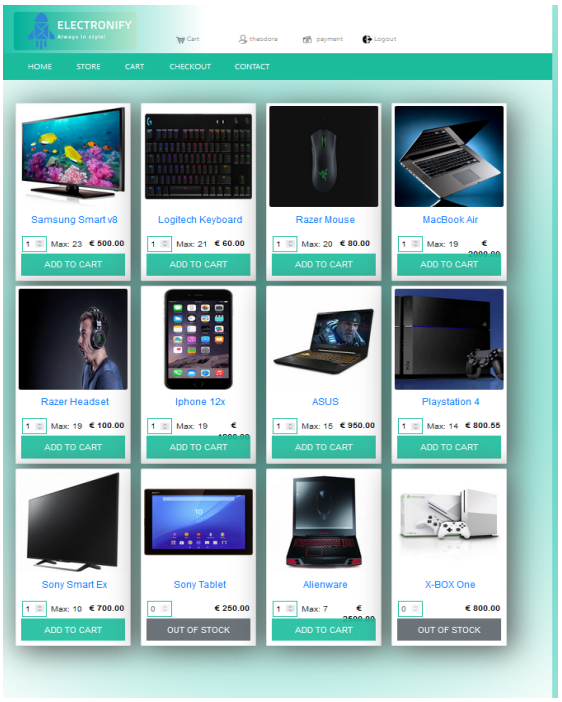
Εικόνα 4.4: Επιτυχία εγγραφής

Τώρα ο χρήστης που εγγράφηκε στην εφαρμογή του εμφανίζει περισσότερες επιλογές και στα δύο μενού και έχει την δυνατότητα αγοράς προϊόντων από το μαγαζί.

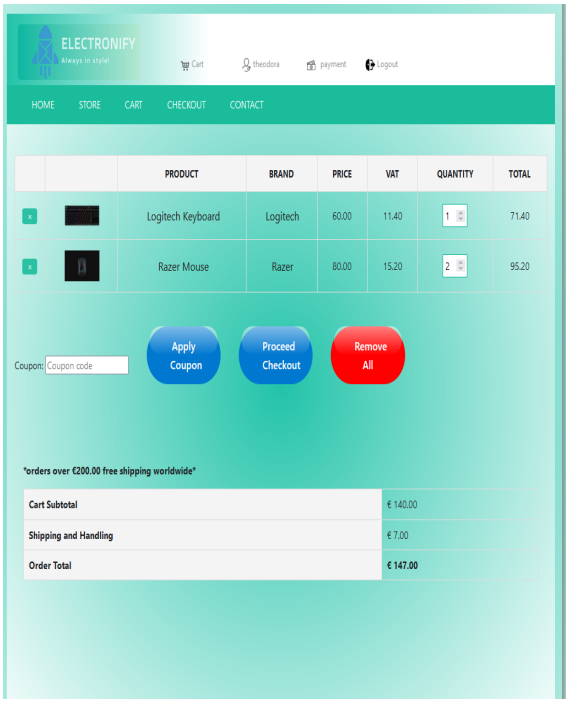
4.4 Προϊόντα και καλάθι αγορών

Εφόσον ο χρήστη είναι συνδεδεμένος στην εφαρμογή πλέον έχει την δυνατότητα να προσθέσει προϊόντα στο καλάθι αγορών του με σκοπό την αγορά τους.

Διακρίνονται λοιπόν σε αυτή την οθόνη όλα τα προϊόντα που διαθέτει η επιχείρηση και με το πάτημα του κουμπιού ADD TO CART προσθέτει τον ανάλογο αριθμό ποσότητας για ένα προϊόν στο καλάθι αγορών του. Επιλέγοντας τώρα ο χρήστη από το μενού την επιλογή CART τον ανακατευθύνει στην σελίδα καλάθι αγορών και μπορεί να δει πλέον τα προϊόντα που πρόσθεσε.



