ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα παραγγελιοληψίας και διαχείριση αποθήκης καταστημάτων εστίασης

Των φοιτητών
Δημητρίου Ορέστη
Αρ. Μητρώου: 164655
Αρκίλε Δάμο
Αρ. Μητρώου: 154589

Επιβλέπων
Χαράλαμπος Μπράτσας
Επίκουρος Καθηγητής

Ημερομηνία 26 Ιανουαρίου 2025

Σύστημα παραγγελιοληψίας και διαχείριση αποθήκης καταστημάτων εστίασης
23252
Δημητρίου Ορέστης
Αρκίλε Δάμο
Χαράλαμπος Μπράτσας
19 Σεπτεμβρίου 2023
Ημερομηνία περάτωσης Δ.Ε. 26 Ιανουαρίου 2025

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

« Στους γονείς μας, που μας στήριξαν και μας δίδαξαν να κυνηγάμε τα όνειρά μας »

Πρόλογος

Η επιλογή του θέματος αυτής της διπλωματικής εργασίας έγινε με έμπνευση από τις προσωπικές μας εμπειρίες κατά τη διάρκεια της εργασίας μας σε διάφορα καταστήματα εστίασης. Σε ορισμένα από αυτά, δεν υπήρχε καθόλου σύστημα παραγγελιοληψίας, γεγονός που προκαλούσε συχνά καθυστερήσεις και μπερδέματα στο προσωπικό λόγω μεγάλης συσσώρευσης από χαρτάκια παραγγελιών. Σε άλλα καταστήματα, αν και υπήρχε σύστημα παραγγελιών, δεν υπήρχε επαρκής έλεγχος των αποθεμάτων, οδηγώντας σε σπατάλη πόρων είτε από κακοδιαχείριση από τη παραγωγή του καταστήματος είτε από εκμετάλλευση του προσωπικού.

Μέσα από την ενασχόλησή μας με αυτό το έργο, κατανοήσαμε βαθύτερα τη σημασία μιας σωστά δομημένης βάσης δεδομένων, καθώς και την αξία της ορθής χρήσης τεχνολογιών, όπως η βελτιστοποίηση SQL ερωτημάτων, για την αποδοτική επίλυση πραγματικών προβλημάτων. Παράλληλα, η ανάπτυξη του συστήματος μας έδωσε την ευκαιρία να ενισχύσουμε τις δεξιότητές μας σε νέες τεχνολογίες, ενώ δημιούργησε προοπτικές για μελλοντική εμπορική αξιοποίηση του έργου μας.

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τη δημιουργία ενός ολοκληρωμένου συστήματος παραγγελιοληψίας και διαχείρισης αποθήκης για καταστήματα εστίασης. Στόχος του συστήματος είναι η βελτίωση της λειτουργίας τέτοιων επιχειρήσεων μέσω της αυτοματοποίησης της λήψης και αποστολής της παραγγελίας, της μείωσης των λαθών και της αποτελεσματικής παρακολούθησης των αποθεμάτων.

Το σύστημα περιλαμβάνει λειτουργίες όπως η λήψη και διαχείριση παραγγελιών σε πραγματικό χρόνο, αυτόματη ενημέρωση του αποθέματος μετά από κάθε πώληση αν το επιθυμεί ο εκάστοτε καταστηματάρχης και σε οποιοδήποτε προϊόν επιθυμεί αυτός, καθώς και η δημιουργία αναφορών για τα έσοδα και τις πωλήσεις.

Η υλοποίηση βασίστηκε σε τεχνολογίες όπως είναι η Laravel για το server side κώδικα και η Vue.js για το client side κώδικα, με έμφαση στη χρήση φιλικών προς τον χρήστη διεπαφών. Το σύστημα διαθέτει μια δομημένη βάση δεδομένων για την αποθήκευση και επεξεργασία των πληροφοριών.

Κατά τη δοκιμή του συστήματος σε καφετέρια, διαπιστώθηκε η δυνατότητά του να μειώσει σημαντικά τους χρόνους εξυπηρέτησης και να ελαχιστοποιήσει τα λάθη που προκύπτουν από χειροκίνητες διαδικασίες όπως είναι η λάθος καταμέτρηση εσόδων και το να χαθούν παραγγελίες. Επιπλέον, το σύστημα προσφέρει τη δυνατότητα ανάλυσης δεδομένων, παρέχοντας πληροφορίες σχετικά με τα ημερήσια έσοδα, τις παραγγελίες ή ακόμα και το συνολικό ιστορικό των υπαλλήλων.

Η εργασία αυτή αποδεικνύει τη σημασία της τεχνολογίας στη βελτίωση της αποδοτικότητας και την εξυπηρέτηση πελατών, προσφέροντας προοπτικές για την περαιτέρω ανάπτυξη παρόμοιων λύσεων στον τομέα της εστίασης.

«Order Management and Inventory Control System for Hospitality Businesses»

Dimitriou Orestis

Arkhile Damo

Abstract

This diploma thesis focuses on the development of an integrated order management and inventory control system for food service businesses. The objective of the system is to enhance the operations of such businesses by automating the process of receiving and sending orders, reducing errors, and effectively monitoring inventory.

The system includes features such as real-time order management, automatic inventory updates after each sale (if desired by the business owner) for any product of their choice, and the generation of reports on revenues and sales performance.

The implementation was based on technologies such as Laravel for server-side code and Vue.js for client-side code, with an emphasis on user-friendly interfaces. The system is equipped with a well-structured database for storing and processing information.

During testing in a café environment, the system demonstrated its ability to significantly reduce service times and minimize errors caused by manual processes, such as incorrect revenue calculations or lost orders. Additionally, the system provides data analysis capabilities, offering information on daily revenues, orders, or even the complete history of employee activities.

This thesis highlights the importance of technology in improving efficiency and customer service, while also opening up opportunities for the further development of similar solutions in the food service industry.

Ευχαριστίες

Η ολοκλήρωση αυτής της πτυχιακής εργασίας δεν θα ήταν εφικτή χωρίς την πολύτιμη βοήθεια και υποστήριξη πολλών ανθρώπων.

Εκφράζουμε την ειλικρινή μας ευγνωμοσύνη στον επιβλέποντα καθηγητή μας, κύριο Χαράλαμπο Μπράτσα, για την ακαδημαϊκή καθοδήγηση, την υπομονή και την εμπιστοσύνη που μας έδειξε καθ' όλη τη διάρκεια της συγγραφής.

Τέλος ευχαριστούμε θερμά τους γονείς μας, για την αμέριστη αγάπη, την ακλόνητη πίστη και την ηθική υποστήριξη που μας παρείχαν σε κάθε βήμα της ακαδημαϊκής μας πορείας. Η παρουσία τους υπήρξε σταθερή πηγή δύναμης και έμπνευσης.

Περιεχόμενα

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
DBMS	Database Management System
ORM	Object Relational Mapper
MVC	Model-View-Controller
CSRF	Cross-Site Request Forgery
IP	Internet Protocol
SQL	Structured Query Language
HTTP	HyperText Transfer Protocol
DAO	Data Access Objects

1. Το πρόβλημα και τα υφιστάμενα συστήματα

1.1 Εισαγωγή

Η αποτελεσματική διαχείριση των παραγγελιών και της αποθήκης αποτελεί ακρογωνιαίο λίθο για την επιτυχημένη λειτουργία ενός καταστήματος εστίασης. Στο σημερινό ανταγωνιστικό περιβάλλον, η ταχύτητα, η ακρίβεια και η οργάνωση είναι απαραίτητα για την ικανοποίηση των πελατών και την οικονομική ευρωστία της επιχείρησης. Στο κεφάλαιο αυτό θα αναλύσουμε το πρόβλημα της αναποτελεσματικής διαχείρισης παραγγελιών και αποθήκης στον χώρο της εστίασης. Θα παρουσιάσουμε κάποια υφιστάμενα συστήματα εξετάζοντας τα πλεονεκτήματα και τις αδυναμίες τους.

2.1 Το πρόβλημα που θέλουμε να λύσουμε

Στον χώρο της εστίασης, η αποτελεσματική διαχείριση των παραγγελιών και της αποθήκης αποτελεί κρίσιμο παράγοντα για την εύρυθμη λειτουργία ενός καταστήματος. Πολλά καταστήματα βασίζονται ακόμα σε αναχρονιστικές μεθόδους όπως η χειρόγραφη καταγραφή παραγγελιών ή η μη αυτοματοποιημένη διαχείριση αποθεμάτων γεγονός που συχνά οδηγεί σε σφάλματα, καθυστερήσεις και οικονομικές απώλειες. Το σύστημα που αναπτύσσουμε στοχεύει να λύσει αυτά τα προβλήματα προσφέροντας ένα ολοκληρωμένο σύστημα παραγγελιοληψίας για τα καταστήματα εστίασης το οποίο θα επιτρέπει όχι μόνο την ταχεία και ακριβή καταχώριση και διαχείριση παραγγελιών αλλά και τον έλεγχο των διαθέσιμων προϊόντων μέσω της αποθήκης. Μέσα από ένα σύγχρονο περιβάλλον οι υπεύθυνοι του καταστήματος θα έχουν τη δυνατότητα να γνωρίζουν σε πραγματικό χρόνο τα υπόλοιπα των διαθέσιμων προϊόντων αποφεύγοντας ελλείψεις ή υπερβολικές προμήθειες. Επιπλέον το σύστημα προσφέρει τόσο απομακρυσμένη πρόσβαση μέσω διαδικτύου για την εύκολη διαχείριση του καταστήματος και των υπαλλήλων από οπουδήποτε όσο και τοπική πρόσβαση για καταστήματα που επιθυμούν περιορισμό της χρήσης σε συγκεκριμένες συσκευές ή δίκτυα. Με αυτόν τον τρόπο επιτυγχάνεται μεγαλύτερη ευελιξία και ασφάλεια στη διαχείριση των επιχειρησιακών διαδικασιών, καλύπτοντας τις ανάγκες μικρών και μεγάλων καταστημάτων εστίασης.

3.2 Υφιστάμενα συστήματα

3.2.1 Kalypso Restaurant

Το σύστημα ασύρματης παραγγελιοληψίας του Kalypso Restaurant[20] αποτελεί μια ολοκληρωμένη και λύση για την οργάνωση των καθημερινών επιχειρησιακών διαδικασιών σε καταστήματα εστίασης. Με τη χρήση προηγμένων τεχνολογιών το σύστημα επιτρέπει στους υπαλλήλους και τους διαχειριστές να λαμβάνουν, να επεξεργάζονται και να παρακολουθούν τις παραγγελίες σε πραγματικό χρόνο μειώνοντας σημαντικά τα σφάλματα και βελτιστοποιώντας την ταχύτητα εξυπηρέτησης των πελατών. Η ασύρματη λειτουργία του συστήματος προσφέρει υψηλή κινητικότητα και ευελιξία, επιτρέποντας την πρόσβαση σε κρίσιμα δεδομένα όπως τα στατιστικά της ημέρας και οι παραγγελίες μέσω ενός φιλικού και διαδραστικού interface. Επιπλέον το σύστημα ενσωματώνεται εύκολα με άλλες επιχειρησιακές πλατφόρμες όπως συστήματα πληρωμών και διαχείρισης αποθεμάτων παρέχοντας μια ολοκληρωμένη λύση που ανταποκρίνεται στις απαιτήσεις των σύγχρονων καταστημάτων εστίασης. Με τον τρόπο αυτό το Kalypso Restaurant βοηθάει τις επιχειρήσεις να βελτιώσουν την αποδοτικότητά τους, να μειώσουν τα κόστη και να ενισχύσουν την ικανοποίηση των πελατών.

3.2.2 OrderSoft

Το σύστημα ασύρματης παραγγελιοληψίας της OrderSoft[21] προσφέρει μία προηγμένη λύση για την αυτοματοποίηση και βελτιστοποίηση των διαδικασιών παραγγελιοληψίας σε καταστήματα εστίασης. Σχεδιασμένο για να ενισχύσει την αποδοτικότητα και την ακρίβεια το σύστημα επιτρέπει στους σερβιτόρους να καταχωρούν τις παραγγελίες απευθείας από τα τραπέζια μέσω κινητών συσκευών ή ταμπλετών μειώνοντας τις καθυστερήσεις και τα σφάλματα στην παραγγελία. Τα δεδομένα αποστέλλονται σε πραγματικό χρόνο στην κουζίνα και στο ταμείο εξασφαλίζοντας άμεση επεξεργασία και εξυπηρέτηση των πελατών. Το σύστημα δεν ενσωματώνει λειτουργίες για τη διαχείριση του αποθέματος. Επιτρέπει την παρακολούθηση των παραγγελιών και την καταγραφή των πληρωμών παρέχοντας πλήρη εικόνα των καθημερινών εργασιών. Έτσι, οι επιχειρηματίες μπορούν να ελέγχουν την κατάσταση του καταστήματος, να εντοπίζουν τυχόν προβλήματα και να παίρνουν πιο ενημερωμένες αποφάσεις, βελτιώνοντας την εμπειρία του πελάτη και αυξάνοντας την κερδοφορία. Με ευκολία στη χρήση και δυνατότητες προσαρμογής στις ανάγκες κάθε επιχείρησης, το σύστημα της OrderSoft είναι ιδανικό για εστιατόρια, καφετέριες και άλλες επιχειρήσεις εστίασης που επιθυμούν να αναβαθμίσουν την καθημερινή τους λειτουργία.

3.2.3 DionServe

Το λογισμικό παραγγελιοληψίας της DionServe[22] προσφέρει μια προηγμένη και ολοκληρωμένη λύση για την αυτοματοποίηση και την εξυπηρέτηση στους χώρους εστίασης. Οι παραγγελίες μεταφέρονται άμεσα στις αντίστοιχες περιοχές του καταστήματος όπως η κουζίνα ή το bar μειώνοντας τον χρόνο αναμονής και αυξάνοντας την αποδοτικότητα. Το σύστημα είναι πλήρως προσαρμόσιμο με δυνατότητες για παραμετροποίηση ανάλογα με τις ανάγκες του κάθε καταστήματος ενώ η ευχρηστία του και η δυνατότητα υποστήριξης διαφορετικών γλωσσών και τύπων συσκευών το καθιστούν ιδανικό για μικρές και μεγάλες επιχειρήσεις. Μια ακόμη σημαντική δυνατότητα που προσφέρει το λογισμικό της DionServe είναι η ενοποίηση με τα συστήματα πληρωμών επιτρέποντας την άμεση καταβολή των λογαριασμών και την ασφαλή επεξεργασία πληρωμών είτε μέσω κάρτας, είτε μέσω άλλων μεθόδων. Το σύστημα υποστηρίζει επίσης την αναφορά και παρακολούθηση των πωλήσεων προσφέροντας έτσι στους ιδιοκτήτες και τους διαχειριστές πολύτιμα δεδομένα για την απόδοση της επιχείρησης και την καλύτερη λήψη στρατηγικών αποφάσεων αλλά δεν επιτρέπει τη πληροφόρηση σχετικά με το υπόλοιπο των διαθέσιμων προϊόντων στη κάβα του καταστήματος

3.2.4 Twinsoft

Το πρόγραμμα παραγγελιοληψίας της Twinsoft γνωστό ως Orexsys[25] θεωρείται μία από τις πιο αξιόπιστες και προηγμένες λύσεις στην αγορά για επιχειρήσεις εστίασης. Σχεδιασμένο για να καλύπτει τις ανάγκες κάθε τύπου καταστήματος προσφέρει πλήρη αυτοματοποίηση στη διαδικασία παραγγελιοληψίας μειώνοντας τον χρόνο εξυπηρέτησης και τα περιθώρια λάθους. Χάρη στη χρήση σύγχρονων τεχνολογιών επιτρέπει στους σερβιτόρους να λαμβάνουν και να διαχειρίζονται παραγγελίες μέσω φορητών συσκευών όπως smartphones και tablets διασφαλίζοντας ταχύτητα και ακρίβεια στη μετάδοση των δεδομένων.

Ένα από τα μεγαλύτερα πλεονεκτήματα του Orexsys είναι η ευκολία χρήσης του. Το περιβάλλον διαχείρισης είναι απλό και κατανοητό ακόμα και για άτομα που δεν έχουν ιδιαίτερη εξοικείωση με τους υπολογιστές. Η εγκατάστασή του είναι γρήγορη ενώ η διαμόρφωση του συστήματος προσαρμόζεται ανάλογα με τις ανάγκες της επιχείρησης. Παράλληλα η Twinsoft παρέχει ολοκληρωμένες λύσεις τόσο για μεμονωμένα καταστήματα όσο και για αλυσίδες εστίασης διασφαλίζοντας την εύκολη κλιμάκωση του συστήματος.

Το Orexsys ενσωματώνεται άψογα με άλλα συστήματα επιτρέποντας τη διασύνδεση με αποθήκες, λογιστικά προγράμματα και άλλες εφαρμογές που χρησιμοποιούνται στον κλάδο της εστίασης. Αυτή η δυνατότητα κάνει την επιχείρηση πιο ευέλικτη και αποδοτική καθώς οι πληροφορίες διακινούνται αυτόματα μεταξύ των διαφόρων τμημάτων χωρίς να απαιτείται χειροκίνητη ενημέρωση.

Ένα από τα βασικά χαρακτηριστικά του συστήματος είναι η αυτοματοποίηση των διαδικασιών. Όταν ένας πελάτης υποβάλει μια παραγγελία το σύστημα ενημερώνει άμεσα την κουζίνα ή το μπαρ μειώνοντας τις καθυστερήσεις και βελτιώνοντας τη συνολική εμπειρία εξυπηρέτησης.

Ωστόσο παρά τα σημαντικά πλεονεκτήματα του Orexsys υπάρχει και ένα αρνητικό σημείο που πρέπει να ληφθεί υπόψη. Το υψηλό κόστος του συστήματος μπορεί να αποτελέσει εμπόδιο για μικρότερες επιχειρήσεις που επιθυμούν να το υιοθετήσουν. Αν και η επένδυση μπορεί να αποδώσει μακροπρόθεσμα μέσω της αυξημένης αποδοτικότητας ο αρχικός προϋπολογισμός που απαιτείται είναι σημαντικός και μπορεί να αποθαρρύνει ορισμένους επιχειρηματίες.

3.3 Επίλογος

Στο κεφάλαιο αυτό, αναδείξαμε τη σημασία της αποτελεσματικής διαχείρισης παραγγελιών και αποθήκης στον απαιτητικό χώρο της εστίασης όπου η ταχύτητα και η ακρίβεια παίζουν καθοριστικό ρόλο στην ομαλή λειτουργία μιας επιχείρησης. Εξετάσαμε τις προκλήσεις που αντιμετωπίζουν τα καταστήματα εστίασης λόγω της εξάρτησης από παραδοσιακές μεθόδους όπως οι χειρόγραφες παραγγελίες και η μη αυτοματοποιημένη καταγραφή αποθέματος οι οποίες συχνά οδηγούν σε λάθη, καθυστερήσεις και μειωμένη παραγωγικότητα.

Μέσα από την ανάλυση υφιστάμενων συστημάτων παρουσιάσαμε τις δυνατότητες που προσφέρουν οι σύγχρονες τεχνολογίες στον κλάδο της εστίασης. Εξετάσαμε τα πλεονεκτήματα και τις αδυναμίες του κάθε συστήματος τονίζοντας τη σημασία της επιλογής του κατάλληλου εργαλείου ανάλογα με τις ανάγκες της κάθε επιχείρησης. Συνολικά η ενσωμάτωση ενός σύγχρονου συστήματος διαχείρισης παραγγελιών και αποθήκης αποτελεί μονόδρομο για τις επιχειρήσεις που θέλουν να παραμείνουν ανταγωνιστικές και να βελτιώσουν την εμπειρία των πελατών τους.

2. Ανάλυση τεχνολογιών

2.1 Εισαγωγή

Η ανάπτυξη ενός ολοκληρωμένου συστήματος παραγγελιοληψίας και διαχείρισης αποθήκης για καταστήματα εστίασης απαιτεί τη χρήση μιας σειράς τεχνολογιών, τόσο για το front-end όσο και για το back-end. Στο κεφάλαιο αυτό, θα αναλύσουμε τις βασικές τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος, εστιάζοντας στα πλεονεκτήματα και τις δυνατότητες που προσφέρουν. Οι τεχνολογίες που χρησιμοποιήθηκαν αναφορικά είναι η **Laravel**, το **Vue.js 3** με **Vite** και **Pinia**, η **Java** για την Android εφαρμογή, το **VUEXY** ως front-end θέμα, καθώς και η **MySQL** για τη βάση δεδομένων.

2.2 Πληροφορίες για τη Laravel

Η Laravel είναι ένα από τα πιο δημοφιλή PHP frameworks, το οποίο χρησιμοποιείται για την ανάπτυξη web εφαρμογών. Προσφέρει έναν εξαιρετικά ευέλικτο και οργανωμένο τρόπο για τη δημιουργία εφαρμογών. Χρησιμοποιήθηκε για την ανάπτυξη του server-side κώδικα, με ιδιαίτερη έμφαση στην ασφάλεια, την αποτελεσματικότητα και την ευχρηστία. Με δυνατότητες όπως το Eloquent ORM (Object-Relational Mapping) για διαχείριση βάσεων δεδομένων, το routing και την υποστήριξη για RESTful APIs, η Laravel επιτρέπει την ταχεία ανάπτυξη εφαρμογών με υψηλή απόδοση και ασφάλεια.

2.2.1 Αυθεντικοποίηση στη Laravel

Για να εισέλθει ο χρήστης στην εφαρμογή αρχικά πρέπει να εισάγει το email και τον κωδικό πρόσβασης του. Στη συνέχεια το Auth::validate($credentials) χρησιμοποιείται για να επαληθεύσει ότι τα διαπιστευτήρια του χρήστη είναι σωστά και αντιστοιχούν με αυτά που είναι αποθηκευμένα στη βάση δεδομένων. Αν τα στοιχεία είναι έγκυρα τότε καλείται η μέθοδος Auth::attempt($credentials) [1] για να πραγματοποιηθεί η αυθεντικοποίηση του χρήστη και το session του χρήστη αποθηκεύεται.[2]

Για κάθε API αίτημα που αποστέλλεται από τον χρήστη προκειμένου να προσδιορίσουμε ποιος είναι ο αποστολέας του αιτήματος αλλά και να ελέγξουμε αν είναι συνδεδεμένος χρησιμοποιούμε το Sanctum[3]. Το Sanctum είναι μια βιβλιοθήκη της Laravel που επιτρέπει την προστασία των routes περιορίζοντας την πρόσβαση σε αυτές μόνο σε αυθεντικοποιημένους χρήστες. Η προστασία επιτυγχάνεται μέσω ενός token που δημιουργείται για τον χρήστη κατά τη διαδικασία σύνδεσης και αποθηκεύεται ως cookie στο localStorage του browser.

Κάθε φορά που ο χρήστης στέλνει ένα αίτημα στο API, το token αποστέλλεται στο header του αιτήματος ως εξής: Authorization: Bearer <token>. Όταν το αίτημα φτάσει στον server, γίνεται έλεγχος της εγκυρότητας του token μέσω του middleware auth:sanctum. Αν το token είναι έγκυρο, ο χρήστης καθίσταται διαθέσιμος μέσω της μεθόδου Auth::user(). Σε αντίθετη περίπτωση, ο server επιστρέφει μία 401 Unauthorized απάντηση.

Όταν ο χρήστης αποσυνδεθεί το token που χρησιμοποιούσε καθίσταται άκυρο και απομακρύνεται από το localStorage. Έτσι δεν μπορεί να χρησιμοποιηθεί για νέες αιτήσεις. Σε περίπτωση που ο χρήστης συνδεθεί εκ νέου δημιουργείται ένα νέο token το οποίο χρησιμοποιείται για τις επόμενες αυθεντικοποιημένες λειτουργίες της εφαρμογής.

2.2.2 Μοντέλα και Eloquent

Στη Laravel, τα μοντέλα είναι κλάσεις PHP που αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων. Το Eloquent ORM (Object-Relational Mapper)[4] είναι μια ενσωματωμένη λειτουργία της Laravel που διευκολύνει την αλληλεπίδραση με τη βάση δεδομένων. Με το Eloquent, μπορούμε να εκτελούμε ερωτήματα SQL με αντικειμενοστραφή τρόπο, χρησιμοποιώντας μεθόδους αντί για απλές εντολές SQL. Αυτό κάνει τον κώδικα πιο ευανάγνωστο και ευκολότερο στη συντήρηση. Για παράδειγμα, αντί να γράψουμε μια εντολή SQL όπως SELECT * FROM users WHERE id = 1, μπορούμε να χρησιμοποιήσουμε το Eloquent ως εξής: User::find(1). Αυτό είναι πολύ πιο απλό και ευανάγνωστο.

