ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Επιλεγμένου Θέματος	Εφαρμογή Πώλησης Μεταχειρισμένων Επίπλων
Ονοματεπώνυμο Φοιτητή	Καϊμάκου Άντζελα
Πατρώνυμο	Εντουάρτ
Αριθμός Μητρώου	Π21039
Επιβλέπων	Ευθύμιος Αλέπης, Καθηγητής

Ημερομηνία Παράδοσης: Σεπτέμβριος 2025

Copyright ©

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου εργασίας και, κατ' επέκταση, των προπτυχιακών μου σπουδών, θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες σε όλους συνέβαλαν με οποιονδήποτε τρόπο στην ολοκλήρωση της παρούσας πτυχιακής εργασίας.

Πρώτα απ' όλα, ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου, κ. Ευθύμιο Αλέπη, για την πολύτιμη καθοδήγηση, και τη συνεχή υποστήριξή του καθ' όλη τη διάρκεια της εκπόνησης της εργασίας.

Ιδιαίτερες ευχαριστίες οφείλω στους φίλους και συνεργάτες μου από τη σχολή, για τις γνώσεις που μοιράστηκαν μαζί μου, τις αμέτρητες ομαδικές εργασίες που ολοκληρώσαμε, και τις ευχάριστες στιγμές που ζήσαμε κατά τη προσπάθεια ολοκλήρωσης τους. Εύχομαι η συνεργασία μας να συνεχιστεί και στον επαγγελματικό χώρο, ως συνάδελφοι πλέον.

Τέλος, θα ήθελα να εκφράσω την απεριόριστη ευγνωμοσύνη μου προς την οικογένεια μου, για την ενθάρρυνση και στήριξη, την κατανόηση και την υπομονή που έδειξαν, όχι μόνο κατά τη διάρκεια της εκπόνησης της εργασίας, αλλά καθ' όλη τη διάρκεια αυτού του ταξιδιού.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας διαδικτυακής εφαρμογής για την πώληση μεταχειρισμένων επίπλων. Μέσω της εφαρμογής, οι χρήστες μπορούν να διαθέτουν αντικείμενα που δεν χρειάζονται πλέον, καθώς και να αγοράζουν μεταχειρισμένα είδη από άλλους χρήστες, σε ένα μοντέλο παρόμοιο με την πλατφόρμα Vinted.

Για την υλοποίηση χρησιμοποιήθηκαν τεχνολογίες όπως το Spring Boot σε Java για το backend, το React.js για το frontend και η βάση δεδομένων MySQL. Η επιλογή αυτών των τεχνολογιών έγινε με γνώμονα την ευελιξία που προσφέρουν, την ευκολία στην ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών και τη γρήγορη εκμάθηση.

Η ανάπτυξη της εφαρμογής βασίστηκε στη προσέγγιση σταδιακής υλοποίησης, ξεκινώντας από ένα γενικό πλάνο και προσθέτοντας σταδιακά λειτουργίες, ενώ παράλληλα πραγματοποιούνταν έλεγχοι και δοκιμές. Η μεθοδολογία αυτή επέτρεψε την προοδευτική βελτίωση του συστήματος.

Η πλατφόρμα έχει ως στόχο να προωθήσει τη βιωσιμότητα και να ενθαρρύνει την επαναχρησιμοποίηση επίπλων, αντί της απόρριψής τους, συμβάλλοντας στη μείωση του περιβαλλοντικού αποτυπώματος.

Λέξεις Κλειδιά: Διαδικτυακή εφαρμογή, Πλατφόρμα πώλησης, Μεταχειρισμένα έπιπλα, Spring Boot, Java, React.js, MySQL

Abstract

This thesis focuses on the development of web application for the sale of second-hand furniture. Through the application, users can list items they no longer need, as well as purchase items from other users, following a model similar to the Vinted platform.

For the implementation, technologies such as Spring Boot in Java for the backend, React.js for the frontend, and MySQL as the database were used. These technologies were chosen for their flexibility, ease of use in modern web application development, and fast learning curve.

The application was developed using a step-by-step implementation, starting with a general plan and gradually adding features, while simultaneously conducting test and evaluations. This methodology allowed for the progressive improvement of the system.

The platform aims to promote sustainability and encourage the reuse of furniture instead of disposal, contributing to the reduction of environmental impact.

Key Words: Web application, Marketplace, Second-hand furniture, Spring Boot, Java, React.js, MySQL

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

1 Εισαγωγή

1.1 Συνοπτική Περιγραφή

Στη σύγχρονη εποχή, η βιωσιμότητα και η εξοικονόμηση πόρων αποτελούν σημαντική προτεραιότητα για ολοένα και περισσότερους καταναλωτές. Τα τελευταία χρόνια, παρατηρείται αυξανόμενο ενδιαφέρον στην χρήση εναλλακτικών λύσεων που περιορίζουν τη σπατάλη και προωθούν την επαναχρησιμοποίηση αγαθών. Τα μεταχειρισμένα έπιπλα, τα οποία παραμένουν σε εξαιρετική κατάσταση, αποτελούν ένα από τα αγαθά που μπορεί κάποιος να τους δώσει «δεύτερη ζωή» μειώνοντας το περιβαλλοντικό αποτύπωμα που έχει η παραγωγή τους.

Παρά την αυξημένη ζήτηση για τέτοιου είδους προϊόντα, παρατηρείται έλλειψη εξειδικευμένων ψηφιακών εφαρμογών που επικεντρώνονται στην αγορά και την πώληση τους, χωρίς πολυπλοκότητα στις διαδικασίες συναλλαγών. Οι υπάρχουσες πλατφόρμες, όπως το Vinted, facebook marketplace, δεν καλύπτουν πλήρως τις ανάγκες των χρηστών που αναζητούν ή να διαθέσουν έπιπλα, καθώς επικεντρώνονται σε άλλες κατηγορίες προϊόντων, καθιστώντας τα έπιπλα ως προϊόν δευτερεύουσας σημασίας.

1.2 Καθορισμός της Εργασίας

Η παρούσα πτυχιακή εργασία έχει ως στόχο την ανάπτυξη μιας ψηφιακής πλατφόρμας, που αφορά αποκλειστικά την αγοραπωλησία μεταχειρισμένων επίπλων διευκολύνοντας την σχετική διαδικασία. Η πλατφόρμα θα επιτρέπει στους χρήστες να δημιουργούν αγγελίες, να περιηγούνται στην εφαρμογή, να χρησιμοποιούν φίλτρα αναζήτησης, και να αγοράζουν έπιπλα από άλλους χρήστες.

Η υλοποίηση της πλατφόρμας πραγματοποιείται χρησιμοποιώντας σύγχρονες τεχνολογίες όπως Spring Boot σε Java για το backend τομέα της εφαρμογής προσφέροντας ευκολία στη δημιουργία RESTful υπηρεσίες, React.js για το frontend με στόχο να δημιουργηθεί ένα εύχρηστο προς τον χρήστη περιβάλλον, και για τη αξιόπιστη διαχείριση δεδομένων χρησιμοποιείται MySQL.

1.3 Επιμέρους Στόχοι

Οι κύριοι επιμέρους στόχοι της εργασίας είναι να μελετηθούν οι ανάγκες της αγοράς στον τομέα των μεταχειρισμένων επίπλων, διερευνώντας πως μπορεί η χρήση της τεχνολογίας να συνεισφέρει σε περιβαλλοντικά ζητήματα, και στη συνέχεια να αναπτυχθεί μια πλήρως λειτουργική διαδικτυακή πλατφόρμα χρησιμοποιώντας σύγχρονες τεχνολογίες στην ανάπτυξη του συστήματος.

Η αναμενόμενη συνεισφορά της εργασίας, δηλαδή, αφορά την χρήση των τεχνολογιών αυτών στην επίλυση ενός προβλήματος με περιβαλλοντικό υπόβαθρο, που αφορά τη σύγχρονη κοινωνία προσφέροντας ένα ολοκληρωμένο και φιλικό προς τον χρήστη περιβάλλον.

1.4 Διάρθρωση Πτυχιακής Εργασίας

Η δομή της εργασίας ορίζεται ως εξής:

- **Κεφάλαιο 1** – Εισαγωγή
  Γίνεται αναφορά στο θέμα της πτυχιακής και στους στόχους.
- **Κεφάλαιο 2** – Σχετική Έρευνα
  Παρουσιάζεται το πρόβλημα, και αναλύονται υπάρχουσες πλατφόρμες που σχετίζονται με την λύση του.
- **Κεφάλαιο 3** – Ανάλυση Απαιτήσεων και Σχεδίαση του Συστήματος
  Περιγράφεται η διαδικασία σχεδίασης της πλατφόρμας με γνώμονα τις ανάγκες των χρηστών, και οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση.

- **Κεφάλαιο 4** – Υλοποίηση της Πλατφόρμας
  Γίνεται τεχνική περιγραφή της υλοποίησης του backend, frontend και της βάσης δεδομένων.
- **Κεφάλαιο 5** – Δοκιμές
  Παρουσιάζονται οι λειτουργείες της εφαρμογής μέσω της χρήσης.
- **Κεφάλαιο 6** – Συμπεράσματα και Μελλοντικές Επεκτάσεις
  Περιλαμβάνει την ανάλυση πιθανών βελτιώσεων και μελλοντικές δυνατότητες επέκτασης που μπορούν να γίνουν.

2 Σχετική Έρευνα

2.1 Ηλεκτρονικό Εμπόριο και Ψηφιακές Πλατφόρμες

Με την εξέλιξη της τεχνολογίας, και την ενσωμάτωση της στον τομέα της οικονομίας, αναπτύχθηκε το ηλεκτρονικό εμπόριο (e-commerce), το οποίο προσφέρει στους καταναλωτές μεγαλύτερη ποικιλία προϊόντων και διευκολύνει και τη διαδικασία αγορών, αφού ο καταναλωτής έχει τη δυνατότητα πραγματοποίησης αγορών οποιαδήποτε στιγμή , από οποιοδήποτε σημείο[1][3].

Το ηλεκτρονικό εμπόριο διακρίνεται σε επιμέρους κατηγορίες, ανάλογα τη φύση της σχέσης μεταξύ πωλητή και αγοραστή. Οι κυριότερες μορφές είναι το B2B (Business-to-Business), B2C (Business-to-Consumer), C2B (Consumer-to-Business), C2C (Consumer-to-Consumer) [2] στην οποία βασίζεται η παρούσα εργασία.

Πιο συγκεκριμένα, στο μοντέλο C2C οι ίδιοι οι καταναλωτές αναλαμβάνουν ρόλο πωλητή, αξιοποιώντας ψηφιακές πλατφόρμες που τους δίνει την δυνατότητα να επικοινωνήσουν με άλλους καταναλωτές-πωλητές [2]. Σε αυτόν τον τύπο βασίζονται και πλατφόρμες όπως το Facebook Marketplace, eBay και Vinted όπου ένας χρήστης μπορεί να καταχωρήσει ένα αγαθό ή υπηρεσία προς πώληση ενώ ταυτόχρονα έχουν πρόσβαση σε προϊόντα άλλων χρηστών και να ολοκληρώσει τις αγορές του μέσα από το σύστημα.

Οι ψηφιακές πλατφόρμες αυτού του είδους συμβάλλουν στην διευκόλυνση των συναλλαγών, μειώνοντας το κόστος, και τον απαιτούμενο χρόνο, ενώ προσφέρουν στον χρήστη μια φιλική εμπειρία, ειδικότερα μέσω των προσωποποιημένων λειτουργιών και φίλτρων αναζήτησης [3]. Με αυτό τον τρόπο ενισχύουν τη τάση για επαναχρησιμοποίηση αγαθών, εντάσσοντας το εμπόριο μεταχειρισμένων προϊόντων σε ένα ψηφιακό και άμεσα προσβάσιμο περιβάλλον.

2.2 Κυκλική Οικονομία και Βιωσιμότητα στο κλάδο των επίπλων

Με τον όρο κυκλική οικονομία αναφερόμαστε σε ένα μοντέλο οικονομίας που υπερβαίνει τη λογική «παράγω, καταναλώνω, απορρίπτω» [4] και βασίζεται σε στρατηγικές όπως η μείωση, η επαναχρησιμοποίηση, η ανακατασκευή, η ανακύκλωση και η επιδιόρθωση» [4][5][6]. Στόχος του μοντέλου είναι η ελαχιστοποίηση του περιβαλλοντικού αποτυπώματος που έχουν οι παραγωγικές δραστηριότητες, μέσω της εξοικονόμησης φυσικών πόρων και της μείωσης αποβλήτων καθιστώντας το άμεσα συνδεδεμένη με την βιωσιμότητα [5].

Η βιομηχανία επίπλων αποτελεί μια από τις βιομηχανίες που επιβαρύνουν σημαντικά το περιβάλλον, καθώς μεγάλο μέρος των επίπλων καταλήγει ως απόβλητα, με μόνο ένα μικρό ποσοστό τους να ανακυκλώνεται [6] [7]. Αυτό το γεγονός προβάλλει την ανάγκη για βιώσιμες λύσεις εντός της αγοράς, όπως η επαναχρησιμοποίηση των επίπλων, η οποία συνεπάγεται με μείωση νέας παραγωγής, δηλαδή μείωση της περιβαλλοντικής επιβάρυνσης [7]

Σε αυτό το πλαίσιο, οι ψηφιακές λύσεις φαίνεται να είναι αποδοτικές, καθώς διευκολύνουν την υλοποίηση του μοντέλου κυκλικής οικονομίας στον κλάδο των επίπλων. Στρατηγικές όπως η επαναχρησιμοποίηση, η ανακατασκευή και η ανακύκλωση, αναπτύσσονται και εφαρμόζονται μέσω ψηφιακών πλατφορμών, επιτρέποντας την επέκταση της βιωσιμότητας [6].

2.3 Υπάρχουσες Πλατφόρμες Πώλησης Μεταχειρισμένων Επίπλων

Στο χώρο του ηλεκτρονικού εμπορίου, υπάρχουν ήδη πλατφόρμες είτε δημιουργημένες με σκοπό την αγοραπωλησία μεταχειρισμένων προϊόντων, είτε χρησιμοποιούνται και γι’ αυτό τον σκοπό.

Κάποιες από αυτές είναι το Vinted και το Facebook Marketplace, οι οποίες δεν καλύπτουν πλήρως τις ανάγκες των καταναλωτών που ενδιαφέρονται για μεταχειρισμένα έπιπλα.

2.3.1 Vinted

Η πλατφόρμα Vinted, που δημιουργήθηκε το 2008, επικεντρώνεται κυρίως στο τομέα του ρουχισμού. Οι χρήστες, μπορούν να ανεβάζουν αγγελίες, να αγοράζουν προϊόντα, να περιηγούνται στην εφαρμογή, να χρησιμοποιούν φίλτρα αναζήτησης, και να επικοινωνούν με άλλους χρήστες [8]. Παρόλο που υπάρχει η δυνατότητα να δημοσιεύονται και μεταχειρισμένα έπιπλα, η χρήση της πλατφόρμας επικεντρώνεται κυρίως στα ρούχα, ενώ δεν παρέχονται εξειδικευμένα φίλτρα για έπιπλα, καθιστώντας πιο δύσκολη την αναζήτηση και την πώληση τους.

2.3.2 Facebook Marketplace

Το Facebook Marketplace επιτρέπει την πώληση διάφορων προϊόντων στα οποία συμπεριλαμβάνονται και τα έπιπλα. Ωστόσο, δεν παρέχει ενσωματωμένη διαδικασία συναλλαγών, παρά μόνο τη δυνατότητα επικοινωνίας μεταξύ χρηστών, αυξάνοντας τον κίνδυνο απάτης. Επιπλέον, δεν παρέχονται φίλτρα εξειδικευμένα για έπιπλα, γεγονός που περιορίζει την αποτελεσματικότητα και την ευχρηστία της πλατφόρμας για τους ενδιαφερόμενους χρήστες [9].

2.3.3 Συμπέρασμα

Οι παραπάνω πλατφόρμες, αποτελούν χαρακτηριστικά παραδείγματα C2C μοντέλου. Παρά την δυνατότητα πώλησης μεταχειρισμένων επίπλων, δεν εξειδικεύονται σε αυτά, δημιουργώντας την ανάγκη για μία πλατφόρμα που επικεντρώνεται στην πώληση τους.

3 Ανάλυση Απαιτήσεων και Σχεδίαση του Συστήματος

Παρακάτω θα αναλυθούν οι λειτουργικές και μη λειτουργικές απαιτήσεις της πλατφόρμας, καθώς και η σχεδίαση και αρχιτεκτονική του.

3.1 Ανάλυση Απαιτήσεων

Το σύστημα δημιουργήθηκε με σκοπό να καλύπτει κάποιες βασικές ανάγκες.

Για τον διαχειριστή

Ο διαχειριστής έχει το πλήρη έλεγχο της πλατφόρμας. Πιο συγκεκριμένα, έχει τη δυνατότητα να βλέπει όλους τους εγγεγραμμένους χρήστες με τα στοιχεία τους, καθώς και τις αγγελίες ή τις παραγγελίες που έχουν. Επίσης, έχει πρόσβαση σε όλες τις αγγελίες και τις παραγγελίες της πλατφόρμας ανεξάρτητα από τον χρήστη στον οποίο ανήκουν. Τέλος, ο διαχειριστής, έχει την επιλογή διαγραφής αγγελιών ή παραγγελιών και απενεργοποίησης ή ενεργοποίησης λογαριασμού χρήστη.

Για τον χρήστη

Ένας απλός χρήστης μπορεί αρχικά να πραγματοποιήσει εγγραφή στο σύστημα παρέχοντας κάποια στοιχεία του, όπως ονοματεπώνυμο, όνομα χρήστη, email και κωδικό και ύστερα να συνδεθεί μέσω του ονόματος χρήστη και του κωδικού. Μετά την είσοδο του στο σύστημα, θα μπορεί να δημοσιεύει αγγελίες, να προβάλλει αγγελίες άλλων χρηστών και τις λεπτομέρειες τους, και να προχωρήσει στην αγορά του προϊόντος εφόσον το επιθυμεί. Έχει τη δυνατότητα να χρησιμοποιεί φίλτρα, να επισκέπτεται προφίλ πωλητών και να αποθηκεύει αγαπημένες αγγελίες. Επιπλέον, υπάρχει η επιλογή επεξεργασίας του προσωπικού του προφίλ, αγγελιών που έχει και η διαγραφή αγγελιών και παραγγελιών του.

Μη λειτουργικές απαιτήσεις

Αναφορικά με τις μη λειτουργικές απαιτήσεις, το σύστημα αποθηκεύει τους κωδικούς πρόσβασης των χρηστών σε κρυπτογραφημένη μορφή για την παροχή ασφάλειας και την αποτροπή πρόσβασης σε προσωπικά δεδομένα από μη εξουσιοδοτημένους χρήστες. Η πλατφόρμα δημιουργήθηκε σε απλό και κατανοητό περιβάλλον ώστε να διευκολυνθεί οποιοσδήποτε χρήστης στην χρήση της.

3.2 Σχεδίαση Συστήματος

Η πλατφόρμα σχεδιάστηκε ακολουθώντας την Three Tier Architecture, η οποία διαχωρίζεται σε τρία επίπεδα, το επίπεδο διεπαφής χρήστη (frontend) όπου πραγματοποιεί η αλληλεπίδραση με τον χρήστη, το επίπεδο λογικής (backend), όπου πραγματοποιούνται οι λειτουργίες, η επεξεργασία δεδομένων και το επίπεδο διαχείρισης δεδομένων (database), δηλαδή ο τομέας αποθήκευσης και ανάκτησης δεδομένων.

3.2.1 Frontend

Η υλοποίηση του frontend, έγινε με τη βιβλιοθήκη React της JavaScript η οποία επιλέχθηκε λόγω της υψηλής απόδοσης και της ευκολίας εκμάθησης και χρήσης της. Η React βασίζεται σε components, τα οποία είναι επαναχρησιμοποιήσιμα κομμάτια διεπαφής, δηλαδή ένα κομμάτι κώδικα μπορεί να γραφεί μια φορά αλλά να χρησιμοποιηθεί πολλαπλές φορές και με διαφορετικά δεδομένα αν είναι επιθυμητό [10]. Παράλληλα, η βιβλιοθήκη χρησιμοποιεί V-DOM ένα εικονικό DOM δηλαδή, το οποίο κρατά στη μνήμη το UI και ενημερώνει αυτόματα μόνο τα απαραίτητα στοιχεία όταν γίνεται αλλαγή σε ένα component. Αυτό συντελεί στην ταχύτητα, την ομαλή αλλαγή του UI και την διευκόλυνση στην χρήση [10][11].

3.2.2 Backend

Το backend κομμάτι της πλατφόρμας πραγματοποιήθηκε με την χρήση Spring Boot με γλώσσα Java, το οποίο είναι ένα εργαλείο που διευκολύνει τη χρήση Java-based frameworks [12]. Η επιλογή της συγκεκριμένης τεχνολογίας έγινε λόγω της ευκολίας χρήσης, της ταχύτητας ανάπτυξης και της υποστήριξης για χρήση άλλων τεχνολογιών. Περιλαμβάνει ενσωματωμένους servers όπως ο Tomcat, οι οποίοι τρέχουν χωρίς να χρειάζονται χειροκίνητο setup [12][13]. Ταυτόχρονα, με τη χρήση των annotations όπως @RestController, @RequestMapping, @GetMapping, επιτρέπει τη δημιουργία REST APIs, το οποίο βοηθά στη διαχείριση HTTPs αιτημάτων από τους χρήστες. Ένα βασικό χαρακτηριστικό της Spring Boot αποτελεί το autoconfiguration, δηλαδή η αυτόματη ρύθμιση διάφορων στοιχείων της εφαρμογής ανάλογα τις εξαρτήσεις που μπαίνουν στον κώδικα, κάνοντας πιο σύντομη και εύκολη τη ρύθμιση του συστήματος [12].

3.2.3 Database

Το σύστημα διαχείρισης βάσης δεδομένων που αξιοποιήθηκε στην υλοποίηση του συστήματος είναι η MySQL. Λόγω της απλής σύνταξης και εύκολης χρήσης του, η MySQL καθίσταται δημοφιλή επιλογή για ανάπτυξη εφαρμογών. Η MySQL παρέχει, επίσης, αξιοπιστία, υψηλή απόδοση και μεγάλη ποικιλία σε τύπους δεδομένων, όπως String, Long, BLOB που χρησιμοποιούνται στην πλατφόρμα [14].

4 Υλοποίηση της Πλατφόρμας

4.1 Database

Η βάση δεδομένων υλοποιήθηκε με τη χρήση MySQL με τέσσερις βασικούς πίνακες, τον **user** για τους χρήστες, τον **furniture** για τις αγγελίες, τον **orders** για τις παραγγελίες και τον πίνακα **user_favorites** που περιλαμβάνει τα αγαπημένα του χρήστη.

4.1.1 Πίνακας user

Ο πίνακας user, περιέχει βασικά χαρακτηριστικά για τους χρήστες της εφαρμογής. Μέσα σε αυτά, περιλαμβάνεται το id το οποίο είναι και primary key τύπου Long. Το username που χρησιμοποιείται για το όνομα χρήστη, το fullName για το ονοματεπώνυμο του, το email, το role για τον ρόλο του (“ADMIN”, “USER”), το account_status για την κατάσταση λογαριασμού του (“ACTIVE”, “INACTIVE”), το password για τον κωδικό του, είναι όλα τύπου String.

Για την αποθήκευση εικόνας προφίλ, υπάρχουν τα profilePicture τύπου MEDIUMBLOB για αποθήκευση της εικόνας, profilePictureName τύπου String που αποθηκεύει το όνομα της εικόνας και profilePictureType τύπου String για τον τύπο.

Ο πίνακας, έχει σχέσεις με όλους τους υπόλοιπους πίνακες. Συγκεκριμένα, συνδέεται με τον πίνακα furnitures, με foreign key το user_id, δηλαδή του id του χρήστη. Με τον πίνακα παραγγελιών, orders, ξανά συνδέεται μέσω του χρήστη με foreign key το id του. Τέλος, με τον πίνακα αγαπημένων, user_favorites, συνδέεται με user_id.

4.1.2 Πίνακας furniture

Ο πίνακας furniture, περιέχει στοιχεία για τα έπιπλα που δημοσιεύονται. Όπως και ο πίνακας User, έχει id (primary key) τύπου Long. Περιλαμβάνει πεδία όπως name για τον τίτλο αγγελίας, description για την περιγραφή, furnitureCondition για την κατάσταση του επίπλου (Καινούριο, Καλή Κατάσταση, Μεταχειρισμένο), furnitureCategory για την κατηγορία "Καρέκλα", "Τραπέζι", "Καναπές", "Ντουλάπα", "Κρεβάτι", "Κομοδίνο", "Γραφείο", "Άλλο"), και state για την κατάσταση (“available”,”unavailable”) όλα τύπου String. Η τιμή αποθηκεύεται στο πεδίο price και είναι τύπου Double.

Για την αποθήκευση εικόνας, υπάρχουν τα image τύπου MEDIUMBLOB που αποθηκεύει την εικόνα, imageName τύπου String που αποθηκεύει το όνομα της εικόνας και imageType τύπου String για τον τύπο.

Συνδέεται με τους υπόλοιπους πίνακες, Ο πίνακας, έχει σχέσεις με όλους τους υπόλοιπους πίνακες. Συγκεκριμένα, συνδέεται με τον πίνακα user, με foreign key το user_id, δηλαδή του id του χρήστη. Με τον πίνακα παραγγελιών, orders, συνδέεται μέσω το id αγγελίας με foreign key το product_id. Τέλος, με τον πίνακα αγαπημένων, user_favorites, συνδέεται με furniture_id.

4.1.3 Πίνακας orders

Ο πίνακας Orders περιέχει τις παραγγελίες και έχει id (primary key) τύπου Long. Περιλαμβάνει στοιχεία του χρήστη όπως fullName, email και username τύπου String και στοιχεία για την διεύθυνση αποστολής, όπως address, city, number, postal τύπου String. Η ημερομηνία και ώρα αποστολής αποθηκεύονται στο πεδίο date τύπου LocalDateTime.

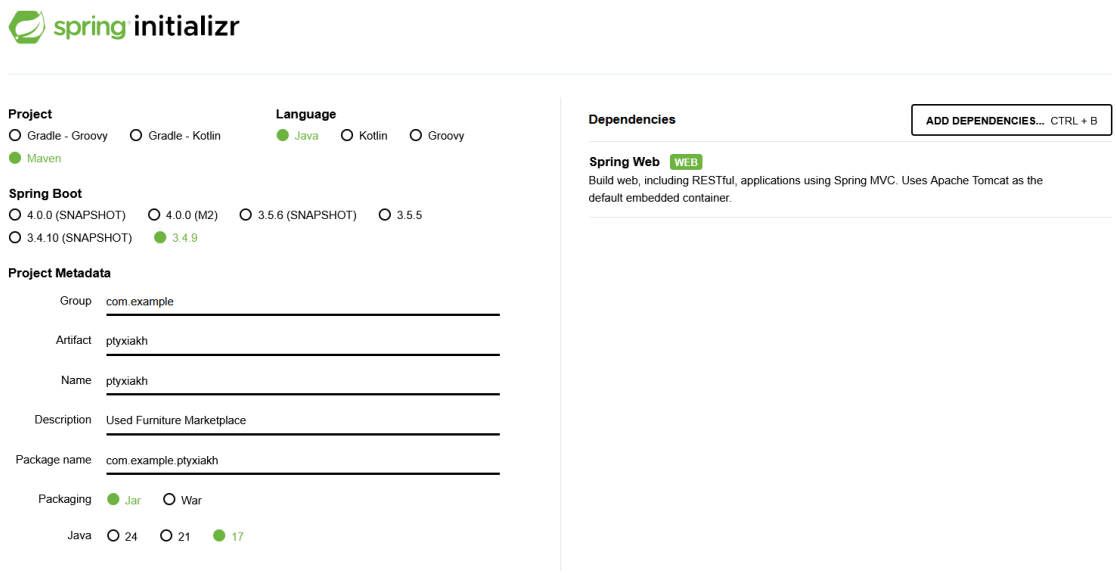
Ο πίνακας, συνδέεται με τον πίνακα User με foreign key το buyer_id και τον πίνακα Furniture με foreign key το product_id.

4.1.4 Πίνακας user_favorites

Ο πίνακας user_favorites, χρησιμοποιείται για τα αγαπημένα του χρήστη. Συνδέει τον πίνακα user με τον πίνακα furniture. Περιέχει ως πεδία, το furniture_id που δείχνει την αγγελία και το user_id που δείχνει τον χρήστη που έχει την αντίστοιχη αγγελία στα αγαπημένα του.

4.2 Backend

Αρχικά, για την υλοποίηση του backend, χρειάστηκε να εγκατασταθεί η τεχνολογία SpringBoot μέσω Spring Initializr, όπου επιλέχθηκε η χρήση Maven. Η γλώσσα προγραμματισμού που επιλέχθηκε είναι Java 17, ενώ η έκδοση της SpringBoot 3.4.3. Για αρχή, προστέθηκε το dependency Spring Web, η οποία περιλαμβάνει όλα τα απαραίτητα για την δημιουργία web εφαρμογών, και βοηθάει στην υλοποίηση μιας RESTful πλατφόρμας μέσω του Spring MVC, με την χρήση Tomcat.



Εικόνα 4.1 Spring Initializer

Κατά τη διάρκεια υλοποίησης της πλατφόρμας, χρησιμοποιήθηκαν επίσης τα εξής dependencies:

spring-boot-starter-security: Προσθέτει Spring Security στην εφαρμογή, και χρησιμοποιείται για authentication των χρηστών, δηλαδή τον έλεγχο ταυτότητας τους, και authorization, δηλαδή τον καθορισμό δικαιωμάτων που έχει ο χρήστης ανάλογα τον ρόλο του.

spring-boot-starter-data-jpa: Περιέχει Spring Data JPA για σύνδεση και τη διαχείριση της βάσης δεδομένων. Υποστηρίζει την δημιουργία των Repositories που επεκτείνουν JpaRepository, και χρησιμοποιείται για CRUD (save, findAll, findById, deleteById) λειτουργίες με απλό τρόπο, χωρίς την ανάγκη για γραφή SQL queries.

spring-boot-starter-test: Περιλαμβάνει βιβλιοθήκες που χρησιμοποιούνται για testing των Spring components, δηλαδή την δοκιμή των Repositories, Services και Controllers.

spring-boot-starter-validation: Παρέχει annotations, όπως @NotNull,@Email, @Size, που χρησιμοποιούνται για τα entities και τα DTOs ώστε τα δεδομένα που φτάνουν στη βάση, είναι σωστά και έγκυρα.