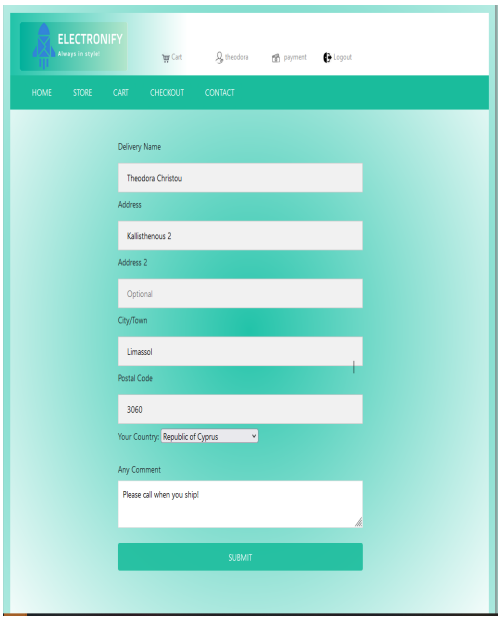
Εικόνα 4.5: Κατάλογος με διαθέσιμα προϊόντα



Εικόνα 4.6: Καλάθι αγορών χρήστη

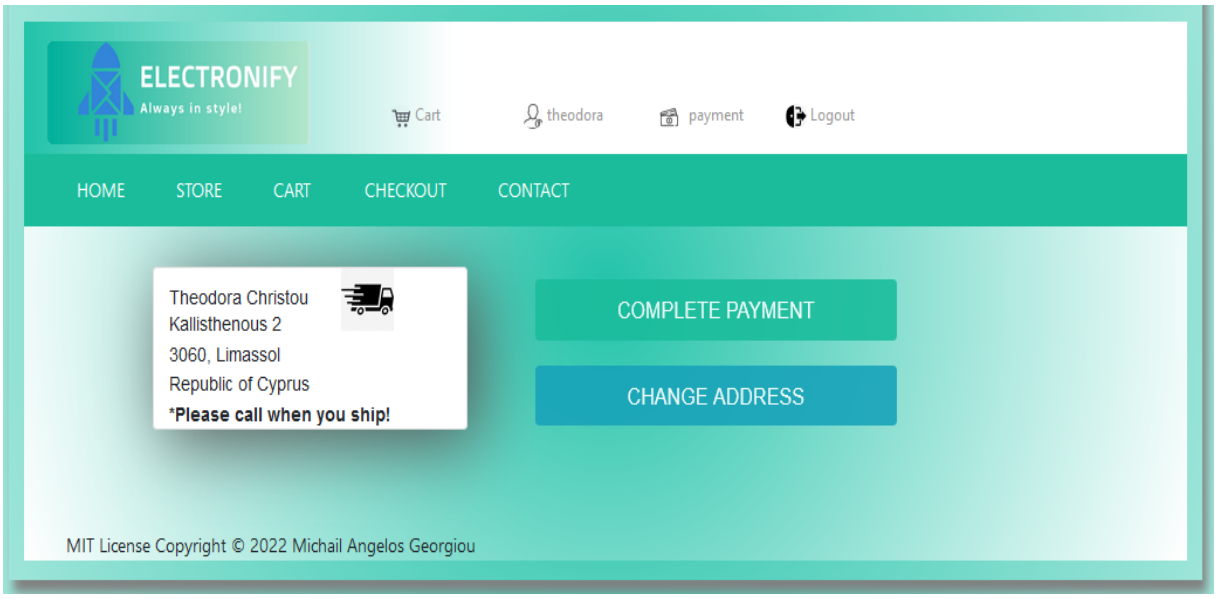
4.5 Διεύθυνση και αγορά

Εφόσον λοιπόν ο χρήστης επιθυμεί να αγοράσει τα προϊόντα που έχει βάλει στο καλάθι του τότε επιλέγει το κουμπί Proceed Checkout όπου εάν ο χρήστης δεν έχει καταχωρημένη διεύθυνση στην εφαρμογή τον ανακατευθύνει στην φόρμα συμπλήρωσης για διεύθυνση.