Επιπλέον, το Eloquent προσφέρει και άλλα πλεονεκτήματα σε σχέση με την απλή PHP και SQL, όπως:

- **Απλότητα και Ευχρηστία:** Η Laravel δημιουργεί αυτόματα τις κατάλληλες SQL εντολές βάσει των μεθόδων που καλούνται από το μοντέλο. Για παράδειγμα, η μέθοδος User::all() θα αναζητήσει όλους τους χρήστες από τον πίνακα users χωρίς να χρειάζεται να γραφτεί κανένα SQL query. Έτσι αποφεύγεται η ανάγκη γνώσης SQL για απλές χρήσεις.
- **Αυτόματη Αντιστοίχιση και Σχέσεις (Relations):** Το Eloquent επιτρέπει την εύκολη διαχείριση των σχέσεων μεταξύ των μοντέλων, όπως η σχέση one-to-many, many-to-many κ.λπ. Η Laravel παρέχει εύχρηστες μεθόδους για την αποθήκευση και ανάκτηση σχετικών δεδομένων μέσω των σχέσεων αυτών χωρίς την ανάγκη των join.
- **Λειτουργίες Σύνθετων Ερωτημάτων:** Το Eloquent παρέχει μεθόδους για την εκτέλεση σύνθετων ερωτημάτων με τρόπο που να είναι εύκολος στην κατανόηση. Οι μέθοδοι όπως where(), orderBy() και groupBy() μπορούν να χρησιμοποιηθούν για να δημιουργηθούν πολύπλοκα ερωτήματα απλά και κατανοητά.
- **Αυξημένη ασφάλεια:** Το Eloquent προστατεύει από επιθέσεις SQL injection καθώς τα δεδομένα που εισάγονται σε ερωτήματα περνάνε μέσω του Eloquent με ασφαλή τρόπο..
- **Καλύτερη οργάνωση κώδικα:** Με το Eloquent, οι προγραμματιστές μπορούν να δουλέψουν με δεδομένα ως αντικείμενα PHP, γεγονός που καθιστά τον κώδικα πιο κατανοητό και ευκολότερο στη συντήρηση. Αντί να πρέπει να χειριστούν raw SQL queries και πίνακες, χρησιμοποιούν απλές μεθόδους που εξασφαλίζουν ότι ο κώδικας είναι καθαρός και αναγνώσιμος.

2.2.3 Gates και AuthServiceProvider

Τα Gates στην Laravel είναι μηχανισμοί που επιτρέπουν τον έλεγχο της πρόσβασης σε διάφορες λειτουργίες της εφαρμογής, με βάση τα δικαιώματα του χρήστη. Τα Gates ορίζουν αν ο χρήστης επιτρέπεται να εκτελέσει μια συγκεκριμένη ενέργεια στην εφαρμογή, όπως για παράδειγμα να δημιουργήσει μια παραγγελία ή να διαγράψει έναν πόρο. Κάθε gate μπορεί να είναι συγκεκριμένο για κάποιο δικαίωμα ή ενέργεια, και η Laravel σας επιτρέπει να τα ορίσετε στον AuthServiceProvider.

Ο AuthServiceProvider[5] είναι υπεύθυνος για την εγγραφή των gates και τη ρύθμιση των μηχανισμών αυθεντικοποίησης και εξουσιοδότησης στην εφαρμογή. Μέσα από το AuthServiceProvider, μπορούμε να καταχωρήσουμε τα gates και να καθορίσουμε τη λογική αυτών, ορίζοντας ποιοι χρήστες έχουν δικαιώματα να εκτελέσουν κάποιες ενέργειες.

Η μέθοδος this->authorize() χρησιμοποιείται σε controllers για τον έλεγχο αν ο χρήστης έχει τα απαραίτητα δικαιώματα (permissions) για να εκτελέσει μια συγκεκριμένη ενέργεια. Όταν καλείται η μέθοδος authorize() η Laravel αναζητά το αντίστοιχο gate στον AuthServiceProvider με την παράμετρο που της δίνεται. Για παράδειγμα, όταν καλούμε την εντολή: $this->authorize('create-order') Η Laravel ψάχνει μέσα στο AuthServiceProvider για ένα gate που έχει καταχωρηθεί με το όνομα 'create-order'. Αν

το gate αυτό υπάρχει, τότε εκτελείται η λογική που έχουμε ορίσει σε αυτό το gate. Αν δεν υπάρχει ή αν ο χρήστης δεν έχει το απαιτούμενο δικαίωμα, επιστρέφεται μια 403 Unauthorized απάντηση.

Στην εφαρμογή μας, χρησιμοποιούμε τη μέθοδο hasPermissionTo() η οποία ελέγχει αν ο χρήστης (μέσω του ρόλου του) έχει το αντίστοιχο δικαίωμα για την ενέργεια που προσπαθεί να εκτελέσει. Αυτός ο έλεγχος βασίζεται στο πίνακα permissions και τον pivot πίνακα permission_role. Ο pivot πίνακας χρησιμοποιείται για να συνδέσει τα permissions με τα roles και να καθορίσει ποιοι ρόλοι έχουν ποια permissions.

Πιο συγκεκριμένα, η μέθοδος hasPermissionTo() ελέγχει αν ο ρόλος του χρήστη έχει τη δυνατότητα να εκτελέσει μια συγκεκριμένη ενέργεια (π.χ., να δημιουργήσει μια παραγγελία). Για παράδειγμα αν ο χρήστης έχει το δικαίωμα create-order τότε η μέθοδος hasPermissionTo() θα επιστρέψει true επιτρέποντας την εκτέλεση της ενέργειας.

2.2.4 Queue, Job και Workers

Η έννοια του work και queue στη Laravel[6] αφορά την εκτέλεση εργασιών (jobs) στο background χωρίς να μπλοκάρουν την κανονική ροή της εφαρμογής. Το Laravel παρέχει έναν ολοκληρωμένο μηχανισμό για την αποθήκευση και εκτέλεση εργασιών που πρέπει να ολοκληρωθούν εκτός από την κύρια διαδικασία του αιτήματος HTTP

- **Queue**: Είναι η σειρά με την οποία διατηρούνται τα jobs που πρέπει να εκτελούνται στο background. Όταν μια εργασία ή job προστίθεται στην queue η εκτέλεσή της αναβάλλεται και θα εκτελεστεί αργότερα από έναν worker.
- **Work**: Είναι η διαδικασία που αναλαμβάνει να εκτελέσει τα jobs που είναι στην queue. Οι workers είναι διεργασίες (ή επιμέρους scripts) που τρέχουν συνεχώς και εξετάζουν την queue για να εκτελέσουν τα jobs.

Η λογική πίσω από αυτό είναι ότι οι χρονοβόρες εργασίες (π.χ. αποστολή email) μπορεί να επηρεάσουν την ταχύτητα απόκρισης του χρήστη αν εκτελούνται αμέσως κατά την διάρκεια του HTTP request. Αντί να εκτελούνται στην ίδια διαδικασία με το αίτημα του χρήστη οι εργασίες προστίθενται σε μια queue και αναλαμβάνονται αργότερα από έναν worker. Με αυτόν τον τρόπο η εφαρμογή παραμένει γρήγορη και ο χρήστης δεν περιμένει για τις χρονοβόρες διαδικασίες. Στην εφαρμογή μας job χρησιμοποιούμε σε δύο περιπτώσεις. Στην αποστολή ενημερωτικού email και αυτόματη αφαίρεση ποσότητας από ένα προϊόν.

Επίσης τα job τα έχουμε ρυθμισμένα με τη μεταβλητή περιβάλλοντος QUEUE_CONNECTION=database[7], αυτό σημαίνει ότι τα jobs θα αποθηκεύονται στη βάση δεδομένων. Άρα όταν ένα job προστίθεται στην queue η Laravel το καταχωρεί στον πίνακα της βάσης δεδομένων. Έτσι workers συνδέονται με τη βάση δεδομένων και αναζητούν τα jobs που περιμένουν για εκτέλεση στον πίνακα. Αν βρουν jobs τα εκτελούν και τα διαγράφουν από την ουρά μόλις ολοκληρωθεί η εκτέλεση τους.

Η χρήση της βάσης δεδομένων για την αποθήκευση των jobs έχει τα εξής πλεονεκτήματα:

- Είναι απλή στην εγκατάσταση, καθώς δεν απαιτεί πρόσθετους εξυπηρετητές ή υπηρεσίες
- Παρέχει ανθεκτικότητα, καθώς τα jobs αποθηκεύονται και μπορούν να ανακτηθούν ακόμα και αν το σύστημα σταματήσει ή επανεκκινηθεί.

Ωστόσο, έχει και μειονεκτήματα:

- Είναι πιο αργό σε σχέση με άλλες τεχνολογίες, όπως η Redis λόγω της ανάγκης για επικοινωνία με τη βάση δεδομένων.

- Υπάρχει πιθανότητα περιορισμών στην απόδοση όταν ο όγκος των jobs είναι μεγάλος.

1.2.5 Laravel Controllers

Στη Laravel οι controllers[8] αποτελούν θεμελιώδες στοιχείο της αρχιτεκτονικής MVC (Model-View-Controller) και διαχειρίζονται τη ροή δεδομένων μεταξύ των μοντέλων (Model) και των προβολών (View). Λειτουργούν ως διαμεσολαβητές που λαμβάνουν τα HTTP αιτήματα των χρηστών, επεξεργάζονται τη λογική της εφαρμογής, αλληλεπιδρούν με τη βάση δεδομένων μέσω των μοντέλων και επιστρέφουν τις κατάλληλες απαντήσεις στο front-end είτε μέσω views είτε σε μορφή JSON ανάλογα με το είδος της εφαρμογής.

Οι κύριες λειτουργίες των controllers περιλαμβάνουν:

- **Διαχείριση των αιτημάτων**: Κάθε controller περιέχει μεθόδους που αντιστοιχούν σε διαφορετικά routes ή ενέργειες. Για παράδειγμα μια μέθοδος store() μπορεί να αποθηκεύει δεδομένα στη βάση ενώ μια index() επιστρέφει μια λίστα δεδομένων.
- **Οργάνωση Κώδικα**: Αντί να γράφουμε όλο τον κώδικα για την επεξεργασία ενός αιτήματος μέσα σε routes ή views, οι controllers προσφέρουν έναν δομημένο τρόπο για να οργανώνουμε τη λογική της εφαρμογής.
- **Αλληλεπίδραση με Μοντέλα**: Οι controllers είναι υπεύθυνοι για την επικοινωνία με τα μοντέλα (models) και την εκτέλεση λειτουργιών όπως η αποθήκευση ανάκτηση ή ενημέρωση δεδομένων.
- **Επιστροφή Απαντήσεων**: Οι controllers επιστρέφουν τις απαντήσεις στον χρήστη, είτε μέσω views (HTML), είτε μέσω JSON για API, είτε μέσω redirects.

Σε τι βοηθούν οι Controllers;

- **Δομημένη Οργάνωση**: Οι controllers βοηθούν στη δημιουργία ενός καθαρού και δομημένου κώδικα διαχωρίζοντας τις διαφορετικές λειτουργίες της εφαρμογής.
- **Επαναχρησιμοποίηση Κώδικα**: Με τη χρήση μεθόδων στους controllers μπορούμε να επαναχρησιμοποιήσουμε κοινές λειτουργίες μειώνοντας τον πλεονασμό κώδικα.
- **Ευκολία Συντήρησης**: Ο διαχωρισμός της λογικής στους controllers καθιστά ευκολότερη τη συντήρηση και την επέκταση του κώδικα.

Στην απλή PHP η λογική επεξεργασίας αιτημάτων και η εμφάνιση δεδομένων βρίσκονται συχνά στο ίδιο αρχείο. Για παράδειγμα ένα αρχείο index.php μπορεί να περιέχει τόσο HTML όσο και PHP κώδικα κάτι που καθιστά τον κώδικα δύσκολο στη συντήρηση και την κατανόηση ειδικά σε μεγαλύτερες εφαρμογές.

Στη Laravel οι controllers διαχωρίζουν τη λογική από την εμφάνιση. Ο κώδικας που διαχειρίζεται τα αιτήματα βρίσκεται στους controllers ενώ η εμφάνιση δεδομένων γίνεται μέσω των views. Αυτός ο διαχωρισμός καθιστά τον κώδικα πιο οργανωμένο κατανοητό και ευκολότερο στη διαχείριση. Οι controllers είναι βασικό μέρος του μοντέλου MVC (Model-View-Controller) που ακολουθεί η Laravel. Οι controllers συνδέουν το model (δεδομένα) με το view (παρουσίαση). Αυτή η αρχιτεκτονική προσφέρει σαφήνεια και διευκολύνει τη συντήρηση του κώδικα. Αντίθετα, στην απλή PHP, δεν υπάρχει

ενσωματωμένο πλαίσιο για το MVC. Ο προγραμματιστής πρέπει να οργανώσει τον κώδικα μόνος του, κάτι που συχνά οδηγεί σε χαοτική και μη δομημένη ανάπτυξη.

2.2.6 Laravel Routing

Το Routing είναι η διαδικασία καθορισμού των διαδρομών (routes) που χειρίζονται τα αιτήματα HTTP (GET, POST, κ.λπ.) προς την εφαρμογή. Στη Laravel το routing καθορίζει πώς η εφαρμογή θα ανταποκριθεί σε συγκεκριμένα αιτήματα διευθύνσεων URL κατευθύνοντας τα αιτήματα σε controllers, views ή κώδικα που εκτελείται απευθείας μέσα στα routes. Το Routing βασίζεται σε ένα αρχείο ρυθμίσεων που βρίσκεται στον φάκελο routes (web.php για routes που εμφανίζουν HTML ή api.php για API routes).

Κάθε route συνδέεται με μία ή περισσότερες HTTP μεθόδους όπως GET, POST, PUT, ή DELETE οι οποίες καθορίζουν τον τύπο του αιτήματος που μπορεί να επεξεργαστεί. Επιπλέον το route καθορίζει την ενέργεια που θα εκτελεστεί όταν ο χρήστης επισκεφθεί τη συγκεκριμένη διαδρομή όπως η επιστροφή ενός view ή η εκτέλεση μιας μεθόδου σε έναν controller. Τα routes μπορούν επίσης να προστατευθούν ή να περιοριστούν μέσω middleware το οποίο εκτελεί ενέργειες πριν από την επεξεργασία του αιτήματος όπως ο έλεγχος αυθεντικοποίησης ή η επικύρωση δικαιωμάτων πρόσβασης.

Το Routing χωρίζεται σε διάφορες κατηγορίες:

- **Basic Routing:** Χρήση απλών routes για να κατευθύνουμε αιτήματα σε closures ή views. Route::get('/', function () { return view('welcome'); });
- **Route Parameters:** Δυνατότητα να περάσουμε δυναμικές τιμές μέσω των URL. Route::post('/settings/role/delete/{role}', [Api\RoleController::class, 'delete']);
- **Controller Routes:** Σύνδεση routes με συγκεκριμένες μεθόδους controllers. Route::get('/orders/fetch', [Api\OrderController::class, 'fetch']);
- **API Routing:** Ειδικά routes για API endpoints (συνήθως βρίσκονται στο routes/api.php
- **Resource Routing:** Αυτόματος καθορισμός routes για CRUD (Create, Read, Update, Delete) λειτουργίες. Route::resource('products', ProductController::class);
- **Route Groups:** Ομαδοποίηση routes με κοινά χαρακτηριστικά όπως middleware, namespace, prefix ή domain. Route::middleware('auth:sanctum')->group(function () {
- **Middleware:** Προσθήκη middleware για να προστατεύσουμε ή να φιλτράρουμε τα requests. Route::middleware('auth:sanctum')->group(function () {
- **Redirects και Fallback Routes:** Αυτόματη ανακατεύθυνση ή διαχείριση μη υπαρκτών routes. Route::redirect('/old-route', '/new-route'); Route::fallback(function () { return 'Page not found!'; });

2.2.7 Middlewear

Τα middleware[10] στη Laravel αποτελούν ένα ενδιάμεσο επίπεδο μεταξύ του αιτήματος και της απόκρισης προσφέροντας τη δυνατότητα να εκτελέσουμε κώδικα πριν ή μετά την επεξεργασία του αιτήματος από τον controller. Είναι εξαιρετικά χρήσιμα για την εφαρμογή λογικής που πρέπει να ισχύει σε πολλά σημεία της εφαρμογής όπως ο έλεγχος αυθεντικοποίησης όπου εξασφαλίζουν ότι ο χρήστης είναι συνδεδεμένος πριν αποκτήσει πρόσβαση σε συγκεκριμένες διαδρομές ή λειτουργίες της εφαρμογής με χαρακτηριστικό παράδειγμα το middleware auth το οποίο ελέγχει αν ο χρήστης είναι αυθεντικοποιημένος. Επιπλέον επικυρώνουν αν ο χρήστης έχει τα απαραίτητα δικαιώματα για να εκτελέσει μια συγκεκριμένη ενέργεια ή να αποκτήσει πρόσβαση σε έναν πόρο, επιτρέπουν την

επικύρωση δεδομένων που αποστέλλονται από τον client (όπως η επαλήθευση ότι όλα τα πεδία είναι σωστά συμπληρωμένα), παρέχουν προστασία από επιθέσεις cross-site request forgery (CSRF) διασφαλίζοντας ότι τα αιτήματα προέρχονται από την εφαρμογή και όχι από κακόβουλες πηγές, καταγράφουν τα αιτήματα και τις αποκρίσεις για σκοπούς εντοπισμού σφαλμάτων ή παρακολούθησης και περιορίζουν τον αριθμό των αιτημάτων που μπορεί να κάνει ένας χρήστης ή ένα IP μέσα σε συγκεκριμένο χρονικό διάστημα (rate limiting).

Η Laravel παρέχει έτοιμα middleware όπως τα auth, verified και throttle αλλά επιτρέπει και τη δημιουργία προσαρμοσμένων middleware για την κάλυψη των αναγκών της εκάστοτε εφαρμογής. Τα middleware καταχωρούνται στον πυρήνα της εφαρμογής (App\Http\Kernel) και μπορούν να εφαρμοστούν σε συγκεκριμένα routes, ομάδες route ή ακόμη και global για ολόκληρη την εφαρμογή. Με αυτόν τον τρόπο διευκολύνουν τη συντήρηση και την επεκτασιμότητα του κώδικα διαχωρίζοντας τη λογική που εφαρμόζεται σε διάφορα επίπεδα της εφαρμογής.

2.2.8 Database Migrations

Τα migrations στη Laravel είναι ένα εργαλείο διαχείρισης της δομής της βάσης δεδομένων που επιτρέπει την εύκολη δημιουργία, τροποποίηση και διαγραφή πινάκων και πεδίων της βάσης με έναν προγραμματιστικό και ελεγχόμενο τρόπο. Αποτελούν μια σημαντική πτυχή της ανάπτυξης καθώς προσφέρουν μια οργανωμένη και ασφαλή μέθοδο για τη διαχείριση αλλαγών στη βάση δεδομένων χωρίς να απαιτείται χειροκίνητη επέμβαση. Ένα migration είναι μια κλάση PHP που περιέχει δύο βασικές μεθόδους: την up που ορίζει τις αλλαγές που πρέπει να γίνουν στη βάση (όπως η δημιουργία ενός πίνακα ή η προσθήκη πεδίων) και την down που καθορίζει πώς να αναιρεθούν αυτές οι αλλαγές, εάν χρειαστεί να γίνει αναίρεση (rollback). Τα migrations αποθηκεύονται στον φάκελο database/migrations και εκτελούνται με τη χρήση της εντολής php artisan migrate.

Η χρησιμότητα των migrations έγκειται στη δυνατότητα που προσφέρουν για την παρακολούθηση και τον έλεγχο όλων των αλλαγών που πραγματοποιούνται στη δομή της βάσης δεδομένων. Επιπλέον, οι αλλαγές είναι αναστρέψιμες, επιτρέποντας την επιστροφή σε προηγούμενη κατάσταση της βάσης με την εντολή php artisan migrate:rollback.

1.2.9 Database Seeders

Οι seeders στη Laravel είναι εργαλεία που χρησιμοποιούνται για την εισαγωγή προκαθορισμένων δεδομένων στη βάση δεδομένων. Επιτρέπουν την εύκολη δημιουργία και διαχείριση δεδομένων που χρειάζονται για τη λειτουργία και τη δοκιμή της εφαρμογής. Τα seeders βρίσκονται στον φάκελο database/seeders και ο κάθε seeder είναι μια κλάση που περιέχει τη μέθοδο run όπου ορίζεται η λογική για την εισαγωγή δεδομένων στη βάση.

Η βασική χρησιμότητα των seeders είναι να αυτοματοποιούν την εισαγωγή δεδομένων εξοικονομώντας χρόνο κατά τη διαδικασία ανάπτυξης. Για παράδειγμα αντί να εισάγονται χειροκίνητα δεδομένα στη βάση οι seeders μπορούν να δημιουργούν δείγματα χρηστών, ρόλων, προϊόντων ή οποιωνδήποτε άλλων δεδομένων απαιτούνται για να δοκιμαστούν οι λειτουργίες της εφαρμογής. Αυτό είναι ιδιαίτερα χρήσιμο όταν η βάση δεδομένων πρέπει να ανανεώνεται συχνά κατά τη διάρκεια της ανάπτυξης ή των δοκιμών.

Οι seeders μπορούν επίσης να χρησιμοποιηθούν για την εισαγωγή βασικών δεδομένων που απαιτούνται για την εκτέλεση της εφαρμογής σε παραγωγή, όπως προεπιλεγμένοι ρόλοι χρηστών, ρυθμίσεις συστήματος ή άλλες στατικές πληροφορίες. Επιπλέον η Laravel προσφέρει τη δυνατότητα χρήσης

εργαλείων όπως το Faker μέσα στους seeders για τη δημιουργία ψεύτικων δεδομένων που βοηθούν στις δοκιμές. Η εκτέλεση των seeders γίνεται με την εντολή php artisan db:seed ή php artisan migrate --seed, η οποία εκτελεί τους seeders μαζί με τα migration. Με αυτόν τον τρόπο, οι seeders συμβάλλουν στη δημιουργία ενός συνεκτικού και επαναλαμβανόμενου περιβάλλοντος ανάπτυξης, ενισχύοντας τη συνεργασία μεταξύ των προγραμματιστών και εξασφαλίζοντας ότι όλοι εργάζονται με τα ίδια δεδομένα στη βάση.

2.2.10 Observers

Ο Observer στη Laravel επιτρέπει την αυτόματη εκτέλεση συγκεκριμένων ενεργειών όταν συμβαίνουν αλλαγές σε ένα μοντέλο. Αντί να τοποθετούμε τον κώδικα διαχείρισης συμβάντων (event handling) μέσα στα ίδια τα μοντέλα ή τους controllers μπορούμε να χρησιμοποιήσουμε έναν Observer για να διαχειριστούμε γεγονότα όπως η δημιουργία, ενημέρωση ή διαγραφή μιας εγγραφής. Η χρήση των Observers βοηθά στη βελτίωση της συνοχής του κώδικα, καθώς διαχωρίζει την επιχειρηματική λογική (business logic) από τη βασική λειτουργικότητα του μοντέλου, καθιστώντας τον κώδικα πιο οργανωμένο, επεκτάσιμο και ευκολότερο στη συντήρηση.

Η χρήση των Observers στη Laravel είναι ιδιαίτερα χρήσιμη σε σενάρια όπου θέλουμε να εκτελούμε συγκεκριμένες λειτουργίες όταν μια εγγραφή δημιουργείται, ενημερώνεται ή διαγράφεται. Ο Observer βοηθά στις παρακάτω περιπτώσεις:

- **Διατήρηση Καθαρού και Οργανωμένου Κώδικα:** Χωρίζει τη λογική από τα μοντέλα και τους controllers διατηρώντας τον κώδικα πιο ευανάγνωστο και διαχειρίσιμο. Αντί να προσθέτουμε κώδικα απευθείας μέσα στο μοντέλο μπορούμε να τον μεταφέρουμε στον Observer.
- **Αυτόματη Εκτέλεση Συγκεκριμένων Ενεργειών:**Μπορεί να χρησιμοποιηθεί για να εκτελεί ενέργειες όπως η καταγραφή αλλαγών στο ιστορικό (logging), η αποστολή ειδοποιήσεων ή email ή και η δημιουργία ή ενημέρωση σχετικών εγγραφών σε άλλους πίνακες.
- **Βελτίωση της Επεκτασιμότητας:** Αν κάποια στιγμή χρειαστεί να αλλάξουμε τη συμπεριφορά ενός event (π.χ., να προσθέσουμε ένα νέο λογισμικό καταγραφής) μπορούμε να το κάνουμε χωρίς να τροποποιήσουμε το ίδιο το μοντέλο.
- **Αποφυγή Επαναλαμβανόμενου Κώδικα (Code Duplication):** Αντί να γράφουμε τον ίδιο κώδικα μέσα σε διαφορετικά μέρη της εφαρμογής μπορούμε να τον συγκεντρώσουμε στον Observer και να τον εκτελούμε αυτόματα όταν χρειάζεται.

Ο Observer παρακολουθεί ένα Eloquent Model και μπορεί να εκτελέσει συγκεκριμένες ενέργειες όταν συμβαίνει κάποιο από τα διαθέσιμα events του Eloquent. Τα πιο συνηθισμένα γεγονότα που υποστηρίζει είναι: created(), updated(), deleted(), restored() και forceDeleted().

Για να λειτουργήσει ο Observer, πρέπει να τον συνδέσουμε με το αντίστοιχο μοντέλο. Αυτό γίνεται μέσα στο AppServiceProvider.php: Ανοίγουμε το αρχείο AppServiceProvider.php και μέσα στη μέθοδο boot(), προσθέτουμε: User::observe(UserObserver::class); Με αυτόν τον τρόπο, κάθε φορά που θα συμβαίνει κάποιο event στο μοντέλο User, ο Laravel θα καλεί αυτόματα τον UserObserver.