4.2.1 Security

Η διαμόρφωση της ασφάλειας της πλατφόρμας, έγινε με την χρήστη Spring Security. Πιο συγκεκριμένα, το @Configuration annotation δηλώνει ότι η κλάση SecurityConfig είναι κλάση

ρυθμίσεων και φορτώνεται κατά την εκκίνηση της πλατφόρμας. Χρησιμοποιείται το PasswordEncoder για το hashing των κωδικών. Αυτό γίνεται με τον αλγόριθμο BCrypt ο οποίος κάνει hash τον κωδικό, χωρίς να μπορεί να μετατραπεί πίσω ο κωδικός από το hash. Η μέθοδος securityFilterChain ρυθμίζει την συμπεριφορά των αιτημάτων όσον αφορά την ασφάλεια. Η πλατφόρμα, προς το παρόν, δεν παρέχει κάποια περαιτέρω ασφάλεια.

4.2.2 Data Initializer

Η κλάση DataInitializer, υλοποιεί το CommandLineRunner ώστε να εκτελείται αυτόματα με την εκκίνηση τους πλατφόρμας. Χρησιμοποιείται για την δημιουργία του διαχειριστή, και εξασφαλίζει ότι πάντα θα υπάρχει admin χρήστης.

Συγκεκριμένα, ελέγχει αν υπάρχει χρήστης με ρόλο “ADMIN”, χρησιμοποιώντας τη μέθοδο findByRole του userRepository. Εάν δεν βρεθεί, δημιουργεί νέο User αντικείμενο στο οποίο του δίνει τα στοιχεία username, fullName, email, role, accountStatus του Admin, και password το οποίο κρυπτογραφεί με την βοήθεια του BcryptPasswordEncoder. Δίνει στον διαχειριστή την default εικόνα προφίλ και ύστερα αποθηκεύει τον νέο χρήστη μέσω τους μεθόδου save.

4.2.3 Entities

Η πλατφόρμα έχει τρείς entity κλάσεις, τους User, Furniture και Orders. Για την δήλωση τους χρησιμοποιείται το annotation **@Entity** πριν την δήλωση τους κλάσης, και ύστερα δίνεται και το όνομα του πίνακα στη βάση μέσω του annotation **@Table** με παράμετρο name.

Για την δημιουργία τους χρησιμοποιούνται annotation για την σωστή δήλωση των πεδίων και την δημιουργία των σχέσεων μεταξύ των αντικειμένων. Συγκεκριμένα:

@Id χρησιμοποιείται σε πεδία για να δηλώσει το primary key του αντικειμένου, δηλαδή οι στήλες που το περιλαμβάνουν θα έχουν μοναδικές τιμές σε κάθε εγγραφή.

@GeneratedValue συνδέεται με το @Id και δηλώνει πως θα γίνει η δημιουργία του. Στα αντικείμενα χρησιμοποιείται η GenerationType.IDENTITY, δηλαδή είναι auto increment και δημιουργούνται αυτόματα οι τιμές για το πεδίο.

@Column χρησιμοποιείται για τα στοιχεία των στηλών στη βάση που αντιστοιχεί στο πεδίο. Χρησιμοποιείται η παράμετρος name για να δώσει όνομα στην στήλη και nullable = false ώστε οι στήλες να μην μπορούν να έχουν null τιμές.

@NotNull χρησιμοποιείται για την πρόληψη αποθήκευσης null τιμών.

@Email χρησιμοποιείται για να ελέγξει ότι το πεδίο θα έχει έγκυρη μορφή email.

@Pattern χρησιμοποιείται για να λάβουν συγκεκριμένη μορφή τα πεδία, τους ο ταχυδρομικός κώδικας και ο αριθμός τους οδού.

@Lob δηλώνει ότι το πεδίο λαμβάνει αντικείμενα με μεγάλο όγκο. Χρησιμοποιείται στα πεδία που περιέχουν τους εικόνες.

@Size ορίζει το μέγεθος τους τιμής τους πεδίου.

@Positive ορίζει ότι ένα πεδίο μπορεί να λάβει μόνο θετικές τιμές

@JoinTable χρησιμοποιείται για να δηλώσει τον ενδιάμεσο πίνακα που συνδέει δύο αντικείμενα.

@JoinColumn δηλώνει ποιο πεδίο τους αντικειμένου συνδέεται με το primary key του αντικειμένου στο οποίο χρησιμοποιείται το annotation.

@ManyToMany δηλώνει την σχέση μεταξύ δύο αντικειμένων, στην οποία τους εγγραφές τους αντικείμενου μπορούν να συνδέονται με τους εγγραφές του άλλου αντικειμένου με το οποίο δηλώνεται η σχέση.

@ManyToOne δηλώνει την σχέση μεταξύ δύο αντικειμένων, στην οποία τους εγγραφές τους αντικειμένου μπορούν να συνδέονται με μία εγγραφή του άλλου αντικειμένου με το οποίο δηλώνεται η σχέση.

@OneToOne δηλώνει την σχέση μεταξύ δύο αντικειμένων, στην οποία μια εγγραφή τους αντικειμένου μπορεί να συνδέεται με μία εγγραφή του άλλου αντικειμένου με το οποίο δηλώνεται η σχέση.

@OneToMany δηλώνει την σχέση μεταξύ δύο αντικειμένων, στην οποία μια εγγραφή τους αντικειμένου μπορεί να συνδέεται με τους εγγραφές του άλλου αντικειμένου με το οποίο δηλώνεται η σχέση.

User Entity

Ο User entity, αναπαριστά έναν χρήστη και περιέχει βασικά του τους το id που δημιουργείται αυτόματα με @GeneratedValue. Τα πεδία username, το οποίο είναι unique για αποφυγή διπλότυπων εγγραφών, με μήκος χαρακτήρων από 4 έως 15 το οποίο δηλώνεται μέσω του @Size, το fullName, email σε μορφή email μέσω @Email, role για τον ρόλο, accountStatus για την κατάσταση του λογαριασμού, και password.

Η εικόνα προφίλ του χρήστη, αποθηκεύεται στο profilePicture με τύπο byte[] και περιέχει @Lob για να δηλώσει ότι το πεδίο θα περιέχει μεγάλα αντικείμενα, ενώ τα πεδία profilePictureName και profilePictureType, αποθηκεύουν το όνομα και τον τύπο τους φωτογραφίας.

Ο πίνακας, συνδέεται με τον πίνακα favorites, δημιουργώντας σχέση @ManyToMany μεταξύ των χρηστών και των αγγελιών επίπλων, δηλαδή ένας χρήστης μπορεί να έχει πολλές αγαπημένες αγγελίες επίπλων και μία αγγελία μπορεί να ανήκει στα αγαπημένα πολλών χρηστών. Υπάρχει επίσης η σχέση με τα έπιπλα, με @OneToMany που σημαίνει ότι ένας χρήστης μπορεί να έχει πολλές αγγελίες και κάθε αγγελία ανήκει σε έναν χρήστη. Οι πίνακες συνδέονται μέσω του πεδίου seller που περιέχει το Furniture entity. Το CascadeType.ALL σημαίνει ότι οι τους οι ενέργειες που εφαρμόζονται στον χρήστη εφαρμόζονται και στα συνδεδεμένα έπιπλα, ενώ το orphanRemoval = true σημαίνει ότι αν ένα έπιπλο αφαιρεθεί από την λίστα, θα διαγραφεί και από την βάση. Τέλος, δημιουργείται και η σχέση @OneToMany με το Orders Entity, δηλαδή ένας χρήστης μπορεί να έχει πολλές παραγγελίες ως αγοραστής και κάθε παραγγελία ανήκει σε έναν μόνο χρήστη. Συνδέονται μέσω του buyer, δηλαδή το Orders περιέχει το πεδίο buyer που δείχνει στον User. Ξανά χρησιμοποιείται το CascadeType.ALL και το orphanRemoval.

Για την πρόληψη αποθήκευσης null τιμών, προστέθηκε το @NotNull, και η παράμετρος nullable=false, εξασφαλίζει ότι τους αντίστοιχες στήλες τους βάσης δεν είναι αποδεκτές οι null τιμές.

Στις μεθόδους, περιλαμβάνονται μέθοδοι διαχείρισης αγαπημένων από την μεριά των χρηστών. Η addFavorite, προσθέτει στον πίνακα favorites μία αγγελία, ενώ η removeFavorite το αφαιρεί.

Furniture Entity

Το Furniture entity, περιέχει στοιχεία για τα έπιπλα που δημοσιεύονται. Όπως και ο πίνακας User, έχει id που δημιουργείται αυτόματα με @GeneratedValue. Περιλαμβάνει πεδία όπως name, description, price με @Positive, furnitureCondition, furnitureCategory, και state. Το κάθε έπιπλο έχει εικόνα, γι' αυτό γίνεται η χρήση των πεδίων image που δηλώνεται με @Lob, imageName και imageType για την αποθήκευση του τίτλου και τύπου τους εικόνας.

Συνδέεται με το αντικείμενο User μέσω του user_id και η σχέση που δημιουργείται είναι @ManyToMany, ενώ με τους παραγγελίες Orders η σχέση είναι @OneToOne και συνδέονται μέσω του id αγγελίας. Εδώ χρησιμοποιείται το CascadeType.ALL, orphanRemoval = true για να αποθηκευτούν τυχόν ενημερώσεις και τους παραγγελίες. Τέλος, τα Furniture, τους συνδέονται με τα αγαπημένα favorited με @ManyToMany μέσω του furniture_id.

Όλα τα πεδία διαθέτουν @NotNull annotation και τη παράμετρο nullable=false για την πρόληψη αποστολής και αποθήκευσης null τιμών στη βάση.

Περιλαμβάνει τους μεθόδους διαχείρισης αγαπημένων από την μεριά των αγγελιών. Η addFavorited που δέχεται user και ενημερώνει ποιοι χρήστες έχουν την αγγελία στα αγαπημένα τους και η removeFavorited που αντίστοιχα αφαιρεί από την favorited.

Orders Entity

Ο πίνακας Orders περιέχει τις παραγγελίες και έχει id που δημιουργείται αυτόματα με @GeneratedValue. Περιλαμβάνει στοιχεία του χρήστη όπως fullName, email και username και στοιχεία για την διεύθυνση αποστολής, όπως address, number (με περιορισμό να επιτρέπονται μόνο αριθμοί), city, postal (με περιορισμό 5 ψηφίων) τύπου String. Η ημερομηνία και ώρα αποστολής αποθηκεύονται στο πεδίο date τύπου LocalDateTime.Τα πεδία διαθέτουν το @NotNull annotation και τη παράμετρο nullable=false

Ο πίνακας, συνδέεται με τον πίνακα User μέσω του buyer_id μέσω του @JoinColumn με σχέση @ManyToOne. Αντίστοιχα, συνδέεται με τον πίνακα Furniture με foreign key το product_id, μέσω του @JoinColumn με σχέση @OneToOne.

4.2.4 DTOs (Data Transfer Objects)

Στην υλοποίηση, χρησιμοποιήθηκαν κλάσεις DTOs (Data Transfer Objects), τα οποία είναι αντικείμενα που χρησιμοποιούνται για τη μεταφορά δεδομένων μεταξύ του πελάτη και του server, χωρίς την μεταφορά ολόκληρων αντικειμένων τους βάσης. Δεν περιέχουν επιχειρησιακή λογική ούτε σχέσεις με άλλα αντικείμενα καθώς περιλαμβάνουν μόνο τα απαραίτητα δεδομένα που πρέπει να σταλούν ή να ληφθούν. Με αυτό επιτυγχάνεται ασφάλεια αφού δεν εκτίθεται ολόκληρη η δομή του entity, και απλότητα καθώς περνιούνται μόνο τα απαραίτητα δεδομένα.

UserDTO

Χρησιμοποιείται για την αποστολή και λήψη πληροφοριών που αφορούν τους χρήστες. Στις πληροφορίες περιλαμβάνονται, το id του χρήστη, το fullName, το email, το username, το role, το accountStatus και το imageUrl.

FurnitureDTO

Χρησιμοποιείται για την αποστολή και την λήψη πληροφοριών που αφορούν τις αγγελίες επίπλων. Περιέχει δεδομένα όπως το id, name, description, price, furnitureCondition, furnitureCategory, state, imageUrl και seller.

OrderDTO

Χρησιμοποιείται για την αποστολή και λήψη πληροφοριών που αφορούν τις παραγγελίες. Περιέχει πεδία όπως το id των παραγγελιών, πληροφορίες που αφορούν τον αγοραστή fullName, email, address, city, postal, number, date, πληροφορίες παραγγελιών όπως το productName με το τίτλο του προϊόντος που πουλήθηκε, productPrice και πληροφορίες του πωλητή, το sellerId και sellerUsername.

LoginRequest

Χρησιμοποιείται για την μεταφορά των στοιχείων που απαιτούνται για τη σύνδεση του χρήστη, περιέχει δηλαδή μόνο το username και το password του. Τα δεδομένα αποστέλλονται στον

controller οπού και γίνονται οι απαραίτητοι έλεγχοι και η είσοδος στην πλατφόρμα, χωρίς να εκτίθενται άλλα πεδία ή πληροφορίες.

RegisterRequest

Χρησιμοποιείται για την εγγραφή του χρήστη. Περιέχει στοιχεία όπως το username, email, fullName, password και confirmPassword, role και accountStatus. Περιλαμβάνει τη μέθοδο toUser() η οποία δημιουργεί ένα νέο User entity με τα δεδομένα του DTO. Αυτά μεταφέρονται στον controller για έλεγχο και δημιουργία χρήστη στη βάση δεδομένων.

4.2.5 Repositories

Η χρήση των Repositories, παρέχει τη δυνατότητα αλληλεπίδρασης με την βάση δεδομένων, χωρίς ανάγκη χειροκίνητης γραφής κώδικα SQL. Κάθε repository, το οποίο αντιστοιχεί σε ένα entity, δηλώνεται ως interface και επεκτείνει την JpaRepository. Ένα interface καθορίζει ποιες μεθόδους πρέπει να υλοποιεί μια κλάση χωρίς να χρειάζεται η υλοποίηση αυτών. Έτσι, παρέχει τους βασικές λειτουργίες CRUD, τους την αποθήκευση save(), ανάκτηση findById() findAll() και διαγραφή deleteById(). Παράλυτά, μπορούν να δημιουργηθούν μέθοδοι τους οποίες υλοποιεί το JPA αυτόματα με βάση το όνομα τους, μειώνοντας την γραφή κώδικα.

UserRepository

Διαχειρίζεται τους εγγραφές του User και κληρονομεί τους βασικές μεθόδους CRUD αφού επεκτείνει την JpaRepository. Περιλαμβάνει μεθόδους για την εύρεση χρηστών με βάση συγκεκριμένα στοιχεία του τους findByUsername για την εύρεση με βάση username, μέθοδο findByEmail για την εύρεση με βάση το email, και μέθοδο findByRole για την εύρεση με βάση το role. Επιστρέφουν Optional <User> καθώς υπάρχει ενδεχόμενο να μην βρεθεί κανένας χρήστης.

FurnitureRepository

Διαχειρίζεται τους εγγραφές των Furniture και επεκτείνει την JpaRepository. Περιέχει, επιπλέον μεθόδους για την εύρεση αγγελιών με βάση τον χρήστη που τη δημοσίευσε findBySeller και μεθόδους που χρησιμοποιούνται στο φιλτράρισμα.

Συγκεκριμένα υπάρχει μέθοδος που δέχεται state, category, condition και accountStatus για να βρει τις αγγελίες με βάση αυτά, μια άλλη που δέχεται state, category και accountStatus, μια που δέχεται state, condition και accountStatus και τέλος μια που δέχεται μόνο state και accountStatus.

OrdersRepository

Διαχειρίζεται τις εγγραφές τους Orders και επεκτείνει την JpaRepository. Περιλαμβάνει μέθοδο εύρεσης παραγγελιών με βάση τον αγοραστή findByBuyer και με βάση το προϊόν findByProduct.

4.2.6 Services

Στα Services, υλοποιείται η επιχειρησιακή λογική, είναι δηλαδή οι κλάσεις όπου γίνεται η επεξεργασία των δεδομένων χρησιμοποιώντας τα repositories για την αποθήκευση ή την λήψη τους. Λειτουργούν ως συνδετικός κρίκος μεταξύ των controllers και των repositories, παρέχοντας στους controllers μεθόδους που μπορούν να χρησιμοποιηθούν για την υλοποίηση των διεργασιών της πλατφόρμας. Η δήλωση τους γίνεται με το **@Service** annotation.

UserService

Παρέχει μεθόδους για την ανάκτηση χρηστών, όπως η getAllUsers για την ανάκτηση όλων των χρηστών, η findById, για την ανάκτηση με βάση το id, και η findByUsername με βάση το username.

Για την διαχείριση της κατάστασης των λογαριασμών, υπάρχει η μέθοδος disableUserById, όπου με βάση το id, δέχεται τον χρήστη, καθαρίζει την λίστα με τα αγαπημένα του και ύστερα θέτει το accountStatus σε “INACTIVE”. Για την ενεργοποίηση του αντίστοιχα, υπάρχει η μέθοδος enableUserById, όπου μέσω του id, αφού βρεθεί ο χρήστης, θέτει το AccountStatus σε "ACTIVE" και αποθηκεύει τις αλλαγές.

Για την εγγραφή του, χρησιμοποιείται η μέθοδος registerUser όπου δέχεται ένα αντικείμενο User. Πραγματοποίει έλεγχο για μοναδικότητα του username και του email, και προχωράει στο hashing του κωδικού πρόσβασης μέσω του passwordEncoder. Κάθε χρήστης που κάνει εγγραφή, λαμβάνει τον ρόλο USER αφού ο διαχειριστής δημιουργείται κατά την εκκίνηση της εφαρμογής, και κατάσταση λογαριασμού ACTIVE, ενώ αποθηκεύεται με default εικόνα προφίλ.

Η πιστοποίηση του χρήστη υλοποιείται στην μέθοδο authenticateUser, όπου γίνεται έλεγχος αν τα στοιχεία που εισάγει ο χρήστης, ταιριάζουν με εγγεγραμμένο χρήστη που έχει ACTIVE κατάσταση λογαριασμού. Ο έλεγχος κωδικού γίνεται κρυπτογραφώντας τον κωδικό που έδωσε ο χρήστης με αυτόν που είναι αποθηκευμένος στην βάση σε hashed μορφή.

Η saveUserWithoutImage μέθοδος, αποθηκεύει τον χρήστη κρατώντας ως εικόνα προφίλ την default εφόσον εκείνος δεν επιθυμεί να την αλλάξει. Χρησιμοποιείται επίσης στην μέθοδο εγγραφής χρήστη. Ελέγχει αρχικά, αν δεν πρόσθεσε εικόνα ο χρήστης, και εφόσον αυτό ισχύει, προσθέτει στα στοιχεία της εικόνας μια default εικόνα και τον αποθηκεύει.

Η μέθοδος saveUser, αποθηκεύει έναν χρήστη με διαφορετική εικόνα προφίλ από την default. Ελέγχεται αρχικά, αν ο χρήστης έχει δώσει αρχείο εικόνας ώστε να αποθηκεύσει τα απαραίτητα στοιχεία όπως την ίδια την εικόνα, τον τύπο αρχείου, και το όνομα του αρχείου.

Τέλος, η addFavorite προσθέτει και αφαιρεί αγγελίες από τα αγαπημένα του χρήστη. Βρίσκει πρώτα τον χρήστη και αντίστοιχα την αγγελία, και αφού πραγματοποιήσει έλεγχο για την εύρεση του προϊόντος μέσα στην λίστα με τα αγαπημένα, τότε αφαιρείται, ενώ αν το προϊόν δεν υπάρχει, τότε το προσθέτει στα αγαπημένα του.

FurnitureService

Περιέχει την μέθοδο getFurnitureBySeller οπού δέχεται ένα αντικείμενο User, και επιστρέφει λίστα, με όλες τις δημοσιευμένες αγγελίες ενός συγκεκριμένου χρήστη. Η εύρεση των αγγελιών γίνεται με την χρήση της findBySeller(seller) του furnitureRepository. Μέσω της findById, όπου δέχεται ένα id, επιστρέφει ένα αντικείμενο Furniture που βρίσκει με την χρήση findById(id) του Repository.

Η εύρεση όλων των αγγελιών γίνεται με τη μέθοδο getAllFurniture, όπου αφού λάβει όλες τις αγγελίες furnitureList μέσω της findAll,επιστρέφει τη λίστα.

Για το φιλτράρισμα των αγγελιών, υπάρχει η getFilteredFurniture που δέχεται category και condition. Επειδή, για την προβολή των αγγελιών, χρειάζεται να εμφανίζονται μόνο οι διαθέσιμες αγγελίες και μόνο των χρηστών με ενεργό λογαριασμό, αρχικοποιείται η state με την τιμή " available ", και το accountStatus με τιμή "ACTIVE". Λαμβάνονται περιπτώσεις, ανάλογα με το ποιες τιμές υπάρχουν, ώστε αν ο χρήστης έχει επιλέξει κατηγορία και κατάσταση, χρησιμοποιεί την αντίστοιχη μέθοδο του furnitureRepository, αν έχει επιλέξει κατηγορία χρησιμοποιεί την μέθοδο που έχει category, state και accountStatus ενώ αν έχει επιλέξει μόνο κατάσταση, χρησιμοποιεί την μέθοδο που δέχεται condition, state και accountStatus. Αν δεν έχει επιλέξει κανένα από τα δύο, τότε επιστρέφει όσα έλαβε από την μέθοδο που δέχεται μόνο state και accountStatus.

Η deleteById, πραγματοποιεί διαγραφή μίας αγγελίας, βρίσκοντας αρχικά την αγγελία μέσω του id, και αφού λάβει όλους τους χρήστες και ελέγξει αν η αγγελία βρίσκεται στα αγαπημένα του χρήστη, την αφαιρεί, αποθηκεύει τον χρήστη με τις αλλαγές και με το delete(furniture) το οποίο δέχεται ως όρισμα το αντικείμενο furniture.

Επιπλέον, με την ίδια λογική όπως στο UserService, υπάρχει και στις αγγελίες μέθοδος saveFurniture που αποθηκεύει την αγγελία με εικόνα, καθώς δέχεται αντικείμενο Furniture και MultipartFile αρχείο και αφού λάβει τα δεδομένα της εικόνας, αποθηκεύει την αγγελία με την μέθοδο save.

Έτσι, υπάρχει και μέθοδος saveFurnitureWithoutImage οπού αποθηκεύει μια αγγελία με την ήδη υπάρχουσα εικόνα. Δέχεται ως όρισμα ένα αντικείμενο Furniture και αφού πραγματοποιήσει ελέγχους για την εύρεση εικόνας ώστε να λάβει τα ήδη υπάρχοντα στοιχεία της, επιστρέφει αποθηκευμένο όλο το furniture μέσω του save.

OrdersService

Οι βασικές λειτουργίες που υλοποιούνται, είναι μέθοδοι επιστροφής παραγγελιών, όπως η getAllOrders που βρίσκει όλες τις παραγγελίες, η findById που βρίσκει με βάση το id, η getOrdersByBuyer, που βρίσκει παραγγελίες με βάση τον αγοραστή και η getOrdersByFurniture που βρίσκει με βάση την αγγελία.

Υπάρχει επίσης μέθοδος διαγραφής παραγγελίας deleteById, και μέθοδος αποθήκευσης saveOrder, η οποία δημιουργεί νέα παραγγελία και η αγγελία που αγοράστηκε αφαιρείται αυτόματα από τη λίστα αγαπημένων όλων των χρηστών που την είχαν αποθηκεύσει.

4.2.7 Controllers

Οι Controllers, αποτελούν το επίπεδο της εφαρμογής που διαχειρίζεται την επικοινωνία του Frontend με την επιχειρησιακή λογική, δηλαδή τα Services. Μέσα από αυτούς πραγματοποιείται η παραλαβή και η επεξεργασία HTTP requests, των αιτημάτων δηλαδή που στέλνει ο χρήστης, και αφού πραγματοποιήσουν ελέγχους, επιστρέφουν την απάντηση στον client μέσω HTTP responses, δηλαδή η απάντηση του server σε μορφή JSON. Το JSON, αποτελεί μια μορφή ανταλλαγής δεδομένων μεταξύ του frontend και του backend.

Η δήλωση τους γίνεται μέσω του **@RestController** annotation, το οποίο επιτρέπει στις κλάσεις να διαχειρίζονται REST API αιτήματα, και παράλληλα περιέχουν το **@CrossOrigin**(“http://localhost:3000”), το οποίο καθιστά δυνατή την επικοινωνία με το frontend (την React) που τρέχει στην διεύθυνση αυτή. Κάθε controller, χρησιμοποιεί **@RequestMapping**, το οποίο ορίζει τη διαδρομή των αιτημάτων που πρόκειται να εξυπηρετήσει και βοηθά στην αντιστοίχιση των αιτημάτων με τις μεθόδους των Controller, ανάλογα τις λειτουργίες που πραγματοποιεί κάθε φορά το API. Δημιουργεί δηλαδή, ένα base URL για όλα τα endpoint των Controller. Τα **endpoint**, αποτελούν την διεύθυνση URL μιας μεθόδου, από την οποία ορίζεται η πρόσβαση της.

Στους Controllers, υπάρχουν διάφορα annotations για τους διαφορετικούς τύπους αιτημάτων.

@PostMapping, το οποίο διαχειρίζεται HTTP POST αιτήματα δίνει το mapping για συγκεκριμένο URL σε μέθοδο όπου σχετίζονται με την αποστολή ή δημιουργία δεδομένων που έχουν επέλθει μέσω των POST requests από τον χρήστη.

@GetMapping, χρησιμοποιείται για αιτήματα τύπου GET, που ανακτά δεδομένα από την βάση με την βοήθεια των Services.

@PutMapping είναι υπεύθυνο για την ενημέρωση υπάρχοντών δεδομένων, με τα πεδία που στάλθηκαν από το HTTP Request του χρήστη.

@DeleteMapping, δίνει την δυνατότητα σε μεθόδους να διαχειρίζονται HTTP DELETE αιτήματα, τα οποία αφορούν την διαγραφή μέσω του αιτήματος που στέλνει ο χρήστης από την χρήση της εφαρμογής.

Οι μέθοδοι, χρησιμοποιούν επίσης annotation για την διαφοροποίηση των ορισμάτων που δέχονται.

@PathVariable χρησιμοποιείται για την λήψη τιμής μέσω του URL ενός endpoint (για παράδειγμα "/users/{id}"). Το endpoint που το περιέχει δηλαδή, αναφέρεται σε συγκεκριμένο αντικείμενο.

@RequestParam χρησιμοποιείται για να λάβει παραμέτρους από το URL.

@RequestPart χρησιμοποιείται για φόρμες με διάφορα είδη δεδομένων. Δίνει την δυνατότητα λήψης μέρος του request.

@RequestBody χρησιμοποιείται για να λάβει δεδομένα από το JSON body. Η χρήση του γίνεται σε μεθόδους που υλοποιούν δημιουργία ή επεξεργασία αντικειμένου.

AdminController

Ο AdminController, έχει τις λειτουργίες που μπορεί να πραγματοποιήσει ο Admin. Ορίζεται με @RequestMapping("/admin") URL και περιλαμβάνει λειτουργίες προβολής, διαγραφής και διαχείρισης των Entities που έχει η πλατφόρμα.

Αναφορικά με τις λειτουργίες που περιλαμβάνουν τον χρήστη, έχει αρχικά την μέθοδο getAllUsers, οπού ορίζεται με @GetMapping("/users") URL, και πραγματοποιεί την εύρεση όλων των χρηστών. Αυτό επιτυγχάνεται μέσω της εύρεσης όλων των χρηστών από την μέθοδο getAllUsers του userService, και ύστερα την δημιουργία μίας UserDTO λίστας, για την επιστροφή όλων των χρηστών της πλατφόρμας.

Επιπλέον, περιέχει την μέθοδο disableUser που δηλώνεται @PutMapping("/usersDisable/{id}") endpoint, με όρισμα @PathVariable id, για την απενεργοποίηση λογαριασμού ενός χρήστη. Έτσι, αφού βρει τον χρήστη μέσω του findById(id), και πραγματοποιήσει έλεγχο εγκυρότητας, χρησιμοποιεί την μέθοδο του UserService την disableUserById(id). Αντίστοιχα λειτουργεί και η μέθοδος enableUser, για την ενεργοποίηση του λογαριασμού, με mapping @PutMapping("/usersEnable/{id}") και @PathVariable id όρισμα. Αυτή τη φορά, καλείται η μέθοδος enableUserById(id) του userService.

Μέσα από τον χρήστη, μπορεί να βρεθούν οι αγγελίες και οι παραγγελίες του χρήστη. Η μέθοδος getUserAds με @GetMapping("/users/{userId}/furniture"), δέχεται το @PathVariable id, βρίσκει τον χρήστη, πραγματοποιεί έλεγχο εγκυρότητας, λαμβάνει τις αγγελίες μέσω της μεθόδου getFurnitureBySeller(user) του furnitureService και δημιουργεί μια λίστα FurnitureDTO όπου αποθηκεύει όλες τις αγγελίες επιστρέφοντας την λίστα στο frontend.

Για την εύρεση παραγγελιών του, με την ίδια λογική, η getUserOrders μέθοδος που δηλώνεται με @GetMapping("/users/{userId}/orders"), δέχεται το @PathVariable userId και χρησιμοποιεί την μέθοδο getOrdersByBuyer(buyer) του ordersService, δημιουργεί μια λίστα OrderDTO, όπου αποθηκεύει όλες τις παραγγελίες επιστρέφοντας την λίστα στο frontend.

Για τις λειτουργίες των αγγελιών, υπάρχουν οι μέθοδοι getAllFurniture() που δηλώνεται @GetMapping("/furniture") και επιστρέφει την λίστα με όλες τις αγγελίες μέσω της getAllFurniture() του furnitureService. Η μέθοδος που αφορά την διαγραφή αγγελιών, η deleteFurniture , ορίζεται με endpoint @DeleteMapping("/furniture/{id}") και δέχεται @PathVariable id το οποίο χρησιμοποιεί για να βρει την αγγελία μέσω findById(id), και να την διαγράψει χρησιμοποιώντας deleteById(id) του furnitureService.