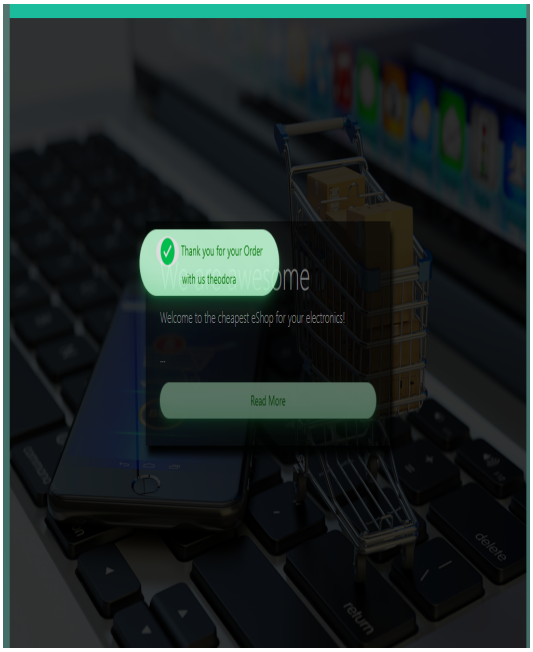
Εικόνα 4.7: Φόρμα στοιχεία διεύθυνσης

Όταν ο χρήστη συμπλήρωση επιτυχώς τα στοιχεία διεύθυνσης. Πατάει το κουμπί SUBMIT και αν όλα πήγαν καλά με την εισαγωγή στοιχείων τον ανακατευθύνει στην τελική σελίδα πριν από την τοποθέτηση παραγγελίας όπου ο χρήστης έχει την ευκαιρία να ελέγξει εάν τα στοιχεία του είναι σωστά ή επιθυμεί να αλλάξει κάτι.



Εικόνα 4.8: Τελική σελίδα πριν την αγορά

Εάν ο χρήστης είναι σίγουρος με την διεύθυνσης και τις οδηγίες που έχει δώσει τότε θα επιλέξει το κουμπί COMPLETE PAYMENT για ολοκλήρωση παραγγελίας και πληρωμής.