2.2.11 Cache

Στη Laravel, η προσωρινή αποθήκευση (cache) είναι ένα ισχυρό εργαλείο που βελτιώνει την απόδοση της εφαρμογής αποθηκεύοντας δεδομένα που χρησιμοποιούνται συχνά ώστε να αποφεύγονται περιττές επεξεργασίες και ερωτήματα στη βάση δεδομένων. Ο ορισμός του CACHE_DRIVER=file στο αρχείο

περιβάλλοντος .env σημαίνει ότι η Laravel αποθηκεύει την προσωρινή μνήμη σε αρχεία μέσα στον φάκελο storage/framework/cache. Αυτή η επιλογή είναι ιδανική για μικρές και μεσαίου μεγέθους εφαρμογές όπου δεν απαιτείται υψηλή απόδοση ή διαμοιρασμένη προσωρινή αποθήκευση όπως σε περιβάλλοντα ανάπτυξης ή εφαρμογές με περιορισμένους πόρους. Ο file-based cache driver λειτουργεί αποθηκεύοντας κάθε στοιχείο της προσωρινής μνήμης ως ένα ξεχωριστό αρχείο JSON, το οποίο ανακτάται γρήγορα χωρίς να χρειάζεται αναζήτηση στη βάση δεδομένων. Παρόλο που είναι εύκολος στην υλοποίηση και δεν απαιτεί πρόσθετες ρυθμίσεις έχει μειονεκτήματα σε εφαρμογές υψηλής επισκεψιμότητας καθώς η διαχείριση μεγάλου αριθμού αρχείων μπορεί να οδηγήσει σε αυξημένη καθυστέρηση λόγω της ανάγκης πρόσβασης στο σύστημα αρχείων. Για καλύτερη απόδοση σε παραγωγικό περιβάλλον προτείνεται η χρήση ταχύτερων cache drivers όπως η redis η οποία λειτουργεί στη μνήμη και επιτρέπει την ταχύτερη ανάκτηση δεδομένων. Ωστόσο, για εφαρμογές που δεν απαιτούν πολύ υψηλή απόδοση η χρήση του CACHE_DRIVER=file παραμένει μια απλή και αξιόπιστη λύση εξασφαλίζοντας ότι η εφαρμογή αποφεύγει περιττές ερωτήσεις στη βάση δεδομένων και λειτουργεί αποδοτικότερα.

Στην εφαρμογή μας, χρησιμοποιούμε την cache για τη βελτιστοποίηση των ερωτημάτων στη βάση δεδομένων και τη μείωση του φόρτου του συστήματος. Η cache μας επιτρέπει να αποθηκεύουμε δεδομένα που δεν αλλάζουν συχνά, ώστε να μπορούμε να τα ανακτούμε πιο γρήγορα χωρίς να εκτελούμε συνεχείς ερωτήσεις στη βάση δεδομένων.

Στο InitializationResource, εφαρμόζουμε caching για διάφορες οντότητες όπως κατηγορίες προϊόντων, υποκατηγορίες, προϊόντα, χαρακτηριστικά, θέσεις τραπεζιών και ρόλους χρηστών. Χρησιμοποιούμε την Cache::rememberForever() η οποία αποθηκεύει τα δεδομένα επ’ αόριστον μέχρι να διαγραφούν χειροκίνητα ή να γίνει εκκαθάριση της cache. Αυτό διασφαλίζει ότι οι πιο συχνά χρησιμοποιούμενες πληροφορίες είναι άμεσα διαθέσιμες και δεν χρειάζεται να εκτελούνται περιττά queries στη βάση.

Για παράδειγμα, όταν γίνεται ένα αίτημα για αρχικοποίηση (InitializationResource) ελέγχουμε αν υπάρχουν ήδη τα δεδομένα στην cache:

- **Αν υπάρχουν:** επιστρέφονται άμεσα από την προσωρινή μνήμη αποφεύγοντας περιττές ερωτήσεις στη βάση.
- **Αν δεν υπάρχουν:** γίνεται ανάκτηση των δεδομένων από τη βάση δεδομένων και η αποθήκευσή τους στην cache για μελλοντική χρήση.

Χρησιμοποιούμε διαφορετικά cache keys (categories, subcategories, products, κ.λπ.), ώστε κάθε σύνολο δεδομένων να μπορεί να ανακτηθεί ανεξάρτητα. Η μέθοδος rememberForever() αποθηκεύει τα δεδομένα στη μνήμη για απεριόριστο χρονικό διάστημα, κάτι που είναι ιδιαίτερα χρήσιμο για στατικά δεδομένα που αλλάζουν σπάνια.

2.2.12 Observers και Cache

Στην εφαρμογή μας, έχουμε ενσωματώσει τον μηχανισμό των **Observers** σε συνδυασμό με τη **Cache** για να εξασφαλίσουμε τη βέλτιστη απόδοση και την αποτελεσματική διαχείριση των δεδομένων. Η χρήση αυτών των μηχανισμών επιτρέπει τη συνεχιζόμενη παρακολούθηση των μοντέλων μας, διασφαλίζοντας την άμεση ανανέωση των δεδομένων χωρίς να επηρεάζουμε την ταχύτητα και τη σταθερότητα του συστήματος.

Η διαδικασία που ακολουθούμε περιλαμβάνει τη χρήση των Observers για να παρακολουθούν τις σημαντικές ενέργειες που γίνονται στα μοντέλα όπως η δημιουργία, η ενημέρωση, η διαγραφή και η

επαναφορά των εγγραφών. Κάθε φορά που συμβαίνει κάποια από αυτές τις ενέργειες οι αντίστοιχοι Observers αναλαμβάνουν να καθαρίσουν την Cache ώστε να εξασφαλίζεται ότι η εφαρμογή έχει πάντα τα πιο πρόσφατα δεδομένα.

Αυτή η προσέγγιση μειώνει την ανάγκη για συνεχείς ερωτήματα στη βάση δεδομένων αφού η Cache παρέχει τα δεδομένα γρήγορα και χωρίς καθυστερήσεις. Ωστόσο για να παραμείνει η Cache επικαιροποιημένη οι Observers διασφαλίζουν ότι τα δεδομένα στη βάση και στην cache είναι συγχρονισμένα.

2.3 Πληροφορίες για τη Vue.js

Το Vue.js[11] είναι ένα JavaScript framework που χρησιμοποιείται για τη δημιουργία user interfaces. Είναι σχεδιασμένο για να είναι προσαρμόσιμο και μπορεί να χρησιμοποιηθεί τόσο για απλές όσο και για πολύπλοκες εφαρμογές . Το Vue.js εστιάζει στην view layer και μπορεί εύκολα να ενσωματωθεί σε υπάρχοντα projects. Επιτρέπει στους developers να δημιουργούν διαδραστικές και αντιδραστικές web εφαρμογές αξιοποιώντας declarative rendering, component-based architecture και two-way data binding. Ένα από τα βασικά χαρακτηριστικά του Vue.js είναι η χρήση του reactivity που επιτρέπει την αυτόματη ενημέρωση του UI όταν αλλάζουν τα δεδομένα. Με αυτόν τον τρόπο ο προγραμματιστής μπορεί να εστιάσει στη λογική της εφαρμογής χωρίς να χρειάζεται να ανησυχεί για τη χειροκίνητη ενημέρωση και διαχείριση του DOM.

Επίσης ενσωματώνει εργαλεία όπως το Vue Router για τη διαχείριση των routes και το Pinia για τη διαχείριση της κατάστασης, προσφέροντας ένα πλήρες οικοσύστημα για την ανάπτυξη εφαρμογών. Υποστηρίζει επίσης τη χρήση των "Components", τα οποία είναι επαναχρησιμοποιήσιμα κομμάτια κώδικα που συμβάλλουν στη δομή και την οργάνωση της εφαρμογής.

Στην εφαρμογή μας κάνουμε χρήση της έκδοσης 3 της Vue.js. Η έκδοση αυτή εισάγει σημαντικές βελτιώσεις στην απόδοση, τη δομή και την ευχρηστία του framework. Μία από τις πιο σημαντικές καινοτομίες είναι το Composition API το οποίο επιτρέπει στους προγραμματιστές να οργανώνουν καλύτερα και να επαναχρησιμοποιούν τη λογική των components με πιο ευέλικτο και καθαρό τρόπο. Το Vue 3 βελτιώνει επίσης την απόδοση του framework μέσω βελτιστοποιήσεων στον πυρήνα του και την υποστήριξη για καλύτερη χειρισμό της ανανέωσης του DOM. Επιπλέον, η νέα έκδοση ενσωματώνει υποστήριξη για TypeScript, καθιστώντας τη χρήση του πιο εύκολη και ασφαλή για μεγάλες εφαρμογές.

2.3.1 Pinia Store

Το Pinia[12] είναι το state management library για το Vue 3 παρέχοντας μια πιο απλή και πιο αποδοτική προσέγγιση για τη διαχείριση της κατάστασης της εφαρμογής. Σχεδιάστηκε για να ενσωματώνεται άψογα με το Vue 3 και να εκμεταλλεύεται τις δυνατότητες του Composition API για καλύτερη οργάνωση και καθαρότερο κώδικα. Το Pinia προσφέρει έναν πιο σύγχρονο τρόπο διαχείρισης της κατάστασης, προσφέροντας παράλληλα βελτιώσεις στην απόδοση και την ευχρηστία. Ένα μεγάλο πλεονέκτημα του Pinia είναι ότι μπορεί να καθαρίσει το κώδικα στα components κάνοντας πολλούς υπολογισμούς στα store του επιστρέφοντας μόνο το αποτέλεσμα στα components.

Η βασική δομή του Pinia χωρίζεται σε stores, τα οποία είναι αντικείμενα που περιέχουν την κατάσταση (state), τις ενέργειες (actions), τους getters. Κάθε store έχει τα εξής κύρια μέρη:

- **Id:** Ένα μοναδικό όνομα για το store.
- **State**: Περιέχει τα δεδομένα ή την κατάσταση που χρειάζεται η εφαρμογή. Αυτά τα δεδομένα είναι προσβάσιμα και ανανεώνονται από τις ενέργειες.

- **Getters**: Παρέχουν έναν τρόπο να εξάγουμε υπολογισμένα δεδομένα από το state. Είναι σαν computed properties, αλλά για ολόκληρο το store.
- **Actions**: Περιέχουν τη λογική για την αλληλεπίδραση με το state όπως η επικοινωνία με APIs ή η τροποποίηση της κατάστασης. Τα actions μπορούν να είναι και ασύγχρονα.
- **Persists:** Υποστηρίζει την αποθήκευση της κατάστασης του store στο localStorage ή sessionStorage, επιτρέποντας την ανάκτηση της κατάστασης ακόμα και μετά την ανανέωση της σελίδας.

Τέλος το pinia μπορεί να επεκταθεί με plugins για να αποκτήσει νέες λειτουργίες όπως για παράδειγμα transactions και είναι μια είναι μια ελαφριά βιβλιοθήκη.

2.3.2 Vite

Το Vite[13] είναι ένα εξαιρετικά προηγμένο εργαλείο για την ανάπτυξη εφαρμογών Vue.js που φέρνει επανάσταση στην εμπειρία του προγραμματιστή και στην απόδοση των εφαρμογών. Αντί να βασίζεται στις παραδοσιακές μεθόδους build όπως το Webpack το Vite εισάγει καινοτόμες τεχνικές που εστιάζουν στην ταχύτητα, την ευχρηστία και την ευελιξία. Η πλατφόρμα επιτρέπει στους προγραμματιστές να επικεντρωθούν στην πραγματική ανάπτυξη των εφαρμογών χωρίς να χρειάζονται να ασχολούνται με τη διαχείριση πολύπλοκων ρυθμίσεων. Βασικά Χαρακτηριστικά του Vite:

- **Γρήγορη Ανάπτυξη και Ανανεώσεις (Hot Module Replacement - HMR)**: Το Vite προσφέρει μία απίστευτα γρήγορη εμπειρία ανάπτυξης μέσω της τεχνολογίας HMR (Hot Module Replacement) η οποία επιτρέπει την άμεση ενημέρωση των τροποποιημένων modules στον browser χωρίς την ανάγκη ανανέωσης της σελίδας. Αυτό σημαίνει ότι οι αλλαγές στον κώδικα εφαρμόζονται σχεδόν άμεσα μειώνοντας δραστικά το χρόνο που απαιτείται για την ανάπτυξη διευκολύνοντας τη διαδικασία debug και επιταχύνοντας την όλη ροή εργασίας.
- **Ταχύτητα στην Εκκίνηση**: Σε αντίθεση με τα παραδοσιακά bundlers που απαιτούν τη σύνθεση και τη δημιουργία ενός πλήρους bundle κατά την εκκίνηση το Vite εκμεταλλεύεται τις δυνατότητες του ES modules για να φορτώσει και να ανανεώσει μόνο τα απαραίτητα κομμάτια του κώδικα σε πραγματικό χρόνο. Αυτό μειώνει σημαντικά τους χρόνους εκκίνησης, καθιστώντας την ανάπτυξη πολύ πιο αποδοτική.
- **Βελτιστοποίηση των Assets για Παραγωγή**: Αν και η ανάπτυξη στο Vite είναι εξαιρετικά γρήγορη η διαδικασία για τη βελτιστοποίηση των assets και των εξαρτημάτων για παραγωγή (production build) είναι εξίσου αποτελεσματική. Η χρήση προηγμένων τεχνικών tree-shaking και code splitting εξασφαλίζει ότι μόνο ο απαραίτητος κώδικας αποστέλλεται στον τελικό χρήστη βελτιώνοντας την απόδοση της εφαρμογής και μειώνοντας το μέγεθος του bundle.
- **Υποστήριξη TypeScript και JSX**: Το Vite προσφέρει πλήρη υποστήριξη για TypeScript και JSX παρέχοντας στους προγραμματιστές ευελιξία στην επιλογή της γλώσσας που επιθυμούν να χρησιμοποιήσουν ενώ ταυτόχρονα διατηρεί την απλότητα και την απόδοση του συστήματος. Η ενσωμάτωση του TypeScript και JSX είναι εξαιρετικά απλή χωρίς την ανάγκη επιπλέον ρυθμίσεων εξοικονομώντας χρόνο και μειώνοντας την πολυπλοκότητα του project.
- **Προκαθορισμένες Ρυθμίσεις για Vue.js**: Το Vite έρχεται με προκαθορισμένες ρυθμίσεις για την υποστήριξη του Vue.js κάνοντάς το ιδανικό για την ανάπτυξη Vue εφαρμογών. Αυτό σημαίνει ότι οι χρήστες μπορούν να ξεκινήσουν άμεσα χωρίς να ανησυχούν για τις απαιτήσεις ρύθμισης αφού το Vite έχει βελτιστοποιήσει την εμπειρία του Vue.js από τη στιγμή της εγκατάστασης.
- **Επεκτασιμότητα και Προσαρμογή**: Ενώ το Vite είναι έτοιμο προς χρήση για τις περισσότερες περιπτώσεις παρέχει και ευέλικτες δυνατότητες προσαρμογής επιτρέποντας στους

προγραμματιστές να τροποποιήσουν τη συμπεριφορά του build system και να ενσωματώσουν επιπλέον plugins ή άλλες βιβλιοθήκες. Επιπλέον με την υποστήριξη plugins και τη δυνατότητα ρύθμισης του config file η πλατφόρμα επιτρέπει την πλήρη προσαρμογή στις ανάγκες του project.

- **Εξαιρετική Υποστήριξη για Περιβάλλον Ανάπτυξης και Παραγωγής**: Η διαχείριση του περιβάλλοντος παραγωγής και ανάπτυξης γίνεται εύκολη με το Vite καθώς προσφέρει εργαλεία για την παρακολούθηση των logs, τη διαχείριση του cache και την παραμετροποίηση για τις διαφορετικές καταστάσεις του έργου. Αυτό καθιστά την όλη διαδικασία ανάπτυξης, καθώς και τη μεταφορά της εφαρμογής στο production, πιο αποτελεσματική.

2.3.3 Δομή ενός vue component

Η δομή ενός Vue component[14] με Vue 3 και Composition API είναι διαφορετική από τη δομή των παλαιότερων εκδόσεων του Vue καθώς εισάγεται μια νέα προσέγγιση για τη διαχείριση της λογικής του component μέσω του Composition API. Αυτό προσφέρει μια πιο οργανωμένη και ευέλικτη μεθοδολογία για την οργάνωση της λογικής του component. Αναλυτικά, η δομή ενός Vue component με Composition API περιλαμβάνει τα εξής βασικά μέρη:

- **Script:** Στη ενότητα script, χρησιμοποιούνται οι δυνατότητες της Composition API για να διαχειριστούμε την κατάσταση, τις μεθόδους και τις εξαρτήσεις του component. Σε αυτήν την ενότητα μπορούμε να χρησιμοποιήσουμε το setup για να δηλώσουμε μεταβλητές με ή χωρίς reactive state, computed properties, watchers, καθώς και μεθόδους.
- **Template:** Η ενότητα template είναι υπεύθυνη για την παρουσίαση του component. Σε αυτήν την ενότητα καθορίζονται τα HTML tags και η εμφάνιση των δεδομένων που θα αποδοθούν στην οθόνη. Μπορούμε επίσης να περάσουμε και να εμφανίσουμε μεταβλητές της vue απευθείας μέσα στο template.
- **Style:** Η ενότητα style χρησιμοποιείται για να ορίσουμε τα στυλ του component. Στη Vue 3, η ενότητα style μπορεί να είναι τοπική (scoped) δηλαδή να επηρεάζει μόνο το συγκεκριμένο component ή παγκόσμια δηλαδή να επηρεάζει και όλα τα υπόλοιπα components.

2.3.4 Axios

Το Axios[15] είναι μια δημοφιλής βιβλιοθήκη JavaScript που χρησιμοποιείται για την αποστολή HTTP αιτημάτων από το frontend προς το backend. Είναι βασισμένο σε Promises και προσφέρεται για την πραγματοποίηση αιτημάτων GET, POST, PUT, DELETE, κ.λπ. Το Axios χρησιμεύει στην αποστολή HTTP αιτημάτων από τη frontend εφαρμογή στην backend επιτρέποντας την ανάκτηση, αποστολή και ενημέρωση δεδομένων από και προς τον server. Είναι ιδιαίτερα χρήσιμο όταν θέλουμε να δημιουργήσουμε διαδραστικές εφαρμογές που απαιτούν επικοινωνία με έναν διακομιστή όπως η υποβολή φορμών, η ανάκτηση δεδομένων από APIs ή η διαχείριση καταστάσεων χρήστη σε πραγματικό χρόνο.

Ορισμένα βασικά χαρακτηριστικά του Axios περιλαμβάνουν:

- Υποστήριξη για όλα τα HTTP αιτήματα (GET, POST, PUT, DELETE).
- Αυτόματη μετατροπή JSON δεδομένων.
- Υποστήριξη για παραμετροποίηση headers.
- Διαχείριση σφαλμάτων και επεξεργασία αποκρίσεων με τη χρήση Promises.
- Υποστήριξη για διαχείριση timeouts και ακύρωση αιτημάτων.

Ο Axios συνδυάζεται με τη Laravel για την επικοινωνία με API και την αποστολή/λήψη δεδομένων από τη βάση δεδομένων ή τους controllers και συνδιάζεται και με το Pinia για την αποθήκευση και διαχείριση αυτών των δεδομένων στην εφαρμογή προσφέροντας έτσι μια κεντρική αποθήκευση καταστάσεων που μπορεί να ενημερώνεται μέσω αιτημάτων Axios και να βλέπει ο χρήστης απευθίας τα αποτελέσματα στην εφαρμογή.

2.4 Kotlin και Android εφαρμογή

Η Kotlin είναι μια σύγχρονη, ισχυρή και ασφαλής γλώσσα προγραμματισμού που υποστηρίζεται επίσημα από την Google για την ανάπτυξη Android εφαρμογών. Σε σύγκριση με την Java, προσφέρει πιο λιτό και κατανοητό κώδικα μειώνοντας τη συνθετότητα και τα πιθανά σφάλματα. Η χρήση του Jetpack Compose σε συνδυασμό με το Kotlin επιτρέπει τη δημιουργία μοντέρνων και αποδοτικών UI ενώ τα Coroutines διευκολύνουν την ασύγχρονη επεξεργασία δεδομένων βελτιώνοντας την απόκριση της εφαρμογής. Επιπλέον με την ενσωμάτωση του MVVM (Model-View-ViewModel) και του LiveData, η διαχείριση δεδομένων γίνεται πιο οργανωμένη και αποδοτική προσφέροντας μια βέλτιστη εμπειρία χρήστη. Η ευελιξία της Kotlin σε συνδυασμό με την πλήρη διαλειτουργικότητά της με την Java τη καθιστά την ιδανική επιλογή για τη σύγχρονη ανάπτυξη Android εφαρμογών.

2.4.1 SharedPreferences

Τα SharedPreferences[17] είναι ένας εξαιρετικά χρήσιμος μηχανισμός αποθήκευσης δεδομένων στο Android για την αποθήκευση μικρών ποσοτήτων δεδομένων σε μορφή ζεύγους "κλειδί-τιμή" (key-value pairs). Αυτή η μέθοδος αποθήκευσης είναι ιδιαίτερα χρήσιμη για δεδομένα που πρέπει να επιβιώσουν από την επανεκκίνηση ή το κλείσιμο της εφαρμογής και πρέπει να είναι άμεσα διαθέσιμα κατά τη διάρκεια της χρήσης της εφαρμογής. Τα SharedPreferences είναι ιδανικά για την αποθήκευση ρυθμίσεων, προτιμήσεων ή μικρών πληροφοριών που σχετίζονται με τον χρήστη όπως το theme της εφαρμογής, οι τελευταίες επιλογές ή ακόμα και το token σύνδεσης.

Η χρηστικότητα τους προκύπτει από την απλότητά τους. Σε αντίθεση με άλλους μηχανισμούς αποθήκευσης όπως οι βάσεις δεδομένων ή τα αρχεία τα SharedPreferences δεν απαιτούν πολύπλοκο σχεδιασμό ή διαδικασίες για να αποθηκεύσει ή να ανακτήσει κάποιος δεδομένα. Το Android παρέχει έτοιμες μεθόδους για την ανάγνωση και τροποποίηση των δεδομένων όπως οι get(), put(), remove() καθιστώντας τα εύκολα και γρήγορα στη χρήση. Αυτές οι μέθοδοι επιτρέπουν την αποθήκευση διαφορετικών τύπων δεδομένων όπως String, int, boolean, float, και long, καλύπτοντας έτσι πολλές ανάγκες.

Τα δεδομένα που αποθηκεύονται στα SharedPreferences είναι συνήθως αποθηκευμένα σε XML αρχεία στη συσκευή του χρήστη προσφέροντας έτσι μια δομημένη αλλά εξαιρετικά αποδοτική προσέγγιση για την αποθήκευση μικρών δεδομένων. Η αποθήκευση σε XML αρχεία εξασφαλίζει ότι τα δεδομένα μπορούν να είναι αναγνώσιμα και κατανοητά τόσο από το Android σύστημα όσο και από τους προγραμματιστές που μπορεί να χρειαστεί να τα επεξεργαστούν.

2.4.2 Retrofit

Το **Retrofit**[23] αποτελεί μία από τις πιο δημοφιλείς βιβλιοθήκες για το Android, σχεδιασμένη για να διευκολύνει την επικοινωνία με RESTful APIs με έναν τρόπο που είναι ταυτόχρονα απλός και κατανοητός. Μέσω του Retrofit οι προγραμματιστές μπορούν να πραγματοποιούν HTTP αιτήματα (όπως GET, POST, PUT, DELETE) χωρίς να χρειάζεται να ασχοληθούν με την πολύπλοκη λογική

της διαχείρισης των αιτήσεων ενώ ταυτόχρονα οι JSON απαντήσεις μετατρέπονται αυτόματα σε Java ή Kotlin αντικείμενα μέσω της χρήσης ειδικών converters όπως ο Converter-Gson. Αυτή η ενσωμάτωση προσφέρει ένα καθαρό και οργανωμένο API το οποίο μειώνει σημαντικά την πολυπλοκότητα του κώδικα και επιτρέπει την ταχύτερη ανάπτυξη των εφαρμογών. Επιπλέον το Retrofit διευκολύνει τον χειρισμό των σφαλμάτων παρέχοντας ενσωματωμένες δυνατότητες για την επεξεργασία των HTTP responses και τη διαχείριση των exceptions κάτι που συμβάλλει στη σταθερότητα και την αξιοπιστία της εφαρμογής. Με τη χρήση του Retrofit οι προγραμματιστές μπορούν να δημιουργήσουν εύκολα και αποδοτικά αιτήματα προς το backend ενώ παράλληλα να επωφεληθούν από την αυτόματη μετατροπή των δεδομένων διασφαλίζοντας ότι κάθε API response μετατρέπεται στα κατάλληλα αντικείμενα σύμφωνα με τις ανάγκες της εφαρμογής. Το Retrofit όχι μόνο εξασφαλίζει αξιόπιστη και αποδοτική επικοινωνία με το backend αλλά επίσης βελτιώνει συνολικά την εμπειρία του χρήστη μειώνοντας τις καθυστερήσεις και αυξάνοντας την απόδοση της εφαρμογής. Με λίγες γραμμές κώδικα το Retrofit συνδυάζει ευκολία στη χρήση, υψηλή απόδοση και ένα καθαρό API καθιστώντας το ένα αναντικατάστατο εργαλείο για την ανάπτυξη σύγχρονων εφαρμογών Android.