Για τις λειτουργίες των παραγγελιών, υπάρχουν οι μέθοδοι η getAllOrders() όπου δηλώνεται με @GetMapping("/orders"), και επιστρέφει την λίστα με όλες τις παραγγελίες μέσω getAllOrders του ordersService. Η μέθοδος που αφορά την διαγραφή παραγγελιών, η deleteOrders, ορίζεται με @DeleteMapping("/orders /{id}") και δέχεται @PathVariable id το οποίο χρησιμοποιεί για να βρει

την παραγγελία μέσω findById(id), και να την διαγράψει χρησιμοποιώντας ordersService.deleteById(id).

AuthController

Ο AuthController, είναι υπεύθυνος για την διαχείριση αυθεντικοποίησης των χρηστών της εφαρμογής, δηλαδή την εγγραφή και την σύνδεση ενός χρήστη. Το URL καθορίζεται μέσω του @RequestMapping("/auth"). Υλοποιούνται δύο βασικές λειτουργίες, η εγγραφή και η σύνδεση.

Πιο συγκεκριμένα, για την εγγραφή registerUser που ορίζεται με @PostMapping(“/register”), δέχεται ως όρισμα ένα RegisterRequest που περιλαμβάνει τα στοιχεία που έχει συμπληρώσει ο χρήστης κατά την εγγραφή, και αρχικά ελέγχει αν ο κωδικός password που συμπληρώθηκε με την επιβεβαίωση confirmPassword, ταιριάζουν. Αν οι κωδικοί ταιριάζουν, προχωράει στην δημιουργία ενός αντικειμένου User μέσω της μεθόδου toUser() του RegisterRequest, και στη συνέχεια γίνεται κλήση της μεθόδου του UserService, την registerUser(user) για να αποθηκεύσει τον νέο χρήστη στη βάση. Το password, ορίζεται null για λόγους ασφαλείας, δηλαδή για να μην επιστραφεί ο κωδικός μαζί με το αντικείμενο, και τελικά επιστρέφει HTTP request ok() μαζί με το αντικείμενο του χρήστη.

Η μέθοδος login που χρησιμοποιείται για την σύνδεση, ορίζεται με @PostMapping(“/login”). Δέχεται ένα LoginRequest αντικείμενο από το οποίο λαμβάνει το username και password που έχει εισάγει ο χρήστης. Καλείται μέσω του UserService η μέθοδος authenticateUser(username,password) και πραγματοποιεί έλεγχο, οπού εφόσον ο user υπάρχει, το password γίνεται null ξανά για λόγους ασφαλείας και επιστρέφει τα δεδομένα του χρήστη σε μορφή UserDTO.

UserController

Ο controller, περιλαμβάνει τις λειτουργίες που αφορούν τον χρήστη και το προφίλ του στην εφαρμογή. Ορίζεται με URL mapping @RequestMapping("/user").

Στις μεθόδους περιλαμβάνεται η getAllUsers με @GetMapping("/users"), η οποία επιστρέφει λίστα με όλους τους χρήστες, χρησιμοποιώντας την μέθοδο getAllUsers του UserService.

Η showProfile μέθοδος με @GetMapping("/profile"), δέχεται ένα id μέσω του @RequestParam, και προχωράει σε έλεγχο, αν ο χρήστης βρεθεί, επιστρέφει το αντικείμενο User.

Για την ενημέρωση προφίλ του χρήστη, υπάρχει η μέθοδος updateUser με @PutMapping ("/updateProfile/{id}"), και ορίσματα @PathVariable Long id και @RequestParam με παράμετρο required=false, οπότε αν κάποιο πεδίο δεν αποσταλεί, τότε δεν ενημερώνεται, για τα στοιχεία username, email, fullName και MultipartFile profilePicture. Έτσι, αφού βρεθεί ο χρήστης από το id, εάν έχει συμπληρώσει τα στοιχεία για αλλαγή, δίνονται τιμές στα πεδία της βάσης, βάζοντας τα καινούρια στοιχεία. Αν ο χρήστης έχει προσθέσει εικόνα, κάνει αποθήκευση μέσω του saveUser, ενώ αν δεν προσθέσει αποθηκεύονται οι αλλαγές μέσω του saveUserWithoutImage.

Για την ανάκτηση της φωτογραφίας προφίλ του χρήστη, υλοποιήθηκε η μέθοδος getUserProfileImage που έχει @GetMapping("/users/{id}/image") και όρισμα @PathVariable id. Η μέθοδος βρίσκει αρχικά τον user μέσω του findById και αποθηκεύει σε μια byte[] μεταβλητή, την εικόνα του χρήστη. Η μέθοδος, φτιάχνει και επιστρέφει ένα HTTP response, το οποίο ορίζει τι τύπος δεδομένων (σε αυτή τη περίπτωση τον τύπο της εικόνας) περιέχει το binary data της εικόνας.

Η μέθοδος ανάκτησης αγαπημένων αγγελιών του χρήστη, getFavorites, με @GetMapping("/users/{id}/favorites"), και όρισμα @PathVariable id, παίρνει το id του χρήστη, και αν υπάρχει, αποθηκεύει σε μια λίστα <Furniture> favorites τα αγαπημένα του χρήστη. Δημιουργεί νέα FurnitureDTO λίστα, την favoritesDTO όπου αποθηκεύει τα αγαπημένα και τα επιστρέφει.

Η προσθήκη ενός επίπλου στα αγαπημένα, γίνεται μέσω του addFavorites με @PostMapping("/users/{userId}/favorites/{furnitureId}") και τα @PathVariable userId, και @PathVariable furnitureId ως ορίσματα. Καλεί την userService.addFavorite(userId, furnitureId), βρίσκει τον χρήστη μέσω του id του, και αποθηκεύει σε μια λίστα <Furniture> favorites τα αγαπημένα του χρήστη, δημιουργεί νέα FurnitureDTO λίστα favoritesDTO, όπου αποθηκεύει τα αγαπημένα και τα επιστρέφει.

FurnitureController

Στον FurnitureController υλοποιούνται οι λειτουργίες που αφορούν τις αγγελίες επίπλων. Ορίζεται με URL @RequestMapping("/furniture").

Για την δημιουργία των αγγελιών, υπάρχει η μέθοδος createFurniture που ορίζεται με @PostMapping("/create") και δέχεται το αντικείμενο furniture και την εικόνα ως MultipartFile μέσω @RequestPart για multipart data requests. Η μέθοδος αρχικά πραγματοποιεί έλεγχο, βρίσκοντας αν υπάρχει ο χρήστης που δημιουργεί την αγγελία. Αφού βρει τον πωλητή, προχωράει δίνοντας τιμή στον seller του furniture, θέτοντας το state σε available και αποθηκεύοντας την αγγελία μέσω furnitureService.saveFurniture. Τελικά επιστρέφει HTTP request ok() μαζί με το αντικείμενο της αγγελίας

Για τις πληροφορίες μιας αγγελίας, υπάρχει η μέθοδος getFurnitureDetails που δηλώνεται με @GetMapping("/details/{id}") και όρισμα το id. Αφού βρεθεί η αγγελία μέσω του findById(id) του furnitureService,. Δημιουργεί ένα αντικείμενο FurnitureDTO για να λάβει τα δεδομένα, και το επιστρέφει στο frontend.

Η λίστα με τις αγγελίες ενός χρήστη, επιστρέφεται μέσω της μεθόδου getMyAds όπου ορίζεται με το @GetMapping("/myAds") και δέχεται το id του χρήστη. Αφού βρεθεί ο χρήστης με την μέθοδο findById του userService, γίνεται έλεγχος εγκυρότητας, και αποθηκεύονται σε μια λίστα myAds, όλες οι δημοσιευμένες αγγελίες του χρήστη με την χρήση getFurnitureBySeller(user) του furnitureService. Τέλος, δημιουργεί μια λίστα FurnitureDTO, όπου κάθε αγγελία την μετατρέπει σε FurnitureDTO, και επιστρέφει την λίστα στο frontend.

Η διαγραφή μιας αγγελίας πραγματοποιείται μέσω της deleteFurniture οπού χρησιμοποιείται το @DeleteMapping("/delete/{id}") λαμβάνοντας για όρισμα το id. Αφού βρεθεί η αγγελία μέσω του findById(id) και γίνει ο έλεγχος εγκυρότητας, χρησιμοποιείται η μέθοδος του Service deleteById(id) για να πραγματοποιηθεί η διαγραφή.

Ο χρήστης έχει την δυνατότητα επεξεργασίας της αγγελίας. Αυτό υλοποιείται με την μέθοδο updateFurniture με endpoint @PutMapping("/updateFurniture/{id}"). Η μέθοδος δέχεται ως ορίσματα το id που λαμβάνει μέσω του mapping, το name, description, price, furnitureCondition, furnitureCategory και MultipartFile image τα οποία έχουν όλα παράμετρο required=false ώστε αν κάποιο πεδίο δεν αποσταλεί, τότε δεν ενημερώνεται. Αρχικά βρίσκεται η αγγελία μέσω findById(id) και γίνεται ο έλεγχος εγκυρότητας. Προχωράει ελέγχοντας για κάθε στοιχείο, αν υπάρχει τιμή συμπληρωμένη από τον χρήστη, ώστε να την ορίσει στο αντίστοιχο πεδίο. Για την αλλαγή της εικόνας, ελέγχεται αν υπάρχει εικόνα, και προχωράει σε αποθήκευση της αγγελίας με την χρήση saveFurniture του furnitureService, ενώ αν η εικόνα δεν αποσταλεί από τον χρήστη, χρησιμοποιεί την μέθοδο saveFurnitureWithoutImage ώστε να αποθηκευτούν τα στοιχεία με την ήδη υπάρχουσα εικόνα.

Ο FurnitureController, περιλαμβάνει μέθοδο για το φιλτράρισμα των αγγελιών getFilteredFurniture με @GetMapping("/filteredAds"). Δέχεται ως παραμέτρους τα category condition, και sortOption χωρίς να είναι required καθώς η χρήση φίλτρων είναι προαιρετική. Η μέθοδος, καλεί την getFilteredFurniture του furnitureService δίνοντας το category και condition που έλαβε από τον χρήστη, και αποθηκεύει τις αγγελίες στο furnitures. Ελέγχει αν ο χρήστης έχει επιλέξει sortOption για την ταξινόμηση, και παίρνει περιπτώσεις, ταξινομώντας την λίστα με τις αγγελίες με βάση την τιμή. Επιστρέφει την λίστα φιλτραρισμένη με χρήση FurnitureDTO.

Τελευταία μέθοδος του Controller, είναι η μέθοδος ανάκτησης εικόνας της αγγελίας, δηλαδή η getImageByFurnitureId με @GetMapping("furnitures/{id}/image"). Όπως και στον user, λαμβάνει

όρισμα το id της αγγελίας την αναζητεί μέσω της μεθόδου findById(furnitureId). Έτσι, αποθηκεύει σε μια byte[] μεταβλητή, την εικόνα του της αγγελίας δημιουργώντας και επιστρέφοντας ένα HTTP response, το οποίο έχει συγκεκριμένο τύπο περιεχομένου που προέρχεται από το furniture.getImageType().

OrdersController

Ο OrdersController, περιέχει τις λειτουργίες που αφορούν τις παραγγελίες και ορίζεται με URL @RequestMapping("/order").

Αρχικά, ο Controller περιέχει μέθοδο για την δημιουργία μιας παραγγελίας, η οποία δηλώνεται με την χρήση του @PostMapping("/createOrder") και δέχεται ένα @RequestBody αντικείμενο Orders. Βρίσκει τον αγοραστή μέσω του userService.findById και ορίζει τον buyer, το username του και την ημερομηνία και ώρα αγοράς μέσω του LocalDateTime.now(). Με την ίδια λογική, βρίσκει το id της αγγελίας από το furnitureService.findById, και θέτει την κατάσταση της αγγελίας σε “unavailable” αποθηκεύοντας το τελικά με την χρήση της μεθόδου saveFurnitureWithoutImage. Τέλος, δίνει στην orders την αγγελία, και το αποθηκεύει μέσω της μεθόδου ordersService.saveOrder(orders). Η μέθοδος επιστρέφει την παραγγελία στο frontend.

Ο OrdersController, έχει επίσης μέθοδο για την εύρεση όλων των αγγελιών ενός χρήστη. Η μέθοδος ορίζεται με endpoint @GetMapping("/myOrders") και δέχεται ως παράμετρο το @RequestParam id του user που πραγματοποιεί το αίτημα. Αφού ο χρήστης βρεθεί με τη βοήθεια του userService.findById(userId), προχωράει σε εύρεση της λίστας με τις παραγγελίες. Η λίστα βρίσκεται μέσω του ordersService.getOrdersByBuyer(buyer), και δημιουργείται μια νέα κενή λίστα orderDTO στην οποία θα μπουν οι παραγγελίες την οποία επιστρέφει στο frontend.

Η μέθοδος getBuyerInfo βρίσκει τις πληροφορίες μιας παραγγελίας μέσω της αγγελίας. Δέχεται ως όρισμα το @RequestParam furnitureId και αφού βρει την αγγελία, βρίσκει την παραγγελία που υπάρχει γι’ αυτό το έπιπλο, και με τη χρήση orderDTO, επιστρέφει τα στοιχεία της παραγγελίας.

Η getOrderDetails, χρησιμοποιείται για να λάβει ο χρήστης λεπτομέρειες παραγγελίας. Ορίζεται με @GetMapping("/details/{id}") και έχει παράμετρο @PathVariable id, βρίσκει την παραγγελία μέσω του id, και δημιουργεί νέο orderDTO το οποίο και επιστρέφει.

Τέλος, για την διαγραφή μιας παραγγελίας, υπάρχει η μέθοδος deleteOrder που ορίζεται με @DeleteMapping("/delete/{id}"). Δέχεται @PathVariable id, και αλλάζει το state της αγγελίας από την οποία πραγματοποιήθηκε η παραγγελία, σε "available", αφού η παραγγελία πλέον δεν θα υπάρχει και ορίζει την μεταξύ τους σχέση σε null. Αποθηκεύει τις αλλαγές που έγιναν στην αγγελία χρησιμοποιώντας saveFurnitureWithoutImage, και προχωράει στην διαγραφή της παραγγελίας.

4.3 Frontend

Σε αυτήν την ενότητα, θα χρησιμοποιηθούν βασικές έννοιες του frontend όπως το JSON, τα Promise, τα async και await, την λειτουργία fetch, axios και χαρακτηριστικά όπως props, states και hooks.

Το **JSON** (JavaScript Object Notation), αποτελεί μια μορφή δεδομένων που χρησιμοποιείται για την ανταλλαγή δεδομένων μεταξύ του frontend και του backend. Η μορφή που έχει είναι σε κείμενο, αλλά μέσα μπορεί να περιλαμβάνει διάφορους τύπους δεδομένων (Αντικείμενα, λίστες, strings, αριθμούς, Boolean και null). Η ανταλλαγή των δεδομένων γίνεται συνήθως με το backend να στέλνει δεδομένα σε JSON μορφή και με την αποστολή JSON σε API.

Το **Promise**, είναι ένα αντικείμενο το οποίο αντιπροσωπεύει την ολοκλήρωση ή την αποτυχία μιας ασύγχρονης ενέργειας και χρησιμοποιείται για τον χειρισμό των αποτελεσμάτων από λειτουργίες όπως τα HTTP requests που παίρνουν χρόνο. Έχει τρείς καταστάσεις, την pending, όπου δείχνει ότι η ενέργεια δεν έχει ολοκληρωθεί, την Fulfilled, που δείχνει ότι ολοκληρώθηκε και την Rejected, που δείχνει ότι η ενέργεια απέτυχε. Λειτουργεί ως δέσμευση προς την συνάρτηση ότι θα ολοκληρωθεί επιτυχώς ή χωρίς επιτυχία το αίτημα.

Το **async**, χρησιμοποιείται πριν από συναρτήσεις για να δηλώσει ότι η συνάρτηση είναι ασύγχρονη. Αυτό σημαίνει ότι αυτή πάντα επιστρέφει ένα Promise και μπορεί να χρησιμοποιεί μέσα της το await.

Το **await** χρησιμοποιείται μέσα σε async συναρτήσεις για να διακόψει προσωρινά την εκτέλεση της μέχρι η κατάσταση του Promise να είναι ολοκληρωμένη.

Η **fetch**, αποτελεί μέθοδο της JavaScript, που χρησιμοποιείται για την εκτέλεση HTTP αιτημάτων προς έναν server. Επιστρέφει ένα αντικείμενο Promise και χρησιμοποιείται για να φέρει δεδομένα από τα APIs.

Το **axios** είναι μια βιβλιοθήκη για HTTP requests που έχει την δυνατότητα να μετατρέπει αυτόματα τα δεδομένα σε JSON, να ακυρώνει αιτήματα και να διαχειρίζεται τα σφάλματα καλύτερα.

Τα **props** είναι αντικείμενα που χρησιμοποιούνται σε React components για τη μεταφορά δεδομένων από component σε component. Αυτά μπορούν μόνο να αναγνωστούν, και ο component που τα χρησιμοποιεί δεν έχει την δυνατότητα να τα αλλάξει.

Τα **states** είναι μεταβλητές οι οποίες κατέχουν τις τιμές που μπορούν να αλλάξουν μέσα σε έναν component. Όταν αυτό αλλάζει, το component ανανεώνεται για να εμφανίσει τα νέα δεδομένα. Δηλώνονται με την χρήση του useState().

Τα **hooks**, είναι συναρτήσεις σε React που επιτρέπουν στα function components να έχουν states και άλλες λειτουργίες. Κάποια από τα hooks που χρησιμοποιούνται είναι τα useState(), για state, useEffect() για ενέργειες όπως το fetch, API calls.

Σε πολλές σελίδες, περιλαμβάνονται επίσης hooks όπως το useEffect(), useNavigate(), useLocation(), useParams() και useCallback().

Το **useEffect()** χρησιμοποιείται για την σύνδεση με το backend.

Το **useParams(),** χρησιμοποιείται για να ληφθεί η παράμετρος από το URL. Θα συναντηθεί δηλαδή, σε σελίδες με δυναμικό URL για τις οποίες χρειάζεται το id του χρήστη, της αγγελίας ή της παραγγελίας.

Το **useNavigate()**, χρησιμοποιείται για την περιήγηση στην πλατφόρμα, αφού ανακατευθύνει τον χρήστη σε άλλη σελίδα.

Το **useLocation()**, επιστρέφει το τωρινό URL, και χρησιμοποιείται σε σελίδες όπου χρειάζεται η το URL για να εκτελέσει διαφορετική λειτουργία με βάση αυτό.

Το **useCallback()**, χρησιμοποιείται για να αποθηκεύσει μια συνάρτηση, χωρίς να την τρέχει κάθε φορά, παρά μόνο όταν αλλάζουν οι εξαρτήσεις της.

4.3.1 Components

Στην React, τα components αποτελούν κομμάτια κώδικα που μπορούν να επαναχρησιμοποιηθούν σε διαφορετικές σελίδες και αρχεία JSX. Τα components μπορούν να δέχονται δεδομένα μέσω των props και να προσαρμόζουν την εμφάνιση και συμπεριφορά τους με βάση αυτά, επιστρέφοντας την περιγραφή του UI που χρειάζεται να εμφανιστεί στην σελίδα.

Στην ανάπτυξη χρησιμοποιούνται κυρίως functional components, δηλαδή συναρτήσεις που μπορούν να κάνουν χρήση React hooks όπως useState, useEffect. Αυτά, επιτρέπουν στα components να διατηρούν state, να εκτελούν ενέργειες ενημερώνουν αλλαγές που γίνονται στα δεδομένα.

Στην πλατφόρμα υπάρχουν διάφορα components με διαφορετικό ρόλο όπως την παρουσίαση δεδομένων, τη διαχείριση φορμών, την εμφάνιση πινάκων και την διαχείριση ενεργειών του χρήστη. Η χρήση τους, γίνεται μέσω των import, όπου δηλώνονται με τη μορφή «import ComponentName from "../components/componentName";»

LogoutButton

Η logoutButton χρησιμοποιείται για τη διαχείριση της αποσύνδεσης ενός χρήστη από τη πλατφόρμα. Αρχικά, ορίζεται η συνάρτηση handleLogout, η οποία διαγράφει τα δεδομένα που είναι αποθηκευμένα στο localStorage του browser που περιέχουν το userId και έπειτα με τη βοήθεια του navigate ανακατευθύνει τον χρήστη στην σελίδα σύνδεσης (“/login”). Επιστρέφει ένα

κουμπί το οποίο όταν πατηθεί θα υλοποιήσει τη handleLogOut συνάρτηση εκτελώντας την αποσύνδεση και ανακατεύθυνση του χρήστη.

AdminTable

Το component αυτό, δημιουργεί δυναμικά πίνακα για την παρουσίαση δεδομένων στις λειτουργίες του admin. Ο σκοπός του είναι να προσαρμόζεται στις ανάγκες προβολής των δεδομένων της πλατφόρμας, ώστε ο διαχειριστής να δει τους χρήστες, τις αγγελίες και τις παραγγελίες.

Δέχεται τρία βασικά props, τα { columns, data, noDataMessage }, όπου το columns είναι πίνακας αντικειμένων ο οποίος καθορίζει τη δομή και το πλήθος των στηλών που θα υπάρχουν ανάλογα με τα δεδομένα που θέλει να δει. Με το μοναδικό key που περιέχει, αναγνωρίζει τη στήλη και το label εμφανίζεται ως τίτλος της στήλης. Το data, αντιπροσωπεύει τις γραμμές του πίνακα, καθώς κάθε αντικείμενο αντιστοιχεί σε μία γραμμή. Τέλος, η noDataMessage, χρησιμοποιείται σε περίπτωση που ο πίνακας δεν έχει διαθέσιμα δεδομένα.

Δημιουργούνται οι τίτλοι των στηλών χρησιμοποιώντας όπου περιέχει το key για την αναγνώριση της στήλης, και το label για τον ορισμό του ονόματος της. Ελέγχει αν υπάρχουν δεδομένα (data.length > 0) ώστε αν υπάρχουν, για κάθε αντικείμενο δημιουργεί μια γραμμή με το και τις στήλες. Σε αντίθετη περίπτωση, εμφανίζει μήνυμα noDataMessage σε μία ενιαία γραμμή. Για το styling, χρησιμοποιείται το CSS module admin.module.css

MenuBar

Για την εμφάνιση μενού πλοήγησης στη πλατφόρμα, δημιουργήθηκε το MenuBar component το οποίο αποτελείται από δύο τμήματα. Το ένα τμήμα, χρησιμοποιείται για την πλοήγηση σε σελίδες όπως η αρχική σελίδα, λειτουργίες που σχετίζονται με το προφίλ του χρήστη, δημιουργία αγγελίας και στοιχεία επικοινωνίας με την πλατφόρμα. Το δεύτερο κομμάτι, περιλαμβάνει τις κατηγορίες επίπλων, οι οποίες οδηγούν απευθείας στην dashboard με τις αγγελίες φιλτραρισμένες ανάλογα την κατηγορία που επέλεξε ο χρήστης.

Αρχικά δηλώνονται σε μια λίστα οι κατηγορίες (categories) επίπλων που υπάρχουν. Το UI του μενού, περιλαμβάνει σύνδεσμο αξιοποιώντας το Link από το react-router-dom, για την ‘Αρχική’ σελίδα (“/dashboard”), μια dropdown λίστα ‘Προφίλ’ με επιλογές ανακατεύθυνσης στις ‘Αγγελίες’ (“/my-ads”), οπού οδηγεί τον χρήστη στο προφίλ του που βρίσκονται οι αγγελίες και κάποια από τα στοιχεία του, ‘Επεξεργασία Προφίλ’ (“/profile”) όπου ο χρήστης οδηγείται στην σελίδα στην οποία μπορεί να επεξεργαστεί τα προσωπικά του στοιχεία, Παραγγελίες (“/myorders”) που ανακατευθύνει τον χρήστη στη σελίδα με τις παραγγελίες που έχει κάνει και τα στοιχεία τους, καθώς και το κουμπί αποσύνδεσης μέσω της LogOutButton. Περιέχει επίσης κουμπί ‘Νέα Αγγελία’ για την δημιουργία νέας αγγελίας που οδηγεί στην σελίδα “/ad” και κουμπί ‘Επικοινωνία’ το οποίο οδηγεί στην σελίδα “/contact” που περιέχει πληροφορίες επικοινωνίας με τους διαχειριστές της πλατφόρμας.

Στο δεύτερο τμήμα, δημιουργείται ένα φίλτρο με τις κατηγορίες μέσω της λίστας categories, οπού για κάθε κατηγορία, δημιουργείται ένας σύνδεσμος που οδηγεί στη dashboard η οποία διαχειρίζεται ποιες αγγελίες εμφανίζονται ανάλογα με αυτή που στέλνει ως παράμετρο. Για το styling του μενού, χρησιμοποιείται το CSS module dashboard.module.css.

AdCard

Ο adCard, χρησιμοποιείται για την απεικόνιση στοιχείων μιας αγγελίας σε μορφή κάρτας. Λαμβάνει ως δεδομένα { ad, favorites, handleFavorites, showFavorites = false, linkTo = null, clickable = false } , όπου ad είναι το αντικείμενο που περιέχει τα δεδομένα της αγγελίας, favorites περιέχει την λίστα με τα id των αγγελιών που ανήκουν στα αγαπημένα του χρήστη, την συνάρτηση handleFavorites που διαχειρίζεται την προσθήκη και αφαίρεση μιας αγγελίας από τα αγαπημένα, showFavorites με προεπιλογή false που καθορίζει αν θα εμφανίζεται το εικονίδιο καρδιάς για τη προσθήκη στα αγαπημένα, clickable με προεπιλογή false, το οποίο αν οριστεί σε true, ο χρήστης θα μπορεί να πατάει την κάρτα ενεργοποιώντας το onClick, και τέλος, onClick με τη συνάρτηση να καλείται όταν ο χρήστης κάνει κλικ στην κάρτα.

Αρχικά στην κάρτα ορίζεται αν είναι clickable, και ύστερα εμφανίζεται η εικόνα της αγγελίας, το όνομα, η περιγραφή, η κατηγορία και η τιμή μέσα στην κάρτα. Αν η επιλογή showFavorites είναι

ενεργοποιημένη, τότε εμφανίζεται ένα εικονίδιο καρδιάς στην κάρτα όπου αν το id της αγγελίας βρίσκεται στη λίστα favorites, η καρδιά εμφανίζεται γεμάτη , ενώ αν δεν βρίσκεται τότε εμφανίζεται κενή. Με το πάτημα των εικονιδίων, καλείται η handleFavorites για την ενημέρωση της κατάστασης των αγαπημένων. Για το styling, χρησιμοποιείται το CSS module dashboard.module.css.

AdInfo

Το AdInfo υλοποιεί τη φόρμα που χρησιμοποιείται είτε για την δημιουργία μιας νέας αγγελίας, είτε για την επεξεργασία υπάρχουσας, δηλαδή λαμβάνει και ενημερώνει τα δεδομένα.

Δέχεται props όπως { formData, handleChange, handleImageChange, handleSubmit, isEdit } όπου ‘formData’ περιέχει όλα τα στοιχεία της φόρμας, ‘handleChange’ είναι η συνάρτηση που ενημερώνει το formData κάθε φορά που αλλάζει κάποιο πεδίο input, η μέθοδος ‘handleImageChange’ που διαχειρίζεται την εισαγωγή ή τροποποίηση εικόνας μιας αγγελίας, την ‘handleSubmit’ που καλείται κατά την υποβολή της φόρμας, και την μεταβλητή isEdit, που καθορίζει αν η φόρμα θα χρησιμοποιείται για επεξεργασία υπάρχουσας αγγελίας ή για δημιουργία νέας αγγελίας. Η συνάρτηση δημιουργεί μια φόρμα <form onSubmit={handleSubmit}>, μέσα σε αυτή, αρχικά ελέγχεται αν είναι true η μεταβλητή isEdit ώστε η φόρμα να λάβει τον κατάλληλο τίτλο ("Επεξεργασία Αγγελίας" αν η τιμή είναι true και "Δημιουργία νέας αγγελίας" αν η τιμή είναι false). Η isEdit χρησιμοποιείται επίσης στο πεδίο της εικόνας, για να δηλώσει ότι σε περίπτωση δημιουργίας καινούριας αγγελίας, η χρήση νέας εικόνας είναι αναγκαία, ενώ σε περίπτωση επεξεργασίας, η υποβολή της είναι προαιρετική. Η φόρμα περιλαμβάνει και τα υπόλοιπα στοιχεία που χρειάζεται μια αγγελία, όπως τον τίτλο, την περιγραφή, την τιμή, την κατηγορία και την κατάσταση του επίπλου. Το κουμπί υποβολής προσαρμόζεται ανάλογα με το αν γίνεται δημιουργία ή επεξεργασία χρησιμοποιώντας το αντίστοιχο κείμενο.

4.3.2 Utilities

Στην React, τα utilities, αποτελούν βοηθητικές συναρτήσεις ή custom hooks, οι οποίες μπορούν να χρησιμοποιηθούν σε διαφορετικές σελίδες και αρχεία JSX. Δεν επιστρέφουν απευθείας το UI, αλλά παρέχουν τη λογική και τη λειτουργικότητα όπως την επικοινωνία με το backend. Τα utilities, μπορούν να λάβουν δεδομένα ως παραμέτρους και να προσαρμόζουν την υλοποίηση τους με βάση αυτά, επιστρέφοντας τα αποτελέσματα των λειτουργιών που υλοποιεί. Η χρήση τους, γίνεται μέσω των import, όπου δηλώνονται με τη μορφή «import { UtilityName } from "../functions/utilityName";».

useAdminData

Η useAdminData είναι ένα custom hook που χρησιμοποιείται για την ανάκτηση και διαχείριση δεδομένων από το backend, δίνοντας παράλληλα τη δυνατότητα για διαγραφή αντικειμένων και επεξεργασία κατάστασης λογαριασμού.

Δέχεται τα endpoint που χρησιμοποιείται για να λάβει τα δεδομένα που χρειάζεται κάθε φορά, το endpointDelete, που είναι το endpoint που χρησιμοποιείται για την διαγραφή δεδομένων, disableEndpoint που χρησιμοποιείται για την απενεργοποίηση χρήστη και enableEndpoint για την ενεργοποίηση του.