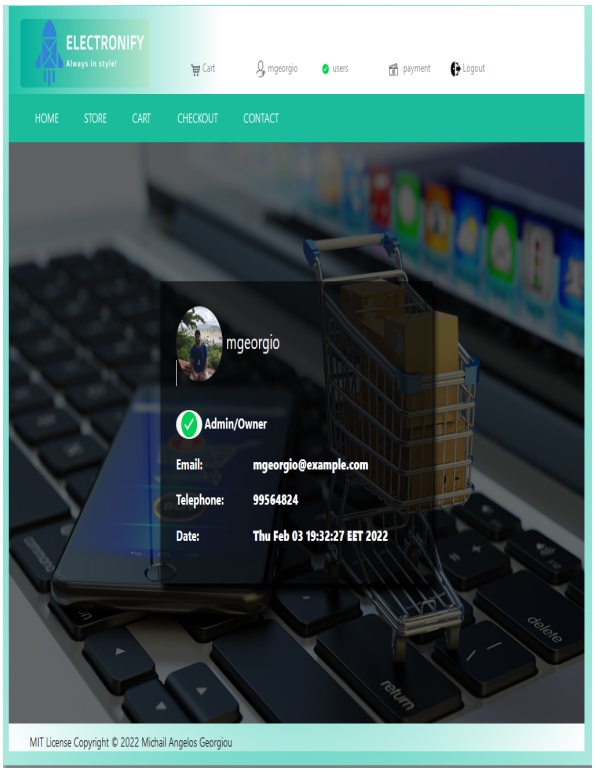


Εικόνα 4.9: Επιτυχία αγοράς

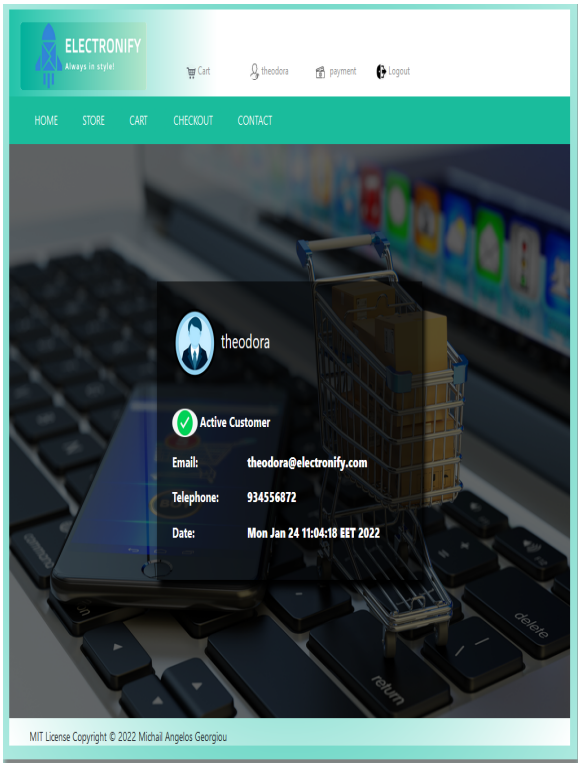
Ο χρήστη εάν έκανα την επιλογή να πατήσει το κουμπί CHANGE ADDRESS γιατί παρατήρησε λάθος στα στοιχεία αποστολής τότε θα τον ανακατεύθυνε ξανά στην φόρμα συμπλήρωσης στοιχείων διεύθυνσης.

4.6 Προφίλ χρήστη

Ο εγγεγραμμένος χρήστης μπορεί να με την επιλογή από το δευτερεύον μενού επιλέγοντας το κουμπί με το username του να δει όλα του στοιχεία και τι επίπεδο χρήστη είναι.