2.4.3 Converter-Gson

Ο Converter-Gson[24] είναι ένας απαραίτητος μετατροπέας για το Retrofit που αξιοποιεί τη βιβλιοθήκη Gson της Google για τη μετατροπή JSON δεδομένων σε αντικείμενα Java ή Kotlin και αντίστροφα. Με αυτόν τον τρόπο οι προγραμματιστές δεν χρειάζεται να ασχοληθούν με το χειροκίνητο parsing των JSON responses καθώς ο Converter-Gson αναλαμβάνει την αυτόματη μετατροπή των δεδομένων σύμφωνα με τις δομές που έχουν οριστεί όπως τα data classes. Αυτό διευκολύνει σημαντικά την ανάπτυξη εφαρμογών, επιτρέποντας την άμεση και ασφαλή χρήση των δεδομένων που λαμβάνονται από RESTful APIs. Ο Converter-Gson παρέχει επίσης προηγμένες δυνατότητες όπως η δυνατότητα δημιουργίας προσαρμοσμένων adapters ώστε να αντιμετωπίζονται ειδικές περιπτώσεις μετατροπής όπως η επεξεργασία ημερομηνιών ή η διαχείριση null τιμών. Μέσω της απλής ενσωμάτωσης του με την εντολή GsonConverterFactory.create(), ο Converter-Gson ενσωματώνεται άμεσα στο Retrofit εξασφαλίζοντας ότι κάθε JSON response μετατρέπεται στα κατάλληλα αντικείμενα, σύμφωνα με τις προδιαγραφές της εφαρμογής. Αυτή η διαδικασία μειώνει σημαντικά τον χρόνο ανάπτυξης και τη συντήρηση καθώς ο κώδικας γίνεται πιο καθαρός και οργανωμένος επιτρέποντας στους προγραμματιστές να εστιάσουν στην επιχειρηματική λογική της εφαρμογής.

2.4.4 Kotlin Data Classes

Στα data classes, ορίζονται οι δομές δεδομένων που αναμένεται να ληφθούν, όπως για παράδειγμα η κλάση EmployeeDashboardResponse. Αυτές τα data classes λειτουργούν ως "κατάλογος" για το πώς θα μοιάζουν τα δεδομένα που λαμβάνονται από το backend.

Ένα από τα μεγαλύτερα πλεονεκτήματα των data classes στη Kotlin είναι ότι παρέχουν αυτόματα βασικές μεθόδους όπως equals(), hashCode(), toString() και άλλες μειώνοντας έτσι τον κώδικα που πρέπει να γραφτεί. Αυτή η αυτοματοποίηση συμβάλλει στη δημιουργία καθαρών και ευανάγνωστων κώδικων που εστιάζουν στην περιγραφή της δομής των δεδομένων.

Είναι σημαντικό να σημειωθεί πως τα data classes δεν χρησιμοποιούνται μόνο με τους converters αλλά χρησιμεύουν ως δομές που ορίζουν την αναμενόμενη μορφή των δεδομένων. Ο μετατροπέας όπως ο Converter-Gson χρησιμοποιεί αυτές τις δομές για να μετατρέψει το JSON που λαμβάνεται από τις

HTTP κλήσεις σε αντικείμενα της Kotlin. Με αυτόν τον τρόπο εξασφαλίζεται ότι τα δεδομένα που λαμβάνονται είναι κατάλληλα δομημένα και έτοιμα για χρήση εντός της εφαρμογής.

Συνολικά, η χρήση των data classes για τον ορισμό των μοντέλων δεδομένων διευκολύνει τη διαχείριση και την μετατροπή των δεδομένων, ενισχύοντας την αξιοπιστία και την επεκτασιμότητα του κώδικα ενώ παράλληλα συμβάλλει στην αποτελεσματική αλληλεπίδραση μεταξύ του frontend και του backend.

2.4.5 RecyclerView

Το RecyclerView[26] είναι ένα ισχυρό και ευέλικτο component στο Android για την εμφάνιση συλλογών δεδομένων. Χρησιμεύει για να παρουσιάζει αποτελεσματικά μεγάλες λίστες δεδομένων ή δυναμικές λίστες (όπως μηνύματα σε μια εφαρμογή chat, λίστες προϊόντων σε ένα ηλεκτρονικό κατάστημα, ή feed social media) με τρόπο που να είναι ομαλός και αποδοτικός στην κατανάλωση πόρων του συστήματος. Αποτελεί μια βελτίωση του παλαιότερου component ListView και GridView, προσφέροντας μεγαλύτερη ευελιξία και καλύτερες επιδόσεις.

Η κύρια χρησιμότητα του RecyclerView είναι να εμφανίζει αποτελεσματικά και αποδοτικά μεγάλες συλλογές δεδομένων. Αυτό επιτυγχάνεται με την ανακύκλωση των views. Σε αντίθεση με το ListView που δημιουργούσε ένα view για κάθε στοιχείο της λίστας ακόμα και αν δεν ήταν ορατό στην οθόνη το RecyclerView δημιουργεί views μόνο για τα στοιχεία που είναι ορατά και αυτά που πρόκειται να γίνουν ορατά. Όταν ένα στοιχείο φεύγει από την οθόνη (λόγω scrolling) το view του δεν καταστρέφεται αλλά ανακυκλώνεται και χρησιμοποιείται ξανά για να εμφανίσει ένα νέο στοιχείο που εισέρχεται στην οθόνη. Αυτή η διαδικασία ανακύκλωσης μειώνει σημαντικά τη δημιουργία και καταστροφή views βελτιώνοντας την απόδοση και την ομαλότητα της κύλισης (scrolling) ειδικά για μεγάλες λίστες. Επιπλέον το RecyclerView είναι πιο ευέλικτο από το ListView επιτρέποντας:

- **Διαφορετικά Layouts:** Μπορεί να εμφανίσει δεδομένα σε μορφή λίστας, πλέγματος (grid), ή ακόμη και πιο πολύπλοκα custom layouts, μέσω των LayoutManagers.
- **Animations:** Εύκολη προσθήκη animations για εισαγωγή, διαγραφή ή μετακίνηση στοιχείων στη λίστα.
- **Customization:** Μεγαλύτερη ευκολία στην προσαρμογή της εμφάνισης και της συμπεριφοράς των views.

2.4.5.1 Πώς λειτουργεί το RecyclerView

Η λειτουργία του RecyclerView βασίζεται σε τρία κύρια στοιχεία[27]:

1. **ViewHolder:** Είναι μια κλάση που "περιτυλίγει" ένα view (π.χ., ένα layout για ένα στοιχείο λίστας) και αποθηκεύει τις αναφορές στα child views μέσα σε αυτό το layout (π.χ., TextViews, ImageViews). Ο ViewHolder είναι υπεύθυνος για την ανακύκλωση του view. Όταν ένα view πρόκειται να ανακυκλωθεί ο ViewHolder ενημερώνεται με τα δεδομένα για το νέο στοιχείο που θα εμφανίσει. Αυτό αποφεύγει την συνεχόμενη αναζήτηση των child views μέσω findViewById βελτιώνοντας την απόδοση.

2. **Adapter:** Ο Adapter είναι ο "μεσάζοντας" μεταξύ των δεδομένων και του RecyclerView. Είναι υπεύθυνος για:
   - Δημιουργία ViewHolders (onCreateViewHolder): Όταν το RecyclerView χρειάζεται ένα νέο view για να εμφανίσει ο Adapter δημιουργεί ένα νέο ViewHolder και "φουσκώνει" το layout για το στοιχείο λίστας μέσα σε αυτό.

- Σύνδεση δεδομένων με ViewHolders (onBindViewHolder): Για κάθε θέση στη λίστα ο Adapter παίρνει τα δεδομένα από την πηγή δεδομένων (π.χ., μια λίστα, μια βάση δεδομένων) και τα συνδέει με το ViewHolder στη συγκεκριμένη θέση. Δηλαδή ενημερώνει τα TextViews, ImageViews κλπ. μέσα στο ViewHolder με τα κατάλληλα δεδομένα.
- Επιστροφή του αριθμού των στοιχείων (getItemCount): Ο Adapter ενημερώνει το RecyclerView για το πόσα στοιχεία δεδομένων υπάρχουν συνολικά.

3. **LayoutManager:** Ο LayoutManager είναι υπεύθυνος για την τοποθέτηση των views μέσα στο RecyclerView και για τον καθορισμό της στρατηγικής κύλισης (scrolling strategy). Το Android παρέχει προκαθορισμένους LayoutManagers όπως:
   - **LinearLayoutManager:** Για εμφάνιση σε μορφή γραμμικής λίστας (κάθετη ή οριζόντια).
   - **GridLayoutManager:** Για εμφάνιση σε μορφή πλέγματος.
   - **StaggeredGridLayoutManager:** Για εμφάνιση σε μορφή πλέγματος όπου τα στοιχεία μπορεί να έχουν διαφορετικό ύψος.

Συνοπτικά, η ροή λειτουργίας είναι:

1. Το RecyclerView ζητάει από τον LayoutManager να τοποθετήσει views.
2. Ο LayoutManager ζητάει από τον Adapter να δημιουργήσει ViewHolders για να εμφανίσει δεδομένα.
3. Ο Adapter δημιουργεί ViewHolders και τα συνδέει με δεδομένα.
4. Ο LayoutManager τοποθετεί τα ViewHolders μέσα στο RecyclerView.
5. Καθώς ο χρήστης κάνει scroll το RecyclerView ανακυκλώνει ViewHolders ζητώντας από τον Adapter να ενημερώσει τα ανακυκλωμένα ViewHolders με νέα δεδομένα.

2.4.6 Room Database

Στο Android το SQLite είναι η ενσωματωμένη βάση δεδομένων για την αποθήκευση δεδομένων τοπικά στη συσκευή. Παρόλο που το SQLite είναι ισχυρό η άμεση χρήση του μέσω των Android SDK APIs (όπως SQLiteOpenHelper, SQLiteDatabase) μπορεί να είναι αρκετά πολύπλοκη και επιρρεπής σε λάθη. Απαιτείται χειρισμός SQL queries, cursors και η διαχείριση της σχέσης αντικειμένων-σχεσιακών βάσεων δεδομένων (Object-Relational Mapping - ORM) γίνεται χειροκίνητα.

Το Room Database[28] έρχεται να απλοποιήσει αυτή τη διαδικασία. Παρέχει ένα ORM layer που επιτρέπει την πρόσβαση στη βάση δεδομένων μέσω annotation processing και Java Data Objects ή Kotlin Data Classes. Αυτό σημαίνει ότι ο οποιοσδήποτε μπορεί να ορίσει τις βάσεις δεδομένων του, τα tables και τα queries σαν κώδικα Java/Kotlin και το Room θα δημιουργήσει τον απαραίτητο κώδικα SQLite για να λειτουργήσει.

Τα κύρια πλεονεκτήματα του Room Database είναι:

- **Συμπυκνωμένος κώδικας και ευκολία χρήσης:** Απλοποιεί την αλληλεπίδραση με τη βάση δεδομένων μειώνοντας την ποσότητα κώδικα που χρειάζεται να γράψει κάποιος.
- **Compile-time query verification:** Τα SQL queries που ορίζει κάποιος μέσω annotations ελέγχονται κατά τη διάρκεια της compilation. Αν υπάρχει κάποιο συντακτικό λάθος ή κάποιο πρόβλημα στη δομή της βάσης θα το εντοπίσει ο compiler αποφεύγοντας runtime errors.
- **Integration with other Architecture Components:** Συνεργάζεται άψογα με άλλα Architecture Components όπως το LiveData και το RxJava, για να υποστηρίξει reactive data access και

observeable data. Αυτό είναι πολύ χρήσιμο για την αυτόματη ενημέρωση του UI όταν τα δεδομένα στη βάση αλλάζουν.

- **Abstraction over SQLite:** Προσφέρει ένα υψηλότερο επίπεδο αφαίρεσης κάνοντας τον κώδικα πιο καθαρό και συντηρήσιμο. Δεν χρειάζεται να ανησυχεί κάποιος για λεπτομέρειες χαμηλού επιπέδου του SQLite.
- **Υποστήριξη συναλλαγών:** Διασφαλίζει την ακεραιότητα των δεδομένων με την υποστήριξη συναλλαγών.

2.4.6.1 Πώς λειτουργεί το Room Database;

Η λειτουργία του Room Database βασίζεται σε τρία κύρια components:

1. **Entity:** Ένα Entity class αναπαριστά έναν πίνακα (table) στην βάση δεδομένων. Είναι μια απλή Java/Kotlin class που έχει annotated με το @Entity. Τα fields (πεδία) της class αντιστοιχούν στις στήλες (columns) του πίνακα. Έχει την εξής δομή:
   - **@Entity(tableName = "users")**: Καθορίζει ότι αυτό το class αντιστοιχεί στον πίνακα με όνομα "users". Αν δεν δοθεί tableName το όνομα του πίνακα θα είναι το όνομα της class.
   - **@PrimaryKey**: Καθορίζει το primary key για τον πίνακα.
   - **@ColumnInfo(name = "first_name")**: Καθορίζει το όνομα της στήλης. Αν δεν δοθεί name το όνομα της στήλης θα είναι το όνομα του field.
   - **@Ignore**: Καθορίζει ένα field που δεν θα αποθηκευτεί στη βάση δεδομένων.

2. **DAO (Data Access Object):** Ένα DAO interface περιέχει τις μεθόδους για την πρόσβαση στα δεδομένα της βάσης. Τα SQL queries μπορούν να οριστούν μέσω annotations μέσα στο DAO. Το Room γεννά τον κώδικα για την υλοποίηση αυτών των μεθόδων. Έχει την εξής δομή:
   - **@Dao**: Δηλώνει ότι αυτό είναι ένα DAO interface.
   - **@Query("SELECT * FROM users")**: Ορίζει ένα SQL SELECT query.
   - **@Insert**: Ορίζει μια μέθοδο για την εισαγωγή δεδομένων.
   - **@Update**: Ορίζει μια μέθοδο για την ενημέρωση δεδομένων.
   - **@Delete**: Ορίζει μια μέθοδος για τη διαγραφή δεδομένων.

3. **Database:** Ένα Database class είναι abstract και επεκτείνει το RoomDatabase. Είναι το κύριο access point στην υποκείμενη σχεσιακή βάση δεδομένων. Πρέπει να περιλαμβάνει:
   - **@Database(entities = [User::class], version = 1)**: Δηλώνει ότι αυτή είναι μια Room Database class. Το entities καθορίζει τις Entity classes που περιλαμβάνονται στην βάση. Το version είναι η έκδοση της βάσης δεδομένων (σημαντικό για migrations).
   - Μια abstract μέθοδος για κάθε DAO interface που έχει οριστεί (π.χ., userDao(): UserDao). Το Room θα δημιουργήσει την υλοποίηση αυτών των μεθόδων.

Συνοπτικά, η ροή λειτουργίας είναι:

1. Ορισμός των Entity classes που αντιστοιχούν στους πίνακες.
2. Ορισμός των DAO interfaces με τις μεθόδους πρόσβασης στα δεδομένα και τα SQL queries.
3. Ορισμός του Database class που συνδέει τα Entities και τα DAOs και δημιουργεί την βάση δεδομένων.
4. Χρησιμοποίηση της Database class για να απόκτηση πρόσβασης στα DAOs και να εκτέλεση operations στη βάση δεδομένων.

2.5 MySql

Η MySQL[18] είναι ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα συστήματα διαχείρισης βάσεων δεδομένων (DBMS) το οποίο βασίζεται στην τεχνολογία των σχεσιακών βάσεων δεδομένων. Είναι ανοιχτού κώδικα και υποστηρίζει τις πιο κοινές λειτουργίες των σχεσιακών βάσεων όπως ερωτήματα SQL, συναρτήσεις και αποθηκευμένες διαδικασίες. Στη MySQL, τα δεδομένα αποθηκεύονται σε πίνακες, οι οποίοι είναι συνδεδεμένοι μεταξύ τους με κλειδιά επιτρέποντας τη διαχείριση και την ανάλυση μεγάλων ποσοτήτων πληροφοριών.

Όσον αφορά τη διαχείριση των σχέσεων μεταξύ των πινάκων στη συγκεκριμένη εφαρμογή χρησιμοποιούνται τόσο κύρια κλειδιά για τη διασφάλιση της μοναδικότητας των εγγραφών όσο και ξένα κλειδιά για τη διατήρηση της ακεραιότητας των δεδομένων. Η επιλογή χρήσης ξένων κλειδιών αποτρέπει την εισαγωγή ασυμβατότητας ή ορφανών εγγραφών καθώς οι σχετικοί πίνακες συνδέονται ρητά μεταξύ τους. Παρόλο που η διαχείριση των ξένων κλειδιών μπορεί να επιφέρει επιπλέον διαδικασίες ιδίως σε μεγάλες βάσεις δεδομένων τα οφέλη που προκύπτουν από τη διατήρηση της συνοχής των δεδομένων αντισταθμίζουν τη μικρή πιθανή επιβάρυνση στις επιδόσεις. Με αυτόν τον τρόπο διασφαλίζεται ότι κάθε εγγραφή σχετίζεται σωστά με τις αντίστοιχες εγγραφές σε άλλους πίνακες συμβάλλοντας στην αποτελεσματική και ασφαλή λειτουργία της εφαρμογής.

2.6 Vuexy

Το Vuexy[19] είναι ένα προηγμένο και υψηλής ποιότητας front-end template ειδικά σχεδιασμένο για εφαρμογές που βασίζονται στο Vue.js. Πρόκειται για μια ισχυρή ευέλικτη και πλήρως responsive λύση η οποία ενσωματώνει μοντέρνες αρχές σχεδιασμού βελτιστοποιημένη απόδοση και ένα ευρύ φάσμα εργαλείων για την επιτάχυνση της ανάπτυξης εφαρμογών.

Χαρακτηριστικά του Vuexy

- **Πλούσια συλλογή από UI components**: Προσφέρει μια εκτενή βιβλιοθήκη από προκατασκευασμένα UI στοιχεία όπως κουμπιά, φόρμες, πίνακες, ειδοποιήσεις και modal dialogs που επιτρέπουν τη γρήγορη υλοποίηση φιλικών προς τον χρήστη διεπαφών.
- **Συμβατότητα με Vue 3 και Composition API**: Το Vuexy εκμεταλλεύεται πλήρως τις δυνατότητες του Vue 3 προσφέροντας βελτιωμένη διαχείριση κατάστασης, αποδοτικότερο rendering και καθαρότερο κώδικα μέσω του Composition API.
- **Υποστήριξη πολλών frameworks και τεχνολογιών**: Εκτός από το Vue.js, το Vuexy διαθέτει εκδόσεις που υποστηρίζουν React, Angular, HTML και Laravel επιτρέποντας την εύκολη προσαρμογή σε διαφορετικά projects.
- **Προσαρμόσιμο και επεκτάσιμο**: Οι προγραμματιστές μπορούν εύκολα να παραμετροποιήσουν το Vuexy, να προσαρμόσουν τα διαθέσιμα στοιχεία UI και να επεκτείνουν τη λειτουργικότητα της εφαρμογής εξασφαλίζοντας ένα εξατομικευμένο και αποδοτικό τελικό προϊόν.
- **Ενσωματωμένα Plugins και Charts**: Περιλαμβάνει ενσωματωμένα γραφικά (charts), widgets, animation libraries και third-party plugins που διευκολύνουν την παρουσίαση δεδομένων με εντυπωσιακό και κατανοητό τρόπο.

2.7 Επίλογος

Η εφαρμογή συνδυάζει την ισχύ του Laravel framework στο backend με την ευελιξία του Vue.js 3 στο frontend. Στο backend, η Laravel διαχειρίζεται κεντρικές λειτουργίες όπως την αυθεντικοποίηση χρηστών, την αλληλεπίδραση με τη βάση δεδομένων MySQL, την εκτέλεση χρονοβόρων εργασιών στο παρασκήνιο και τον έλεγχο πρόσβασης σε λειτουργίες. Παράλληλα, το Vue.js 3 χρησιμοποιείται για τη δημιουργία ενός δυναμικού και reactive UI, με το Pinia να διαχειρίζεται αποτελεσματικά τα δεδομένα της εφαρμογής και το Vite να βελτιστοποιεί την ταχύτητα ανάπτυξης. Η επικοινωνία μεταξύ frontend και backend γίνεται μέσω Axios.

Στην Android εφαρμογή, η ανάπτυξη έχει πραγματοποιηθεί με τη γλώσσα Kotlin, η οποία προσφέρει πιο σύγχρονο και ασφαλή κώδικα σε σύγκριση με την Java. Για την επικοινωνία με τα RESTful APIs, χρησιμοποιούμε το Retrofit σε συνδυασμό με τον Gson, που επιτρέπουν την αυτόματη μετατροπή των JSON απαντήσεων σε Kotlin αντικείμενα. Αυτή η προσέγγιση όχι μόνο βελτιστοποιεί την ταχύτητα και την ακρίβεια στην επεξεργασία των δεδομένων αλλά καθιστά και ευκολότερη τη συντήρηση της εφαρμογής. Για την εμφάνιση και διαχείριση των δεδομένων στο Android, χρησιμοποιούμε το RecyclerView το οποίο παρέχει έναν αποδοτικό και ευέλικτο τρόπο για την παρουσίαση μεγάλων ποσοτήτων δεδομένων σε μια λίστα ή πλέγμα. Για την αποθήκευση δεδομένων τοπικά στην εφαρμογή χρησιμοποιούμε το Room Database το οποίο παρέχει μια πιο σύγχρονη και εύχρηστη προσέγγιση στην αποθήκευση και διαχείριση των δεδομένων σε σχέση με το παραδοσιακό SQLite.

Τέλος, το Vuexy template επιτάχυνε σημαντικά την ανάπτυξη του UI, παρέχοντας ένα πλήρες και ευέλικτο σύνολο από προσχεδιασμένα UI components που επέτρεψαν τη γρήγορη υλοποίηση του frontend.

3. Παρουσίαση Λειτουργιών

1.1. Εισαγωγή

Στο παρόν κεφάλαιο θα περιγράψουμε αναλυτικά τις κύριες λειτουργίες του συστήματος εξηγώντας τον τρόπο με τον οποίο κάθε λειτουργία εξυπηρετεί τις ανάγκες της εφαρμογής. Επιπλέον θα αναλύσουμε τις απαιτήσεις που χρειάζονται για την εγκατάσταση του συστήματος, τη διαδικασία εγκατάστασης βήμα προς βήμα, καθώς και την υλοποίηση και τις λειτουργίες της Android εφαρμογής που συνοδεύει το σύστημα. Μέσα από αυτή την παρουσίαση στόχος μας είναι να δοθεί μια σαφής εικόνα για τη συνολική λειτουργικότητα και την τεχνική αρτιότητα της εφαρμογής.

3.2 Διαδικασία Εγκατάστασης του Συστήματος

Η εγκατάσταση του συστήματος απαιτεί την ικανοποίηση ορισμένων προϋποθέσεων και την εκτέλεση συγκεκριμένων βημάτων για την ορθή λειτουργία του. Αρχικά, είναι απαραίτητο να εγκατασταθούν η PHP έκδοσης 8.2, η MySQL για τη διαχείριση της βάσης δεδομένων, και η Node.js σε έκδοση μεγαλύτερη της 17. Αφού εγκατασταθούν οι απαραίτητες τεχνολογίες μεταβαίνουμε στο directory του Project. Εκεί μετονομάζουμε το .env.example σε .env και προσαρμόζουμε τα κλειδιά μέσα με βάση τα στοιχεία της βάσης και της του email Server / provider έπειτα εκτελούμε τις παρακάτω εντολές:

- **composer install** για την εγκατάσταση των εξαρτήσεων της Laravel.
- **npm i** για την εγκατάσταση των απαιτούμενων πακέτων Node.js.
- **php artisan key:generate** για τη δημιουργία του application key.
- **php artisan migrate:fresh --seed** για τη δημιουργία και την αρχικοποίηση της βάσης δεδομένων με τα seeders.

Αφού ολοκληρωθούν τα παραπάνω για να ξεκινήσει το σύστημα εκτελούμε την εντολή **php artisan serve** για την έναρξη του backend Server και την εντολή **npm run dev** για τη δημιουργία του frontend. Με την ολοκλήρωση αυτών των βημάτων, το σύστημα είναι έτοιμο προς χρήση και διαθέσιμο στη διεύθυνση 127.0.0.1:8000 αν το τρέχουμε τοπικά.

3.3 Λειτουργίες του συστήματος

Το σύστημα που αναπτύχθηκε περιλαμβάνει ένα σύνολο λειτουργιών σχεδιασμένων να καλύψουν τις ανάγκες των επιχειρήσεων εστίασης και να προσφέρουν μια ολοκληρωμένη εμπειρία χρήστη. Κάθε λειτουργία έχει σχεδιαστεί με γνώμονα τη βέλτιστη απόδοση και την ευκολία χρήσης. Σε αυτή την ενότητα, θα παρουσιαστούν αναλυτικά οι βασικές λειτουργίες του συστήματος.