Περιλαμβάνει τα states,data που αποθηκεύονται τα φορτωμένα δεδομένα που λαμβάνονται από το API, loading και αρχικοποιημένη τιμή useState(true) που δείχνει εάν αυτά είναι σε διαδικασία φόρτωσης.

Χρησιμοποιείται η useEffect, που καλείται όταν φορτώνει για πρώτη φορά το hook και κάθε φορά που αλλάζει η τιμή του endpoint. Κάνει GET αίτημα στο API, και αν πετύχει, βάζει τα δεδομένα στο setData. Μέσω του setLoading(false) δηλώνει ανεξαρτήτως το αποτέλεσμα, ότι η φόρτωση τελείωσε.

Η συνάρτηση deleteItem, στέλνει στο endpoint αίτημα DELETE για αντικείμενο με το id που του δίνει. Αν η διαγραφή είναι επιτυχής, τότε το αφαιρεί από το state data για να ενημερωθεί αμέσως με μήνυμα επιτυχίας, ενώ αν αποτύχει εμφανίζει μήνυμα σφάλματος.

Η συνάρτηση disableItem, δέχεται id του χρήστη και κάνει PUT αίτημα στο disableEndpoint που έλαβε, από το οποίο παίρνει τα δεδομένα του χρήστη και αλλάζει το accountStatus σε "INACTIVE" στο data για την άμεση εμφάνιση των αλλαγών. Εμφανίζεται μήνυμα επιτυχίας μετά την ολοκλήρωση.

Αντίστοιχα, η enableItem, με τον ίδιο τρόπο, χρησιμοποιεί το id, και κάνει PUT αίτημα στο enableEndpoint οπού ξανά, αλλάζει το accountStatus σε "ACTIVE" για τον συγκεκριμένο χρήστη στο data για την άμεση εμφάνιση των αλλαγών εμφανίζοντας μήνυμα επιτυχίας.

Το hook τελικά, επιστρέφει τα δεδομένα data, το loading και τις συναρτήσεις για διαγραφή, ενεργοποίηση και απενεργοποίηση.

useFavoritesFetch

Η useFavoritesFetch αποτελεί μια συνάρτηση η οποία χρησιμοποιείται για τη προβολή και διαχείριση των αγαπημένων του χρήστη.

Δέχεται ως props, τα setFavorites, που είναι η συνάρτηση ενημέρωσης της λίστας με τα id των αγαπημένων και το id του χρήστη.

Περιέχει δύο async συναρτήσεις την fetchFavorites και την handleFavorites. Η fetchFavorites, χρησιμοποιεί την useCallback ώστε να μην ανανεώνεται κάθε φορά που φορτώνεται η σελίδα που την καλεί, λαμβάνει την λίστα με τις αγαπημένες αγγελίες του χρήστη κάνοντας αρχικά fetch στο (/user/users/${userId}/favorites?id=${userId}`) endpoint, δηλαδή στέλνει αίτημα στην συνάρτηση getFavorites της UserController όπου εάν το αίτημα πετύχει τότε παίρνει τα δεδομένα σε JSON. εξάγει τα id των αγγελιών. Ενημερώνει το state μέσω της setFavorites και επιστρέφει τα ids, ενώ σε περίπτωση αποτυχίας εμφανίζει μήνυμα σφάλματος.

Η handleFavorites, η οποία αποτελεί συνάρτηση για την διαχείριση των αγαπημένων του χρήστη, δέχεται το furnitureId και το event e από την ενέργεια του χρήστη. Κάνει fetch στο (/user/users/${userId}/favorites/${furnitureId}`) δηλαδή καλεί την συνάρτηση addFavorites της UserController με μέθοδο POST, καθώς γίνονται αλλαγές στα δεδομένα και καλεί την fetchFavorites() για να συγχρονιστεί η λίστα αγαπημένων. Σε περίπτωση αποτυχίας του αιτήματος εμφανίζει μήνυμα αποτυχίας.

Τελικά, το useFavoritesFetch, επιστρέφει τις δύο συναρτήσεις.

4.3.3 Jsx

Οι σελίδες, αποτελούν τα components όπου περιέχουν την γενική δομή συνδυάζοντας components και utilities για την υλοποίηση των λειτουργιών. Αυτά, συνδέονται μέσω routes που γίνονται στο κομμάτι των import. Το styling των σελιδών, έγινε μέσω css.

Register

Η συνάρτηση Register() υλοποιεί την σελίδα εγγραφής και δημιουργεί το Register component. Αρχικά ορίζεται η useNavigate() και η formData με στοιχεία όπως το fullName, username, email, password, confirmPassword το οποίο αποθηκεύει τα δεδομένα της φόρμας. Δηλώνονται States όπως showPassword για έλεγχο εμφάνισης ή απόκρυψης του κωδικού πρόσβασης και τα error και success για την εμφάνιση μηνυμάτων κατάστασης εγγραφής.

Μετά τις δηλώσεις, ακολουθούν οι συναρτήσεις όπως η togglePassword η οποία χρησιμοποιείται για την εμφάνιση του κωδικού πρόσβασης, δηλαδή το showPassword αλλάζει τιμή όταν πατηθεί το εικονίδιο ματιού, μέσω του setShowPassword , και εναλλάσσει την εμφάνιση/απόκρυψη.

Η handleChange, χρησιμοποιείται για την ενημέρωση του formData, όποτε αλλάζει κάποιο input δέχεται ως όρισμα το προηγούμενο (prev) formData και παίρνει το input που προκάλεσε την αλλαγή [e.target.name] και την τιμή που πρόσθεσε ο χρήστης [e.target.value].

Η υποβολή της φόρμας γίνεται μέσω της handleSubmit, η οποία είναι async συνάρτηση και αφού ελέγξει αν το username που συμπλήρωσε ο χρήστης πληροί τις προϋποθέσεις , κάνει αίτημα με

μέθοδο POST προς το endpoint (/auth/register), δηλαδή την μέθοδο registerUser του AuthController με τα δεδομένα του χρήστη να στέλνονται σε μορφή JSON. Σε περίπτωση σφάλματος εμφανίζεται το κατάλληλο μήνυμα ενώ αν επιτύχει τότε λαμβάνει τα δεδομένα του χρήστη και τα αποθηκεύει στο localStorage εμφανίζοντας το κατάλληλο μήνυμα και στη συνέχεια ανακατευθύνει τον χρήστη στην σελίδα σύνδεσης “/login”.

Στην απεικόνιση της σελίδας, περιλαμβάνεται μια φόρμα onSubmit={handleSubmit} με input το username, fullName, email, password, confirm password. Τα πεδία ενημερώνονται μέσω της handleChange μόλις ο χρήστης πληκτρολογήσει σε κάποιο input. Για τα πεδία κωδικού και επιβεβαίωσης κωδικού, υπάρχει η δυνατότητας εμφάνισης και απόκρυψης κειμένου μέσω της συνάρτησης togglePassword. Τέλος χρησιμοποιεί τα error και success για την εμφάνιση μηνυμάτων, και δίνει στον χρήστη την επιλογή ανακατεύθυνσης προς την σελίδα σύνδεσης σε περίπτωση που έχει ήδη λογαριασμό.

Login

Η Login() υλοποιεί την σελίδα σύνδεσης και δημιουργεί το Login component.

Αρχικά ορίζεται η useNavigate() και η formData με username και password όπου αποθηκεύει τα δεδομένα της φόρμας. Δηλώνονται States όπως showPassword για έλεγχο εμφάνισης ή απόκρυψης του κωδικού πρόσβασης και τα error και success για την εμφάνιση μηνυμάτων κατάστασης εγγραφής.

Όπως και το register, υπάρχουν οι συναρτήσεις, όπως η togglePassword η οποία χρησιμοποιείται για την εμφάνιση του κωδικού πρόσβασης και η handleChange, χρησιμοποιείται για την ενημέρωση του formData η οποία λειτουργεί με τον ίδιο τρόπο.

Η υποβολή της φόρμας γίνεται μέσω της handleSubmit, η οποία είναι async συνάρτηση κάνει αίτημα fetch με μέθοδο POST προς το endpoint (/auth/login) του AuthController στην μέθοδο loginUser δηλαδή, με τα δεδομένα του χρήστη να στέλνονται σε μορφή JSON. Σε περίπτωση σφάλματος εμφανίζεται το κατάλληλο μήνυμα ενώ αν επιτύχει τότε λαμβάνει τα δεδομένα του χρήστη και τα αποθηκεύει στο localStorage εμφανίζοντας το κατάλληλο μήνυμα. Στην συνέχεια, ελέγχει αν ο ρόλος του χρήστη είναι ADMIN ώστε να τον ανακατευθύνει στην /adminHome, ενώ αν είναι απλός χρήστης τότε πηγαίνει στο /dashboard. Αν το αίτημα αποτύχει, εμφανίζεται το κατάλληλο μήνυμα στον χρήστη.

Στην απεικόνιση της σελίδας, περιλαμβάνεται μια φόρμα onSubmit={handleSubmit} με input το username, με χρήση της handleChange μόλις ο χρήστης πληκτρολογήσει στο πεδίο. Αντίστοιχα λειτουργεί και το πεδίο για το password με την προσθήκη του εικονιδίου, το οποίο μόλις πατηθεί χρησιμοποιεί την συνάρτηση togglePassword. Τέλος χρησιμοποιεί τα error και success για την εμφάνιση μηνυμάτων, και δίνει στον χρήστη την επιλογή ανακατεύθυνσης προς την σελίδα εγγραφής σε περίπτωση που δεν έχει λογαριασμό.

AdminHomePage

Η σελίδα AdminHomePage αποτελεί την αρχική σελίδα που εμφανίζεται στον Admin μετά την σύνδεση του στην πλατφόρμα.

Γίνεται χρήση του useNavigate() για την ανακατεύθυνση στις σελίδες διαχείρισης των αντικειμένων. Χρησιμοποιείται το component LogOutButton, το οποίο παρέχει τη λειτουργία αποσύνδεσης του διαχειριστή. Το περιεχόμενο της σελίδας περιλαμβάνει είναι δομημένο σε τρείς ενότητες, οι οποίες περιλαμβάνουν κουμπί με τον αντίστοιχο τίτλο.

Συγκεκριμένα, εάν ο Admin επιθυμεί την διαχείριση των χρηστών της εφαρμογής, θα επιλέξει το αντίστοιχο κουμπί που με το πάτημα του κουμπιού (onClick) ‘Διαχείριση Αγγελιών’, μέσω του navigate θα τον οδηγήσει στην σελίδα /adminUsers. Για την διαχείριση των αγγελιών, με τον ίδιο τρόπο θα ανακατευθυνθεί στην σελίδα /adminFurniture ενώ για τη διαχείριση των παραγγελιών θα ανακατευθυνθεί στην σελίδα /adminOrders.

AdminUserList

Η AdminUserList σελίδα, αποτελεί τη σελίδα όπου ο Admin μπορεί να υλοποιήσει την διαχείριση των χρηστών, δηλαδή τη σελίδα όπου εμφανίζονται όλοι οι χρήστες της πλατφόρμας και οι επιλογή για προβολή αγγελιών και παραγγελιών συγκεκριμένου χρήστη.

Λαμβάνει όλους τους χρήστες αφού εκτελέσει ένα αίτημα μέσω της συνάρτησης useAdminData στο endpoint(/admin/users) και παρέχει τη συνάρτηση disableUser για απενεργοποίηση λογαριασμού και enableUser για ενεργοποίηση.

Η συνάρτηση disableUserEndpoint (id) δημιουργεί το κατάλληλο URL για την απενεργοποίηση του χρήστη με βάση το id και η enableUserEndpoint(id) για την ενεργοποίηση του.

Για την εμφάνιση των δεδομένων, χρησιμοποιεί το AdminTable, το οποίο παρουσιάζει τους χρήστες σε μορφή πίνακα. Οι στήλες του πίνακα περιλαμβάνουν το fullName, το email, το username καθώς και μια στήλη ‘Ενέργειες’, για τον κάθε χρήστη. Σε αυτή τη στήλη, για κάθε χρήστη που δεν έχει ρόλο ADMIN, του δίνει την δυνατότητα προβολής όλων των αγγελιών ενός συγκεκριμένου χρήστη και όλων των παραγγελιών, ενώ αντίστοιχα για τον χρήστη την επιλογή απενεργοποίησης ή ενεργοποίησης λογαριασμού.

Συγκεκριμένα, για την διαχείριση αγγελιών ενός συγκεκριμένου χρήστη, ανακατευθύνει μέσω του navigate στην σελίδα /adminUserFurniture/${user.id} και για την διαχείριση των παραγγελιών στην σελίδα /adminUserOrders/{user.id}. Για την απενεργοποίηση ή ενεργοποίηση, ελέγχει αν το accountStatus του user είναι "ACTIVE" ώστε να καλέσει το disableUser δίνοντας το id του user με την επιλογή «Απενεργοποίηση», ενώ σε αντίθετη περίπτωση, να καλέσει την enableUser ξανά με το id του user, με «Ενεργοποίηση» ως επιλογή.

Η σελίδα εμφανίζει ένα κουμπί αποσύνδεσης ‘LogOutButton’, τον τίτλο ‘Χρήστες’, και μήνυμα σε περίπτωση που δεν υπάρχουν διαθέσιμα δεδομένα.

AdminUserFurnitureList

Η σελίδα αυτή, υλοποιεί τη διαχείριση των αγγελιών επίπλων που ανήκουν σε έναν συγκεκριμένο χρήστη.

Αρχικά, γίνεται χρήση του useParams από το react-router-dom ώστε να ανακτηθεί το userId που μεταφέρεται μέσω του URL για την δημιουργία του κατάλληλου endpoint. Ορίζεται το (/admin/users/${userId}/furniture) endpoint, για να επικοινωνήσει με τη μέθοδο getUserAds του AdminController για να λάβει όλες τις αγγελίες του συγκεκριμένου χρήστη. Χρησιμοποιεί το custom hook useAdminData, δίνοντας το endpoint για την ανάκτηση των δεδομένων και το endpoint (/admin/furniture/${id}) για την διαγραφή της αγγελίας μέσω της deleteFurniture του controller.

Τα δεδομένα εμφανίζονται μέσω του AdminTable component, όπου ορίζονται οι στήλες του πίνακα, τίτλος name, περιγραφή ‘description’, κατηγορία ‘furnitureCategory’, κατάσταση ‘furnitureCondition’, διαθεσιμότητα ‘state’, τιμή ‘price’ και μια στήλη με ενέργειες για κάθε αγγελία. Στη στήλη ενεργειών, εμφανίζεται το κουμπί ‘Διαγραφής’ που καλεί το ‘deleteFurniture (furniture.id)’ για την διαγραφή της αντίστοιχης αγγελίας. Η σελίδα περιλαμβάνει επιπλέον το component LogOutButton για την αποσύνδεση του διαχειριστή, και έναν τίτλο ‘Αγγελίες Χρήστη’. Σε περίπτωση που δεν υπάρχουν δεδομένα, εμφανίζει το κατάλληλο μήνυμα.

AdminUserOrdersList

Η σελίδα χρησιμοποιείται για την υλοποίηση της διαχείριση των παραγγελιών, που ανήκουν σε έναν συγκεκριμένο χρήστη.

Αρχικά, γίνεται χρήση του useParams ώστε να ανακτηθεί το userId που μεταφέρεται μέσω του URL για την δημιουργία του κατάλληλου endpoint. Ορίζεται το (/admin/users/${userId}/orders) endpoint, ώστε να επικοινωνήσει με τη μέθοδο getUserOrders του AdminController για να λάβει όλες τις παραγγελίες του συγκεκριμένου χρήστη. Χρησιμοποιεί το custom hook useAdminData, δίνοντας το endpoint για την ανάκτηση των δεδομένων και το endpoint (/admin/orders/${id}) για την διαγραφή της παραγγελίας μέσω της deleteOrders του controller.

Τα δεδομένα εμφανίζονται μέσω του AdminTable component, όπου ορίζονται οι στήλες του πίνακα, ονοματεπώνυμο fullName, fullName, διεύθυνση 'address, αριθμός 'number, πόλη 'city, ταχυδρομικός κώδικας 'postal', ημερομηνία 'date' και μια στήλη με ενέργειες για κάθε παραγγελία. Στη στήλη ενεργειών, εμφανίζεται το κουμπί 'Διαγραφής' που καλεί το 'deleteOrders(orders.id)' για την αφαίρεση της αντίστοιχης παραγγελίας. Τέλος, χρησιμοποιεί το component LogOutButton για την αποσύνδεση του διαχειριστή, και έναν τίτλο 'Παραγγελίες Χρήστη'. Σε περίπτωση που δεν υπάρχουν δεδομένα, εμφανίζει το κατάλληλο μήνυμα.

AdminFurnitureList

Η AdminFurnitureList σελίδα, αποτελεί τη σελίδα του Admin, που μπορεί να υλοποιήσει την διαχείριση των αγγελιών, δηλαδή τη σελίδα όπου εμφανίζονται όλες οι αγγελίες της πλατφόρμας και η επιλογή διαγραφής τους.

Χρησιμοποιεί το custom hook useAdminData, δίνοντας ένα endpoint (/admin/furniture) για την ανάκτηση των δεδομένων μέσω της getAllFurniture του AdminController και το deleteFurnitureEndpoint (/admin/furniture/${id}) για την διαγραφή της αγγελίας μέσω της deleteFurniture του controller.

Για την εμφάνιση των δεδομένων, χρησιμοποιεί το AdminTable, το οποίο παρουσιάζει τους χρήστες σε μορφή πίνακα με στήλες, τον τίτλο name, την περιγραφή 'description', την κατηγορία 'furnitureCategory', την κατάσταση 'furnitureCondition', τη διαθεσιμότητα 'state', τη τιμή 'price' και μια στήλη με ενέργειες για κάθε αγγελία. Στη στήλη ενεργειών, εμφανίζεται το κουμπί 'Διαγραφής' που καλεί το 'deleteFurniture (furniture.id)' για την αφαίρεση της αντίστοιχης αγγελίας. Η σελίδα περιλαμβάνει επιπλέον το component LogOutButton για την αποσύνδεση του διαχειριστή, και έναν τίτλο 'Αγγελίες. Σε περίπτωση που δεν υπάρχουν δεδομένα, εμφανίζει το κατάλληλο μήνυμα.

AdminOrdersList

Στην AdminOrdersList σελίδα, βρίσκεται η σελίδα του Admin, που μπορεί να υλοποιήσει την διαχείριση των παραγγελιών, δηλαδή τη σελίδα όπου εμφανίζονται όλες οι παραγγελίες της πλατφόρμας και παρέχεται η δυνατότητα διαγραφής τους.

Χρησιμοποιεί το custom hook useAdminData, δίνοντας ένα endpoint (/admin/orders) για την ανάκτηση των δεδομένων μέσω της getAllOrders του AdminController και το deleteOrdersEndpoint (/admin/orders/${id}) για την διαγραφή της παραγγελίας μέσω της deleteOrders του controller.

Για την εμφάνιση των δεδομένων, χρησιμοποιεί το AdminTable, το οποίο παρουσιάζει τις παραγγελίες σε μορφή πίνακα με στήλες, ονοματεπώνυμο fullName, διεύθυνση 'address, αριθμός 'number, πόλη 'city, ταχυδρομικό κώδικα 'postal', ημερομηνία 'date' και μια στήλη με ενέργειες για κάθε παραγγελία. Στη στήλη ενεργειών, εμφανίζεται το κουμπί 'Διαγραφής' που καλεί το 'deleteOrders(orders.id)' για την αφαίρεση της αντίστοιχης παραγγελίας. Τέλος, χρησιμοποιεί το component LogOutButton για την αποσύνδεση του διαχειριστή, και έναν τίτλο 'Παραγγελίες'. Σε περίπτωση που δεν υπάρχουν δεδομένα, εμφανίζει το κατάλληλο μήνυμα.

Dashboard

Το Dashboard αντιπροσωπεύει την αρχική σελίδα που συναντάει ένας χρήστης όταν εισέρχεται στην πλατφόρμα και μπορεί να δει τις διαθέσιμες αγγελίες, να χρησιμοποιήσει φίλτρα, και να διαχειριστεί τα αγαπημένα του.

Ως states, έχει την ads για αποθήκευση αγγελιών που έρχονται από το backend, selectedCategory για κατηγορία που έχει επιλέξει ο χρήστης, sortOption για ταξινόμηση, sortCondition για το φίλτρο κατάστασης και favorites για να κρατάει τα αγαπημένα του χρήστη. Η userId έχει το id του χρήστη που έλαβε από το localStorage , και οι fetchFavorites, handleFavorites είναι οι συναρτήσεις που παίρνει από το useFavoritesFetch custom hook που διαχειρίζεται τις αγαπημένες αγγελίες. Χρησιμοποιεί τις useNavigate() και useLocation(). Η queryParams χρησιμοποιείται για να ψάξει URL με βάση το categoryFromUrl που του δίνει τη κατηγορία. Για το φιλτράρισμα των αγγελιών υπάρχει η availableAdsExcludingUser που παίρνει μόνο τις αγγελίες όπου ο πωλητής δεν είναι ο συνδεδεμένος χρήστης.

Αρχικά, ελέγχεται αν έχει επιλεχθεί προεπιλεγμένη κατηγορία κατά την μεταφορά του χρήστη στην σελίδα και την αποθηκεύει στο selectedCategory.

Η useEffect ελέγχει αν υπάρχει το id του χρήστη και αν δεν υπάρχει, οδηγεί τον χρήστη στην σελίδα της σύνδεσης “/login”. Περιέχει ασύγχρονη συνάρτηση fetchAds και στέλνει στο backend την επιλεγμένη κατηγορία, κατάσταση και ταξινόμηση κάνοντας αίτημα στο endpoint (/furniture/filteredAds?${params.toString()}) στην μέθοδο filteredAds του FurnitureController. Λαμβάνει τα δεδομένα και τα αποθηκεύει στο data. Καλεί την fetchAds() για να φέρει τις αγγελίες και την fetchFavorites() για να υπάρχουν τα αγαπημένα του χρήστη.

Η δεύτερη useEffect, ενεργοποιείται μόνο όταν ο χρήστης αλλάξει κατηγορία, και δίνει στην selectedCategory την επιλεγμένη κατηγορία του χρήστη.

Η συναρτήση handleSortChange, δίνει στην sortOption την επιλογή ταξινόμησης του χρήστη, και η handleConditionChange δίνει στην sortCondition την επιλογή κατάστασης. Και οι δύο χρησιμοποιούν το e.target.value η οποία παίρνει την τιμή από τις dropdown λίστες που υπάρχουν.

Η σελίδα εμφανίζει το μενού όπως είναι στο MenuBar και παράλληλα έχει την επιλογή φίλτρων, δηλαδή μια dropdown list για την ταξινόμηση (προτεινομένη, αύξουσα, φθίνουσα) που χρησιμοποιεί την onChange={handleSortChange} για την διαχείριση της, και φίλτρο για την κατάσταση onChange={handleConditionChange} με επιλογές (Όλες οι καταστάσεις, Καινούριο, Καλή κατάσταση, Μεταχειρισμένο). Έτσι αν το μέγεθος λίστας αγγελιών που είναι διαθέσιμες, είναι μηδέν, εμφανίζει μήνυμα ότι δεν υπάρχουν αγγελίες, ενώ αν υπάρχουν χρησιμοποιεί την AdCard για να τις εμφανίσει δίνοντας τα στοιχεία και με παραμέτρους clickable με τιμή true ώστε ο χρήστης να μπορεί να πατάει πάνω και με την (`/adDetails/${ad.id}`) να οδηγείται στην σελίδα με τις λεπτομέρειες αγγελίας.

UserFavorites

Η σελίδα χρησιμοποιείται για να εμφανίσει όλες τις αγγελίες που βρίσκονται στα αγαπημένα του χρήστη αφού εκείνος πατήσει το κουμπί που οδηγεί σε αυτή από τη σελίδα MyAds.

Περιέχει τα state, favorites που είναι λίστα με τα id των αγαπημένων αγγελιών και favoriteAds που είναι η λίστα με τα αντικείμενα των αγγελιών. Χρησιμοποιεί την useNavigate() και αποθηκεύει το userId στο localStorage. Περιλαμβάνει, επίσης, τη συνάρτηση που δημιουργήθηκε για την λήψη των αγαπημένων αγγελιών, την useFavoritesFetch fetchFavorites, handleFavorites , δηλαδή τις συναρτήσεις για την λήψη αγαπημένων και την διαχείριση τους.

Στο useEffect, γίνεται αρχικά έλεγχος αν βρέθηκε το userId και αν δεν βρεθεί εμφανίζει μήνυμα στον χρήστη, και τον ανακατευθύνει στην σελίδα σύνδεσης. Η ασύγχρονη συνάρτηση fetchFavoriteAds φέρνει ids των αγαπημένων αγγελιών, κάνει ένα Promise.all, δηλαδή στέλνει πολλαπλά αιτήματα με όλα τα ids των αγγελιών για να ληφθούν όλα τα αντικείμενα αγγελιών. Το αίτημα που πραγματοποιεί, είναι GET αίτημα στην getFurnitureDetails μέθοδο της

FurnitureController κλάσης. Αποθηκεύει την λίστα με τα αντικείμενα αγγελιών καλεί την fetchFavoriteAds.

Στην σελίδα, υπάρχει το MenuBar, και ελέγχει αν το πλήθος αγγελιών είναι μηδενικό ώστε να εμφανίσει αντίστοιχο μήνυμα, ενώ αν υπάρχουν αγγελίες τότε για κάθε αγγελία καλείται το AdCard με τα κατάλληλα στοιχεία για την εμφάνιση τους.

AdDetails

Αυτό το component εμφανίζει λεπτομέρειες μιας αγγελίας, καθώς και στοιχεία του πωλητή στο dashboard. Πατώντας πάνω σε μία αγγελία που βρίσκεται στο Dashboard, ο χρήστης πηγαίνει στην σελίδα αυτή.

Περιέχει το useParams { id }, και το ad state, στο οποίο αποθηκεύει όλα τα στοιχεία της αγγελίας.

Έχει την ασύγχρονη συνάρτηση fetchAdDetails η οποία κάνει GET αίτημα στο endpoint (/furniture/details/${id}), δηλαδή στη μέθοδο getFurnitureDetails του controller, για να λάβει την αγγελία, και την αποθηκεύει στο ad. Σε περίπτωση σφάλματος, εμφανίζει μήνυμα στον χρήστη.

Στην σελίδα εμφανίζεται το MenuBar και δημιουργεί το πλαίσιο εμφάνισης των πληροφοριών. Ελέγχει αν υπάρχει η αγγελία και στη συνέχεια προβάλει την εικόνα της αγγελίας, μαζί με τα στοιχεία και το κουμπί αγοράς που οδηγεί στην σελίδα {`/purchase/${id}`}. Παρακάτω, υπάρχει το πλαίσιο πληροφοριών του πωλητή που ξανά ελέγχει αν υπάρχει και εμφανίζει την εικόνα προφίλ του, μαζί με στοιχεία όπως το fullName, username, email. Αν ο πωλητής δεν βρεθεί εμφανίζει μήνυμα.

SellerProfile

Η SellerProfile χρησιμοποιείται για την προβολή προφίλ του πωλητή μιας αγγελίας, μαζί με τις αγγελίες του. Η σελίδα μπορεί να προβληθεί πατώντας πάνω στο πλαίσιο με τις πληροφορίες του πωλητή που βρίσκονται στην σελίδα AdDetails.

Συγκεκριμένα, χρησιμοποιεί το useNavigate(), έχει useParams() το id του πωλητή και useState το seller για τα στοιχεία του που θα εμφανίζονται στο προφίλ. Περιέχει επίσης τα useState furnitureList για την προβολή των αγγελιών του και favorites για τα αγαπημένα του συνδεδεμένου χρήστη. Ο χρήστης, βρίσκεται μέσω του localStorage και αποθηκεύεται στο userId. Τέλος, τα availableAds, που φιλτράρουν από τις αγγελίες του ποιες έχουν κατάσταση state που είναι 'available'.

Στην useEffect() πραγματοποιεί έλεγχο εύρεσης χρήστη, ώστε αν δεν βρεθεί να προβάλει μήνυμα. Περιέχει την ασύγχρονη συνάρτηση fetchProfileData που κάνει αίτημα GET στη μέθοδο showProfile του UserController, παίρνοντας τα στοιχεία σε JSON και αποθηκεύοντας τα στο sellerData.

Στο δεύτερο useEffect(), που χρησιμοποιείται για την ανάκτηση των αγγελιών, υπάρχει fetchMyAds που είναι ασύγχρονη συνάρτηση. Πραγματοποιεί GET αίτημα στην getMyAds του FurnitureController και αν βρεθούν οι αγγελίες τότε τις αποθηκεύει σε JSON. Τέλος, καλεί την fetchMyAds() και την fetchFavorites() για την προβολή αγγελιών του χρήστη που έχει επισκεφτεί και των αγαπημένων αγγελιών που μπορεί να υπάρχουν από αυτές τις αγγελίες.

Στην σελίδα εμφανίζεται το MenuBar και παρακάτω περιέχει την εικόνα προφίλ του χρήστη μαζί με το fullName, username. Εμφανίζονται όλες οι διαθέσιμες αγγελίες, μέσω του availableAds, δηλαδή για κάθε μια από τις αγγελίες που βρέθηκαν ότι είναι διαθέσιμες, εμφανίζει ένα AdCard, το οποίο μπορεί να πατήσει και να οδηγηθεί στην (`/adDetails /${ad.id}`)}.

MyAds

Η MyAds χρησιμοποιείται για την προβολή προφίλ του χρήστη μαζί με τις αγγελίες του.

Συγκεκριμένα περιέχει useState για το userId, furnitureList, user που περιέχει τα στοιχεία του χρήστη που εμφανίζονται στο προφίλ, χρησιμοποιεί την useNavigate() για ανακατευθύνσεις και τα availableAds, unavailableAds που φιλτράρουν από τις αγγελίες του χρήστη ποιες έχουν κατάσταση state που είναι 'available' και ποιες που είναι 'unavailable'.