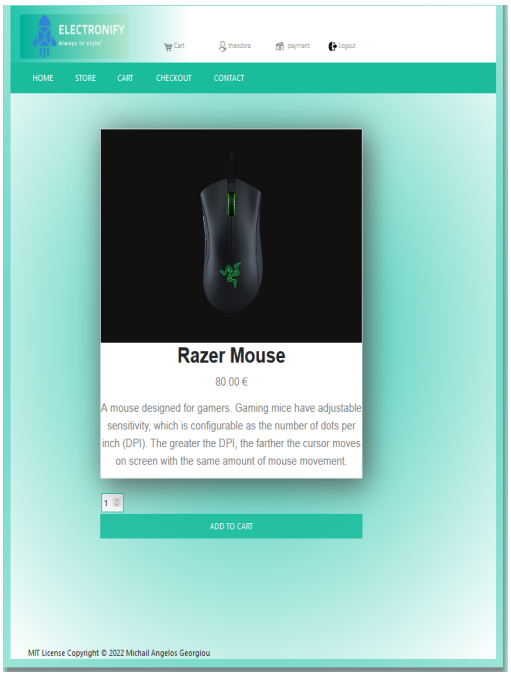
Εικόνα 4.10: Profile admin



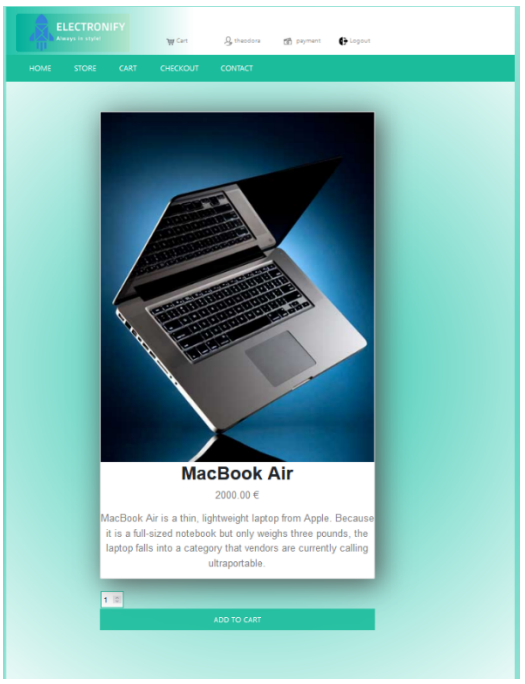
Εικόνα 4.11: Profile active customer

4.7 Περιγραφή προϊόντος

Ο χρήστης έχει την δυνατότητα αν θέλει να μάθει περισσότερες πληροφορίες για ένα προϊόν να επιλέξει από τον κατάλογο προϊόντων ένα αντικείμενο και πατώντας πάνω στο όνομα του τότε τον ανακατευθύνει στην σελίδα με το προϊόν με αναλυτική περιγραφή για αυτό και δυνατότητα πρόσθεσης στο καλάθι.



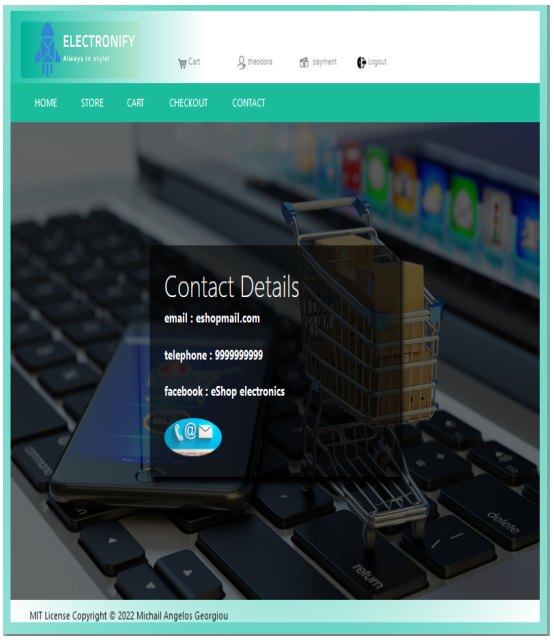
Εικόνα 4.12: Περιγραφή προϊόντος ποντίκι



Εικόνα 4.13: Περιγραφή προϊόντος laptop

4.8 Σελίδα στοιχείων επικοινωνίας

Με την επιλογή από το βασικό μενού CONTACT ένας χρήστης της εφαρμογής μπορεί να δεί τα στοιχεία επικοινωνίας με της επιχείρησης.



Εικόνα 4.14: Κάρτα επικοινωνίας

4.9 Επίλογος

Σε αυτό το κεφάλαιο έγινε ανάλυση των σελίδων της εφαρμογής και πως ένας χρήστης αλληλοεπιδρά με αυτές.

Κεφάλαιο 5ο: Σύνοψη

5.1 Σύνοψη

Η παρούσα Π.Ε. είχε σκοπό την υλοποιήσει μιας εφαρμογής ιστού με σύγχρονες τεχνολογίες παρέχοντας δυνατότητα χρήσης της και από χρήστες με ελάχιστες γνώσεις χρήσης τέτοιου είδους εφαρμογών. Η εφαρμογή δίνει την δυνατότητα σε χρήστες να μπορούν να κάνουν εύκολα , γρήγορα και όσο το δυνατό πιο ασφαλή τις αγορές τους μέσω μιας εφαρμογής που ο σχεδιασμός του U.I. έγινε με τον πιο απλό δυνατό τρόπο αλλά η ασφάλεια και η προστασία των δεδομένων έγινε με υπευθυνότητα. Κάθε εγγεγραμμένος χρήστης στην εφαρμογή μπορεί να συνδεθεί και να απολαύσεις τις αγορές των ηλεκτρονικών του συσκευών από την άνεση του σπιτιού όπου και αν βρίσκεται στον κόσμο.