3.3.1 Αρχικοποίηση συστήματος

Κατά τη διάρκεια αυτής της αρχικοποίησης, πραγματοποιείται μια κλήση προς το backend προκειμένου να φορτωθούν τα απαραίτητα δεδομένα που απαιτούνται για τη σωστή και απρόσκοπτη λειτουργία του συστήματος. Αυτά τα δεδομένα περιλαμβάνουν τις κατηγορίες, τις υποκατηγορίες, τα προϊόντα, τα χαρακτηριστικά των προϊόντων, τις τοποθεσίες των τραπεζιών, τις ειδοποιήσεις, τους ρόλους χρηστών καθώς και την κατάσταση του ανοίγματος της ημέρας.

Εφόσον έχει γίνει σύνδεση χρήστη, το σύστημα προχωρά στη φόρτωση των βασικών στοιχείων του χρήστη, του ρόλου του και των δικαιωμάτων που του έχουν εκχωρηθεί. Όλα αυτά τα δεδομένα από την

αρχικοποίηση αποθηκεύονται στο αντίστοιχο Pinia store επιτρέποντας γρήγορη και εύκολη πρόσβαση σε αυτά κατά τη διάρκεια της χρήσης της εφαρμογής.

Η λήψη αυτών των δεδομένων κατά την αρχική φόρτωση του συστήματος αποσκοπεί στην εξάλειψη της ανάγκης για συχνές αιτήσεις προς το backend. Με αυτόν τον τρόπο, εξασφαλίζεται μια πιο αποδοτική και γρήγορη εμπειρία χρήστη καθώς μειώνεται σημαντικά ο αριθμός των αιτήσεων που πρέπει να γίνουν κατά τη διάρκεια της λειτουργίας της εφαρμογής. Επιπλέον, περιορίζεται η ανάγκη για συνεχιζόμενη επικοινωνία με τον server και μειώνεται η φόρτωση στο σύστημα διασφαλίζοντας την ομαλή και αποδοτική λειτουργία του.

Αυτή η στρατηγική για τη φόρτωση των απαραίτητων δεδομένων από την αρχή, όχι μόνο ενισχύει την ταχύτητα και την απόδοση της πλατφόρμας, αλλά δημιουργεί επίσης ένα ισχυρό θεμέλιο για μια ομαλή, συνεχιζόμενη εμπειρία χρήστη, μειώνοντας τις καθυστερήσεις και τις αβεβαιότητες που μπορεί να προκύψουν κατά τη διάρκεια της χρήσης της εφαρμογής.

3.3.2 Σύνδεση

Κατά την έναρξη της εφαρμογής ο χρήστης οδηγείται στη σελίδα σύνδεσης. Στη φόρμα αυτή εισάγει τα στοιχεία πρόσβασής του τα οποία αποστέλλονται στον αντίστοιχο controller μέσω ενός αιτήματος. Ο controller συγκρίνει τα εισαχθέντα στοιχεία με εκείνα που είναι αποθηκευμένα στη βάση δεδομένων.

Σε περίπτωση που τα στοιχεία είναι λανθασμένα εμφανίζεται ένα μήνυμα λάθους που ενημερώνει τον χρήστη για το πρόβλημα δίνοντάς του τη δυνατότητα να προσπαθήσει ξανά. Αν τα στοιχεία είναι σωστά δημιουργείται ένα session για τον χρήστη και ένα μοναδικό token αποθηκεύεται στο browser του. Το token αυτό χρησιμοποιείται για τη διαχείριση της αυθεντικοποίησης σε επόμενα αιτήματα του χρήστη.Αφού ολοκληρωθεί επιτυχώς η διαδικασία σύνδεσης, ο χρήστης μεταφέρεται στην αρχική σελίδα του συστήματος, το Dashboard, όπου μπορεί να ξεκινήσει τη χρήση των λειτουργιών του συστήματος.

3.3.3 Dashboard

Ανάλογα με τον ρόλο που έχει ο χρήστης το dashboard που εμφανίζεται είναι προσαρμοσμένο στις ανάγκες και τις αρμοδιότητές του:

- **Σερβιτόροι:** Οι σερβιτόροι έχουν πρόσβαση σε ένα απλοποιημένο dashboard στο οποίο εμφανίζονται πληροφορίες σχετικά με το σημερινό τους ταμείο το πλήθος των σημερινών ανοιχτών παραγγελιών τους καθώς και το πλήθος των συνολικών παραγγελιών που έχουν διαχειριστεί την τρέχουσα ημέρα.
- **Μπαρ:** Οι χρήστες που ανήκουν στον ρόλο του μπαρ βλέπουν στο dashboard μόνο τον αριθμό των παραγγελιών που βρίσκονται σε κατάσταση αναμονής ώστε να γνωρίζουν ποιες παραγγελίες χρειάζεται να ετοιμάσουν.
- **Διαχειριστές / Moderators:** Το dashboard για αυτούς τους ρόλους είναι πιο σύνθετο και προσαρμοσμένο στις αυξημένες ανάγκες τους. Ανάλογα με τα δικαιώματά τους μπορεί να περιλαμβάνει διάφορα τμήματα και λειτουργίες. Αρχικά, εμφανίζεται μια ειδοποίηση για να κάνουν άνοιγμα της ημέρας ώστε να ξεκινήσει η καταγραφή παραγγελιών στο σύστημα. Αν υπάρχουν ειδοποιήσεις από την αποθήκη εμφανίζονται στο dashboard, παρέχοντας ενημέρωση για πιθανές ελλείψεις. Επιπλέον έχουν πρόσβαση στα στατιστικά της ημέρας από τη στιγμή που έγινε η έναρξη. Πιο αναλυτικά, μπορούν να παρακολουθούν:
    - Όλους όσους έχουν συνδεθεί σήμερα

- Τους σερβιτόρους που έχουν συνδεθεί σήμερα
- Τα σημερινά συνολικά έσοδα
- Τις σημερινές παραγγελιές
- Τις ανοιχτές παραγγελίες
- Την αξία ανοιχτών παραγγελιών
- Τις συνολικές διαγραμμένες παραγγελίες
- Την αξία των διαγραμμένων παραγγελιών
- Το πλήθος των διαγραμμένων προϊόντων από τις παραγγελίες
- Την αξία των διαγραμμένων προϊόντων από τις παραγγελίες

Επιπλέον, οι διαχειριστές και οι moderators έχουν τη δυνατότητα να δουν έναν πίνακα που περιέχει πληροφορίες για τους σερβιτόρους που έχουν στείλει παραγγελίες κατά τη διάρκεια της ημέρας. Ο πίνακας αυτός παρέχει τα εξής δεδομένα: το πλήθος των συνολικών παραγγελιών που έχει καταχωρίσει κάθε σερβιτόρος, τη συνολική αξία των παραγγελιών του, καθώς και την αξία των παραγγελιών που παραμένουν ανοιχτές. Αυτή η λειτουργία διευκολύνει την παρακολούθηση της απόδοσης των σερβιτόρων, την ανάλυση της δραστηριότητάς τους και τη διαχείριση του καθημερινού κύκλου εργασιών.

3.3.4 Παραγγελίες

Στην ενότητα "Παραγγελίες" ο χρήστης έχει πρόσβαση σε μια σελίδα που περιλαμβάνει έναν πίνακα με όλες τις τρέχουσες παραγγελίες. Από τον πίνακα αυτόν παρέχονται σημαντικές πληροφορίες όπως η ώρα αποστολής της παραγγελίας, το τραπέζι στο οποίο ανήκει, η συνολική αξία της παραγγελίας, ο σερβιτόρος που την καταχώρισε και η τρέχουσα κατάστασή της.

Ο χρήστης έχει τη δυνατότητα να διαχειριστεί τις παραγγελίες με δύο βασικές λειτουργίες. Μπορεί να διαγράψει μια παραγγελία εντελώς ή να την προβάλει αναλυτικά. Στην αναλυτική προβολή, παρουσιάζονται όλες οι λεπτομέρειες της παραγγελίας και δίνεται η επιλογή είτε να διαγραφεί ολόκληρη η παραγγελία είτε να αφαιρεθούν συγκεκριμένα προϊόντα από αυτήν.

Επιπλέον η σελίδα παραγγελιών προσφέρει τη δυνατότητα φιλτραρίσματος με βάση την κατάσταση των παραγγελιών. Αυτό επιτρέπει στον χρήστη να εντοπίσει εύκολα τις παραγγελίες που τον ενδιαφέρουν όπως παραγγελίες που βρίσκονται σε αναμονή, ολοκληρωμένες ή ακυρωμένες. Αυτή η λειτουργία διευκολύνει σημαντικά τη διαχείριση, επιτρέποντας την καλύτερη οργάνωση και παρακολούθηση των παραγγελιών σε πραγματικό χρόνο.

3.3.5 Διαγραφές

Στο μενού "Διαγραφές" υπάρχουν δύο υπομενού που παρέχουν πληροφορίες σχετικά με τις διαγραφές: το ένα αφορά τα προϊόντα και το άλλο τις παραγγελίες.

Στην ενότητα των προϊόντων εμφανίζεται ένας πίνακας που περιλαμβάνει πληροφορίες για κάθε διαγραφή προϊόντος. Συγκεκριμένα εμφανίζονται η ημερομηνία και η ώρα διαγραφής του προϊόντος το ID της παραγγελίας στην οποία ανήκε το προϊόν ο σερβιτόρος που σχετίζεται με την παραγγελία το όνομα του προϊόντος που διαγράφηκε καθώς και το άτομο που προχώρησε στη διαγραφή.

Στην ενότητα των παραγγελιών εμφανίζεται ένας ξεχωριστός πίνακας με πληροφορίες για τις διαγραφές παραγγελιών. Ο πίνακας αυτός περιλαμβάνει την ημερομηνία και την ώρα διαγραφής το ID της

διαγραμμένης παραγγελίας τον σερβιτόρο που σχετίζεται με την παραγγελία και το άτομο που προχώρησε στη διαγραφή.

Και στις δύο ενότητες, οι πίνακες παρέχουν τη δυνατότητα φιλτραρίσματος με βάση την ημερομηνία. Αυτό διευκολύνει τον εντοπισμό συγκεκριμένων διαγραφών επιτρέποντας στους χρήστες να παρακολουθούν με ακρίβεια τις αλλαγές που έχουν γίνει στο σύστημα.

3.3.6 Νέα Παραγγελία

Η δυνατότητα δημιουργίας νέας παραγγελίας είναι ένα από τα πιο σημαντικά χαρακτηριστικά του συστήματος και έχει σχεδιαστεί για να παρέχει ευχρηστία και αποτελεσματικότητα στους σερβιτόρους. Η διαδικασία δημιουργίας μιας παραγγελίας χωρίζεται σε δύο κύριες κάρτες καθεμία με τη δική της λειτουργικότητα και σκοπό.

Στην πρώτη κάρτα εμφανίζονται συνοπτικές πληροφορίες για την παραγγελία όπως το πλήθος των προϊόντων που έχουν προστεθεί και η συνολική αξία τους. Επιπλέον παρέχονται τρία κουμπιά λειτουργιών: το κουμπί για την προβολή της παραγγελίας και την τυχόν διαγραφή κάποιου προϊόντος, το κουμπί για την ολική διαγραφή της παραγγελίας και το κουμπί για την αποστολή της παραγγελίας.

Στην κάτω κάρτα αρχικά εμφανίζονται τα άδεια τραπέζια ή τα τραπέζια για τα οποία ο σερβιτόρος έχει ήδη στείλει παραγγελίες. Επιλέγοντας το τραπέζι της επιλογής του ο σερβιτόρος έχει τη δυνατότητα να δει τις κύριες κατηγορίες προϊόντων. Κάθε κατηγορία μπορεί να αποκαλύψει τις σχετικές υποκατηγορίες ενώ οι υποκατηγορίες περιέχουν τα προϊόντα που είναι διαθέσιμα για παραγγελία. Αυτή η ιεραρχική διάταξη κατηγοριών και προϊόντων έχει σχεδιαστεί για να διατηρεί την εφαρμογή καθαρή και εύχρηστη διευκολύνοντας την γρήγορη πλοήγηση και την αναζήτηση των προϊόντων.

Όταν ο σερβιτόρος επιλέξει ένα προϊόν ανοίγει ένα παράθυρο για την επεξεργασία του όπου μπορεί να επιλέξει την ποσότητα του προϊόντος και να προσθέσει χαρακτηριστικά που έχουν καθοριστεί από τον διαχειριστή (όπως ζάχαρη ή γάλα για έναν φραπέ). Επιπλέον, ο σερβιτόρος έχει τη δυνατότητα να προσθέσει σημειώσεις σχετικά με τις ειδικές απαιτήσεις του πελάτη εάν αυτές δεν καλύπτονται από τα χαρακτηριστικά του προϊόντος.

Με το πάτημα του κουμπιού "Προσθήκη" το προϊόν και όλα τα χαρακτηριστικά του προστίθενται στο Pinia Store (order) όπου αποθηκεύονται στην παραγγελία (new_order). Για λόγους ασφάλειας, το state της παραγγελίας παραμένει αποθηκευμένο στο Pinia μέχρι να σταλεί η παραγγελία ή μέχρι να λήξει το session του σερβιτόρου. Αυτό διασφαλίζει ότι οι παραγγελίες δεν θα χαθούν ακόμα και αν ο χρήστης αποχωρήσει από τη σελίδα ή κάνει ανανέωση. Επιπλέον προκειμένου να αποτραπούν λάθη ή κακόβουλες ενέργειες αν υπάρχουν προϊόντα προς αποστολή ο σερβιτόρος δεν μπορεί να επιλέξει νέο τραπέζι ή να επιστρέψει στην προηγούμενη οθόνη με τα τραπέζια χωρίς πρώτα να στείλει την παραγγελία. Ενημερώνεται σχετικά με αυτό με την εμφάνιση του αντίστοιχου μηνύματος.

Τέλος, όταν η παραγγελία αποστέλλεται στο backend καταχωρείται με την κατάσταση "pending" και υπολογίζεται εκ νέου το κόστος της παραγγελίας για λόγους ασφαλείας. Αυτός ο έλεγχος αποτρέπει την παραποίηση της αξίας της παραγγελίας κατά την αποστολή του αιτήματος εξασφαλίζοντας την ορθότητα των δεδομένων.

3.3.7 Μπαρ

Αφού καταχωρηθεί μια παραγγελία, αυτή αποστέλλεται στο μπαρ, όπου οι υπάλληλοι παρακολουθούν και διαχειρίζονται τις παραγγελίες. Στο μπαρ οι παραγγελίες εμφανίζονται με μορφή καρτών σε

αύξουσα σειρά, διασφαλίζοντας ότι οι νέες παραγγελίες τοποθετούνται στο τέλος. Κάθε 5 δευτερόλεπτα πραγματοποιείται αυτόματος έλεγχος για νέες παραγγελίες, διασφαλίζοντας ότι το μπαρ είναι πάντα ενημερωμένο.

Κάθε κάρτα περιλαμβάνει βασικές πληροφορίες για την παραγγελία, όπως το ID της, το συνολικό ποσό, τον σερβιτόρο που την καταχώρησε, το τραπέζι στο οποίο ανήκει και την ώρα που πραγματοποιήθηκε η παραγγελία. Επιπλέον η κάρτα περιλαμβάνει αναλυτικά τα προϊόντα που περιέχονται στην παραγγελία, τα χαρακτηριστικά τους και τυχόν σημειώσεις που μπορεί να έχει προσθέσει ο σερβιτόρος.

Στο κάτω μέρος της κάρτας υπάρχουν δύο κουμπιά το "Αποδοχή" και το "Απόρριψη". Αν η παραγγελία απορριφθεί, το status της παραγγελίας αλλάζει σε "denied" και το πεδίο deleted_at στη βάση δεδομένων ενημερώνεται μέσω του SoftDeletes. Αυτό σημαίνει ότι η παραγγελία θεωρείται διαγραμμένη αλλά τα δεδομένα δεν διαγράφονται πραγματικά από τη βάση. Ωστόσο τα προϊόντα που ανήκουν στην απορριφθείσα παραγγελία διαγράφονται κανονικά από τη βάση χρησιμοποιώντας forceDelete καθώς δεν έχουν σημαντική πληροφοριακή αξία για τις απορριπτόμενες παραγγελίες.

Αν η παραγγελία γίνει αποδεκτή το status της αλλάζει σε "completed" και καλείται το DeductProductQuantity job. Αυτό το job ελέγχει τα προϊόντα της παραγγελίας και υπολογίζει την ποσότητα κάθε προϊόντος που πρέπει να αφαιρεθεί από το απόθεμα με κάθε παραγγελία που το περιέχει. Αφού γίνει ο υπολογισμός της αφαίρεσης της ποσότητας, γίνεται σύγκριση με το όριο προειδοποίησης για το απόθεμα που έχει καθορίσει ο διαχειριστής. Εάν η ποσότητα πέσει κάτω από το όριο, δημιουργείται μια νέα ειδοποίηση productwarning στη βάση δεδομένων, αλλά μόνο εάν η προηγούμενη ειδοποίηση έχει διαβαστεί ή δεν υπάρχει δημιουργείται καινούργια.

Η ειδοποίηση για το χαμηλό απόθεμα εμφανίζεται στο dashboard ή στην status bar για να ενημερώσει τους υπεύθυνους. Τέλος το SendWarningLimitEmail Job εκτελείται και στέλνει email σε όλα τα άτομα που έχει ορίσει ο διαχειριστής ενημερώνοντάς τα για το χαμηλό απόθεμα του προϊόντος ώστε να ληφθούν έγκαιρα μέτρα για την ανανέωση του αποθέματος.

3.3.8 Τραπέζια

Στο μενού "Τραπέζια" εμφανίζονται όλα τα τραπέζια του καταστήματος με διαφορετική απόχρωση για να υποδεικνύεται η κατάσταση κάθε τραπεζιού. Τα τραπέζια που δεν έχουν παραγγελία εμφανίζονται με γκρίζο χρώμα, αυτά που έχουν παραγγελία σε αναμονή εμφανίζονται πορτοκαλί, τα τραπέζια με ολοκληρωμένη παραγγελία είναι πράσινα και τα τραπέζια με πληρωμένη παραγγελία είναι μπλε. Επιλέγοντας ένα τραπέζι, ανοίγει μια κάρτα με τις πληροφορίες του όπως το όνομα του τραπεζιού και τη δυνατότητα να διαγραφεί ή να μεταφερθεί σε άλλη τοποθεσία.

Αν το τραπέζι περιέχει παραγγελίες το σύστημα δεν επιτρέπει τη διαγραφή του. Επίσης η κάρτα εμφανίζει τον σερβιτόρο που ανέλαβε την παραγγελία, την κατάσταση της παραγγελίας και τις δυνατότητες για τη διαγραφή ή το κλείσιμο του τραπεζιού. Όταν το τραπέζι κλείνει, το σύστημα ελέγχει αν το τραπέζι ανήκει σε άλλον χρήστη. Αν ανήκει επιστρέφει σφάλμα. Διαφορετικά το status της παραγγελίας ενημερώνεται σε "closed", και το σύστημα προχωράει στη διαγραφή των προϊόντων που περιλαμβάνονται στην παραγγελία καθώς και της ίδιας της παραγγελίας (μέσω Soft Deletes).

Επιπλέον παρέχεται η δυνατότητα να φιλτράρουμε τα τραπέζια με βάση την κατάσταση της παραγγελίας και την τοποθεσία τους. Στο status bar υπάρχουν δύο κουμπιά. Το πρώτο κουμπί επιτρέπει στο διαχειριστή να προσθέσει νέα τραπέζια, δίνοντας ένα όνομα και επιλέγοντας μια τοποθεσία. Το δεύτερο κουμπί είναι για τη διαχείριση των τοποθεσιών, επιτρέποντας στον διαχειριστή να προσθέσει νέες τοποθεσίες, να επεξεργαστεί τις υπάρχουσες ή να διαγράψει κάποιες που δεν είναι πια απαραίτητες. Στη περίπτωση κάποιας διαγραφής διαγράφονται και όλα τα τραπέζια που ανήκουν σε εκείνη τη τοποθεσία.

3.3.9 Σερβιτόροι

Στο μενού "Σερβιτόροι" εμφανίζονται όλοι οι χρήστες με ρόλο σερβιτόρου σε μορφή κάρτας. Κάθε κάρτα περιλαμβάνει το όνομα και το επώνυμο του σερβιτόρου, την ημερομηνία της τελευταίας σύνδεσης και τον συνολικό αριθμό των ημερών που έχει εργαστεί στην επιχείρηση. Για κάθε σερβιτόρο υπάρχουν δύο κουμπιά: το κουμπί διαγραφής το οποίο χρησιμοποιείται σε περίπτωση που ο σερβιτόρος έχει σταματήσει να δουλεύει στην επιχείρηση και το κουμπί "Προβολής" το οποίο ανοίγει ένα νέο παράθυρο με βασικά στοιχεία του σερβιτόρου όπως το όνομα, επώνυμο, τηλέφωνο, κωδικό και ημερομηνία εγγραφής.

Στο νέο παράθυρο εμφανίζονται αναλυτικά οι ημερομηνίες που ο σερβιτόρος εργάστηκε και το ταμείο που έκανε κάθε ημέρα καθώς και το συνολικό του ταμείο και τις συνολικές μέρες εργασίας του. Επιπλέον παρέχεται η δυνατότητα επεξεργασίας και ενημέρωσης αυτών των στοιχείων ώστε να είναι πάντοτε ενημερωμένα και ακριβή. Στο status bar υπάρχει και το κουμπί "Προσθήκης σερβιτόρου" το οποίο επιτρέπει στο διαχειριστή να προσθέσει έναν νέο σερβιτόρο στην πλατφόρμα με τα απαραίτητα στοιχεία του.

3.3.10 Στατιστικά

Στο μενού "Στατιστικά" υπάρχει ένα υπομενού που περιλαμβάνει τρεις ενότητες τα Ημερήσια Στατιστικά, τα Στατιστικά Σερβιτόρων και τα Στατιστικά Προϊόντων. Κάθε ενότητα παρέχει χρήσιμες πληροφορίες και δυνατότητες φιλτραρίσματος για την παρακολούθηση των επιδόσεων της επιχείρησης.

- **Στατιστικά Ημέρας**: Σε αυτή την ενότητα εμφανίζεται ένας πίνακας που περιλαμβάνει την ώρα έναρξης και λήξης της λειτουργίας της επιχείρησης, τα συνολικά έσοδα της ημέρας, τον αριθμό των συνολικών παραγγελιών, τις διεγραμμένες παραγγελίες και τον αριθμό των σερβιτόρων που εργάστηκαν εκείνη την ημέρα.
- **Στατιστικά Σερβιτόρων**: Ο πίνακας αυτής της ενότητας παρουσιάζει τα ονόματα χρηστών των σερβιτόρων, τα συνολικά τους έσοδα, και τις συνολικές παραγγελίες που έχουν διαχειριστεί. Οι πληροφορίες αυτές βοηθούν στην αξιολόγηση των επιδόσεων των σερβιτόρων.
- **Στατιστικά Προϊόντων**: Στη συγκεκριμένη ενότητα εμφανίζεται ένας πίνακας που περιλαμβάνει το όνομα κάθε προϊόντος, τη συνολική ποσότητα του προϊόντος που έχει πουληθεί και τις συνολικές παραγγελίες που περιείχαν το συγκεκριμένο προϊόν. Αυτά τα στατιστικά βοηθούν στην παρακολούθηση των πωλήσεων και των προϊόντων που είναι πιο δημοφιλή.

Τέλος σε όλες τις ενότητες των στατιστικών παρέχεται η δυνατότητα να φιλτραριστούν δυναμικά οι εγγραφές επιλέγοντας την ημερομηνία και την ώρα έναρξης και λήξης για το συγκεκριμένο διάστημα που επιθυμεί ο χρήστης.

3.3.11 Αποθήκη

Το μενού “Αποθήκη” είναι και αυτό ένα από τα πιο σημαντικά χαρακτηριστικά του συστήματος και περιλαμβάνει υπομενού για την εύκολη διαχείριση των κατηγοριών, των υποκατηγοριών, των προϊόντων, των χαρακτηριστικών και των αποθεμάτων.

- **Κατηγορίες**: Σε αυτή την ενότητα εμφανίζεται ένας πίνακας που περιλαμβάνει το όνομα της κάθε κατηγορίας. Ο χρήστης έχει τη δυνατότητα να προσθέσει νέες κατηγορίες, να επεξεργαστεί υπάρχουσες κατηγορίες ή να τις διαγράψει.