Στην useEffect() αρχικά λαμβάνει το userId μέσω του localStorage και πραγματοποιεί έλεγχο ώστε αν δεν βρεθεί χρήστη, να τον ανακατευθύνει στη “/login” σελίδα. Περιέχει την ασύγχρονη συνάρτηση fetchProfileData που κάνει αίτημα GET στη μέθοδο showProfile του UserController παίρνοντας τα στοιχεία σε JSON και αποθηκεύοντας τα στο formData.

Στο δεύτερο useEffect() που χρησιμοποιείται για την ανάκτηση των αγγελιών αφού έχουν ληφθεί οι πληροφορίες του χρήστη, υπάρχει η fetchMyAds που είναι ασύγχρονη συνάρτηση. Πραγματοποιεί GET αίτημα στην getMyAds του FurnitureController και αν βρεθούν οι αγγελίες τότε τις αποθηκεύει σε JSON. Αν το userId υπάρχει, καλεί την fetchMyAds().

Στην σελίδα εμφανίζεται το MenuBar και παρακάτω περιέχει την εικόνα προφίλ του χρήστη μαζί με το fullName, username, το πλήθος των διαθέσιμων αγγελιών και το πλήθος των πουλημένων, και ένα κουμπί που οδηγεί τον χρήστη στη σελίδα με τις αγαπημένες αγγελίες του /userFavorites. Η σελίδα χωρίζεται σε δύο μέρη όπου αρχικά στο πρώτο εμφανίζονται όλες οι διαθέσιμες αγγελίες του μέσω του availableAds, εμφανίζοντας τις μέσω του AdCard και navigate(`/furnitureDetails/${ad.id}`)} καθώς ο χρήστης μπορεί να δει, να επεξεργαστεί ή να διαγράψει την αγγελία. Παρακάτω, έχει πλαίσιο με όλες τις αγγελίες που έχουν πουληθεί που με αντίστοιχο τρόπο μέσω του unavailableAds χρησιμοποιεί την AdCard για την εμφάνιση τους και οδηγεί στην (`/buyerInfo/${ad.id}`) σελίδα από την οποία μπορεί να δει τις πληροφορίες του αγοραστή.

BuyerInfo

Η σελίδα, χρησιμοποιείται για να δει ο χρήστης ποιος αγόρασε το έπιπλο του. Παρέχει όλες τις πληροφορίες της παραγγελίας και τα στοιχεία του χρήστη που την αγόρασε. Ο χρήστης οδηγείται στην σελίδα, πατώντας πάνω σε ένα έπιπλο στο προφίλ του που βρίσκεται στην κατηγορία “Πουλήθηκαν”.

Χρησιμοποιεί το navigate και το order state όπου θα αποθηκεύονται οι παραγγελίες και παίρνει τον χρήστη μέσω του localStorage. Δέχεται επίσης την παράμετρο id από το useParams που αντιπροσωπεύει την αγγελία.

Η συνάρτηση useEffect,αρχικά ελέγχει αν ο χρήστης υπάρχει, ώστε αν δεν υπάρχει να τον ανακατευθύνει στην σελίδα σύνδεσης. Η μέθοδος fetchOrder, πραγματοποιεί ένα αίτημα GET προς την μέθοδο getBuyerInfo του OrdersController δίνοντας το furnitureId ως παράμετρο. Αποθηκεύει τα δεδομένα στο order και εμφανίζει μήνυμα στον χρήστη σε περίπτωση αποτυχίας.

Στη σελίδα, αν δεν έχουν φορτωθεί ακόμα οι παραγγελίες εμφανίζει μήνυμα φόρτωσης. Περιλαμβάνει το MenuBar, και εμφανίζει σε ένα πλαίσιο τον τίτλο του προϊόντος, την ημερομηνία της παραγγελίας, το email το όνομα και την διεύθυνση του αγοραστή και την τιμή του προϊόντος.

Profile

Το component αυτό αποτελεί την σελίδα επεξεργασίας προφίλ του χρήστη.

Χρησιμοποιεί την useNavigate() για ανακατεύθυνση που θα χρειαστεί στην σελίδα και έχει useState τα formData που αποθηκεύονται τα στοιχεία του χρήστη, selectedImage για την αλλαγή της εικόνας, και showPassword για αλλαγή εικονιδίου εμφάνισης κωδικού.

Στην useEffect, αρχικά λαμβάνει το userId μέσω του localStorage και πραγματοποιεί έλεγχο ώστε αν δεν βρεθεί χρήστης, να ανακατευθύνεται στη /login. Κάνει αίτημα GET, στη μέθοδο showProfile του UserController όπου παίρνει JSON αποτέλεσμα τα στοιχεία του χρήστη και τα αποθηκεύει στο formData.

Η handleChange, ενημερώνει το formData δηλαδή όταν αλλάζει κάποιο input, δέχεται ως όρισμα το προηγούμενο (prev) formData και παίρνει το input που προκάλεσε την αλλαγή [e.target.name] και την τιμή που πρόσθεσε ο χρήστης e.target.value.

Τέλος η ασύγχρονη handleSubmit συνάρτηση δημιουργεί νέο formData στο οποίο δίνει τα στοιχεία fullName username, email, password και το selectedImage, εφόσον έχει δώσει εικόνα ο χρήστης. Πραγματοποιεί αίτημα POST προς την μέθοδο updateUser του UserController και αν είναι επιτυχής εμφανίζει στον χρήστη μήνυμα επιτυχίας και τον ανακατευθύνει στην /my-ads. Αν ο χρήστης είχε επιλέξει να αλλάξει εικόνα προφίλ, τότε δημιουργεί νέο URL προσωρινό. Σε περίπτωση σφάλματος του εμφανίζει μήνυμα.

Ο χρήστης έχει επίσης την επιλογή ακύρωσης αλλαγών, το οποίο διαχειρίζεται από το handleCancel, με την οποία πατώντας το κουμπί, οδηγείται πίσω στην σελίδα “/my-ads”.

Στην σελίδα εμφανίζεται το MenuBar, και δημιουργείται η φόρμα επεξεργασίας προφίλ όπου δείχνει αρχικά την εικόνα προφίλ του χρήστη όπου ανάλογα την εικόνα που έχει ή θέλει να βάλει ο χρήστης, αυτό αλλάζει μέσω του formData.imageUrl. Έχει ως input τα fullName, username, email, password που χρησιμοποιούν την handleChange με κουμπί αποθήκευσης, και ένα κουμπί ακύρωσης που χρησιμοποιεί το handleCancel.

CreateAd

Η σελίδα αυτή υλοποιεί τη σελίδα δημιουργίας νέας αγγελίας επίπλου. Ο χρήστης μπορεί να δώσει τίτλο, περιγραφή, τιμή, κατηγορία, κατάσταση και εικόνα.

Αρχικά χρησιμοποιούνται state για όλα τα στοιχεία της φόρμας (name, description, price, furnitureCategory, furnitureCondition) στο formData. Χρησιμοποιεί το navigate για πλοήγηση και το image state όπου θα αποθηκεύεται η εικόνα που θα ανεβάζει ο χρήστης, καθώς και state για το userId.

Όσον αφορά τις συναρτήσεις, υπάρχει η handleChange, η οποία ενημερώνει το formData με βάση τα input που δέχεται ως όρισμα κάθε φορά που συμπληρώνει κάποιο στοιχείο ο χρήστης, και αντίστοιχα η handleImageChange παίρνει το αρχείο που ανέβασε ο χρήστης.

Η useEffect, πραγματοποιεί έλεγχο, ώστε αν ο χρήστης δεν είναι συνδεδεμένος να κάνει ανακατεύθυνση στην σελίδα σύνδεσης.

Η συνάρτηση handleSubmit η οποία είναι async, υλοποιείται όταν ο χρήστης πατήσει αποθήκευση της αγγελίας. Δημιουργεί το αντικείμενο της αγγελίας και προσθέτει τον πωλητή seller στο id και ύστερα φτιάχνει νέο FormData ώστε να στείλει JSON μαζί με το αρχείο εικόνας. Στέλνει το POST αίτημα στον server με μέθοδο POST όπου χρησιμοποιεί την μέθοδο createFurniture του FurnitureController για να δημιουργηθεί η αγγελία. Αν το αίτημα ήταν επιτυχές, τότε εμφανίζει μήνυμα στον χρήστη και ανακατευθύνεται στην /my-ads, και σε περίπτωση σφάλματος εμφανίζει μήνυμα αποτυχίας στον χρήστη.

Στη σελίδα, χρησιμοποιείται το MenuBar και AdInfo component στο οποίο περνάει όλα τα props που χρειάζεται το formData, τις παραπάνω συναρτήσεις και το isEdit με τιμή false για να χρησιμοποιηθεί το component στην σωστή φόρμα, δηλαδή αυτή της δημιουργίας αγγελίας.

FurnitureDetails

Το FurnitureDetails component, αποτελεί το component που δείχνει τις λεπτομέρειες μιας αγγελίας του χρήστη αφού εκείνος πατήσει πάνω της.

Παίρνει το id της αγγελίας μέσω του useParams , χρησιμοποιεί την navigate,το furniture state όπου θα αποθηκεύεται η αγγελία και το loading state, δείχνει αν φορτώνουν τα δεδομένα.

Η useEffect, περιέχει την ασύγχρονη συνάρτηση fetchFurniture η οποία κάνει ένα αίτημα για να λάβει τα στοιχεία της αγγελίας, δημιουργεί ένα αντικείμενο adForCard με αυτά, και τα βάζει στο furniture. Σε περίπτωση σφάλματος βγάζει το αντίστοιχο μήνυμα και ανακατευθύνει στη σελίδα /my-ads ενώ αν είναι επιτυχές ορίζει τη φόρτωση σε loading σε false.

Η συνάρτηση handleDelete, η οποία είναι επίσης ασύγχρονη, προβάλει αρχικά στον χρήστη μήνυμα επιβεβαίωσης ώστε αν θέλει να διαγράψει την αγγελία να προχωρήσει στην διαγραφή. Αν το confirmDelete είναι αληθές στέλνει αίτημα στη μέθοδο deleteFurniture του FurnitureController με μέθοδο 'DELETE', και αν το αίτημα είναι επιτυχές εμφανίζει μήνυμα στον χρήστη και τον ανακατευθύνει στην /my-ads σελίδα. Στην αντίθετη περίπτωση βγάζει μήνυμα αποτυχίας. Στην σελίδα, περιλαμβάνεται και ο έλεγχος φόρτωσης αγγελιών ώστε να εμφανίζει μήνυμα φόρτωσης, όπως και έλεγχος εύρεσης αγγελίας όπου βγάζει μήνυμα ότι δεν βρέθηκε.

Τελικά, στην σελίδα χρησιμοποιείται το MenuBar και δημιουργείται το πλαίσιο εμφάνισης του περιεχομένου στο οποίο αξιοποιείται η συνάρτηση AdCard δίνοντας τα στοιχεία που χρειάζεται για να προσαρμοστεί στο περιεχόμενο δηλαδή id, furniture, showFavorites με τιμή false clickable και false. Στο πλαίσιο υπάρχουν δύο κουμπιά από τα οποία το ένα είναι το κουμπί επεξεργασίας που οδηγεί τον χρήστη στη σελίδα επεξεργασίας μαζί με το id της αγγελίας {`/edit-ad/${furniture.id}`} και το κουμπί διαγραφής που χρησιμοποιεί την μέθοδο handleDelete.

EditAd

Η EditAd αποτελεί το component που χρησιμοποιείται για την επεξεργασία μιας αγγελίας. Ο χρήστης οδηγείται στην σελίδα, πατώντας το κουμπί “Επεξεργασία” που βρίσκεται στην σελίδα FurnitureDetails.

Συγκεκριμένα χρησιμοποιεί την useNavigate() και τα state formData για την αποθήκευση των στοιχείων αγγελίας, και selectedImageFile για την αποθήκευση της εικόνας αγγελίας. Έχει επίσης, { id } useParams() και βρίσκει το userId του χρήστη μέσω του localStorage.

Στη useEffect γίνεται αρχικά έλεγχος εύρεσης του χρήστη, ώστε αν δεν βρεθεί τον ανακατευθύνει στην σελίδα σύνδεσης. Αν βρεθεί, κάνει έλεγχο για την εύρεση της αγγελίας προχωρώντας με GET αίτημα στην μέθοδο getFurnitureDetails του FurnitureController. Παίρνει σε JSON τα στοιχεία και τα αποθηκεύει στο formData.

Η συνάρτηση handleChange, ενημερώνει το formData με βάση τα στοιχεία που συμπλήρωσε ο χρήστης.

Η handleImageChange, παίρνει το αρχείο που ανέβασε ο χρήστης, αν εκείνος επιλέξει να αλλάξει την εικόνα αγγελίας.

Για την αποθήκευση των αλλαγών, υπάρχει η handleSubmit, που είναι async συνάρτηση, δημιουργεί ένα νέο FormData στο οποίο του δίνει τα στοιχεία της αγγελίας name, description, price, furnitureCondition, furnitureCategory και selectedImageFile. Κάνει αίτημα τύπου PUT στην μέθοδο updateFurniture του FurnitureController και αν αυτό επιτύχει εμφανίζει μήνυμα και οδηγεί τον χρήστη στην /my-ads ενώ αν αποτύχει εμφανίζει μήνυμα αποτυχίας.

Τέλος, η handleCancel, εμφανίζει στον χρήστη μήνυμα για να επιβεβαιώσει την ακύρωση αλλαγών, ανακατευθύνοντας τον στην σελίδα '/my-ads' αν επιβεβαιώσει την ακύρωση.

Στην σελίδα, εμφανίζεται το MenuBar και το πλαίσιο AdInfo με παραμέτρους όπως τη συνάρτηση handleChange για την αλλαγή των στοιχείων, τη handleImageChange, για την αλλαγή της εικόνας handleSubmit για την αποθήκευση των νέων στοιχείων και isEdit με τιμή true για να δώσει την κατάλληλη μορφοποίηση στην σελίδα. Τέλος, υπάρχει κουμπί ακύρωσης που χρησιμοποιεί την handleCancel.

Purchase

Η Purchase component απευθύνεται στην σελίδα που χρησιμοποιείται για την πραγματοποίηση μιας παραγγελίας. Ο χρήστης οδηγείται σε αυτήν, πατώντας το κουμπί “Αγορά” που βρίσκεται στην σελίδα AdDetails.

Περιλαμβάνει την useNavigate() και το useParams με id, ενώ για useState έχει την ad για αποθήκευση της αγγελίας, formData για αποθήκευση της παραγγελίας που θα δημιουργηθεί και isSubmitting για την υποβολή της. Κρατάει το userId μέσω της localStorage.

Η συνάρτηση handleChange, η οποία ενημερώνει το formData με τα στοιχεία που συμπλήρωσε ο χρήστης στο αντίστοιχο πεδίο.

Στην useEffect γίνεται αρχικά έλεγχος εύρεσης του χρήστη μέσω του id του, και εάν δεν βρεθεί τον ανακατευθύνει στην σελίδα σύνδεσης. Αν βρεθεί, τότε κάνει αίτημα GET στην μέθοδο getFurnitureDetails του FurnitureController λαμβάνοντας JSON αποτέλεσμα το οποίο αποθηκεύει στην ad. Δείχνει μήνυμα φόρτωσης αν δεν βρεθεί αγγελία.

Η handleCashOnDelivery, κάνει POST αίτημα, στην μέθοδο createOrder του OrdersController δίνοντας το JSON με τα στοιχεία που έχει συμπληρώσει ο χρήστης και δείχνει μήνυμα υποβολής.

Η σελίδα εμφανίζει το MenuBar και δημιουργείται η φόρμα που δείχνει τον τίτλο αγγελίας με την τιμή, και έχει ως input fullName, email, address, number, city, postal μήνυμα ότι η πληρωμή γίνεται με αντικαταβολή και κουμπί αποθήκευσης παραγγελίας.

MyOrders

Το component χρησιμοποιείται για την προβολή των παραγγελιών που έχει κάνει ένας χρήστης.

Συγκεκριμένα, χρησιμοποιεί το navigate και το order όπου θα αποθηκεύονται οι παραγγελίες και παίρνει τον χρήστη μέσω του localStorage.

Η συνάρτηση useEffect,αρχικά ελέγχει αν ο χρήστης είναι συνδεδεμένος ώστε να οδηγηθεί στην /login. Πραγματοποιεί ένα αίτημα GET προς την μέθοδο getMyOrders του OrdersController δίνοντας το userId ως παράμετρο. Αν το αίτημα αποτύχει εμφανίζει το κατάλληλο μήνυμα, αλλιώς παίρνει τις παραγγελίες σε JSON μορφή και τις βάζει στο order.

Στη σελίδα, αν δεν έχουν φορτωθεί ακόμα οι παραγγελίες εμφανίζει μήνυμα φόρτωσης. Περιλαμβάνει το MenuBar, και ελέγχει αν ο πίνακας order είναι άδειος, δείχνει ότι δεν υπάρχουν παραγγελίες, διαφορετικά για κάθε αγγελία που βρίσκεται σε αυτό, εμφανίζει σε ένα πλαίσιο τον τίτλο του προϊόντος, την ημερομηνία της παραγγελίας, και την τιμή του προϊόντος.

OrderDetails

Το OrderDetails, είναι η σελίδα προβολής λεπτομερειών παραγγελίας. Ο χρήστης οδηγείται σε αυτήν πατώντας πάνω σε μία παραγγελία του. Περιλαμβάνει πληροφορίες όπως τα στοιχεία που συμπλήρωσε στην φόρμα Purchase, και το username του χρήστη από τον οποίο αγόρασε το έπιπλο.

Περιλαμβάνει useParams το id, useNavigate, και useState το order για την αποθήκευση πληροφοριών παραγγελίας και loading για μήνυμα φόρτωσης.

Η useEffect, περιέχει την ασύγχρονη μέθοδο fetchOrder, στην οποία πραγματοποιεί αίτημα (/order/details/${id}) στη μέθοδο orderDetails του OrdersController, και τα δεδομένα που έλαβε σε JSON, τα αποθηκεύει στο order. Μετά την ολοκλήρωση του αιτήματος, ορίζει σε false τη loading για την προβολή μηνύματος.

Η ασύγχρονη handleDelete, υλοποιεί την διαγραφή παραγγελίας, όπου αρχικά εμφανίζει στον χρήστη μήνυμα για να επιβεβαιώσει η να ακυρώσει την διαγραφή. Αν επιβεβαιώσει, προχωράει

στο fetch του (/order/delete/${id}) endpoint με μέθοδο DELETE. Αν η διαγραφή επιτύχει, εμφανίζεται μήνυμα στον χρήστη.

Στην σελίδα, εμφανίζεται μήνυμα φόρτωση, αν τα δεδομένα φορτώνονται, και το MenuBar. Εμφανίζονται οι πληροφορίες της αγγελίας, όπως ο τίτλος προϊόντος, το όνομα χρήστη του πωλητή, τα στοιχεία που συμπληρώθηκαν, δηλαδή ονοματεπώνυμο, email, διεύθυνση και την τιμή και ημερομηνία υποβολής αγγελίας. Τέλος, υπάρχει και το κουμπί διαγραφής, που χρησιμοποιεί την handleDelete.

Contact

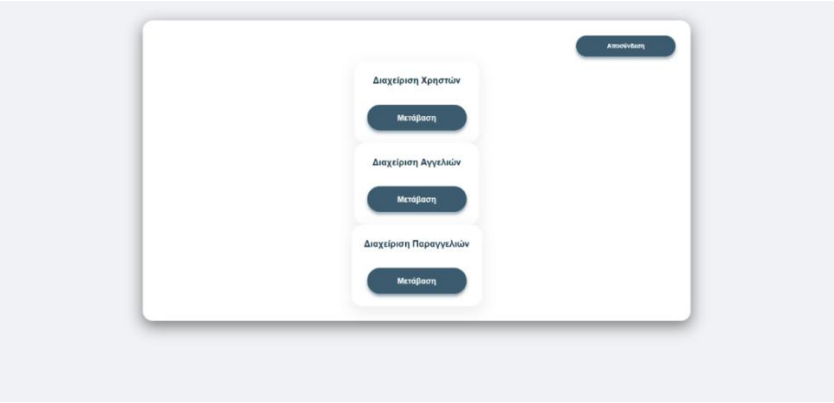
Η σελίδα contact, περιέχει πληροφορίες με τις οποίες μπορεί ο χρήστης να επικοινωνήσει με τον διαχειριστή της πλατφόρμας. Περιλαμβάνει το menuBar component, και ένα πλαίσιο με τις πληροφορίες.

5 Δοκιμές και Αξιολόγηση

5.1 Λειτουργίες Διαχειριστή

5.1.1 Αρχική σελίδα

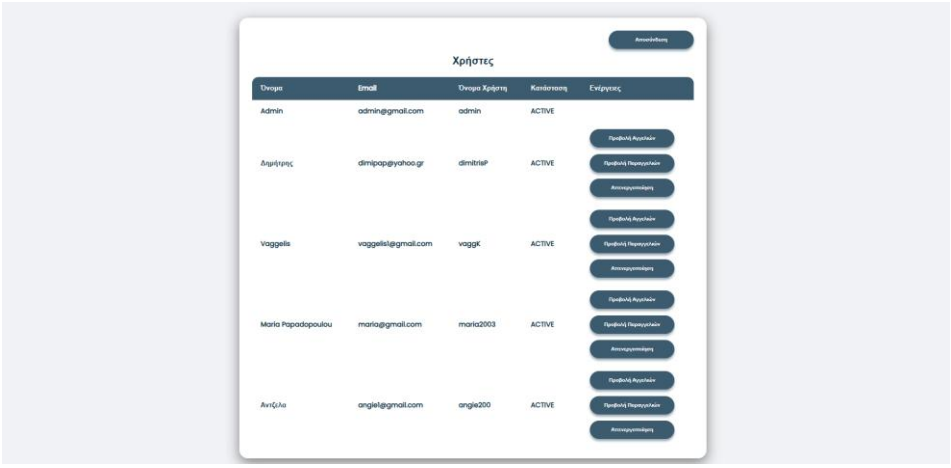
Η αρχική σελίδα που έχει ο διαχειριστής μετά την είσοδο του στο σύστημα, περιλαμβάνει ένα πλαίσιο από το οποίο μπορεί να επιλέξει την προβολή όλων των χρηστών της πλατφόρμας, όλων των αγγελιών και όλων των παραγγελιών. Έχει επίσης το κουμπί αποσύνδεσης.



Εικόνα 5.1 Αρχική σελίδα διαχειριστή

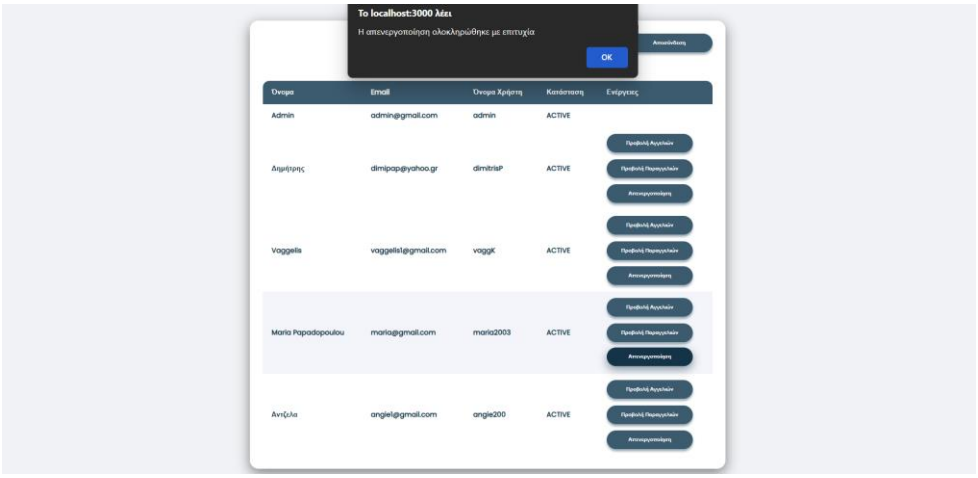
5.1.2 Προβολή Χρηστών

Ο διαχειριστής, πατώντας το κουμπί προβολής χρηστών, μεταφέρεται στη σελίδα διαχείρισης τους. Εκεί, έχει την επιλογή προβολής των αγγελιών τους, των παραγγελιών τους και την απενεργοποίηση ή ενεργοποίηση του λογαριασμού τους.



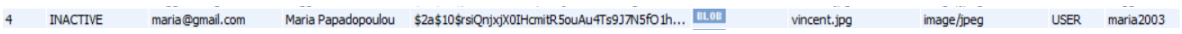
Εικόνα 5.2 Σελίδα διαχείρισης χρηστών

Πατώντας το κουμπί απενεργοποίησης, ο λογαριασμός του χρήστη απενεργοποιείται. Έτσι βγάζει μήνυμα επιτυχίας.



Εικόνα 5.3 Απενεργοποίηση χρήστη

Τα αποτελέσματα έχουν αποθηκευτεί στην βάση.

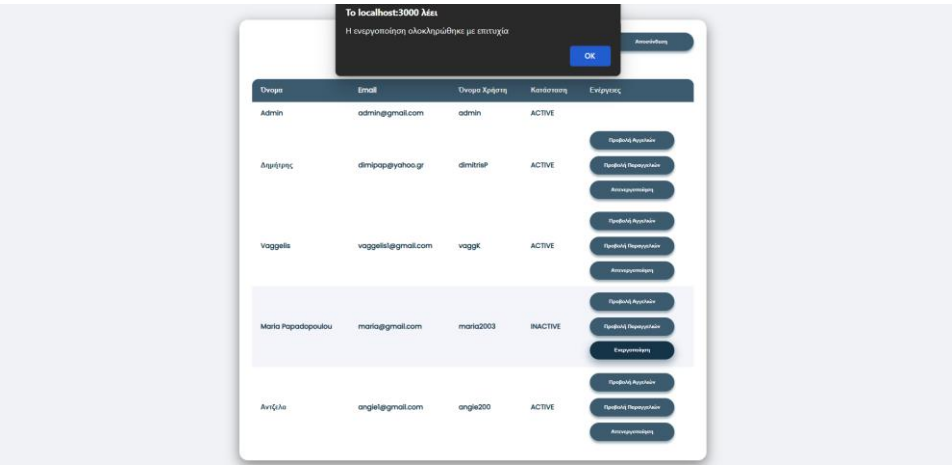


Εικόνα 5.4 Ενημερωμένη βάση μετά την απενεργοποίηση

Ο χρήστης πλέον δεν μπορεί να συνδεθεί με τα στοιχεία του.

Εικόνα 5.5 Αποτυχία σύνδεσης χρήστη μετά την απενεργοποίηση του

Η ενεργοποίηση του γίνεται με το πάτημα του κουμπιού ενεργοποίησης, εμφανίζοντας μήνυμα επιτυχίας.



Εικόνα 5.6 Ενεργοποίηση χρήστη

Οι αλλαγές αποθηκεύτηκαν στη βάση.

Εικόνα 5.7 Ενημερωμένη βάση μετά την ενεργοποίηση

Τώρα μπορεί ξανά να συνδεθεί.

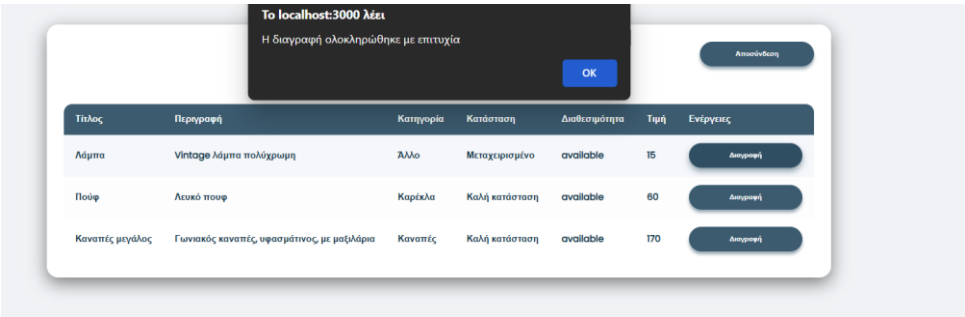
Εικόνα 5.8 Επιτυχία σύνδεσης μετά την ενεργοποίηση

5.1.3 Προβολή αγγελιών χρήστη

Πατώντας στο κουμπί «Προβολή Αγγελιών», εμφανίζονται οι αγγελίες του χρήστη και η δυνατότητα διαγραφής τους.

Εικόνα 5.9 Σελίδα διαχείρισης αγγελιών χρήστη

Πατώντας την διαγραφή, εμφανίζει μήνυμα επιτυχίας και διαγράφει την αγγελία.



Εικόνα 5.10 Διαγραφή αγγελίας χρήστη

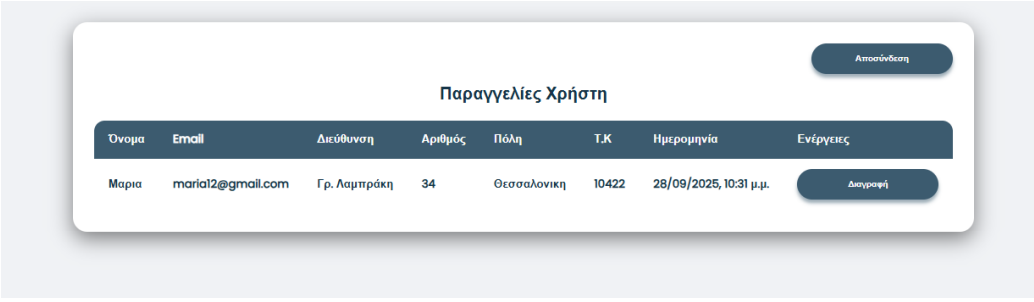
Οι αλλαγές έχουν αποθηκευτεί στη βάση.

	id	description	furniture_category	furniture_condition	image	image_name	image_type	name	price	state	user_id
▶	14	Λευκό πουφ	Καρέκλα	Καλή κατάσταση	BLOB	pouf.jpg	image/jpeg	Πούφ	60	available	4
	15	Γωνιακός καναπές, υφασμάτινος, με μαξιλάρια	Καναπές	Καλή κατάσταση	BLOB	usedcouch.jpg	image/jpeg	Καναπές μεγάλος	170	available	4
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Εικόνα 5.11 Ενημερωμένη βάση μετά την διαγραφή

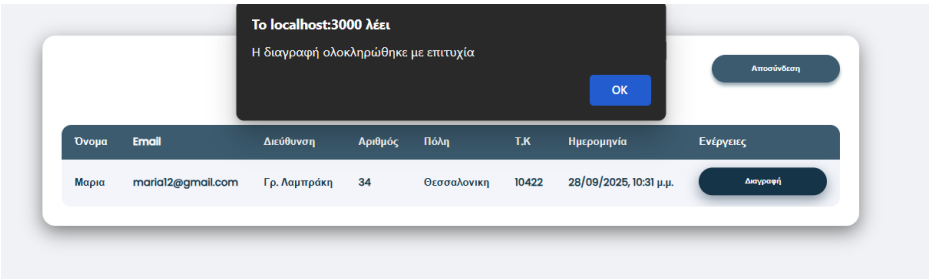
5.1.4 Προβολή παραγγελιών χρήστη

Αντίστοιχα με τις αγγελίες, το κουμπί «Προβολή Παραγγελιών», εμφανίζει τις παραγγελίες ενός χρήστη και τη δυνατότητα διαγραφής τους.