Το μέλλον της εφαρμογής θα ήθελα να εμπλουτιστεί με περισσότερες λειτουργίες και δυνατότητες. Δουλεύοντας με αυτές τις νέες τεχνολογίες έμαθα πολλά καινούργια εργαλεία σχετικά με τον σχεδιασμό εφαρμογών ιστού και πως αυτές μπορούν να αλλάξουν την καθημερινότητα μας δίνοντας μας μια νέα δυναμική για το πως να χρησιμοποιούμε το διαδίκτυο. Θα ήθελα να συνεχίσω να δουλεύω στο συγκεκριμένο Project έτσι ώστε να καταφέρω να φτάσω στο επιθυμητό επίπεδο έτσι να γίνει μια ανταγωνιστική e -commerce εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Martin Odersky, Lez Spoon and Bill Venners, *Programming in Scala, Fourth Edition*, Artima, Inc 2020.

[2] Robert C. Martin, *Clean Cod*e, Prentice Hall 2009.

[3] Julien Richard-Foy, *Play Framework Essentials*, Packt Publishing 2014.

[4] Richard Dallaway and Jonathan Ferguson, *Essential Slick*, eBook 2015.

[5] Play Framework Documentation, https://www.playframework.com/documentation/2.8.x/Home

(Visited on Nov. 25, 2021)

[6] Flyway Documentation, https://flywaydb.org/documentation/

(Visited on Jan. 22, 2022)

[7] Learn Scala, https://docs.scala-lang.org/

(Visited on Oct. 18, 2021)

[8] Bootstrap, https://getbootstrap.com/docs/4.1/getting-started/introduction/

(Visited on Oct. 7, 2021)

[9] W3Schools, https://www.w3schools.com/

(Visited on Sept. 22, 2021)

[10] eCommerce trends for 2021,

https://www.globalbankingandfinance.com/ecommerce-trends-for-2021-the-boom-in-online-competition-due-to-ecommerce-increase/

(Visited on Aug. 8, 2021)

[11] E-Commerce Statistics 2022, https://www.drip.com/blog/e-commerce-statistics

(Visited on Aug. 10, 2022)

[12] User Experience Basics, https://www.usability.gov/what-and-why/user-experience.html

(Visited on May 15, 2022)

[13] How to write a Software Requirements Specification, https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

(Visited on April 10, 2021)

[14] Ecommerce Security: Importance, Issues & Protection Measures, https://www.getastra.com/blog/knowledge-base/ecommerce-security/

(Visited on Jul. 15, 2022)

[15] Play Framework, https://en.wikipedia.org/wiki/Play_Framework

(Visited on Jul. 20, 2022)

[16] Akka, https://akka.io/

(Visited on Dec. 15, 2021)

[17] Tour of Scala Introduction, https://docs.scala-lang.org/tour/tour-of-scala.html

(Visited on Oct. 12, 2021)

[18] Nilanjan Raychaudhuri, Scala in Action, Meap 2020.

[19] Performance, https://levelup.gitconnected.com/6-reasons-scala-is-better-than-java-c328cfb410d1

(Visited on Sept. 1, 2021)

[20] Why use PostgreSQL as database, https://fulcrum.rocks/blog/why-use-postgresql-database

(Visited on Jan 8, 2022)

[21] Getting Started with Flyway and Version-Based Database Migration, https://thorben-janssen.com/flyway-getting-started/

(Visited on Jan 10, 2022)

[22] Why web security is so important, https://strategynewmedia.com/why-web-security-is-important/

(Visited on Nov. 2, 2021)

[23] Play documentation, https://www.playframework.com/documentation/2.8.x/Introduction

(Visited on Oct. 16, 2021)

[24] Why is Play framework so fast? https://www.lightbend.com/blog/why-is-play-framework-so-fast

(Visited on May 20, 2022)

[25] JavaScript, https://en.wikipedia.org/wiki/JavaScript

(Visited on Jul. 10, 2022)

[26] Bootstrap, https://en.wikipedia.org/wiki/Bootstrap

(Visited on Oct. 5, 2021)