- **Υποκατηγορίες**: Ο πίνακας της ενότητας υποκατηγοριών εμφανίζει τα ονόματα των υποκατηγοριών και την κύρια κατηγορία στην οποία ανήκουν. Ο χρήστης μπορεί να προσθέσει νέες υποκατηγορίες επιλέγοντας τη κύρια κατηγορία στην οποία ανήκουν, να επεξεργαστεί υπάρχουσες υποκατηγορίες ή να τις διαγράψει. Επιπλέον υπάρχει η δυνατότητα φιλτραρίσματος των περιεχομένων του πίνακα με βάση τη κύρια κατηγορία.
- **Προϊόντα**: Ο πίνακας προϊόντων περιλαμβάνει τα ονόματα των προϊόντων, την τιμή τους, την ποσότητα που διατίθεται στο κατάστημα, το όριο προειδοποίησης της ποσότητας, καθώς και την υποκατηγορία και την κατηγορία στην οποία ανήκουν. Ο χρήστης έχει τη δυνατότητα να προσθέσει νέα προϊόντα, να επεξεργαστεί ή να διαγράψει υπάρχοντα προϊόντα. Η ποσότητα και το όριο προειδοποίησης δεν είναι υποχρεωτικά και δεν επηρεάζουν την ομαλή λειτουργία του συστήματος, αλλά παρέχουν χρήσιμες πληροφορίες για την παρακολούθηση των αποθεμάτων.
- **Χαρακτηριστικά**: Ο πίνακας των χαρακτηριστικών περιλαμβάνει το όνομα κάθε χαρακτηριστικού, την κατηγορία στην οποία ανήκει και την επιπλέον χρέωση που μπορεί να έχει το χαρακτηριστικό. Τα χαρακτηριστικά μπορεί να είναι γενικά, όπως π.χ. ο "πάγος", ή πιο συγκεκριμένα, όπως "μέτριος" για το καφέ. Αυτά τα χαρακτηριστικά εμφανίζονται για επιλογή κατά την επεξεργασία των προϊόντων στη νέα παραγγελία. Ανάλογα με την κατηγορία του προϊόντος, το σύστημα θα εμφανίσει όλα τα χαρακτηριστικά που ανήκουν σε αυτήν. Από το μενού των χαρακτηριστικών ο χρήστης έχει τη δυνατότητα να προσθέσει νέα χαρακτηριστικά, να επεξεργαστεί υπάρχοντα ή να διαγράψει χαρακτηριστικά που δεν είναι πλέον απαραίτητα.
- **Διαχείριση Αποθεμάτων**: Ο πίνακας της διαχείρισης αποθεμάτων περιλαμβάνει τα προϊόντα και την ποσότητα τους. Η ποσότητα αυτή αναφέρεται στην ποσότητα που θα αφαιρείται από το προϊόν σε κάθε παραγγελία. Ο χρήστης έχει τη δυνατότητα να προσθέσει, να επεξεργαστεί ή να διαγράψει ποσότητες για τα προϊόντα. Μια ιδιαιτερότητα στην προσθήκη ποσότητας είναι ότι δεν επιτρέπεται να αναθέσουμε δύο διαφορετικές ποσότητες στο ίδιο προϊόν. Αυτό διασφαλίζει την ορθότητα των αποθεμάτων και αποτρέπει τυχόν σφάλματα ή διπλές καταχωρήσεις.

Σε περίπτωση διαγραφής μιας κατηγορίας ή υποκατηγορίας διαγράφονται και όλα τα στοιχεία που βρίσκονται από κάτω. Για παράδειγμα αν διαγραφεί μια κατηγορία θα διαγραφούν όλες οι υποκατηγορίες της και τα προϊόντα των υποκατηγοριών της. Αυτός ο μηχανισμός διαγραφής διασφαλίζει την ακεραιότητα των δεδομένων και αποτρέπει τυχόν ατέλειες ή ασυμβατότητες που μπορεί να προκύψουν από την ύπαρξη ανεξάρτητων κατηγοριών, υποκατηγοριών ή προϊόντων.

3.3.12 Ρυθμίσεις

Η ενότητα “Ρυθμίσεις” επιτρέπει στους διαχειριστές και τους υπεύθυνους του συστήματος να προσαρμόσουν διάφορες παραμέτρους για τη διαχείριση των χρηστών, των ρόλων και των δικαιωμάτων, καθώς και για τις επικοινωνίες σχετικά με τα αποθέματα προϊόντων. Υπάρχουν τα εξής υπομενού στις ρυθμίσεις:

- **Ρυθμίσεις:** Ο χρήστης έχει τη δυνατότητα να αλλάξει τον κωδικό πρόσβασης του. Αλλά του παρέχεται επίσης η δυνατότητα στον χρήστη να διαγράψει τον λογαριασμό του.
- **Ρόλοι:** Στο πίνακα εμφανίζονται οι ρόλοι χρηστών με το όνομα του ρόλου, το slug και τον αριθμό των χρηστών που ανήκουν σε αυτόν. Ο χρήστης μπορεί να δημιουργήσει νέους ρόλους καθώς και να επεξεργαστεί ή να διαγράψει υπάρχοντες ρόλους, με την προϋπόθεση ότι δεν υπάρχουν χρήστες σε αυτούς. Οι ρόλοι "σερβιτόρος" και "μπαρ" δεν μπορούν να διαγραφούν προκειμένου να διατηρηθεί η σωστή λειτουργία του συστήματος.

- **Υπεύθυνοι:** Εμφανίζεται ένας πίνακας με τα στοιχεία των υπευθύνων χρηστών όνομα χρήστη, όνομα, επώνυμο, ρόλο, τηλέφωνο και τελευταία σύνδεση. Ο χρήστης μπορεί να προσθέσει νέους υπεύθυνους, να επεξεργαστεί υπάρχοντες ή να τους διαγράψει.
- **Δικαιώματα:** Χωρίζονται σε δύο κάρτες. Στην πρώτη κάρτα ο διαχειριστής επιλέγει τον ρόλο που θέλει να επεξεργαστεί. Στη δεύτερη κάρτα εμφανίζονται όλα τα δικαιώματα που σχετίζονται με τον επιλεγμένο ρόλο και ο χρήστης μπορεί να επιλέξει ή να αφαιρέσει δικαιώματα για αυτόν τον ρόλο. Κατά την αποθήκευση των αλλαγών η σελίδα επαναφορτώνεται για να ενημερωθούν τα δικαιώματα.
- **Email Επικοινωνίας:** Εμφανίζεται ένας πίνακας με όλα τα emails επικοινωνίας που έχει ορίσει ο χρήστης. Αυτά τα emails χρησιμοποιούνται για να ενημερωθούν οι υπεύθυνοι όταν ένα προϊόν βρίσκεται σε έλλειψη. Ο χρήστης μπορεί να προσθέσει, να επεξεργαστεί ή να διαγράψει τα emails επικοινωνίας.

3.3.13 Δικαιώματα στο frontend

Στο προηγούμενο κεφάλαιο εξηγήσαμε πώς μέσω της χρήσης των Laravel Gates περιορίζουμε τα δικαιώματα και τις ενέργειες του κάθε ρόλου στο backend. Ωστόσο, προκειμένου να διασφαλίσουμε τη συνοχή της εφαρμογής και στο frontend είναι απαραίτητο να μην εμφανίζονται όλα τα μενού σε όλους τους χρήστες. Για το λόγο αυτό ο έλεγχος και ο περιορισμός του περιεχομένου του frontend βασίζονται στα δικαιώματα του κάθε χρήστη.

Κατά την είσοδο του χρήστη στην εφαρμογή τα διαθέσιμα δικαιώματα του χρήστη αποθηκεύονται στο Pinia για να είναι συνέχεια διαθέσιμα απλά και γρήγορα. Έτσι, για κάθε στοιχείο του frontend όπως μενού, κουμπιά ή ακόμα και μηνύματα πραγματοποιείται αρχικά έλεγχος προκειμένου να διαπιστωθεί αν το αντίστοιχο δικαίωμα υπάρχει στη λίστα των διαθέσιμων δικαιωμάτων του χρήστη. Αν δεν υπάρχει το αντίστοιχο στοιχείο εξαφανίζεται απο το DOM.

Σε περίπτωση που κάποιος χρήστης προσπαθήσει να αποκτήσει πρόσβαση σε μια σελίδα (route) που δεν του είναι επιτρεπτή, παρά το γεγονός ότι η σελίδα αυτή είναι κρυφή στο μενού, το σύστημα πραγματοποιεί έναν έλεγχο δικαιωμάτων πριν φορτώσει η σελίδα. Εάν ο χρήστης δεν έχει τα απαραίτητα δικαιώματα ανακατευθύνεται στο dashboard και εμφανίζεται το κατάλληλο μήνυμα σφάλματος.

Αυτή η προσέγγιση περιορισμού του περιεχομένου στο frontend, σε συνδυασμό με την χρήση της μεθόδου this->authorized στο backend, παρέχει ισχυρή προστασία ενάντια σε κακόβουλες προσπάθειες μη εξουσιοδοτημένης πρόσβασης είτε από υπαλλήλους είτε από εξωτερικούς παράγοντες που ενδέχεται να προσπαθήσουν να εκμεταλλευτούν τα API endpoints της εφαρμογής.

3.3.14 Κλείσιμο ημέρας

Η λειτουργία "Κλείσιμο Ημέρας" επιτρέπει στο κατάστημα να κλείσει την ημέρα και να υπολογίσει τα συνολικά αποτελέσματα του. Αυτή η διαδικασία πρέπει να εκτελείται όταν το κατάστημα ολοκληρώνει τη λειτουργία του για την ημέρα και επιθυμεί να συγκεντρώσει τα δεδομένα ώστε να μπορέσει να αξιολογήσει την απόδοση του.

Κατά την εκτέλεση της διαδικασίας κλεισίματος της ημέρας το σύστημα αναλαμβάνει αυτόματα να κλείσει όλες τις παραγγελίες που παραμένουν ανοιχτές και έχουν δημιουργηθεί εντός των χρονικών ορίων της ημέρας (δηλαδή, από την ώρα του ανοίγματος μέχρι την ώρα του κλεισίματος), χρησιμοποιώντας τα αντίστοιχα timestamps. Στη συνέχεια, για κάθε παραγγελία που κλείνει, το

σύστημα τη διαγράφει (SoftDeletes) όπως και τα προϊόντα που περιλαμβάνονται σε αυτήν, ακριβώς όπως συμβαίνει και κατά το κλείσιμο ενός τραπεζιού.

Μετά την ολοκλήρωση της διαδικασίας κλεισίματος των παραγγελιών, το σύστημα προχωρά στον υπολογισμό των συνολικών εσόδων της ημέρας προσφέροντας έτσι μια πλήρη εικόνα των οικονομικών αποτελεσμάτων του καταστήματος. Επιπλέον, καταγράφει το συνολικό πλήθος των παραγγελιών που πραγματοποιήθηκαν εντός της ημέρας.

Στη συνέχεια ομαδοποιούνται οι παραγγελίες με βάση τον σερβιτόρο που τις εξυπηρέτησε και υπολογίζει το συνολικό ταμείο που αποδόθηκε στον κάθε σερβιτόρο το οποίο αποθηκεύεται στον πίνακα των δεδομένων των χρηστών. Παράλληλα, υπολογίζεται ο αριθμός των διαγραμμένων παραγγελιών καθώς και ο αριθμός των σερβιτόρων που εργάστηκαν κατά τη διάρκεια της ημέρας και είχαν τουλάχιστον μία παραγγελία.

Η διαδικασία αυτή αν και αποτελεσματική είναι από τις πιο απαιτητικές για το σύστημα ιδιαίτερα όταν υπάρχει μεγάλο πλήθος δεδομένων και παραγγελιών στη βάση. Η εκτέλεση αυτών των υπολογισμών και η διαχείριση ενός μεγάλου όγκου δεδομένων απαιτεί ιδιαίτερη προσοχή προκειμένου να διασφαλιστεί η απόδοση και η ακεραιότητα των αποτελεσμάτων χωρίς να υπάρξουν καθυστερήσεις ή προβλήματα κατά την εκτέλεση της.

3.3.15 Ειδοποιήσεις

Στο σύστημα διαχείρισης παραγγελιών η διαθεσιμότητα των προϊόντων ελέγχεται δυναμικά ώστε να αποφεύγονται ελλείψεις και να διασφαλίζεται η ομαλή λειτουργία της επιχείρησης. Κάθε φορά που μια παραγγελία ολοκληρώνεται η ποσότητα του κάθε προϊόντος μειώνεται αυτόματα σύμφωνα με τον αριθμό των μονάδων που έχει οριστεί στην παραγγελία.

Όταν η διαθέσιμη ποσότητα ενός προϊόντος μειωθεί στο προκαθορισμένο όριο αποθέματος ή πέσει κάτω από αυτό εκτελείται ένα job το οποίο δημιουργεί μια νέα εγγραφή στον πίνακα product_warnings. Αυτός ο μηχανισμός διασφαλίζει ότι οι υπεύθυνοι διαχείρισης αποθέματος λαμβάνουν έγκαιρες ειδοποιήσεις για τα προϊόντα που πλησιάζουν σε κρίσιμα επίπεδα διαθεσιμότητας.

Για την αποφυγή περιττών και επαναλαμβανόμενων ειδοποιήσεων το σύστημα ελέγχει αν υπάρχει ήδη προειδοποίηση για το ίδιο προϊόν που δεν έχει ακόμα διαβαστεί. Σε αυτή την περίπτωση δεν δημιουργείται νέα ειδοποίηση διατηρώντας έτσι μια καθαρή και εύχρηστη λίστα ειδοποιήσεων χωρίς περιττή πληροφορία που μπορεί να οδηγήσει σε σύγχυση.

Οι ειδοποιήσεις εμφανίζονται σε δύο βασικά σημεία του συστήματος για τη διευκόλυνση των χρηστών:

- **Στο εικονίδιο του status bar** όπου ο χρήστης μπορεί άμεσα να δει ότι υπάρχουν νέες ειδοποιήσεις που απαιτούν προσοχή.
- **Στο dashboard** μέσα σε ένα εμφανές **κόκκινο πλαίσιο** ώστε οι υπεύθυνοι να έχουν άμεση ορατότητα στα προϊόντα που χρειάζονται αναπλήρωση.

Με αυτόν τον τρόπο οι διαχειριστές έχουν μια σαφή εικόνα των προϊόντων που βρίσκονται σε κρίσιμο επίπεδο διαθεσιμότητας μπορούν να προγραμματίσουν έγκαιρα νέες προμήθειες και να διασφαλίσουν ότι η επιχείρηση λειτουργεί απρόσκοπτα χωρίς ελλείψεις και καθυστερήσεις.

3.3.16 Ειδοποιήσεις και email

Το σύστημα ειδοποιήσεων περιλαμβάνει έναν μηχανισμό αποστολής email ώστε οι υπεύθυνοι να λαμβάνουν ενημερώσεις όταν ένα προϊόν φτάνει σε χαμηλό απόθεμα. Αυτό διασφαλίζει ότι η διαχείριση των αποθεμάτων γίνεται έγκαιρα αποτρέποντας ελλείψεις που μπορεί να επηρεάσουν την ομαλή λειτουργία της επιχείρησης.

Κάθε φορά που ολοκληρώνεται μια παραγγελία, το σύστημα ελέγχει αν η διαθέσιμη ποσότητα του προϊόντος έχει πέσει κάτω από το όριο που έχει οριστεί (warning_limit). Αν συμβεί αυτό, τότε:

- Ελέγχεται αν υπάρχει ήδη μια εγγραφή στον πίνακα product_warnings για το συγκεκριμένο προϊόν.
- Αν η ειδοποίηση υπάρχει αλλά έχει διαβαστεί δημιουργείται νέα εγγραφή και στέλνεται email.
- Αν δεν υπάρχει προηγούμενη ειδοποίηση δημιουργείται μία νέα και ενεργοποιείται η αποστολή email.

Για να διασφαλιστεί η σωστή διαχείριση της αποστολής email χωρίς να επηρεάζεται η απόδοση του συστήματος χρησιμοποιείται ένα job με την ονομασία SendWarningLimitEmail. Αυτό το job εκτελείται στο παρασκήνιο και αποστέλλει email σε μια λίστα παραληπτών. Οι διαχειριστές μπορούν να ορίσουν ποιοι θα λαμβάνουν τις ειδοποιήσεις μέσω email μέσα από το μενού Ρυθμίσεις > Email Επικοινωνίας. Τα emails αποθηκεύονται στον πίνακα warning_email_list επιτρέποντας εύκολη προσθήκη ή αφαίρεση παραληπτών. Για τη σωστή λειτουργία της αποστολής email πρέπει να διαμορφωθούν οι κατάλληλες ρυθμίσεις στο .env αρχείο της εφαρμογής. Ο διαχειριστής μπορεί να ορίσει τον SMTP server, το email αποστολής και τα διαπιστευτήρια.

3.4 Εφαρμογή κινητού

Η εφαρμογή κινητού αναπτύχθηκε σε Java για τηλέφωνα Android και προσφέρει μια άμεση και γρήγορη λύση σε σχέση με τη κλασσική πρόσβαση μέσω browser διευκολύνοντας την εμπειρία χρήστη.

3.4.1 Πρώτη εκκίνηση εφαρμογής

Κατά την πρώτη εκκίνηση της εφαρμογής πραγματοποιείται έλεγχος στα SharedPreferences για την ύπαρξη του κλειδιού “INITIAL_SETUP”. Αν το συγκεκριμένο κλειδί δεν είναι καταχωρημένο η εφαρμογή ενεργοποιεί αυτόματα τον Setup Wizard έναν φιλικό προς τον χρήστη οδηγό αρχικοποίησης. Ο οδηγός αυτός δίνει τη δυνατότητα στο χρήστη να εισάγει τα απαραίτητα στοιχεία για τη σύνδεση με τον διακομιστή.

Πιο συγκεκριμένα ο χρήστης μπορεί να προσθέσει είτε το domain name της ιστοσελίδας σε περίπτωση που το σύστημα φιλοξενείται σε έναν εξωτερικό web server είτε τη τοπική διεύθυνση IP του διακομιστή αν το σύστημα λειτουργεί σε τοπικό δίκτυο. Αυτή η ευελιξία εξασφαλίζει ότι η εφαρμογή είναι συμβατή τόσο με απομακρυσμένα όσο και με τοπικά περιβάλλοντα φιλοξενίας.

3.4.2 Σύνδεση σερβιτόρου

Η σύνδεση του σερβιτόρου μέσω της Android εφαρμογής υλοποιείται με τη χρήση της βιβλιοθήκης Retrofit η οποία διευκολύνει την επικοινωνία με το Laravel backend. Όταν ο σερβιτόρος εισάγει τα στοιχεία του (username και password) αυτά συσκευάζονται σε ένα αντικείμενο LoginRequest και αποστέλλονται στο backend μέσω ενός HTTP αιτήματος. Το Laravel backend αναλαμβάνει να επεξεργαστεί το αίτημα πραγματοποιώντας τον έλεγχο των διαπιστευτηρίων. Αν τα στοιχεία είναι

έγκυρα επιστρέφει ένα αντικείμενο LoginResponse που περιέχει το access token, το όνομα χρήστη και επιπλέον πληροφορίες για τον χρήστη.

Μόλις η Android εφαρμογή λάβει την επιτυχημένη απόκριση από το backend ο κώδικας στη Kotlin χειρίζεται το response εμφανίζοντας ένα μήνυμα επιτυχίας στον χρήστη μέσω ενός Toast. Το access token το οποίο είναι κρίσιμο για την ταυτοποίηση και την ασφαλή επικοινωνία με το backend για τις επόμενες ενέργειες αποθηκεύεται στις SharedPreferences της συσκευής. Αυτή η προσέγγιση επιτρέπει στην εφαρμογή να διατηρεί μια συνεχιζόμενη σύνδεση χωρίς την ανάγκη επαναλαμβανόμενων ελέγχων ταυτοποίησης εξασφαλίζοντας έτσι ότι μόνο οι εξουσιοδοτημένοι χρήστες έχουν πρόσβαση στις περαιτέρω λειτουργίες του συστήματος.

Στη συνέχεια το access token χρησιμοποιείται για τη διατήρηση της αυθεντικοποίησης στην εφαρμογή ενώ το username ανακτάται από την απόκριση για να παρέχει μια πιο εξατομικευμένη εμπειρία χρήστη. Μετά την αποθήκευση του token ο χρήστης μεταφέρεται στην οθόνη DashboardActivity όπου έχει πλήρη πρόσβαση σε όλες τις λειτουργίες της εφαρμογής παραγγελιοληψίας. Η μετάβαση γίνεται μέσω ενός Intent που μεταφέρει το όνομα χρήστη και το token στην επόμενη δραστηριότητα εξασφαλίζοντας τη συνεχιζόμενη σύνδεση και αυθεντικοποίηση.

3.4.3 Ενημέρωση δεδομένων

Μετά τη σύνδεση η εφαρμογή εκκινεί τη διαδικασία φόρτωσης και ενημέρωσης των δεδομένων από τον initialization resource του backend (Laravel). Αυτή η διαδικασία επιτρέπει στη συσκευή να λάβει τα πιο πρόσφατα δεδομένα που απαιτούνται για την ομαλή λειτουργία της εφαρμογής.

Το πρώτο βήμα περιλαμβάνει την αποστολή ενός αιτήματος στο backend για την ανάκτηση των απαραίτητων δεδομένων. Στην απόκριση το backend επιστρέφει τα δεδομένα σε μορφή JSON τα οποία περιλαμβάνουν χρήστες, κατηγορίες προϊόντων, υποκατηγορίες, χαρακτηριστικά προϊόντων και τα ίδια τα προϊόντα. Αυτά τα δεδομένα αποθηκεύονται στη βάση δεδομένων Room η οποία χρησιμοποιείται για τη τοπική αποθήκευση και γρήγορη ανάκτηση δεδομένων στο Android.

Πριν από την αποθήκευση των νέων δεδομένων η εφαρμογή κάνει ένα truncate στους πίνακες της Room δηλαδή διαγράφει όλα τα υπάρχοντα δεδομένα για να εξασφαλίσει ότι η βάση θα περιέχει μόνο τα πιο πρόσφατα και ακριβή δεδομένα από το backend. Με αυτόν τον τρόπο αποφεύγονται τυχόν παλιά ή λανθασμένα δεδομένα που θα μπορούσαν να προκαλέσουν σφάλματα ή καθυστερήσεις.

Τα δεδομένα που αποθηκεύονται περιλαμβάνουν:

- **User**: Πληροφορίες για τον χρήστη που έχει συνδεθεί όπως το όνομα και άλλες προσωπικές ρυθμίσεις.
- **Category_names**: Κατηγορίες προϊόντων.
- **Subcategory_names**: Υποκατηγορίες των προϊόντων.
- **Attributes**: Τα χαρακτηριστικά των προϊόντων που επιτρέπουν την εξατομίκευση των παραγγελιών.
- **Products**: Τα προϊόντα που είναι διαθέσιμα για παραγγελία περιλαμβάνοντας λεπτομέρειες όπως όνομα και τιμή.

Αυτή η διαδικασία εξασφαλίζει ότι η εφαρμογή θα έχει πάντα διαθέσιμα τα πιο πρόσφατα δεδομένα για να λειτουργεί απρόσκοπτα και καθιστά τη διαδικασία παραγγελιοληψίας γρήγορη και ακριβή για τον σερβιτόρο.

3.4.4 Dashboard

Το dashboard υποδέχεται τον σερβιτόρο με ένα μήνυμα καλωσορίσματος που περιλαμβάνει το όνομα χρήστη δημιουργώντας μια φιλική και προσωποποιημένη εμπειρία. Κάτω από το μήνυμα εμφανίζονται τρία βασικά στατιστικά προσφέροντας μια γρήγορη εικόνα της ημερήσιας δραστηριότητας: το συνολικό ταμείο, ο συνολικός αριθμός των παραγγελιών και οι ανοιχτές παραγγελίες που έχει ο σερβιτόρος. Αυτή η προβολή διευκολύνει τον σερβιτόρο στην παρακολούθηση των πωλήσεων και των εκκρεμών υποχρεώσεων χωρίς να χρειάζεται να περιηγηθεί σε διαφορετικά μενού.

Στην επάνω αριστερή γωνία της οθόνης, βρίσκεται ένα burger menu μέσω του οποίου ο χρήστης έχει πρόσβαση σε διάφορες λειτουργίες της εφαρμογής. Με ένα πάτημα ανοίγει η πλαϊνή μπάρα πλοήγησης η οποία περιλαμβάνει τις εξής επιλογές:

- Dashboard: Επιστροφή στην κεντρική οθόνη για άμεση πρόσβαση στα στατιστικά.
- Νέα παραγγελία: Δημιουργία και καταχώριση νέων παραγγελιών για τους πελάτες.
- Τραπέζια: Διαχείριση της κατάστασης και των παραγγελιών των τραπεζιών καθώς και παρακολούθηση διαθεσιμότητας.
- Ρυθμίσεις: Η εφαρμογή περιλαμβάνει μια ειδική σελίδα ρυθμίσεων
- Αποσύνδεση: Έξοδος από την εφαρμογή, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση στις λειτουργίες της.

Αυτός ο σχεδιασμός επιτρέπει στον σερβιτόρο να έχει ανά πάσα στιγμή μια σαφή εικόνα της δραστηριότητας του καταστήματος και να μεταβαίνει γρήγορα σε όλες τις κρίσιμες λειτουργίες, διατηρώντας την εφαρμογή εύχρηστη και αποτελεσματική.