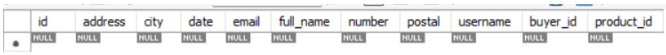
Εικόνα 5.12 Σελίδα διαχείρισης παραγγελιών χρήστη

Πατώντας το κουμπί διαγραφής, εμφανίζεται μήνυμα και διαγράφεται η παραγγελία.



Εικόνα 5.13 Διαγραφή παραγγελίας χρήστη

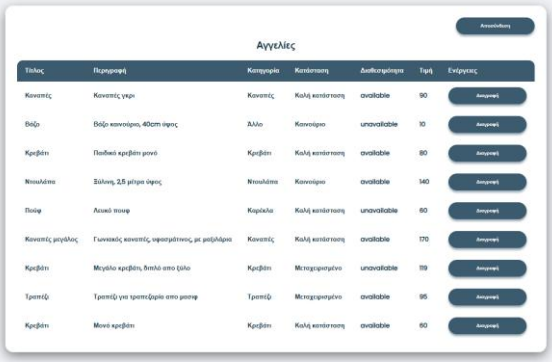
Οι αλλαγές αποθηκεύτηκαν στη βάση.



Εικόνα 5.14 Ενημερωμένη βάση μετά την διαγραφή

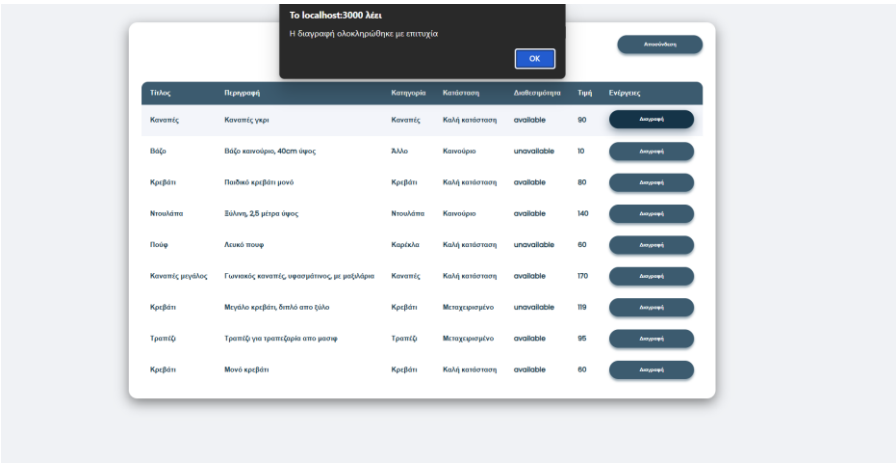
5.1.5 Προβολή Αγγελιών

Με το πάτημα του κουμπιού «Προβολή Αγγελιών» από το μενού του χρήστη, μπορεί να δει και να διαχειριστεί όλες τις αγγελίες της πλατφόρμας.



Εικόνα 5.15 Σελίδα διαχείρισης αγγελιών της πλατφόρμας

Πατώντας διαγραφή, η αγγελία διαγράφεται και δείχνει μήνυμα επιτυχίας.



Εικόνα 5.16 Διαγραφή αγγελίας

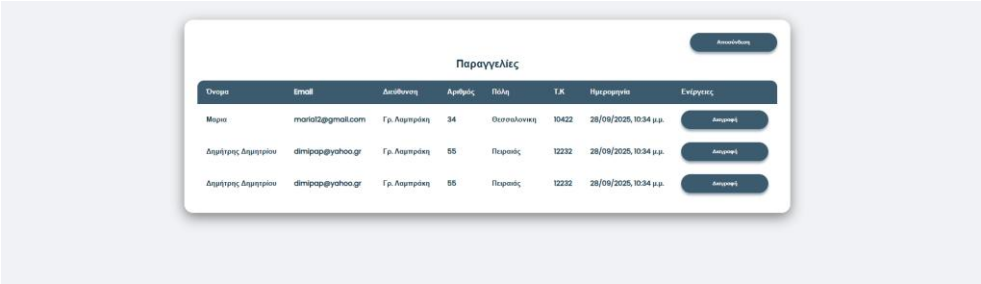
Η αγγελία έχει διαγραφτεί από την βάση.

id	description	furniture_category	furniture_condition	image	image_name	image_type	name	price	state	user_id
10	Βάζο καινούριο, 40cm ύψος	Άλλο	Καινούριο	BLOB	vase.jpg	image/jpeg	Βάζο	10	unavailable	2
12	Παιδικό κρεβάτι μονό	Κρεβάτι	Καλή κατάσταση	BLOB	singlekidsbed.jpg	image/jpeg	Κρεβάτι	80	available	3
13	Ξύλινη, 2,5 μέτρα ύψος	Ντουλάπα	Καινούριο	BLOB	closet.jpg	image/jpeg	Ντουλάπα	140	available	3
14	Λευκό πουφ	Καρέκλα	Καλή κατάσταση	BLOB	pouf.jpg	image/jpeg	Πούφ	60	unavailable	4
15	Γωνιακός καναπές, υφασμάτινος, με μαξιλάρια	Καναπές	Καλή κατάσταση	BLOB	usedcouch.jpg	image/jpeg	Καναπές μεγάλος	170	available	4
16	Μεγάλο κρεβάτι, διπλό απο ξύλο	Κρεβάτι	Μεταχειρισμένο	BLOB	bedwooden.jpg	image/jpeg	Κρεβάτι	119	unavailable	3
17	Τραπέζι για τραπεζαρία απο μασιφ	Τραπέζι	Μεταχειρισμένο	BLOB	table.jpg	image/jpeg	Τραπέζι	95	available	2
18	Μονό κρεβάτι	Κρεβάτι	Καλή κατάσταση	BLOB	krev.webp	image/webp	Κρεβάτι	60	available	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Εικόνα 5.17 Ενημερωμένη βάση μετά την διαγραφή αγγελίας

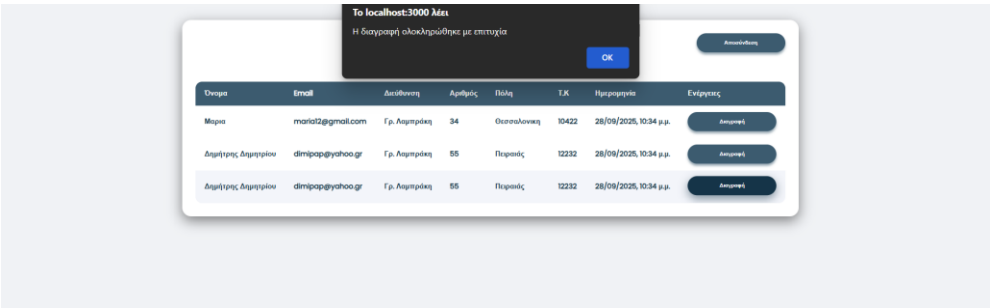
5.1.6 Προβολή Παραγγελιών

Πατώντας το κουμπί «Προβολή Παραγγελίας», ο διαχειριστής μπορεί να δει και να διαγράψει παραγγελίες.



Εικόνα 5.18 Σελίδα διαχείρισης παραγγελιών της πλατφόρμας

Το κουμπί «Διαγραφή», διαγράφει μια παραγγελία εμφανίζοντας μήνυμα.



Εικόνα 5.19 Διαγραφή παραγγελίας

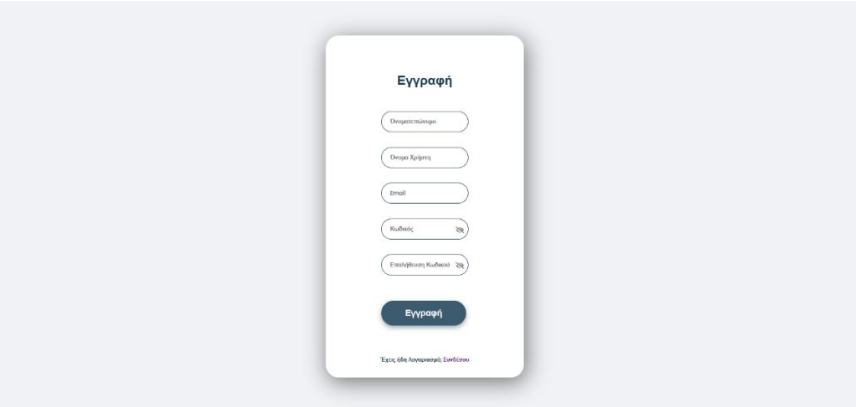
Η παραγγελία έχει διαγραφτεί από την βάση.

id	address	city	date	email	full_name	number	postal	username	buyer_id	product_id
13	Γρ. Λαμπράκη	Θεσσαλονικη	2025-09-28 22:34:06.419931	maria12@gmail.com	Μαρια	34	10422	maria2003	4	10
14	Γρ. Λαμπράκη	Πειραιάς	2025-09-28 22:34:25.841209	dimipap@yahoo.gr	Δημήτρης Δημητρίου	55	12232	dimitrisP	2	16
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Εικόνα 5.20 Ενημερωμένη βάση μετά την διαγραφή

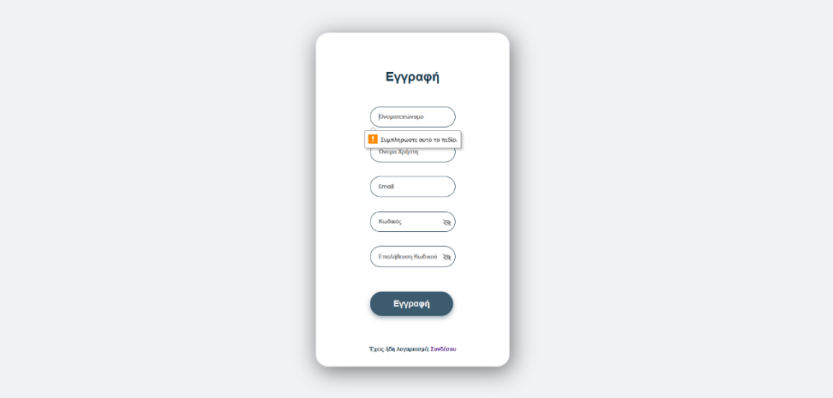
5.2 Εγγραφή χρήστη

Ο χρήστης, μόλις εισέλθει στην πλατφόρμα, έχει την δυνατότητα εγγραφής όπου του παρουσιάζεται η παρακάτω φόρμα για την συμπλήρωση των στοιχείων του. Στα στοιχεία περιλαμβάνεται το ονοματεπώνυμο του, το όνομα χρήστη, το email και κωδικός πρόσβασης μαζί με την επιβεβαίωση. Εφόσον ο χρήστης έχει ήδη λογαριασμό μπορεί να πατήσει στο σύνδεσμο “Συνδέσου” για να μεταφερθεί στη σελίδα σύνδεσης.



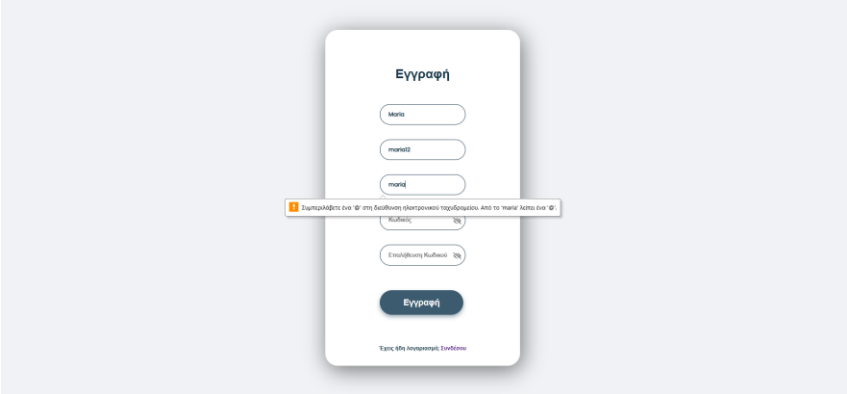
Εικόνα 5.21 Φόρμα εγγραφής

Αν προσπαθήσει να κάνει εγγραφή χωρίς τα στοιχεία να είναι συμπληρωμένα, θα εμφανιστεί το μήνυμα συμπλήρωσης των πεδίων καθώς όλα τα πεδία είναι required.



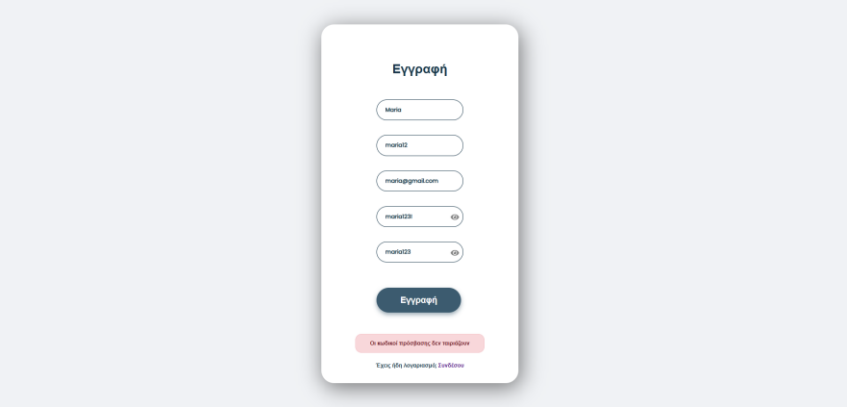
Εικόνα 5.22 Μήνυμα συμπλήρωσης πεδίων

Στην περίπτωση που το email δεν συμπληρωθεί με την κατάλληλη μορφή, ξανά εμφανίζει μήνυμα συμπλήρωσης του πεδίου με την μορφή email.



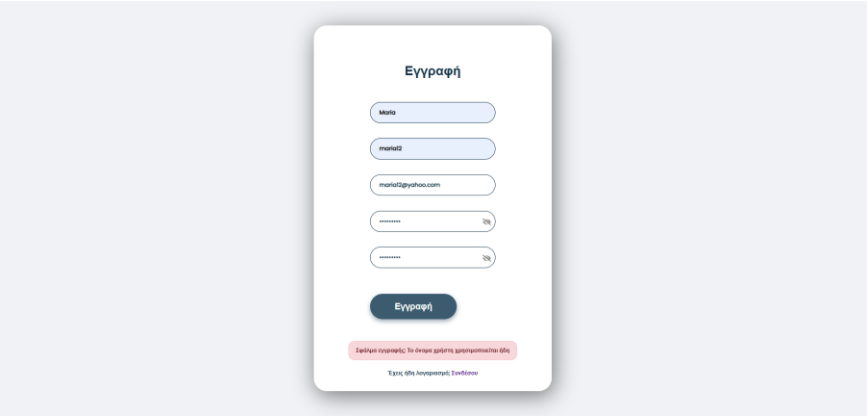
Εικόνα 5.23 Μήνυμα συμπλήρωσης email

Παρακάτω φαίνεται και η περίπτωση που ο χρήστης συμπληρώσει διαφορετικούς κωδικούς στα δύο πεδία, εμφανίζεται το μήνυμα ότι οι κωδικοί δεν ταιριάζουν.



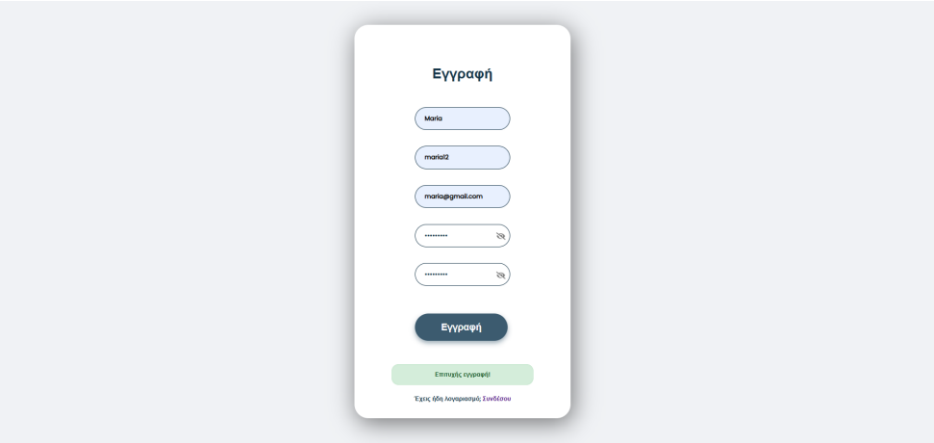
Εικόνα 5.24 Μήνυμα σφάλματος για μη ταύτιση κωδικού και επιβεβαίωσης

Αν ο χρήστης προσπαθήσει να κάνει εγγραφή με όνομα χρήστη που χρησιμοποιείται ήδη, η πλατφόρμα θα του εμφανίσει μήνυμα σφάλματος και η εγγραφή θα αποτύχει.



Εικόνα 5.25 Μήνυμα σφάλματος για υπάρχοντα χρήστη

Η εγγραφή είναι επιτυχής εφόσον όλα τα πεδία είναι συμπληρωμένα και με σωστό τύπο.



Εικόνα 5.26 Επιτυχής εγγραφή

Ο χρήστης δημιουργήθηκε και αποθηκεύτηκε στην βάση με κρυπτογραφημένο κωδικό πρόσβασης.

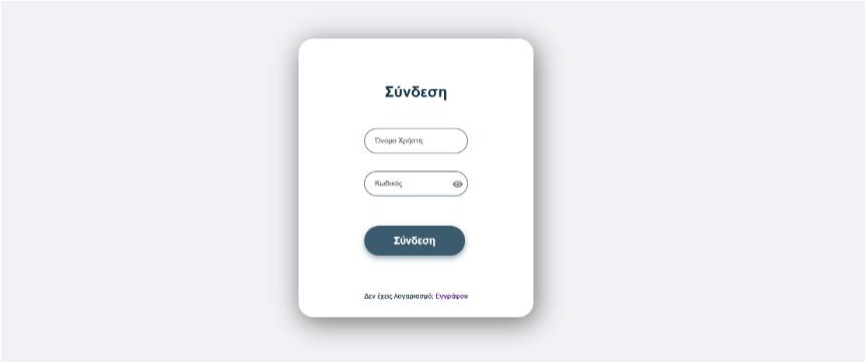
4	ACTIVE	maria@gmail.com	Maria	$2a$10$VF.epdj8aDDHOS9s09njzut2riagQLP72...	BLOB	default.jpg	image/jpg	USER	maria12

Εικόνα 5.27 Δημιουργημένος χρήστης στη βάση

Εφόσον η εγγραφή είναι επιτυχής, ο χρήστης μεταφέρεται στην σελίδα σύνδεσης.

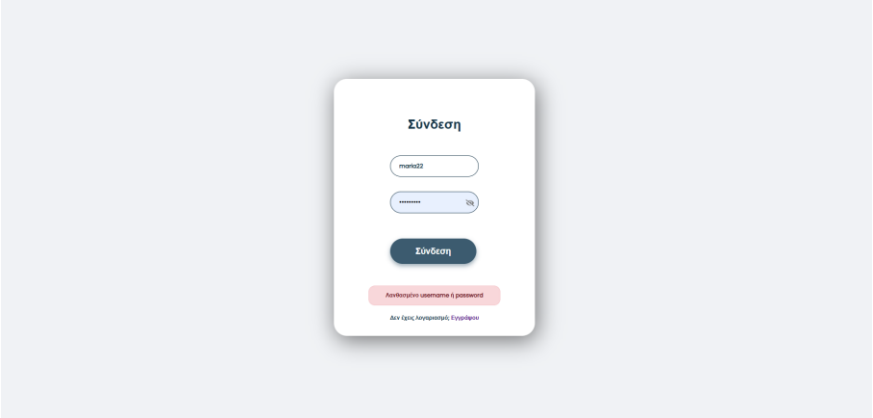
5.3 Σύνδεση χρήστη

Η φόρμα σύνδεσης του χρήστη, απαιτεί το όνομα χρήστη του και τον κωδικό.



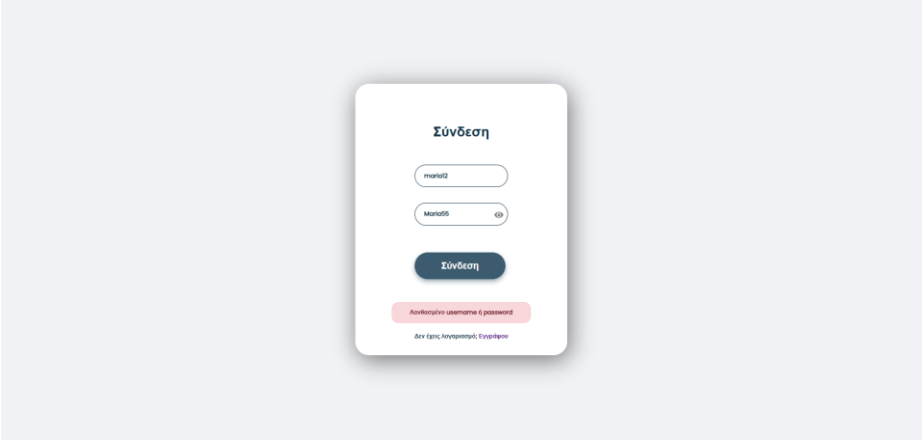
Εικόνα 5.28 Φόρμα σύνδεσης

Σε περίπτωση που ο χρήστης εισάγει λάθος όνομα χρήστη, εμφανίζει μήνυμα σφάλματος.



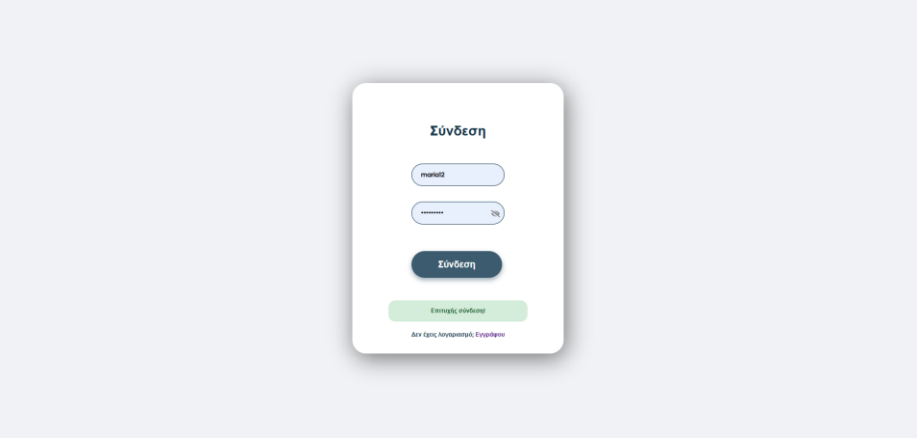
Εικόνα 5.29 Αποτυχία σύνδεσης λόγω λανθασμένου username

Σε περίπτωση που συμπληρώσει λάθος κωδικό, ξανά εμφανίζεται μήνυμα σφάλματος.



Εικόνα 5.30 Αποτυχία σύνδεσης λόγω λανθασμένου κωδικού

Αν τα στοιχεία είναι σωστά, γίνεται η αυθεντικοποίηση, και εμφανίζεται μήνυμα επιτυχίας.

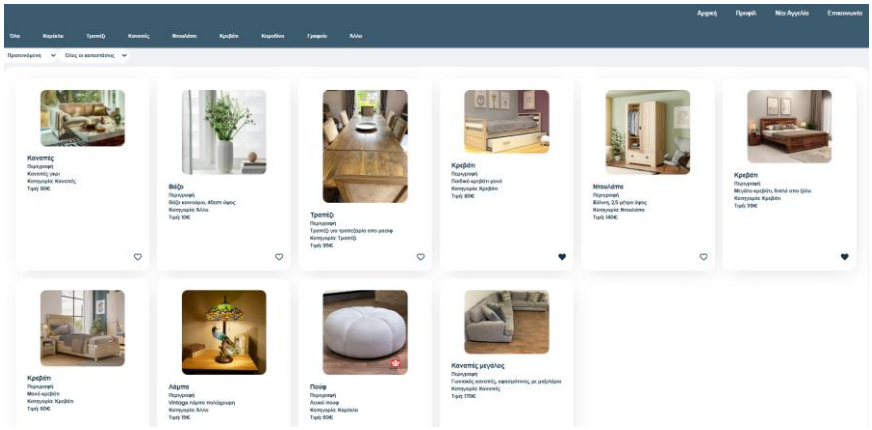


Εικόνα 5.31 Επιτυχής σύνδεση

5.4 Λειτουργίες Χρήστη

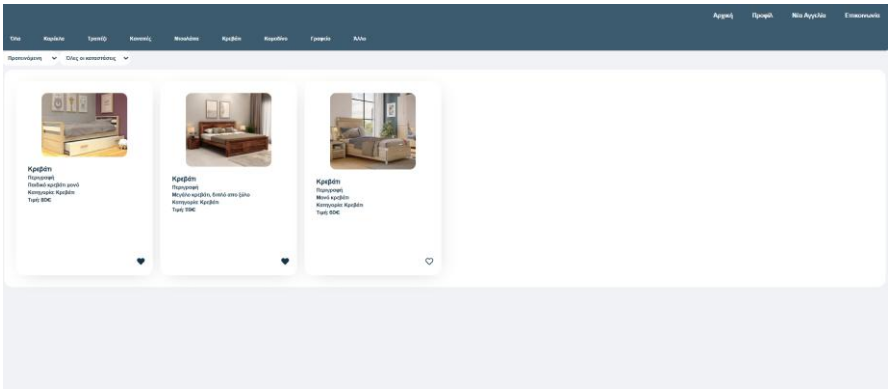
5.4.1 Αρχική σελίδα

Ο χρήστης μετά την σύνδεση του, μεταφέρεται στην αρχική σελίδα της εφαρμογής που περιέχει το μενού και τις αγγελίες, μαζί με τα φίλτρα. Αυτά μπορούν να εφαρμοστούν από οποιαδήποτε σελίδα βρίσκεται ο χρήστης καθώς το μενού με τις κατηγορίες βρίσκεται σε όλες τις σελίδες της πλατφόρμας, και από εκεί μπορεί στην συνέχεια να εφαρμόσει και τα φίλτρα ταξινόμησης και κατάστασης.



Εικόνα 5.32 Αρχική σελίδα

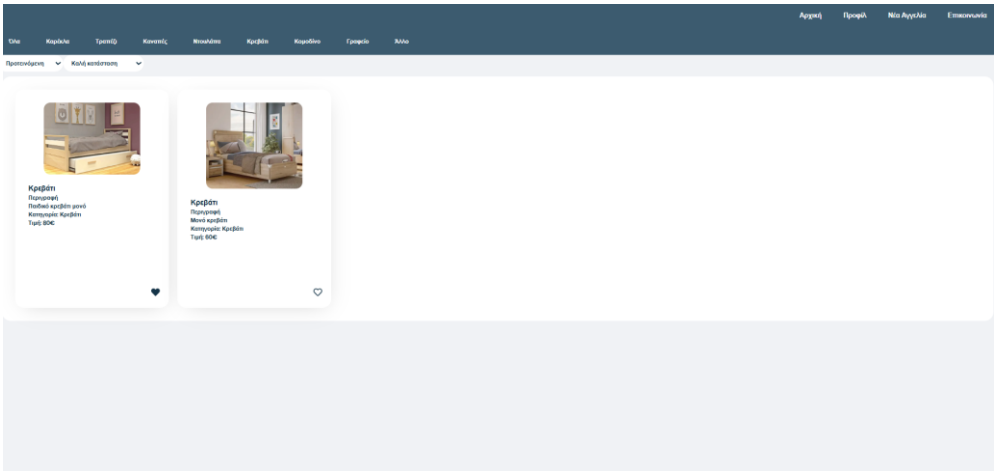
Με το πάτημα των κατηγοριών, εμφανίζονται οι αγγελίες που ανήκουν σε αυτή την κατηγορία



Εικόνα 5.33 Φίλτρο κατηγορίας «Κρεβάτια»

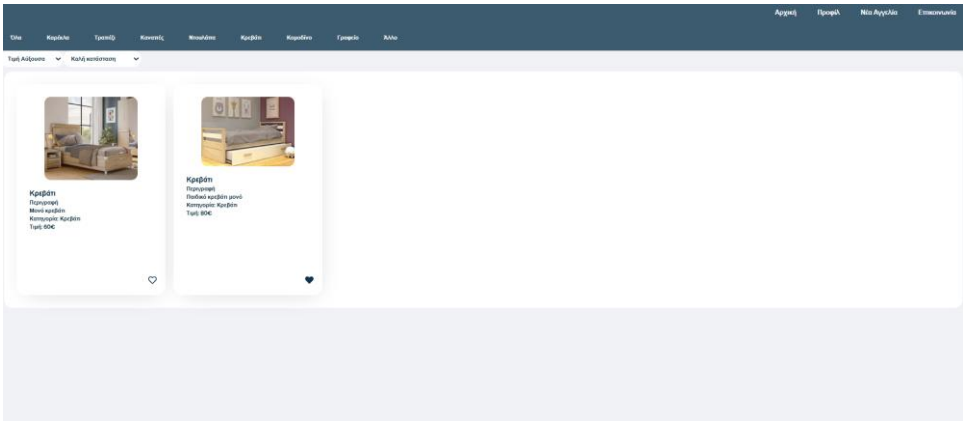
Αντίστοιχα, με το πάτημα της κατηγορίας ‘Άλλο’ εμφανίζονται όσες αγγελίες δεν ανήκουν σε κάποια από τις άλλες κατηγορίες.