Για την δημιουργία αυτού του dashboard, έχει υλοποιηθεί η λειτουργικότητα για την εμφάνιση του ονόματος του χρήστη, την διαχείριση του πλευρικού μενού πλοήγησης που ανοίγει με το hamburger menu, και την αυτόματη ανάκτηση και παρουσίαση των στατιστικών δεδομένων, όπως το συνολικό ταμείο, οι συνολικές και ανοιχτές παραγγελίες, μέσω κλήσεων σε ένα API. Ο κώδικας χρησιμοποιεί τα εργαλεία της Android πλατφόρμας όπως το AppCompatActivity για την δημιουργία της οθόνης και το NavigationView για το πλευρικό μενού. Επιπλέον χρησιμοποιεί ένα Toolbar για την δημιουργία της μπάρας εφαρμογών και το εικονίδιο του hamburger menu. Για την ανάκτηση δεδομένων από το backend χρησιμοποιείται η βιβλιοθήκη Retrofit κάνοντας μια ασύγχρονη κλήση σε ένα API και εμφανίζοντας τα δεδομένα στην οθόνη μόλις αυτά ανακτηθούν. Τα δεδομένα που λαμβάνονται από το API έχουν δομή η οποία ορίζεται από τις κλάσεις EmployeeDashboardResponse και EmployeeDashboard. Αυτές οι κλάσεις περιγράφουν πως η εφαρμογή αναμένει να λάβει την πληροφορία – συγκεκριμένα περιμένει ένα αντικείμενο EmployeeDashboardResponse το οποίο περιέχει μέσα του ένα EmployeeDashboard αντικείμενο. Το EmployeeDashboard αντικείμενο με τη σειρά του περιέχει ακριβώς τις τρεις τιμές που εμφανίζονται στο dashboard: το total_price, το total_orders, και το total_open_orders. Σε περίπτωση αποτυχίας φόρτωσης των δεδομένων ή προβλημάτων σύνδεσης με το API εμφανίζονται ενημερωτικά μηνύματα προς τον χρήστη εξασφαλίζοντας την καλή λειτουργία της εφαρμογής ακόμα και σε περιπτώσεις σφαλμάτων.

3.4.5 Νέα παραγγελία

Η οθόνη "Νέα Παραγγελία" παρέχει έναν εξαιρετικά διαισθητικό και γρήγορο τρόπο για την καταχώρηση παραγγελιών, που καθιστά την εμπειρία του σερβιτόρου πιο άνετη και αποτελεσματική. Κατά την είσοδο στην οθόνη, ο σερβιτόρος βλέπει αμέσως ένα καλά οργανωμένο πλέγμα με τα

διαθέσιμα τραπέζια. Κάθε τραπέζι έχει μια ευδιάκριτη ετικέτα όπως “A1”, “A2” ή “ΠΑ1” ώστε η αναγνώριση του να είναι άμεση και χωρίς καθυστερήσεις. Με ένα απλό άγγιγμα σε οποιοδήποτε τραπέζι ανοίγει το παράθυρο "Νέα Παραγγελία" δίνοντας τη δυνατότητα στον σερβιτόρο να επιλέξει γρήγορα τα προϊόντα που επιθυμεί.

Η διαδικασία παραγγελιοληψίας είναι πλήρως ιεραρχική και οργανωμένη με σαφήνεια με την αρχή να γίνεται από την επιλογή της κατηγορίας προϊόντων όπως "Καφέδες", "Ποτά" ή "Γλυκά". Κάθε κατηγορία αναπτύσσεται σε υποκατηγορίες όπως "Εσπρέσο", "Φρέσκοι Χυμοί", ή "Σοκολάτα" και στη συνέχεια παρουσιάζονται τα επιμέρους προϊόντα π.χ. "Φραπέ", "Καπουτσίνο", "Φρέσκος Χυμός Πορτοκάλι". Σε κάθε προϊόν ο σερβιτόρος μπορεί να προσαρμόσει την παραγγελία με επιλογές όπως "Με πάγο", "Με γάλα", ή "Χωρίς ζάχαρη" να ορίσει την ποσότητα και να προσθέσει οποιεσδήποτε ειδικές σημειώσεις όπως "Ελαφρύ γάλα" ή "Χωρίς καφεΐνη".

Η προσθήκη προϊόντων στην παραγγελία είναι απλή και γίνεται μέσω του κουμπιού "Προσθήκη", επιτρέποντας την εύκολη και γρήγορη ολοκλήρωση της παραγγελίας. Επίσης, η εφαρμογή παρέχει ευχέρεια για προσθήκη πολλαπλών παραλλαγών του ίδιου προϊόντος (π.χ. διαφορετική ποσότητα ή γεύση) ή προϊόντων από άλλες κατηγορίες χωρίς να χρειάζεται να επαναλάβει τη διαδικασία επιλογής κατηγοριών.

Πριν την τελική καταχώρηση της παραγγελίας η οθόνη "Σύνοψη Παραγγελίας" παρέχει μια αναλυτική προβολή όλων των επιλεγμένων προϊόντων ώστε ο σερβιτόρος να ελέγξει τη σωστή καταχώρηση των στοιχείων. Από εκεί με το πάτημα του κουμπιού "Αποστολή" η παραγγελία αποστέλλεται άμεσα στο σύστημα για επεξεργασία και καταχωρείται στην βάση δεδομένων.

Η υλοποίηση της λειτουργικότητας στηρίζεται στην δυναμική ανάκτηση δεδομένων από την ενσωματωμένη βάση δεδομένων εξασφαλίζοντας γρήγορη και αξιόπιστη πληροφορία για τα τραπέζια, τις κατηγορίες προϊόντων, τις υποκατηγορίες και τα προϊόντα. Στην πλευρά του χρήστη χρησιμοποιούνται ισχυρά Android components όπως το RecyclerView για την εμφάνιση των τραπεζιών σε μορφή “λίστας” διασφαλίζοντας γρήγορη πλοήγηση και το AlertDialog για την παρουσίαση διαλόγων επιλογής προϊόντων με μία εξαιρετικά ευέλικτη και ευχάριστη εμπειρία χρήστη.

Αυτή η υλοποίηση προσφέρει μια εξαιρετική και αποτελεσματική διαδικασία παραγγελιοληψίας, εξοικονομώντας χρόνο και μειώνοντας τα σφάλματα ενώ παράλληλα εξασφαλίζει ότι κάθε παραγγελία καταχωρείται με ακρίβεια και ταχύτητα.

3.4.6 Τραπέζια

Στην οθόνη "Τραπεζια" όπως φαίνεται στην εικόνα παρουσιάζεται ένα πλέγμα που απεικονίζει όλα τα τραπέζια του καταστήματος και τις ενεργές παραγγελίες τους. Κάθε κουμπί φέρει μια ετικέτα όπως “A1”, “A2” ή “ΠΑ1” επιτρέποντας στον υπάλληλο να εντοπίζει εύκολα το τραπέζι που θέλει να διαχειριστεί. Ο σχεδιασμός της οθόνης προσφέρει μια άμεση επισκόπηση της συνολικής αξίας των παραγγελιών σε κάθε τραπέζι διευκολύνοντας τη γρήγορη και αποδοτική διαχείριση των παραγγελιών.

Τα δεδομένα για τα τραπέζια και τις παραγγελίες τους έρχονται μέσω API από το Laravel backend. Η εφαρμογή χρησιμοποιώντας τη βιβλιοθήκη Retrofit πραγματοποιεί αιτήματα για την ανάκτηση των πληροφοριών των τραπεζιών και των παραγγελιών τους σε πραγματικό χρόνο. Αυτά τα δεδομένα περιλαμβάνουν τη λίστα των τραπεζιών, την κατάσταση κάθε παραγγελίας (π.χ., πληρωμένη ή εκκρεμής) και την τρέχουσα αξία των παραγγελιών κάθε τραπεζιού.

Για την αποτελεσματική παρουσίαση των τραπεζιών και των παραγγελιών χρησιμοποιείται το RecyclerView το οποίο επιτρέπει την αποδοτική αναπαράσταση μεγάλων συνόλων δεδομένων. Κάθε τραπέζι απεικονίζεται σε ένα στοιχείο της λίστας, το οποίο περιλαμβάνει τις πληροφορίες για την τρέχουσα κατάσταση των παραγγελιών. Κάθε κουμπί τραπεζιού, όπως “Α1”, “Α2”, ή “ΠΑ1”, ενσωματώνει τις αντίστοιχες πληροφορίες παραγγελιών και της αξίας τους εξασφαλίζοντας έτσι γρήγορη και ευέλικτη πλοήγηση. Ο RecyclerView.Adapter χειρίζεται τα δεδομένα και τα εμφανίζει δυναμικά στην οθόνη ενημερώνοντας τα πάντα χωρίς να απαιτείται ανανέωση ολόκληρης της οθόνης.

Μόλις ο υπάλληλος επιλέξει ένα τραπέζι αποστέλλεται ένα νέο αίτημα στο backend μέσω του API για να ληφθούν αναλυτικές πληροφορίες για το συγκεκριμένο τραπέζι όπως το σύνολο της αξίας όλων των παραγγελιών. Τα δεδομένα αυτά επιστρέφουν μέσω της απάντησης του API και η εφαρμογή τα χειρίζεται ανάλογα εμφανίζοντας τα στην οθόνη "Διαχείριση Τραπεζιού". Η λογική για την εμφάνιση των στοιχείων περιλαμβάνει τη διαχείριση της κατάστασης της παραγγελίας, της αξίας της και άλλων σχετικών πληροφοριών. Σε περίπτωση αποτυχίας της κλήσης, εμφανίζεται κατάλληλο μήνυμα λάθους για να ενημερωθεί ο χρήστης για οποιοδήποτε πρόβλημα στην επικοινωνία με το API.

Τα δεδομένα περιλαμβάνουν, για παράδειγμα: τη Σύνολο Αξίας Παραγγελιών για κάθε τραπέζι, η οποία ανανεώνεται δυναμικά όταν υπάρχουν νέες παραγγελίες ή ακυρώσεις καθώς και στοιχεία παραγγελιών για κάθε προϊόν στο τραπέζι (π.χ., όνομα προϊόντος, ποσότητα και κατάσταση της παραγγελίας).

Έτσι, με τη χρήση του RecyclerView και της επικοινωνίας μέσω API διασφαλίζεται η ομαλή λειτουργία της οθόνης "Τραπεζια" προσφέροντας στον υπάλληλο πλήρη έλεγχο της διαδικασίας από την παρακολούθηση της συνολικής αξίας μέχρι την πληρωμή, τη διαγραφή ή το κλείσιμο του τραπεζιού. Με αυτόν τον τρόπο διασφαλίζεται μια γρήγορη αποδοτική και οργανωμένη εξυπηρέτηση των πελατών.

3.4.7 Υποστήριξη πολυγλωσσίας

Η υποστήριξη πολυγλωσσίας στην εφαρμογή επιτυγχάνεται μέσω του αρχείου strings.xml το οποίο αποτελεί το βασικό εργαλείο για τη διαχείριση και την αποθήκευση όλων των μεταφράσεων που απαιτούνται για την εφαρμογή. Με την αποθήκευση των κειμένων της εφαρμογής σε αυτό το αρχείο η εφαρμογή καθίσταται εύκολα μεταφράσιμη σε πολλές γλώσσες χωρίς να χρειάζεται να γίνουν αλλαγές στον κώδικα της εφαρμογής.

Το strings.xml είναι ένα αρχείο το οποίο περιέχει όλα τα στατικά κείμενα της εφαρμογής όπως κουμπιά, ειδοποιήσεις, ετικέτες και μηνύματα. Για κάθε γλώσσα που θέλουμε να υποστηρίξουμε δημιουργούμε ένα αντίστοιχο αρχείο strings.xml στο φάκελο της κάθε γλώσσας. Για παράδειγμα, το αρχείο για τα αγγλικά θα βρίσκεται στον φάκελο res/values-en/strings.xml, για τα ελληνικά στον φάκελο res/values-el/strings.xml και ούτω καθεξής.

Κάθε φάκελος περιέχει το ίδιο σύνολο από κλειδιά (keys) για τα κείμενα της εφαρμογής αλλά με διαφορετικές τιμές (τις μεταφράσεις για κάθε γλώσσα). Όταν ο χρήστης αλλάζει τη γλώσσα της συσκευής του το Android σύστημα φορτώνει αυτόματα το σωστό αρχείο strings.xml που αντιστοιχεί στη γλώσσα του χρήστη και εμφανίζει τα κείμενα στην κατάλληλη γλώσσα.

Αυτή η μέθοδος επιτρέπει στην εφαρμογή να υποστηρίζει πολλές γλώσσες με ελάχιστο κόστος συντήρησης καθώς οι μεταφράσεις είναι απομονωμένες από τον κώδικα και διαχειρίζονται εύκολα μέσω των αντίστοιχων αρχείων γλώσσας.

3.4.8 Ρυθμίσεις

Η εφαρμογή περιλαμβάνει μια ειδική σελίδα Ρυθμίσεων η οποία προσφέρει στον χρήστη την ευχέρεια να προσαρμόσει ορισμένες βασικές παραμέτρους της εφαρμογής. Μέσω αυτής της σελίδας ο χρήστης μπορεί να αλλάξει τη διεύθυνση URL που καταχωρήθηκε κατά τη διάρκεια της διαδικασίας του Setup Wizard. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη σε περιπτώσεις που αλλάξει το domain ή η τοποθεσία φιλοξενίας της εφαρμογής επιτρέποντας στον χρήστη να ενημερώσει τη σύνδεση με το backend χωρίς να απαιτείται επαναφορά ή επανεγκατάσταση της εφαρμογής.

Επιπλέον η σελίδα ρυθμίσεων παρέχει πληροφορίες για την τρέχουσα έκδοση της εφαρμογής επιτρέποντας στον χρήστη να ελέγξει εάν είναι εγκατεστημένη η πιο πρόσφατη και ενημερωμένη έκδοση. Αυτή η λειτουργία είναι σημαντική για να διασφαλίζεται ότι η εφαρμογή παραμένει πάντα ενημερωμένη με τις τελευταίες βελτιώσεις και διορθώσεις σφαλμάτων.

Η σελίδα ρυθμίσεων υλοποιείται ως Fragment το οποίο παρέχεται από το Android Studio και ενσωματώνεται ομαλά στην εφαρμογή. Το Fragment αυτό είναι εύκολα επεκτάσιμο και παραμετροποιήσιμο επιτρέποντας την προσθήκη νέων ρυθμίσεων ή την τροποποίηση των υπαρχουσών ώστε να ανταποκρίνεται στις ανάγκες της εφαρμογής και των χρηστών της.

3.5 Υλοποίηση Συστήματος σε Web Server ή Τοπικό Επίπεδο

Το σύστημα που αναπτύχθηκε προσφέρει ευελιξία και μπορεί να εγκατασταθεί είτε σε έναν απομακρυσμένο web server είτε σε τοπικό επίπεδο χρησιμοποιώντας ένα Raspberry Pi, ανάλογα με τις ανάγκες και την υποδομή του χρήστη. Η επιλογή της πλατφόρμας εξαρτάται από παράγοντες όπως η διαθεσιμότητα πόρων, οι απαιτήσεις πρόσβασης, αλλά και το επίπεδο ελέγχου και ασφάλειας που επιθυμείται.

Προσφέρει τα εξής πλεονεκτήματα:

- Επιτρέπει την απομακρυσμένη πρόσβαση στο σύστημα από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο. Ευκολία πρόσβασης από οποιαδήποτε τοποθεσία.
- Επεκτασιμότητα και δυνατότητα υποστήριξης πολλαπλών χρηστών ταυτόχρονα.
- Διαχείριση backup μέσω εργαλείων που προσφέρει ο πάροχος του server.

Για μικρές επιχειρήσεις ή χρήστες που επιθυμούν μεγαλύτερο έλεγχο στο σύστημα, το Raspberry Pi μπορεί να αποτελέσει μια οικονομική και ευέλικτη λύση για τοπική υλοποίηση αν δεν επιθυμεί το κατάστημα να επενδύσει πολλά χρήματα σε υπολογιστή.

Προσφέρει τα εξής πλεονεκτήματα

- Απόλυτος έλεγχος του συστήματος χωρίς εξάρτηση από τρίτους παρόχους.
- Χαμηλό κόστος υλοποίησης.
- Κατάλληλο για τοπικά δίκτυα χωρίς ανάγκη για συνεχή σύνδεση στο διαδίκτυο.

Συγκρίνοντας και τους δύο τρόπους καταλήγουμε στο συμπέρασμα ότι ο web server συνιστάται για επιχειρήσεις που απαιτούν πρόσβαση από πολλαπλές τοποθεσίες και έχουν αυξημένες ανάγκες απομακρυσμένης διαχείρισης ενώ η τοπική λύση είναι ιδανική για μικρές επιχειρήσεις ή και ως εφεδρική λύση για την αποφυγή εξάρτησης από το διαδίκτυο.

3.6 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια λεπτομερής περιγραφή για την εγκατάσταση, τις λειτουργίες και την υλοποίηση του συστήματος. Το σύστημα περιλαμβάνει μια σειρά από λειτουργίες που εξασφαλίζουν την αποτελεσματική διαχείριση δεδομένων και τη βελτιστοποίηση των διαδικασιών. Αρχικά κατά την

εκκίνηση πραγματοποιείται φόρτωση των απαραίτητων δεδομένων. Τα δεδομένα αυτά αποθηκεύονται στο Pinia store, προσφέροντας γρήγορη και εύκολη πρόσβαση σε αυτά.

Η σύνδεση του χρήστη γίνεται μέσω εισαγωγής των στοιχείων του τα οποία ελέγχονται για την εγκυρότητά τους. Εφόσον είναι σωστά δημιουργείται session και token αυθεντικοποίησης διασφαλίζοντας την ασφαλή πρόσβαση στο σύστημα. Το dashboard του συστήματος προσαρμόζεται ανάλογα με τον ρόλο του χρήστη προσφέροντας μια εξατομικευμένη εμπειρία. Για παράδειγμα οι διαχειριστές έχουν τη δυνατότητα να βλέπουν ειδοποιήσεις, στατιστικά της ημέρας (όπως έσοδα και παραγγελίες), καθώς και πληροφορίες σχετικά με τους σερβιτόρους.

Οι παραγγελίες παρουσιάζονται με λεπτομερή προβολή δυνατότητα διαγραφής και φίλτρα βάσει κατάστασης. Υπάρχει επίσης ξεχωριστή ενότητα για διαγραφές προϊόντων και παραγγελιών όπου παρέχονται σχετικές πληροφορίες και δυνατότητα φιλτραρίσματος βάσει ημερομηνίας. Η δημιουργία νέας παραγγελίας είναι μια διαδικασία δύο καρτών. Η πρώτη κάρτα εμφανίζει σύνοψη της παραγγελίας ενώ η δεύτερη επιτρέπει την επιλογή τραπεζιού, κατηγορίας, υποκατηγορίας και προϊόντος. Οι χρήστες μπορούν να προσθέσουν ποσότητες, χαρακτηριστικά και σημειώσεις στην παραγγελία, η οποία αποθηκεύεται προσωρινά στο Pinia μέχρι να σταλεί ή να λήξει το session.

Στο μπαρ οι παραγγελίες εμφανίζονται σε μορφή καρτών με βασικές πληροφορίες και κουμπιά "Αποδοχή" ή "Απόρριψη". Η απόρριψη οδηγεί σε διαγραφή της παραγγελίας ενώ η αποδοχή αλλάζει την κατάσταση της παραγγελίας σε "completed" και εκτελείται job για την αφαίρεση των ποσοτήτων από το απόθεμα. Η διαχείριση των τραπεζιών περιλαμβάνει την προβολή τους με χρωματική κωδικοποίηση ανάλογα με την κατάστασή τους καθώς και τη δυνατότητα διαγραφής, μεταφοράς ή κλεισίματός τους.

Η ενότητα των σερβιτόρων παρέχει πληροφορίες για κάθε σερβιτόρο όπως οι ημερομηνίες εργασίας και το ταμείο ενώ δίνεται και η δυνατότητα διαγραφής ή προβολής λεπτομερειών. Στατιστικά στοιχεία όπως ημερήσια δεδομένα, στατιστικά σερβιτόρων και προϊόντων μπορούν να προβληθούν με δυνατότητα φιλτραρίσματος βάσει ημερομηνίας, εξασφαλίζοντας πλήρη εικόνα της απόδοσης.

Η διαχείριση της αποθήκης περιλαμβάνει την οργάνωση κατηγοριών, υποκατηγοριών, προϊόντων, χαρακτηριστικών και αποθεμάτων. Παράλληλα η ενότητα των ρυθμίσεων επιτρέπει τη διαχείριση του κωδικού πρόσβασης του λογαριασμού χρήστη, των ρόλων, των υπευθύνων, των δικαιωμάτων και των email επικοινωνίας. Επιπλέον το περιεχόμενο του frontend περιορίζεται ανάλογα με τα δικαιώματα του χρήστη, τα οποία αποθηκεύονται στο Pinia.

Τέλος, κατά το κλείσιμο της ημέρας το σύστημα φροντίζει για το κλείσιμο όλων των ανοιχτών παραγγελιών, τον υπολογισμό των εσόδων, την καταγραφή του πλήθους των παραγγελιών και την ομαδοποίησή τους ανά σερβιτόρο, προσφέροντας μια ολοκληρωμένη αναφορά της ημέρας.

4. Σχεδιασμός Βάσης Δεδομένων

4.1 Εισαγωγή

Η βάση δεδομένων αποτελεί τον πυρήνα του συστήματος παραγγελιοληψίας και διαχείρισης αποθήκης. Σκοπός της είναι η οργάνωση και αποθήκευση όλων των δεδομένων που σχετίζονται με τις παραγγελίες, τα προϊόντα, τα αποθέματα, τους χρήστες και τις λειτουργίες του συστήματος, εξασφαλίζοντας την ορθότητα, ακεραιότητα και αποτελεσματικότητα στην επεξεργασία των δεδομένων.

Κατά το σχεδιασμό της βάσης δεδομένων λήφθηκαν υπόψη οι βασικές ανάγκες του συστήματος:

- Αποθήκευση πληροφοριών για τα προϊόντα (όνομα, τιμή, ποσότητα, κατηγορία).
- Καταγραφή παραγγελιών (ημερομηνία, κατάσταση, συσχετισμός με τραπέζια και σερβιτόρους).
- Παρακολούθηση αποθεμάτων για τη σωστή διαχείριση της αποθήκης.
- Καταγραφή χρηστών του συστήματος (διαχειριστές, σερβιτόροι, υπεύθυνοι αποθήκης) και των ρόλων τους.
- Εξασφάλιση ακεραιότητας των δεδομένων με σχέσεις μεταξύ πινάκων.

4.2 Περιγραφή πινάκων

Παρακάτω αναλύονται οι πιο βασικοί πίνακες, οι οποίοι αποτελούν τον πυρήνα της λειτουργικότητας.

4.2.1 Πίνακας χρηστών (Users)

Ο πίνακας Users διαχειρίζεται τα στοιχεία των χρηστών του συστήματος, συμπεριλαμβανομένων των διαχειριστών, σερβιτόρων και υπευθύνων.

Πεδία και Περιγραφή

- **id (Primary Key):** Είναι ο μοναδικός αναγνωριστικός αριθμός κάθε χρήστη. Αυτό το πεδίο είναι ακέραιος αριθμός και δημιουργείται αυτόματα με αύξουσα αρίθμηση.
- **deleted_at:** Είναι ένα πεδίο για τη διαχείριση λογικών διαγραφών (soft deletes). Εάν η τιμή είναι null, η εγγραφή θεωρείται ενεργή. Διαφορετικά, περιέχει την ημερομηνία και ώρα που ο χρήστης έχει διαγραφεί λογικά από το σύστημα.
- **last_login_date:** Καταγράφει την τελευταία ημερομηνία και ώρα που ο χρήστης συνδέθηκε στο σύστημα.
- **username:** Ένα μοναδικό πεδίο που περιέχει το όνομα χρήστη. Το πεδίο αυτό διευκολύνει την ταυτοποίηση χρηστών κατά τη σύνδεση.
- **firstname και lastname:** Αυτά τα πεδία αποθηκεύουν το όνομα και το επώνυμο του χρήστη, αντίστοιχα, και χρησιμοποιούνται για την προσωπική αναγνώριση και την εμφάνιση του ονόματος στο frontend.
- **email:** Περιέχει τη διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη.
- **phone:** Αποθηκεύει τον αριθμό τηλεφώνου του χρήστη. Το πεδίο είναι προαιρετικό και επιτρέπει μεγαλύτερη ευελιξία στην καταγραφή στοιχείων.
- **role_id:** Πρόκειται για ένα ακέραιο πεδίο που συνδέεται με τον πίνακα Roles, αντιπροσωπεύοντας τον ρόλο του χρήστη (π.χ., διαχειριστής, σερβιτόρος, υπεύθυνος

αποθήκης). Η σχέση αυτή είναι one-to-many, όπου πολλοί χρήστες μπορούν να έχουν τον ίδιο ρόλο.

- **password:** Περιέχει τον κρυπτογραφημένο κωδικό πρόσβασης του χρήστη, εξασφαλίζοντας την ασφάλεια του συστήματος.