Σε οποιαδήποτε κατηγορία, ο χρήστης μπορεί να επιλέξει φίλτρα. Παρακάτω φαίνεται να έχει επιλεχθεί η κατάσταση ‘Καλή κατάσταση’, όπου δείχνει μόνο τα κρεβάτια, εφόσον βρίσκεται στην κατηγορία με τα κρεβάτια, που έχουν αυτή την κατάσταση.



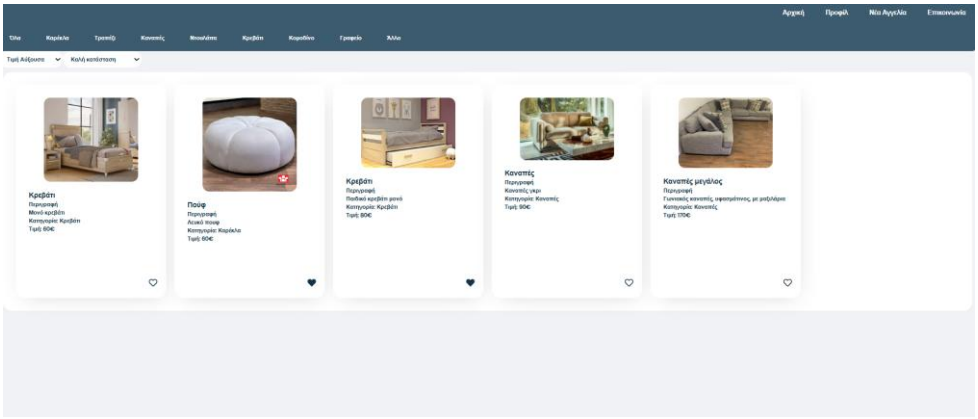
Εικόνα 5.34 Φίλτρο κατηγορίας και κατάστασης

Μπορεί επίσης να εφαρμόσει φίλτρο ταξινόμησης, όπου όπως φαίνεται ταξινομεί ανάλογα. Ο χρήστης επέλεξε ‘Τιμή αύξουσα’



Εικόνα 5.35 Φίλτρο κατηγορίας, κατάστασης και ταξινόμησης

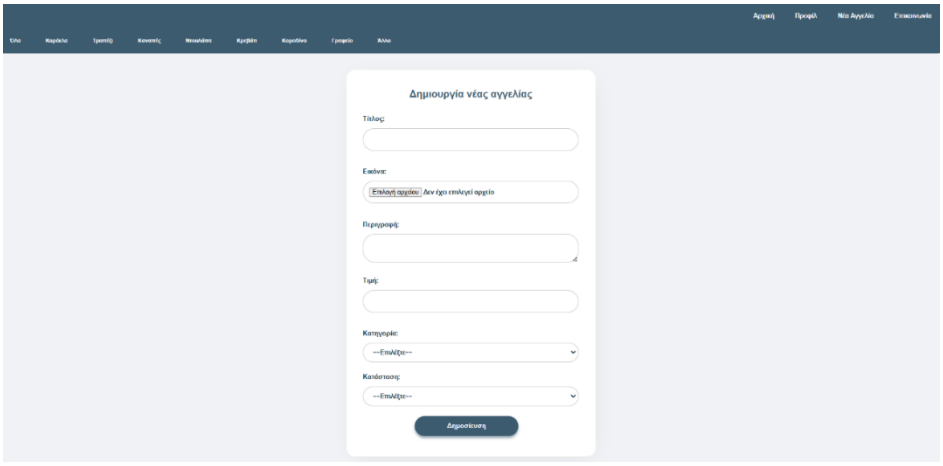
Τα φίλτρα μπορούν να παραμείνουν και να εφαρμοστούν ακόμα και με την αλλαγή της κατηγορίας. Όπως φαίνεται, επιλέγοντας μόνο κατάσταση και ταξινόμηση, δείχνει αγγελίες από οποιαδήποτε κατηγορία.



Εικόνα 5.36 Υπάρχοντα φίλτρα στην κατηγορία «Όλα»

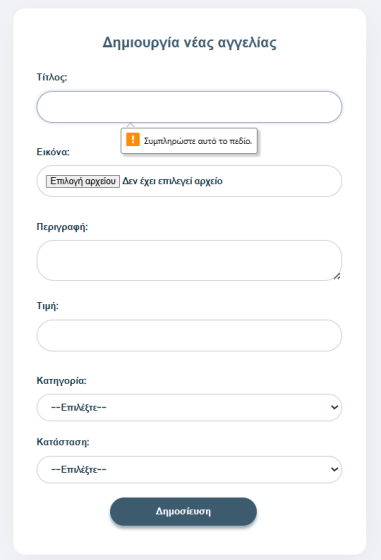
5.4.2 Προσθήκη Αγγελίας

Στην παρακάτω εικόνα, φαίνεται η φόρμα που εμφανίζεται στον χρήστη με το πάτημα του κουμπιού “Νέα Αγγελία” που βρίσκεται στο μενού. Ο χρήστης πρέπει να συμπληρώσει τον τίτλο της αγγελίας, την εικόνα, την περιγραφή, την τιμή, και να επιλέξει κατηγορία και κατάσταση.



Εικόνα 5.37 Φόρμα δημιουργίας αγγελίας

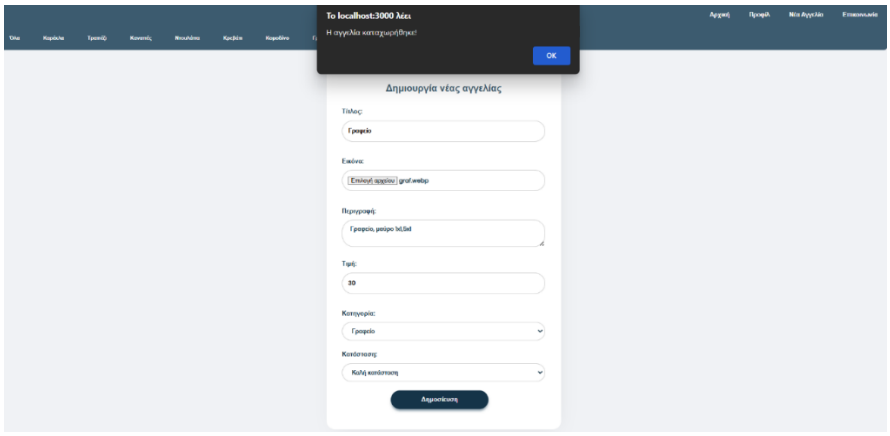
Αν ο χρήστης δεν συμπληρώσει τα στοιχεία, και προσπαθήσει να δημιουργήσει την αγγελία, του εμφανίζεται μήνυμα για την συμπλήρωση πεδίων.



Εικόνα 5.38 Μήνυμα συμπλήρωσης πεδίων

Ο χρήστης έχει την δυνατότητα να επιλέξει ανάμεσα σε κάποιες κατηγορίες, ενώ αν το έπιπλο δεν ταυτίζεται με καμία από αυτές μπορεί να επιλέξει την κατηγορία “Άλλο”

Με το πάτημα του κουμπιού `Δημοσίευση`, δημιουργείται η αγγελία και εμφανίζεται μήνυμα επιτυχίας.



Εικόνα 5.39 Επιτυχής δημιουργία αγγελίας

Η αγγελία έχει δημιουργηθεί επιτυχώς όπως φαίνεται και στην βάση.

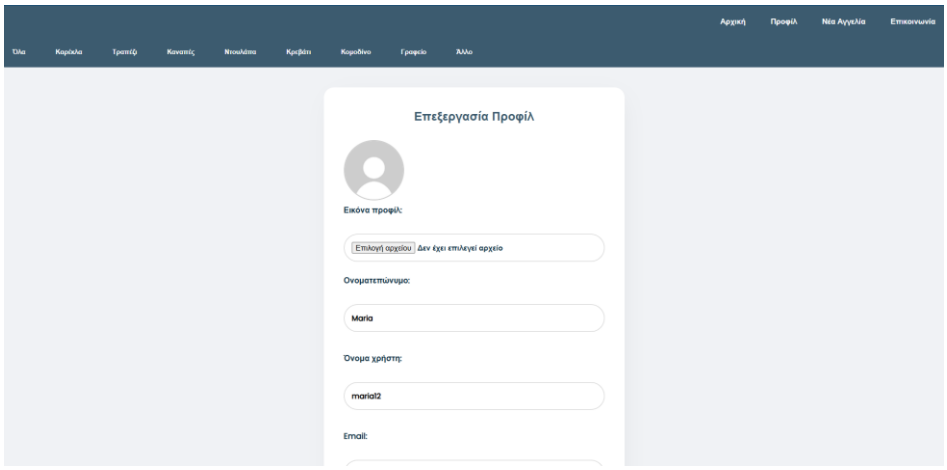
Εικόνα 5.40 Αποθήκευση αγγελίας στη βάση

5.4.3 Επεξεργασία προφίλ

Για την επεξεργασία των στοιχείων του χρήστη, θα πρέπει να εισέλθει στην σελίδα που οδηγεί το κουμπί `Επεξεργασία Προφίλ`

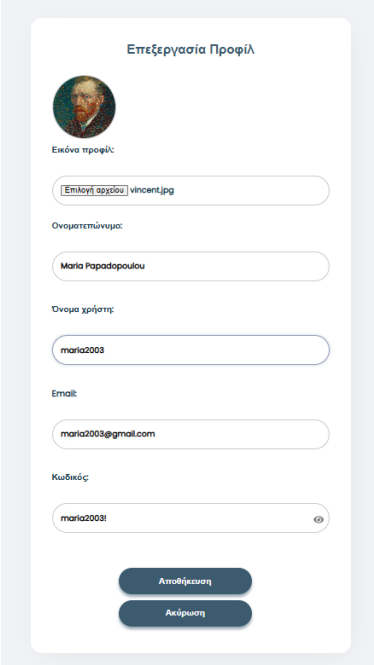
Εικόνα 5.41 Επιλογή «Επεξεργασία Προφίλ» στο μενού

Παρακάτω, φαίνεται η φόρμα με τα στοιχεία του χρήστη.



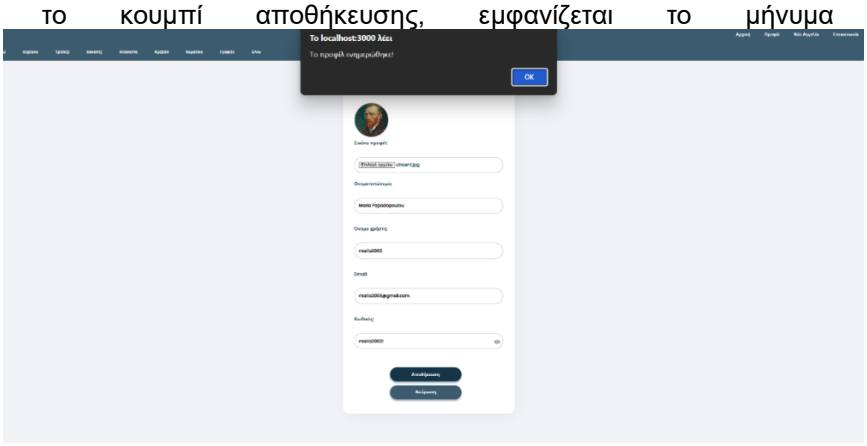
Εικόνα 5.42 Φόρμα επεξεργασίας προφίλ

Αν ο χρήστης επιλέξει να αλλάξει εικόνα, έχει την δυνατότητα να την δει προτού αποθηκεύσει τις αλλαγές.



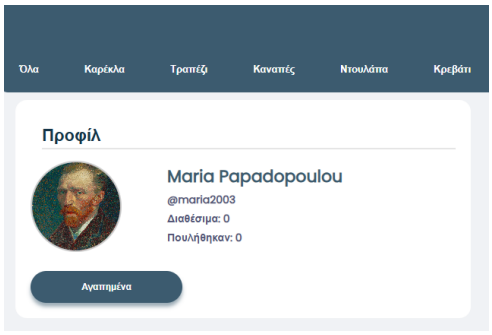
Εικόνα 5.43 Εμφάνιση επιλεγμένης εικόνας πριν την αποθήκευση

Πατώντας το κουμπί αποθήκευσης, εμφανίζεται το μήνυμα επιτυχίας.



Εικόνα 5.44 Αποθήκευση αλλαγών

Ο χρήστης μπορεί πλέον να δει την καινούρια εικόνα προφίλ του, καθώς και τα υπόλοιπα στοιχεία που άλλαξε.



Εικόνα 5.45 Εμφάνιση αλλαγών στη σελίδα του χρήστη

Όπως φαίνεται, τα στοιχεία άλλαξαν και στην βάση.

4	ACTIVE	maria@gmail.com	Maria Papadopoulou	$2a$10$rsiQnjxjX0IHcmitR.5ouAu4Ts9J7N5fO.1h...	BLOB	vincent.jpg	image/jpeg	USER	maria2003

Εικόνα 5.46 Αποθήκευση αλλαγών στη βάση

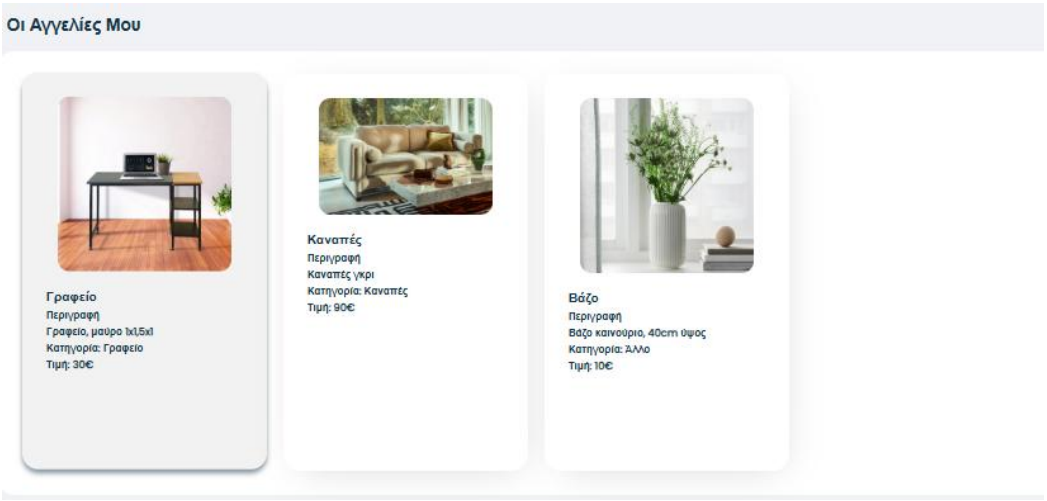
Ο χρήστης πλέον δεν μπορεί να εισέλθει με τα παλιά του στοιχεία.

Εικόνα 5.47 Αποτυχία σύνδεσης μετά την αλλαγή κωδικού

Σε περίπτωση που δεν θελήσει πλέον να αλλάξει τα στοιχεία αφού βρεθεί στην φόρμα, πατώντας το κουμπί ακύρωσης, εμφανίζεται μήνυμα επιβεβαίωσης χωρίς να γίνει κάποια αλλαγή.

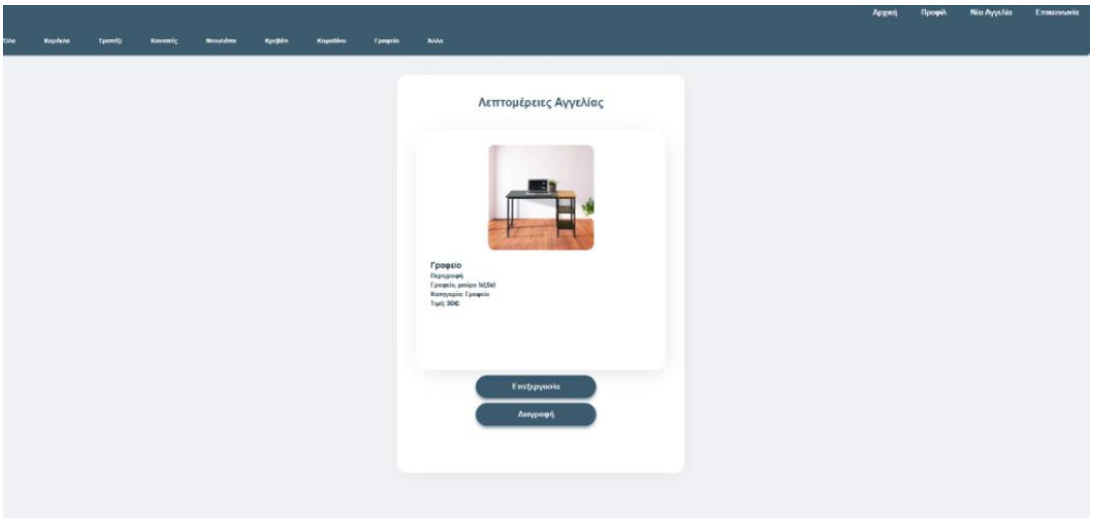
5.4.4 Επεξεργασία αγγελίας

Στο προφίλ του χρήστη, μπορεί να επιλέξει κάποια από τις διαθέσιμες αγγελίες του για επεξεργασία ή διαγραφή.



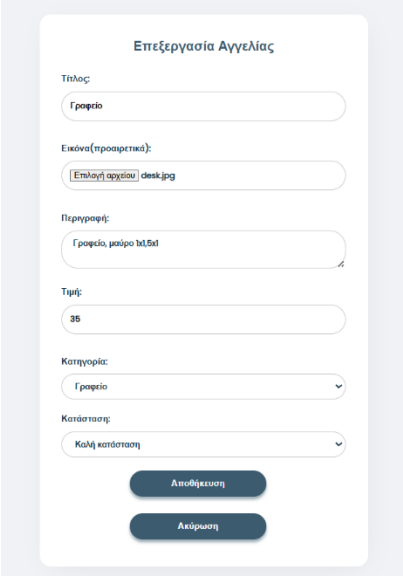
Εικόνα 5.48 Διαθέσιμες αγγελίες χρήστη

Πατώντας πάνω στην αγγελία, οδηγείται στην σελίδα με τις επιλογές.



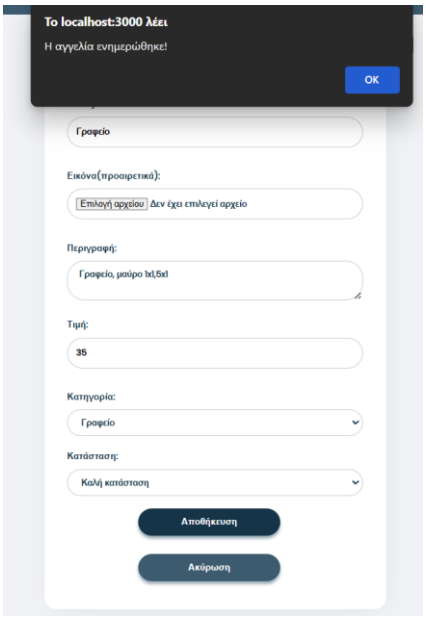
Εικόνα 5.49 Λεπτομέρειες αγγελίας και διαθέσιμες ενέργειες

Ο χρήστης οδηγείται στην φόρμα επεξεργασίας, όπου εμφανίζονται όλα τα στοιχεία της αγγελίας, και μπορεί να επιλέξει να αλλάξει κάποιο από αυτά εάν θελήσει. Όπως και στην επεξεργασία προφίλ, έχει την δυνατότητα να ακυρώσει τις αλλαγές με το πάτημα του αντίστοιχου κουμπιού.



Εικόνα 5.50 Φόρμα επεξεργασίας αγγελίας

Παρακάτω φαίνεται ότι ο χρήστης μπορεί να επεξεργαστεί τα στοιχεία της αγγελίας με επιτυχία, χωρίς να αλλάξει υποχρεωτικά την εικόνα της.

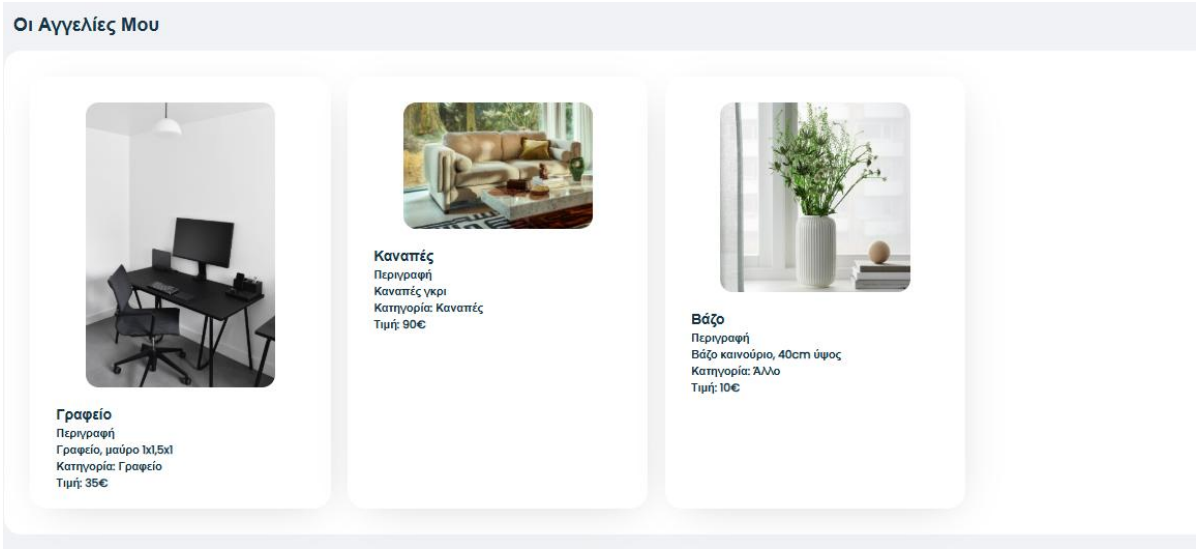


Εικόνα 5.51 Αποθήκευση αλλαγών

Η αλλαγή έγινε με επιτυχία.

Εικόνα 5.52 Ενημέρωση αλλαγών αγγελίας στη βάση

Παρακάτω φαίνεται η αλλαγή της τιμής και της εικόνας μετά την επιλογή αλλαγής της.

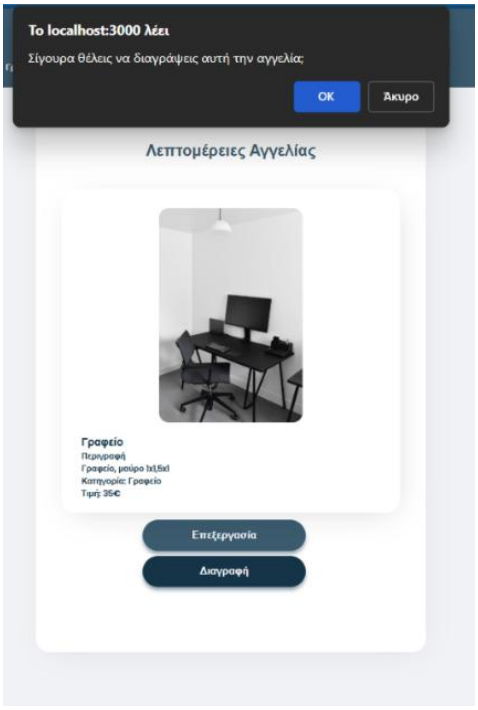


Εικόνα 5.53 Αλλαγές στη κάρτα αγγελίας

5.4.5 Διαγραφή αγγελίας

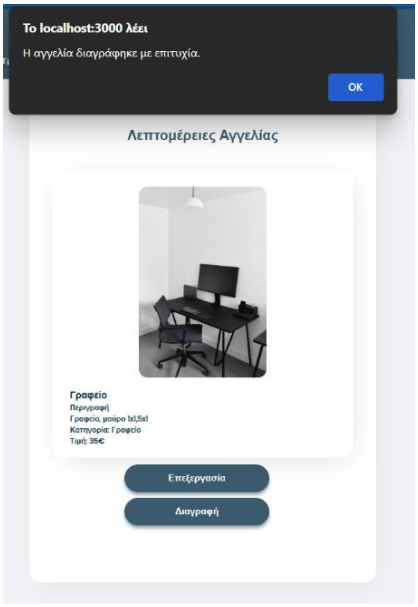
Στην σελίδα με τις ενέργειες αγγελιών, ο χρήστης με το πάτημα του κουμπιού διαγραφής, μπορεί να διαγράψει την επιλεγμένη αγγελία.

Έτσι εμφανίζεται το μήνυμα επιβεβαίωσης.



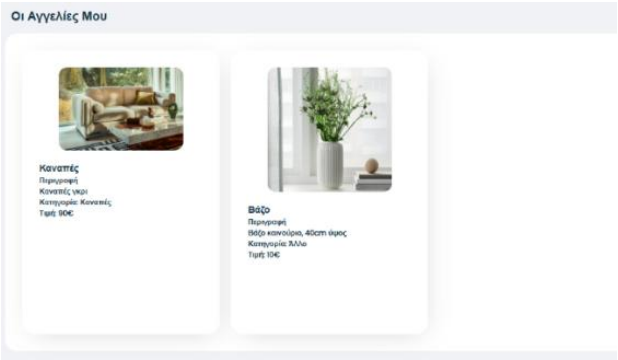
Εικόνα 5.54 Επιβεβαίωση διαγραφής

Αν επιλέξει να το διαγράψει εμφανίζει το μήνυμα επιτυχίας.



Εικόνα 5.55 Επιτυχής διαγραφή

Παρακάτω φαίνεται ότι η αγγελία που διαγράφηκε δεν υπάρχει πλέον στο προφίλ του χρήστη.



Εικόνα 5.56 Διαγραφή αγγελίας από το προφίλ χρήστη

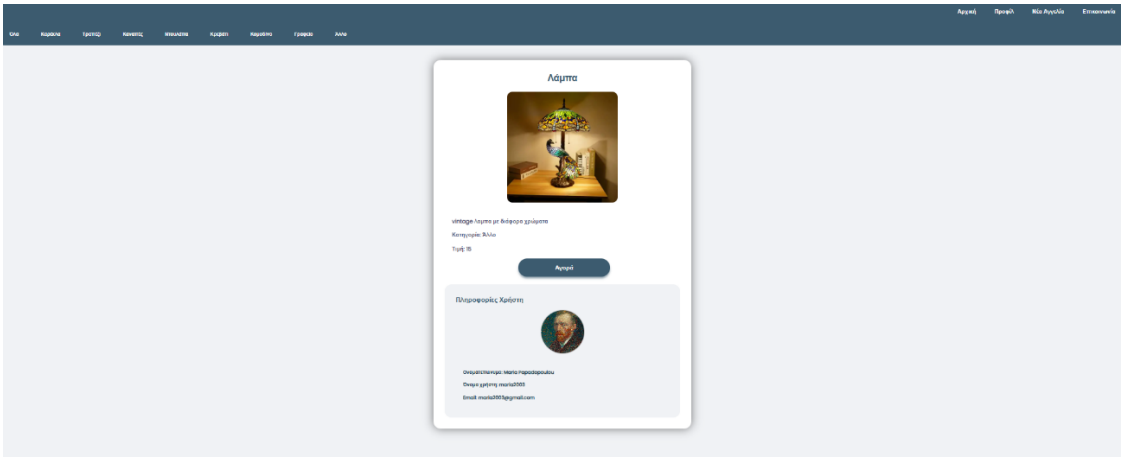
Η διαγραφή, έγινε με επιτυχία και στην βάση.

	id	description	furniture_category	furniture_condition	image	image_name	image_type	name	price	state	user_id
▶	9	Καναπές γκρι	Καναπές	Καλή κατάσταση	BLOB	sofa.webp	image/webp	Καναπές	90	available	2
	10	Βάζο καινούριο, 40cm ύψος	Άλλο	Καινούριο	BLOB	vase.jpg	image/jpeg	Βάζο	10	available	2

Εικόνα 5.57 Διαγραφή αγγελίας από τη βάση

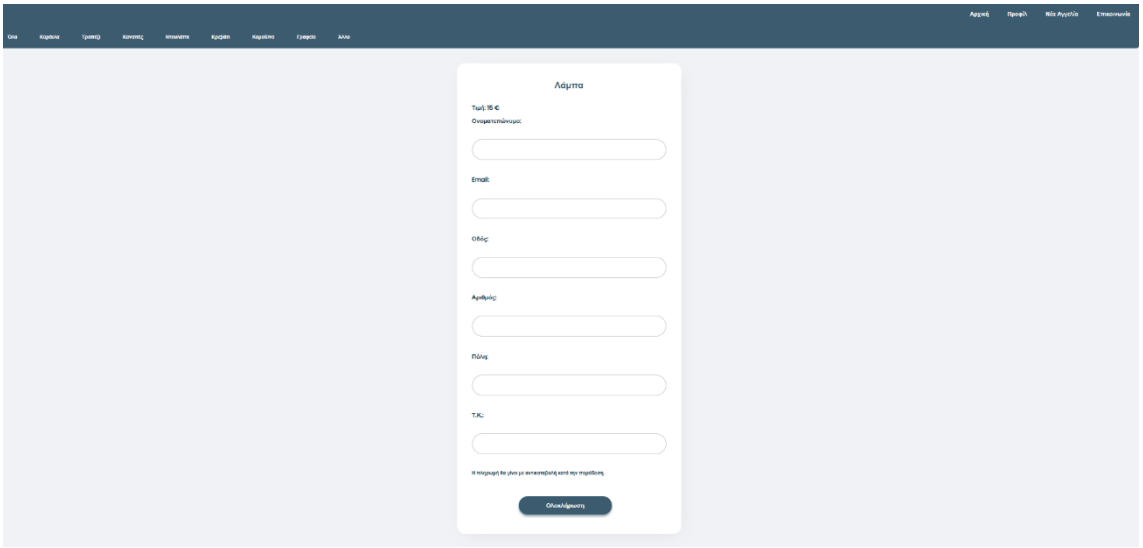
5.4.6 Αγορά επίπλου

Για την αγορά επίπλου, ο χρήστης μπορεί πατώντας το κουμπί αγοράς να προχωρήσει σε αυτήν.



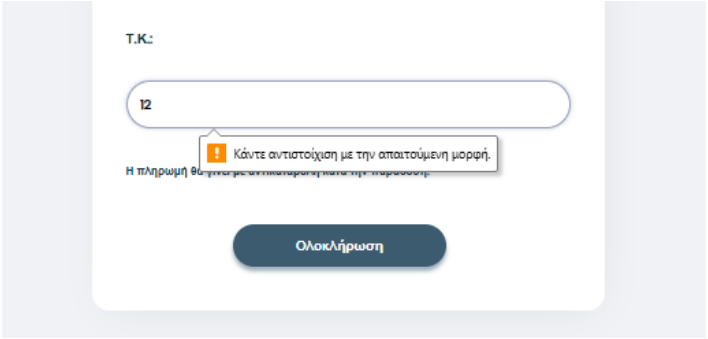
Εικόνα 5.58 Σελίδα λεπτομερειών αγγελίας προς πώληση

Εμφανίζεται η φόρμα συμπλήρωσης στοιχείων, όπου ζητείται το ονοματεπώνυμο, το email, η οδός, ο αριθμός οδού, πόλη, και ταχυδρομικός κώδικας.



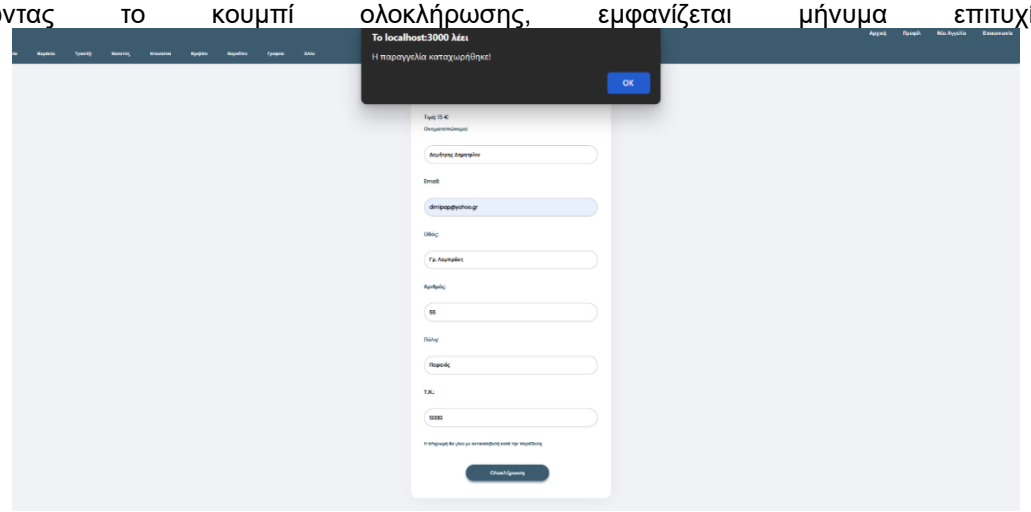
Εικόνα 5.59 Φόρμα αγοράς

Σε περίπτωση που δεν εισάγει κάποιο από τα στοιχεία, ή δεν τα εισάγει σε σωστή μορφή, εμφανίζεται μήνυμα.



Εικόνα 5.60 Μήνυμα συμπλήρωσης πεδίων με τον ζητούμενο τύπο

Πατώντας το κουμπί ολοκλήρωσης, εμφανίζεται μήνυμα επιτυχίας.



Εικόνα 5.61 Δημιουργία παραγγελίας

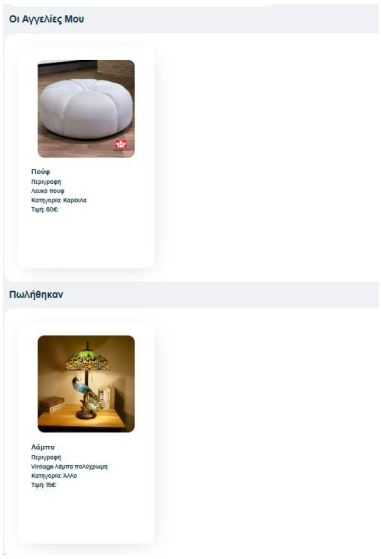
Αμέσως μετά οδηγείται στην σελίδα προβολής των παραγγελιών του.

Φαίνεται ότι η παραγγελία δημιουργήθηκε και στην βάση

9	Γρ. Λαμπράκη	Πειραιάς	2025-09-28 20:24:30.446997	dimipap@yahoo.gr	Δημήτρης Δημητρίου	55	12232	dimitrisP	2	11

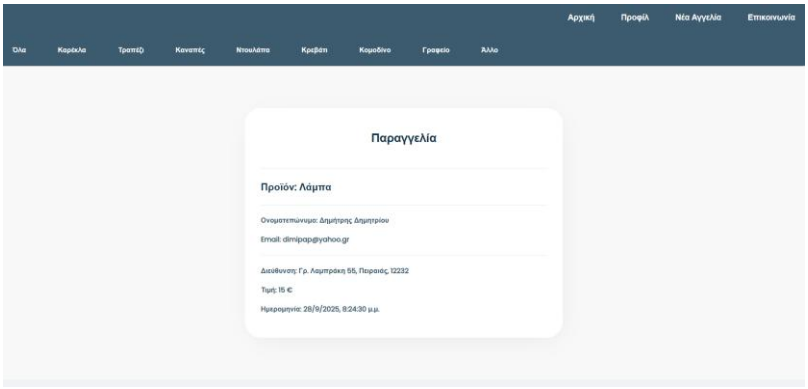
Εικόνα 5.62 Παραγγελία στην βάση

Στο προφίλ του πωλητή, φαίνεται ότι πλέον η αγγελία είναι στην κατηγορία ‘Πωλήθηκαν’.



Εικόνα 5.63 Ενημέρωση σελίδας πωλητή μετά την αγορά

Από εδώ, ο πωλητής μπορεί πατώντας πάνω στην αγγελία, να δει τις λεπτομέρειες παραγγελίας με τα στοιχεία του αγοραστή.



Εικόνα 5.64 Προβολή λεπτομερειών παραγγελίας από την μεριά του πωλητή

Η κατάσταση της αγγελίας είναι πλέον unavailable.

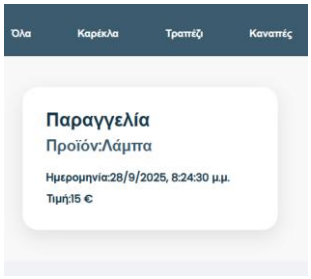
Εικόνα 5.65 Κατάσταση αγγελίας στη βάση μετά την αγορά

5.4.7 Διαγραφή Παραγγελίας

Ο χρήστης έχει την δυνατότητα να διαγράψει μια παραγγελία του.

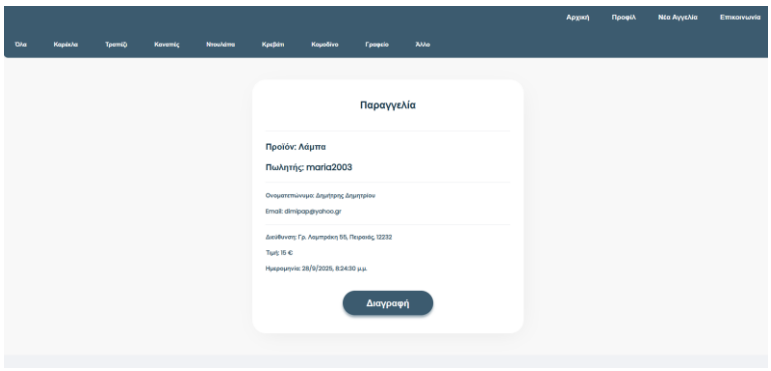
Πατώντας στις «Παραγγελίες», οδηγείται στην σελίδα με τις παραγγελίες του.

Παρακάτω φαίνεται η παραγγελία που πραγματοποίησε πριν, και πατώντας πάνω της, οδηγείται στις λεπτομέρειες της παραγγελίας.



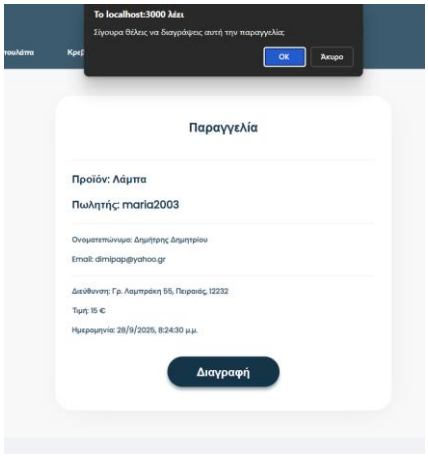
Εικόνα 5.66 Παραγγελίες χρήστη

Στην παρακάτω εικόνα, φαίνεται η σελίδα με τις λεπτομέρειες παραγγελίας, με την επιλογή διαγραφής της.



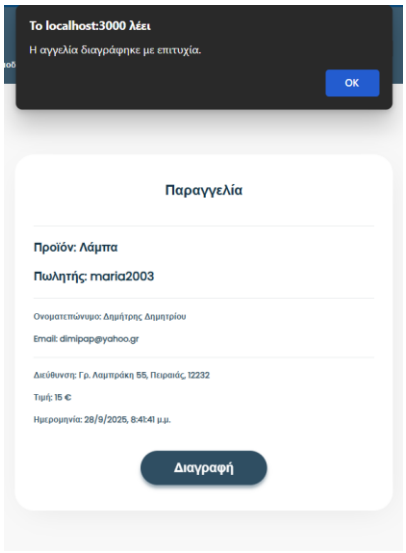
Εικόνα 5.67 Λεπτομέρειες παραγγελίας από τη μεριά του αγοραστή

Πατώντας στο κουμπί διαγραφής, εμφανίζεται μήνυμα επιβεβαίωσης στον χρήστη.



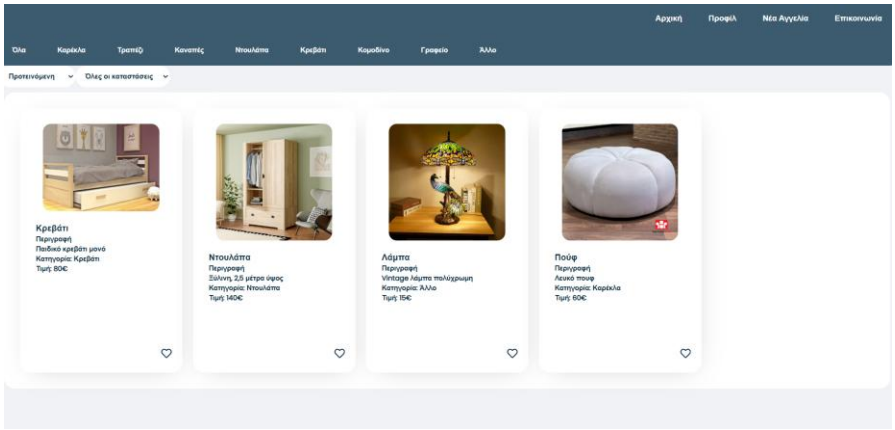
Εικόνα 5.68 Επιβεβαίωση διαγραφής

Αφού επιβεβαιώσει την διαγραφή, εμφανίζεται μήνυμα επιτυχίας.



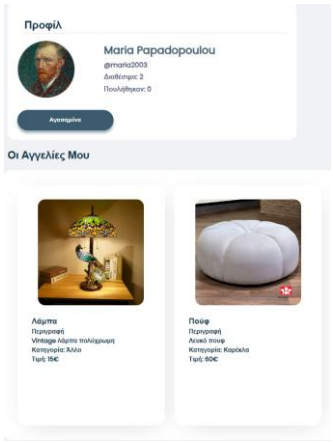
Εικόνα 5.69 Διαγραφή παραγγελίας

Η αγγελία πλέον εμφανίζεται και πάλι στην αρχική σελίδα, καθώς πλέον είναι ξανά διαθέσιμη για αγορά.



Εικόνα 5.70 Επανεμφάνιση αγγελίας στην αρχική σελίδα

Αντίστοιχα, στο προφίλ του πωλητή, πλέον μπορεί να δει την αγγελία στις «Αγγελίες μου», και όχι στην κατηγορία «Πουλήθηκαν».



Εικόνα 5.71 Αλλαγή αγγελίας σε διαθέσιμη στη σελίδα του πωλητή

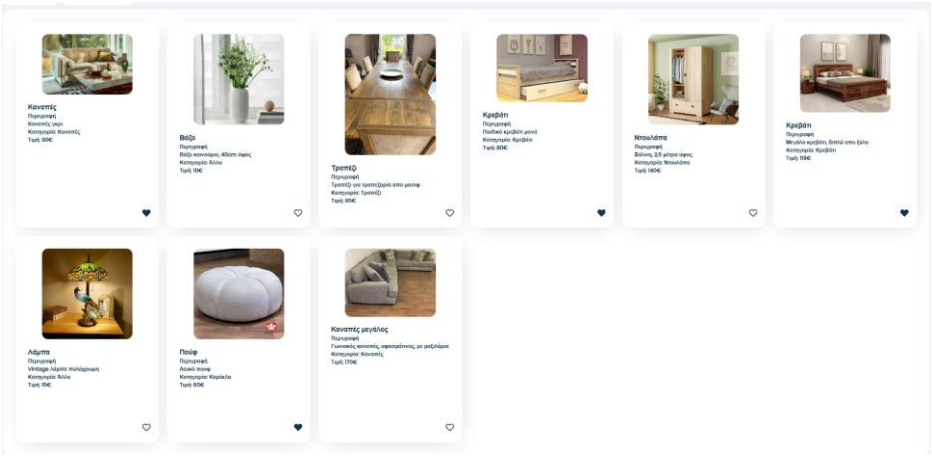
Η κατάσταση της αγγελίας άλλαξε στην βάση και είναι “available”.

11	Vintage λάμπα πολύχρωμη	Άλλο	Μεταχειρισμένο	BLOB	vintage.webp	image/webp	Λάμπα	15	available	4

Εικόνα 5.72 Αλλαγή κατάστασης αγγελίας στη βάση

5.4.8 Αγαπημένα χρήστη

Παρακάτω φαίνεται η αρχική σελίδα, όπου ο χρήστης πρόσθεσε αγγελίες στη λίστα με τα αγαπημένα του.



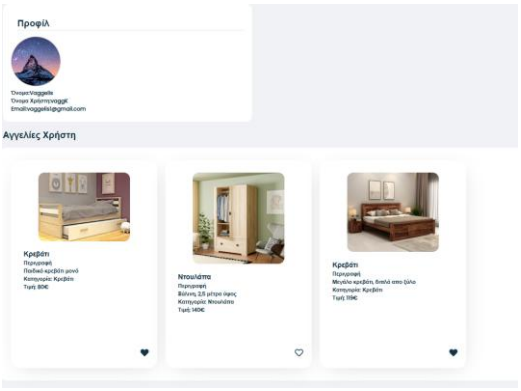
Εικόνα 5.73 Αρχική σελίδα με τις αγαπημένες αγγελίες του χρήστη

Ο πίνακας με τα αγαπημένα, ανανεώθηκε προσθέτοντας τις αγγελίες και το id του χρήστη.

user_id	furniture_id
5	9
5	12
5	14
5	16
NULL	NULL

Εικόνα 5.74 Πίνακας αγαπημένων αγγελιών χρήστη

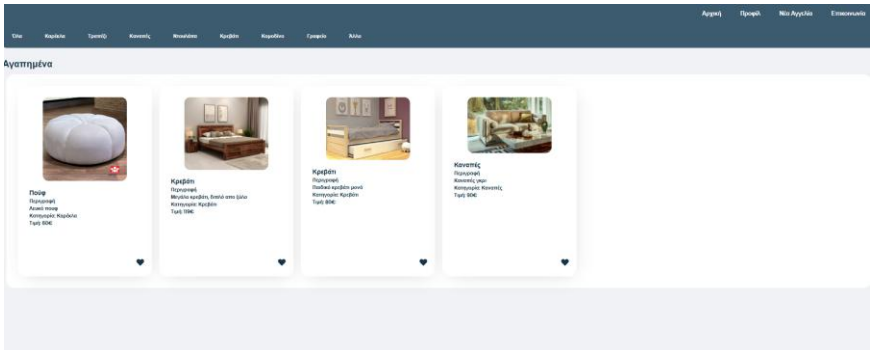
Αν ο χρήστης, επισκεφτεί το προφίλ ενός πωλητή, μπορεί να δει ποιες αγγελίες του περιλαμβάνονται στα αγαπημένα του



Εικόνα 5.75 Προβολή αγαπημένων αγγελιών του χρήστη από το προφίλ του πωλητή

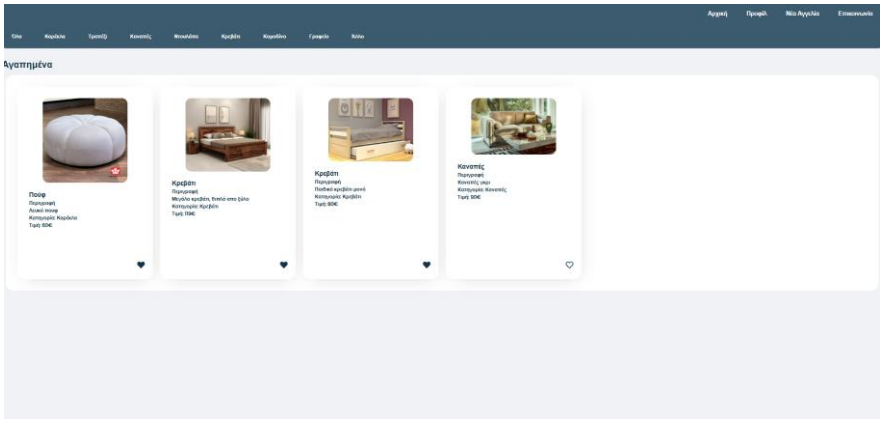
Ο χρήστης πατώντας στο κουμπί «Αγαπημένα» που βρίσκεται στο προφίλ του οδηγείται στην σελίδα με τις αγαπημένες αγγελίες του.

Παρακάτω φαίνεται η σελίδα με τις αγαπημένες αγγελίες του χρήστη.



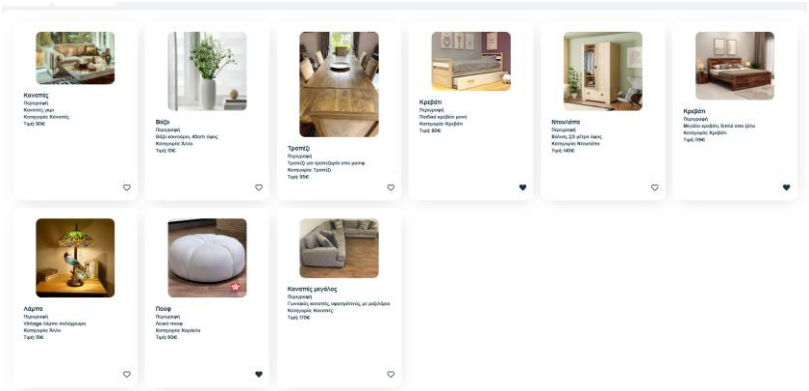
Εικόνα 5.76 Αγαπημένες αγγελίες χρήστη

Ο χρήστης μπορεί από οποιαδήποτε σελίδα, να επεξεργαστεί τις αγαπημένες αγγελίες του. Έτσι βγάζοντας μια αγγελία από αυτή τη σελίδα, οι αλλαγές θα παραμείνουν και στις υπόλοιπες.



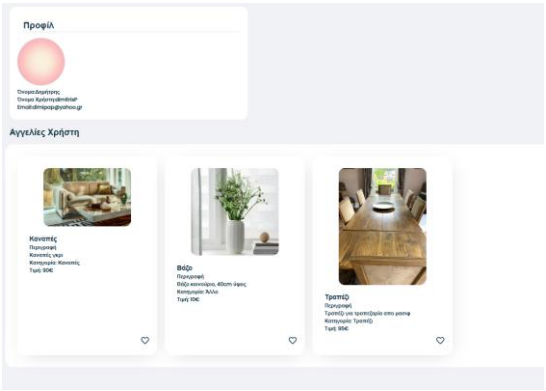
Εικόνα 5.77 Αφαίρεση αγαπημένης αγγελίας

Η αγγελία, φαίνεται και στην αρχική σελίδα ότι πλέον δεν είναι στα αγαπημένα του χρήστη.



Εικόνα 5.78 Ενημέρωση αφαίρεσης στην αρχική σελίδα

Έτσι, φαίνεται και στο προφίλ του πωλητή της αγγελίας ότι πλέον δεν είναι στα αγαπημένα.



Εικόνα 5.79 Ενημέρωση αφαίρεσης στη σελίδα πωλητή

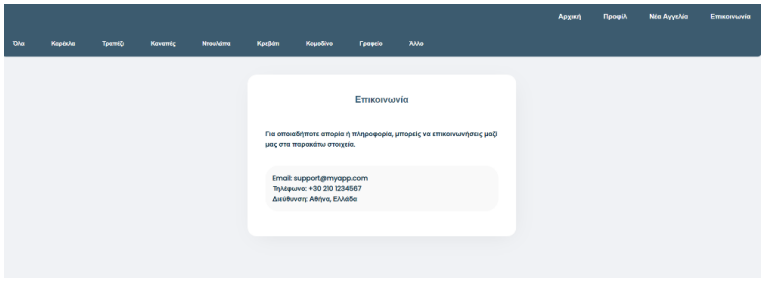
Οι αλλαγές έχουν αποθηκευτεί και στην βάση.

	user_id	furniture_id
▶	5	12
	5	14
	5	16
*	NULL	NULL

Εικόνα 5.80 Ενημέρωση αφαίρεσης στη βάση

5.4.9 Σελίδα Επικοινωνίας

Παρακάτω, φαίνεται η σελίδα με τα στοιχεία επικοινωνίας της πλατφόρμας πατώντας το κουμπί “Επικοινωνία” που υπάρχει στο μενού.



Εικόνα 5.81 Σελίδα στοιχείων επικοινωνίας με τη πλατφόρμα

6 Μελλοντικές Επεκτάσεις και Συμπεράσματα

Στο κεφάλαιο αυτό, θα αναπτυχθούν οι τρόποι με τους οποίους η λειτουργικότητα της πλατφόρμα μπορεί να διευρυνθεί σε διάφορα τμήματα της, καθώς και τα συμπεράσματα που προκύπτουν από την έρευνα, την υλοποίηση, και τις βελτιώσεις της.

6.1 Μελλοντικές Επεκτάσεις

Παρά την επιτυχή υλοποίηση της πλατφόρμας για αγορά και πώληση μεταχειρισμένων επίπλων, υπάρχουν περιθώρια επέκτασης και βελτίωσης των ήδη υπαρχόντων λειτουργιών, με στόχο την αύξηση της ασφάλειας, την επέκταση των δυνατοτήτων της εφαρμογής και την βελτίωση της συνολικής εμπειρίας του χρήστη.

Συγκεκριμένα, όσον αφορά της διαδικασία της αγοράς, η πλατφόρμα μπορεί να υποστηρίζει με πολλαπλές μεθόδους πληρωμής, εισάγοντας και τις ηλεκτρονικές πληρωμές (PayPal, Google/Apple pay) για την διευκόλυνση της διαδικασίας και της ευελιξίας των αγοραστών.

Πέρα από το email που εμφανίζεται στο προφίλ των χρηστών, η υλοποίηση ενός live chat, όπου ο χρήστης θα μπορεί άμεσα να επικοινωνεί μέσω της εφαρμογής με τον πωλητή για τυχόν απορίες ή προβλήματα, αλλά και την επικοινωνία μεταξύ των χρηστών εντός της πλατφόρμας συμβάλλοντας στην άμεση επικοινωνία.

Για την βελτιστοποίηση της περιήγησης του χρήστη στην πλατφόρμα, η προσθήκη αξιολογήσεων, θα επιτρέψει στους χρήστες να μπορούν να βλέπουν τις αξιολογήσεις άλλων χρηστών, καθώς και να προσθέτουν οι ίδιοι κριτικές δείχνοντας την αντίστοιχα εμπειρία που είχε αγοράζοντας από κάποιον συγκεκριμένο πωλητή. Παράλληλα η υλοποίηση καλαθιού, με αγγελίες που προέρχονται από τον ίδιο χρήστη, θα διευκολύνει την διαχείριση πολλαπλών αγορών.

Τέλος, όσον αφορά την διαχείριση αγγελιών, η πλατφόρμα μπορεί να επεκταθεί με την προσθήκη πολλαπλών φωτογραφιών ανά αγγελία που δημοσιεύεται, δίνοντας την δυνατότητα στον χρήστη να παρουσιάζει πολλαπλές εικόνες για το έπιπλο που θέλει να πουλήσει ώστε ένας αγοραστής να έχει μια ολοκληρωμένη εικόνα του προϊόντος.

6.2 Συμπεράσματα

Από την ανάλυση και την μελέτη που παρουσιάστηκε μπορεί να εξαχθεί το συμπέρασμα ότι η ανάπτυξη μιας εξειδικευμένης πλατφόρμας για την αγοραπωλησία μεταχειρισμένων επίπλων αποτελεί μια αναγκαία και χρήσιμη καινοτομία που καλύπτει τις ανάγκες των σύγχρονων κοινωνιών και του κενού που υπάρχει στην αγορά. Παρόλο που υπάρχουν ήδη γνωστές πλατφόρμες με C2C μοντέλο, όπως το Vinted και το Facebook Marketplace, η έλλειψη εξειδίκευσης σε μεταχειρισμένα έπιπλα περιορίζει την ευχρηστία και την αποτελεσματική διαδικασία αγοραπωλησίας για αυτά. Η παρούσα πτυχιακή εργασία, παρουσιάζει την δημιουργία μιας πλατφόρμας, με εξειδικευμένα προς τα έπιπλα φίλτρα αναζήτησης αγγελιών, και σύστημα πληρωμών εντός της πλατφόρμας καθώς και την δημοσίευση και διαχείριση των αγγελιών.

Με τη δημιουργία της πλατφόρμας, βελτιώνεται η εμπειρία του χρήστη και ενισχύει την τάση για επαναχρησιμοποίηση. Έτσι, τα οφέλη της πλατφόρμας, δεν περιορίζονται μόνο στην τεχνολογικό κομμάτι, αλλά διευρύνονται και στο περιβαλλοντικό τομέα. Η κυκλική οικονομία που είναι άμεσο αποτέλεσμα της χρήσης της πλατφόρμας, μειώνει την ανάγκη για νέα παραγωγή επίπλων, και κατά συνέπεια την επιβάρυνση του περιβάλλοντος από τις βιομηχανίες επίπλων. Η πλατφόρμα αποτελεί ένα εργαλείο που συμβάλλει άμεσα στην προώθηση πρακτικών που συνδέονται άμεσα με τη βιωσιμότητα και την μείωση του κόστους και του χρόνου που απαιτούνται για την αγορά επίπλων.

Η πτυχιακή εργασία, αποδεικνύει την σημασία που έχει η δημιουργία μιας τέτοιας πλατφόρμας που βασίζεται στα μεταχειρισμένα έπιπλα, καλύπτοντας το κενό που υπάρχει στην αγορά σε αυτόν τον τομέα και στηρίζοντας την κυκλική οικονομία, ως μέσο αντιμετώπισης των περιβαλλοντικών προβλημάτων που έχουν προκύψει στην σύγχρονη εποχή λόγω των βιομηχανιών και των αποβλήτων που αυτές δημιουργούν. Η λειτουργικότητα της πλατφόρμας

επιτρέπει στους χρήστες να την χρησιμοποιούν και να περιηγούνται σε αυτή με ευκολία, ενώ οι μελλοντικές βελτιώσεις που μπορούν να υλοποιηθούν, αποτελούν θεμέλιο για την ανάπτυξη αυτού του τομέα. Η παροχή πολλαπλών τρόπων συναλλαγών, πέρα από την ήδη υπάρχουσα, η ενσωμάτωση επικοινωνίας εντός της πλατφόρμας και το σύστημα αξιολογήσεων, αποτελούν σημαντικά στοιχεία για την ευχρηστία της.

Βιβλιογραφία

[1] Yaser Ahangari Nanehkaran INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 2, ISSUE 4, APRIL 2013

[2] Laudon, K. C., & Traver, C. G. (2020). *E-Commerce: Business, Technology, Society* (15th ed.). Pearson. https://books.google.gr/books?hl=el&lr=&id=MwEB8LuK0P0C&oi=fnd&pg=PA1&dq=e+commerce&ots=fayyd5u0lr&sig=Pao3H3G3p-rwDkN_U-wki_GvCs0&redir_esc=y#v=onepage&q&f=false

[3] Niranjanamurthy M., Kavyashree N., Jagannath S., Chahar D., *Analysis of E-Commerce and M-Commerce: Advantages, Limitations and Security Issues*, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 6, June 2013.

[4] Upadhyay, A., Mukhuty, S., Kumar, V., & Kazancoglu, Y. (2021). Blockchain technology and the circular economy: Implications for sustainability and social responsibility. *Technological Forecasting and Social Change, 164*, 120409.

[5] Geissdoerfer, M., Savaget, P., Bocken, N. M. P., & Hultink, E. J. (2017). The Circular Economy – A new sustainability paradigm? *Journal of Cleaner Production, 143*, 757–768.

[6] Wulf, F., Hagedorn, L., Pfriem, A., Munier, L., Balder, J., & Mathi, C. (έτος). *Towards digitalization of the circular economy in the furniture industry*.

[7] Hrafnkelsdóttir, K. *Second-hand furniture and climate impact: LCA modeling to explore potential emission savings of reused furniture* (AL227X Degree project in Industrial Ecology, Second cycle, 30 credits).

[8] Vinted | Όλα τα μεταχειρισμένα είδη σε μια εφαρμογή

[9] Facebook Marketplace: Αγοράστε και πουλήστε προϊόντα τοπικά ή μέσω αποστολής | Facebook

[10] 8 Top Advantages of Using React for Development

[11] Virtual DOM and Internals – React

[12] What is Java Spring Boot?—Intro to Spring Boot | Microsoft Azure

[13] 10 Reasons Why You Should Use Spring Boot - Sigma Solve Inc

[14] 8 Big Advantages of Using MySQL | Datamation

[15] MySQL

[16] Αλέπης, Ε., & Παναγιωτόπουλος,Ι-Χ.(2019).*Αντικειμενοστραφείς γλώσσες προγραμματισμού Java*.

[17] Abraham Silberschatz, Henry F. Korth, S. Sudarshan (2021). *Συστήματα Βάσεων Δεδομένων 7η Έκδοση*. Εκδόσεις: Μ.Γκιούρδας

[18] Spring Boot :: Spring Boot

[19] react-router | React Router API Reference

[20] React Reference Overview – React