4.2.2 Πίνακας δεδομενων χρηστών (user_data)

Ο πίνακας user_data είναι σχεδιασμένος για την αποθήκευση επιπλέον στοιχείων σχετικά με τους χρήστες του συστήματος τα οποία σχετίζονται τα έσοδα. Είναι χρήσιμος για την παρακολούθηση των εσόδων που έχει συγκεντρώσει κάθε σερβιτόρο2, με στόχο την αποδοτική διαχείριση των εσόδων και των επιδόσεων. Σε αυτό το πίνακα αποθηκεύονται μόνο δεδομένα απο χρήστες που έχουν ρόλο σερβιτόρος. (Σχήμα 4.1)

- **id (Primary Key):** Ο μοναδικός αναγνωριστικός αριθμός για κάθε εγγραφή στον πίνακα. Αυτό το πεδίο είναι ακέραιο και δημιουργείται αυτόματα με αύξουσα αρίθμηση διασφαλίζοντας τη μοναδικότητα της εγγραφής.
- **timestamps:** Δημιουργούνται δύο στήλες: created_at και updated_at, που καταγράφουν την ημερομηνία και ώρα δημιουργίας και τελευταίας τροποποίησης της εγγραφής, αντίστοιχα. Η στήλη created_at είναι χρήσιμη γιατί καταγράφει την ημέρα που ο σερβιτόρος έστειλε κάποια παραγγελία.
- **user_id:** Ακέραιο πεδίο που αναφέρεται στον μοναδικό αναγνωριστικό αριθμό του χρήστη από τον πίνακα users. Συνδέει τα δεδομένα οικονομικών κερδών με συγκεκριμένο χρήστη.
- **income:** Πεδίο που καταγράφει τα έσοδα του σερβιτόρου εκείνης της ημέρας συνολικά. Είναι τύπου double για να εξασφαλίζεται η ακριβής αποθήκευση αριθμητικών δεδομένων με δεκαδική ακρίβεια.

4.2.3 Πίνακας παραγγελιών (orders)

Ο πίνακας orders είναι σχεδιασμένος για την αποθήκευση πληροφοριών που σχετίζονται με τις παραγγελίες που πραγματοποιούνται στο κατάστημα. Κάθε παραγγελία συνδέεται με έναν χρήστη (π.χ., σερβιτόρος) και ένα τραπέζι, και καταγράφονται οι σχετικές λεπτομέρειες της παραγγελίας, όπως η τιμή και η κατάσταση. (Σχήμα 4.2)

Ο πίνακας περιέχει τα εξής πεδία:

- **id (Primary Key):** Ο μοναδικός αναγνωριστικός αριθμός για κάθε παραγγελία στον πίνακα. Είναι τύπου auto-increment και εξασφαλίζει ότι κάθε εγγραφή παραγγελίας έχει έναν μοναδικό αριθμό.
- **timestamps:** Δημιουργούνται οι στήλες created_at και updated_at που καταγράφουν την ημερομηνία και ώρα δημιουργίας και τελευταίας τροποποίησης της παραγγελίας. Αυτά τα δεδομένα είναι χρήσιμα για τον εντοπισμό της χρονικής στιγμής κατά την οποία έγινε η παραγγελία και τυχόν τροποποιήσεις της.
- **deleted_at:** Υποστηρίζει τη λογική διαγραφή (soft delete). Όταν η τιμή είναι null, η παραγγελία θεωρείται ενεργή. Όταν η παραγγελία διαγράφεται λογικά η ημερομηνία και ώρα της διαγραφής καταγράφεται σε αυτό το πεδίο.
- **table_id:** Ακέραιο πεδίο που αναφέρεται στον αναγνωριστικό αριθμό του τραπεζιού για το οποίο έγινε η παραγγελία. Αυτό το πεδίο χρησιμοποιείται για να συσχετίσει μια παραγγελία με το συγκεκριμένο τραπέζι στο κατάστημα.

- **user_id:** Ακέραιο πεδίο που αναφέρεται στον αναγνωριστικό αριθμό του χρήστη (π.χ., σερβιτόρου) που δημιούργησε την παραγγελία. Αυτό το πεδίο συνδέει τις παραγγελίες με τους χρήστες του συστήματος και είναι χρήσιμο για τη διαχείριση του προσωπικού.
- **price:** Πεδία τύπου float για την καταγραφή της τιμής της παραγγελίας. Εάν δεν έχει καθοριστεί τιμή, το πεδίο επιτρέπει την τιμή NULL. Η τιμή αυτή μπορεί να περιλαμβάνει το σύνολο της παραγγελίας ή τυχόν εκπτώσεις/προσαυξήσεις που εφαρμόζονται.
- **status:** Πεδίο τύπου enum, το οποίο ορίζει την κατάσταση της παραγγελίας. Η κατάσταση μπορεί να είναι μία από τις εξής:
    - pending: Η παραγγελία είναι σε εκκρεμότητα και δεν έχει ολοκληρωθεί.
    - completed: Η παραγγελία έχει ολοκληρωθεί και είναι έτοιμη.
    - paid: Η παραγγελία έχει πληρωθεί από τον πελάτη.
    - denied: Η παραγγελία έχει απορριφθεί.
    - closed: Η παραγγελία έχει κλείσει, αφορά την ολοκλήρωση της παραγγελίας που πραγματοποιείται όταν κλείσει το τραπέζι.

4.2.4 Πίνακας προιόντων παραγγελίας (order_items)

Ο πίνακας order_items είναι σημαντικός για την καταγραφή των ατομικών στοιχείων κάθε παραγγελίας. Στην ουσία, κάθε εγγραφή στον πίνακα αναφέρεται σε ένα μεμονωμένο προϊόν που περιλαμβάνεται σε μια παραγγελία, καθώς και στις σχετικές πληροφορίες που συνδέονται με το προϊόν αυτό όπως η ποσότητα, η τιμή, τυχόν σημειώσεις και τα χαρακτηριστικά του προϊόντος. Αυτός ο πίνακας συνδέει τα προϊόντα με τις παραγγελίες και επιτρέπει την καταγραφή λεπτομερών στοιχείων για κάθε προϊόν ξεχωριστά. (Σχήμα 4.3)

Ο πίνακας περιέχει τα εξής πεδία:

- **id (Primary Key):** Ο μοναδικός αναγνωριστικός αριθμός για κάθε στοιχείο παραγγελίας στον πίνακα. Η τιμή αυξάνεται αυτόματα με κάθε νέα εγγραφή και εγγυάται ότι κάθε στοιχείο παραγγελίας έχει μοναδική αναγνώριση.
- **deleted_at:** Πεδία τύπου timestamp που επιτρέπει την λογική διαγραφή του στοιχείου παραγγελίας. Όταν το πεδίο είναι null, το στοιχείο θεωρείται ενεργό. Όταν η τιμή του είναι μια ημερομηνία/ώρα, τότε το στοιχείο έχει διαγραφεί λογικά και δεν εμφανίζεται στην εφαρμογή, αλλά παραμένει στη βάση δεδομένων για ιστορικούς σκοπούς.
- **order_id:** Ακέραιο πεδίο που αναφέρεται στον αναγνωριστικό αριθμό της παραγγελίας στην οποία ανήκει το συγκεκριμένο στοιχείο παραγγελίας. Το πεδίο αυτό δημιουργεί μια σύνδεση μεταξύ του πίνακα order_items και του πίνακα orders που καταγράφεται η παραγγελία στην οποία ανήκει το προϊόν.
- **product_id:** Ακέραιο πεδίο που αναφέρεται στον αναγνωριστικό αριθμό του προϊόντος που περιλαμβάνεται στην παραγγελία. Σύνδεση με τον πίνακα products όπου καταγράφονται οι λεπτομέρειες για τα προϊόντα που διατίθενται στο κατάστημα.
- **quantity:** Ακέραιο πεδίο που καθορίζει την ποσότητα του προϊόντος που παραγγέλθηκε. Χρησιμοποιείται για να καταγραφεί πόσα τεμάχια του συγκεκριμένου προϊόντος περιλαμβάνονται στην παραγγελία.
- **price:** Πεδίο τύπου float που καταγράφει την τιμή του προϊόντος στην παραγγελία. Η τιμή αυτή μπορεί να διαφέρει από την κανονική τιμή του προϊόντος αν υπάρχουν εκπτώσεις ή προσαυξήσεις για την παραγγελία. Αν η τιμή δεν έχει οριστεί, επιτρέπεται η τιμή NULL.

- **note:** Πεδίο τύπου text που επιτρέπει την αποθήκευση σημειώσεων που σχετίζονται με το προϊόν στην παραγγελία. Για παράδειγμα αν υπάρχει κάποια ειδική επιθυμία του πελάτη ή κάποια άλλη πληροφορία που πρέπει να καταγραφεί για το προϊόν.
- **attributes:** Πεδίο τύπου text που επιτρέπει την αποθήκευση χαρακτηριστικών του προϊόντος. Αυτό μπορεί να περιλαμβάνει μεταβλητά χαρακτηριστικά ή επιλογές προϊόντων, όπως μέγεθος, χρώμα ή άλλες προσαρμοσμένες παραμέτρους.

4.2.5 Πίνακες του συστήματος αποθήκης

Η αποθήκη εξασφαλίζει ότι οι παραγγελίες μπορούν να εξυπηρετηθούν με τα κατάλληλα προϊόντα και με τη σωστή ποσότητα. Για να κατανοήσουμε πλήρως τη λειτουργία του συστήματος αποθήκης είναι απαραίτητο να αναλύσουμε όλους τους πίνακες που σχετίζονται με τη διαχείριση των προϊόντων και των αποθεμάτων. Αυτοί οι πίνακες περιλαμβάνουν τις πληροφορίες για τα προϊόντα, τις κατηγορίες προϊόντων, τις υποκατηγορίες, τα χαρακτηριστικά, και τα αποθέματα.

Οι πίνακες categories και subcategories συνεργάζονται για την κατηγοριοποίηση και οργάνωση των προϊόντων. Η δομή αυτή εξασφαλίζει την ιεραρχική οργάνωση των προϊόντων σε κατηγορίες και υποκατηγορίες, διευκολύνοντας την αναζήτηση και την ομαδοποίηση των προϊόντων ανάλογα με την κατηγορία τους. Και οι δύο πίνακες περιέχουν τα ίδια σχεδόν χαρακτηριστικά οπότε θα τους αναλύσουμε μαζί: (Σχήμα 4.4)

- **id (Primary Key):** Μοναδικό αναγνωριστικό για κάθε κατηγορία ή υποκατηγορία.
- **name:** Το όνομα της κατηγορίας (π.χ. "Ποτά") ή της υποκατηγορίας (π.χ. “Gin”).
- **timestamps:** Στοιχεία για την ημερομηνία δημιουργίας και τελευταίας ενημέρωσης της εγγραφής.
- **category_id:** Ακέραιο πεδίο που αναφέρεται στον αναγνωριστικό αριθμό του της κύρια κατηγορία που ανοίκει αυτή η υποκατηγορία. Αυτό το πεδίο το συναντάμε μόνο στο πίνακα subcategories.

Ο πίνακας products καταγράφει τα προϊόντα που διαχειρίζεται το σύστημα αποθήκης και συνδέεται άμεσα με τις υποκατηγορίες των προϊόντων μέσω του πεδίου subcategory_id. Αυτός ο πίνακας περιλαμβάνει σημαντικά πεδία που βοηθούν στην παρακολούθηση, τιμολόγηση και απογραφή των προϊόντων διασφαλίζοντας τη σωστή διαχείριση των αποθεμάτων. Πιο αναλυτικά: (Σχήμα 4.5)

- **id**: Το μοναδικό αναγνωριστικό για κάθε προϊόν. Κάθε προϊόν καταχωρείται με ένα μοναδικό ID που χρησιμοποιείται για την αναγνώριση και αναφορά του σε άλλους πίνακες του συστήματος.
- **timestamps**: Καταγράφει την ημερομηνία και ώρα δημιουργίας και τελευταίας ενημέρωσης του προϊόντος. Αυτά τα δεδομένα είναι χρήσιμα για την παρακολούθηση των αλλαγών στα προϊόντα.
- **deleted_at**: Επιτρέπει την εφαρμογή της λειτουργικότητας "soft delete" όπου το προϊόν δεν διαγράφεται από τη βάση δεδομένων αλλά απλώς σημαίνεται ως διαγραμμένο. Αυτό επιτρέπει την αναίρεση της διαγραφής σε περίπτωση λάθους ή αν χρειαστεί για λόγους αναφοράς.
- **name**: Το όνομα του προϊόντος το οποίο χρησιμοποιείται για να αναγνωρίσει το προϊόν στον κατάλογο και να γίνει αναφορά σε αυτό σε παραγγελίες και άλλες ενέργειες του συστήματος.
- **quantity**: Η διαθέσιμη ποσότητα του προϊόντος στο απόθεμα. Αυτό το πεδίο είναι σημαντικό για την παρακολούθηση των αποθεμάτων.
- **warning_limit**: Το όριο προειδοποίησης για την ποσότητα του προϊόντος. Αν η ποσότητα πέσει κάτω από το όριο αυτό το σύστημα μπορεί να ενεργοποιήσει μια προειδοποίηση για την έλλειψη αποθέματος.

- **price**: Η τιμή του προϊόντος. Αυτό το πεδίο καθορίζει την τιμή πώλησης του προϊόντος και είναι απαραίτητο για την τιμολόγηση των παραγγελιών.
- **subcategory_id**: Το αναγνωριστικό της υποκατηγορίας στην οποία ανήκει το προϊόν. Αυτό το πεδίο συνδέει το προϊόν με τη συγκεκριμένη υποκατηγορία του.

Ο πίνακας attributes καταγράφει τα χαρακτηριστικά που σχετίζονται με τα προϊόντα στο σύστημα αποθήκης. Τα χαρακτηριστικά μπορούν να είναι προσθήκες ή παραλλαγές προϊόντων ή άλλες ιδιότητες που επηρεάζουν την τιμή ή την κατηγοριοποίηση του προϊόντος. (Σχήμα 4.6)

Ο πίνακας attributes περιέχει τα εξής πεδία:

- **id**: Το μοναδικό αναγνωριστικό για κάθε χαρακτηριστικό.
- **timestamps**: Καταγράφει την ημερομηνία και ώρα δημιουργίας και τελευταίας ενημέρωσης του χαρακτηριστικού.
- **name**: Το όνομα του χαρακτηριστικού.
- **price**: Η τιμή του χαρακτηριστικού, αν υπάρχει. Εδώ μπορεί να καταχωρηθεί η τιμή που προστίθεται στο βασικό προϊόν αν το χαρακτηριστικό επιλεγεί. Αν δεν υπάρχει επιπλέον κόστος για το χαρακτηριστικό, η τιμή μπορεί να παραμείνει null.
- **category_id**: Το αναγνωριστικό της κατηγορίας στην οποία ανήκει το χαρακτηριστικό. Αυτό το πεδίο συνδέει το χαρακτηριστικό με μια συγκεκριμένη κατηγορία και επιτρέπει την ομαδοποίηση των χαρακτηριστικών με βάση τις κατηγορίες προϊόντων.

Τελευταίος πίνακας για τη διαχείριση της αποθήκης είναι ο inventory_deductions. Ο πίνακας inventory_deductions καταγράφει τη ποσότητα που θα αφαίρείται από το απόθεμα προϊόντων, γεγονός που είναι χρήσιμο για τη διαχείριση των πωλήσεων. (Σχήμα 4.7)

Πεδία του πίνακα inventory_deductions

- **id**: Το μοναδικό αναγνωριστικό για κάθε καταγραφή αφαίρεσης από το απόθεμα.
- **product_id**: Το αναγνωριστικό του προϊόντος από το οποίο αφαιρείται ποσότητα. Αυτό το πεδίο συνδέει την αφαίρεση με ένα συγκεκριμένο προϊόν στον πίνακα products επιτρέποντας τον προσδιορισμό του προϊόντος που επηρεάζεται από την αφαίρεση.
- **deduction_number**: Η ποσότητα που αφαιρείται από το απόθεμα του προϊόντος. Αυτή η τιμή δηλώνει την ακριβή ποσότητα του προϊόντος που μειώθηκε από το απόθεμα. Μπορεί να είναι ένας ακέραιος ή δεκαδικός αριθμός ανάλογα με τις ανάγκες του συστήματος.

4.2.6 Πίνακες δικαιωμάτων

Η διαχείριση των δικαιωμάτων σε ένα σύστημα είναι κρίσιμη για την ασφάλειας του και τη σωστή λειτουργία του. Στο συγκεκριμένο σύστημα, οι πίνακες roles, permissions και permission_role συνεργάζονται για να καθορίσουν ποια δικαιώματα έχει κάθε χρήστης, ανάλογα με τον ρόλο του.

Ο πίνακας roles περιέχει τους διάφορους τύπους ρόλων που μπορεί να έχει ένας χρήστης μέσα στο σύστημα. Κάθε ρόλος αντιπροσωπεύει μία ομάδα χρηστών με κοινές εξουσιοδοτήσεις και δικαιώματα. Για παράδειγμα, ρόλοι όπως διαχειριστής, σερβιτόρος, ή λογιστής καθορίζουν τι μπορεί να κάνει ο χρήστης με αυτόν τον ρόλο.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε ρόλο στο σύστημα.

- **timestamps**: Τα πεδία **created_at** και **updated_at** καταγράφουν την ημερομηνία δημιουργίας και τελευταίας ενημέρωσης του ρόλου.
- **name**: Το όνομα του ρόλου, όπως "Διαχειριστής", "Σερβιτόρος" κ.ά.
- **slug**: Μια σύντομη, μοναδική αναπαράσταση του ονόματος του ρόλου.

Ο πίνακας permissions περιγράφει τα διάφορα δικαιώματα ή ενέργειες που μπορούν να εκτελούν οι χρήστες όπως η δημιουργία παραγγελιών, η προβολή στατιστικών και άλλα. Κάθε δικαίωμα συνδέεται με συγκεκριμένες ενέργειες στο σύστημα που επηρεάζουν τη ροή της εργασίας και τη δυνατότητα πρόσβασης σε διάφορες περιοχές του συστήματος.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε δικαίωμα στο σύστημα.
- **group**: Κατηγοριοποιεί τα δικαιώματα σε ομάδες, για παράδειγμα, "Διαχείριση Παραγγελιών", "Προβολή Στατιστικών" κ.λπ. Αυτό βοηθά στην καλύτερη και πιο ξεκάθαρη προβολή των δικαιωμάτων στο frontend.
- **name**: Το όνομα του δικαιώματος που περιγράφει την ενέργεια που επιτρέπεται, όπως "Δημιουργία Παραγγελίας" κ.ά.
- **slug**: Μια σύντομη μοναδική αναπαράσταση του ονόματος του δικαιώματος η οποία χρησιμοποιείται για την αναγνώριση του δικαιώματος στο σύστημα. Το πεδίο αυτό είναι μοναδικό και βοηθά στην αποφυγή συγκρούσεων.

Σχέση Ρόλων και Δικαιωμάτων (permission_role pivot table)

Η διασύνδεση μεταξύ των ρόλων και των δικαιωμάτων γίνεται μέσω του pivot πίνακα permission_role (Σχήμα 4.8) ο οποίος συνδέει τους ρόλους με τα αντίστοιχα δικαιώματα. Κάθε καταχώρηση στον πίνακα αυτό αναπαριστά τη σύνδεση ενός ρόλου με ένα δικαίωμα, επιτρέποντας την ευέλικτη και κλιμακούμενη χορήγηση δικαιωμάτων στους χρήστες ανάλογα με τον ρόλο τους.

Αυτή η διάρθρωση επιτρέπει στο σύστημα να έχει μια πιο οργανωμένη και επεκτάσιμη διαχείριση χρηστών καθώς η προσθήκη νέων δικαιωμάτων ή ρόλων μπορεί να γίνει εύκολα χωρίς να επηρεάζει τις υπόλοιπες λειτουργίες του συστήματος. Με αυτόν τον τρόπο εξασφαλίζεται η ασφαλής και ευέλικτη διαχείριση της πρόσβασης στις διάφορες λειτουργίες του συστήματος.

4.2.7 Πίνακας ειδοποιήσεων (product_warnings)

Ο πίνακας product_warnings (Σχήμα 4.9) χρησιμοποιείται για την καταγραφή ειδοποιήσεων που σχετίζονται με προϊόντα των οποίων το απόθεμα έχει πέσει κάτω από ένα προκαθορισμένο όριο, το οποίο ορίζουν οι διαχειριστές. Οι ειδοποιήσεις αυτές βοηθούν στη διαχείριση του αποθέματος, επιτρέποντας στους διαχειριστές να λαμβάνουν έγκαιρα μέτρα για την αναπλήρωση των προϊόντων.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε ειδοποίηση αποθέματος.
- **product_id**: Αναφέρεται στο προϊόν για το οποίο έχει δημιουργηθεί η ειδοποίηση.
- **read**: Boolean τιμή που δηλώνει αν η ειδοποίηση έχει διαβαστεί (true) ή παραμένει μη αναγνωσμένη (false). Η προεπιλεγμένη τιμή είναι false.
- **created_at**: Ημερομηνία και ώρα δημιουργίας της ειδοποίησης.
- **updated_at**: Ημερομηνία και ώρα τελευταίας ενημέρωσης της ειδοποίησης.

4.2.8 Πίνακας διαγραμμένων γεγονότων (delete_events)

Ο πίνακας delete_events (Σχήμα 4.10) χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικά με τις παραγγελίες που έχουν διαγραφεί. Συγκεκριμένα καταγράφει ποια παραγγελία διαγράφηκε, σε ποιον ανήκε, ποιος την διέγραψε καθώς και ένα περιγραφικό γεγονός της διαγραφής. Αυτό επιτρέπει τη διατήρηση ιστορικού αλλαγών και την παρακολούθηση διαγραφών για λόγους διαφάνειας και ελέγχου.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε εγγραφή διαγραφής παραγγελίας.
- **order_id**: Αναφέρεται στην παραγγελία που έχει διαγραφεί.
- **employee**: Το όνομα του υπαλλήλου στον οποίο ανήκει η διαγεγραμμένη παραγγελία.
- **event**: Κείμενο που περιγράφει το γεγονός της διαγραφής όπως ο λόγος ή οι συνθήκες κάτω από τις οποίες έγινε.
- **deleted_by**: Το όνομα του χρήστη που διέγραψε την παραγγελία.
- **created_at**: Ημερομηνία και ώρα καταγραφής της διαγραφής.
- **updated_at**: Ημερομηνία και ώρα τελευταίας ενημέρωσης της εγγραφής.

4.2.9 Πίνακας διαγραμμένων αντικειμένων (delete_item_events)

Ο πίνακας delete_item_events (Σχήμα 4.11) χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικά με τα προϊόντα που έχουν αφαιρεθεί από παραγγελίες. Συγκεκριμένα καταγράφει από ποια παραγγελία αφαιρέθηκε το προϊόν σε ποιον ανήκε, ποιος το διέγραψε, καθώς και μια περιγραφή του γεγονότος. Αυτό επιτρέπει την καταγραφή του ιστορικού αλλαγών και τη διατήρηση ελέγχου για λόγους διαφάνειας και διαχείρισης παραγγελιών.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε εγγραφή διαγραφής προϊόντος από μια παραγγελία.
- **order_id**: Αναφέρεται στην παραγγελία από την οποία διαγράφηκε το προϊόν.
- **employee**: Το όνομα του υπαλλήλου στον οποίο ανήκει η διαγεγραμμένη παραγγελία.
- **event**: Κείμενο που περιγράφει το γεγονός της διαγραφής, όπως ο λόγος ή οι συνθήκες κάτω από τις οποίες έγινε.
- **deleted_by**: Το όνομα του χρήστη που διέγραψε το προϊόν από την παραγγελία.
- **created_at**: Ημερομηνία και ώρα καταγραφής της διαγραφής.
- **updated_at**: Ημερομηνία και ώρα τελευταίας ενημέρωσης της εγγραφής.

4.2.10 Πίνακας τοποθεσιών (positions)

Ο πίνακας positions χρησιμοποιείται για την αποθήκευση των τοποθεσιών στις οποίες ανήκουν τα τραπέζια του καταστήματος. Οι τοποθεσίες μπορεί να αναφέρονται σε διαφορετικούς χώρους όπως "Αίθουσα 1", "Βεράντα", "Ισόγειο" κ.λπ. Η μοναδικότητα του ονόματος διασφαλίζει ότι κάθε τοποθεσία είναι μοναδική και δεν δημιουργούνται διπλές εγγραφές.

Ο πίνακας αυτός περιέχει τα παρακάτω πεδία:

- **id**: Κύριο πεδίο του πίνακα που αναγνωρίζει μοναδικά κάθε τοποθεσία.

- **name**: Το όνομα της τοποθεσίας όπου ανήκουν τα τραπέζια. Είναι μοναδικό, ώστε να μην υπάρχουν διπλές εγγραφές για την ίδια τοποθεσία.
- **created_at**: Ημερομηνία και ώρα καταγραφής της τοποθεσίας.
- **updated_at**: Ημερομηνία και ώρα τελευταίας ενημέρωσης της εγγραφής.

4.2.11 Πίνακας τραπεζιών (tables)

Ο πίνακας tables (Σχήμα 4.12) χρησιμοποιείται για την αποθήκευση των τραπεζιών ενός καταστήματος. Κάθε τραπέζι έχει ένα μοναδικό όνομα και συνδέεται με μία τοποθεσία (position_id) ώστε να γνωρίζουμε πού βρίσκεται μέσα στον χώρο. Η σύνδεση με τις τοποθεσίες επιτρέπει την καλύτερη διαχείριση και οργάνωση των τραπεζιών όπως για παράδειγμα ομαδοποίηση ανά αίθουσα ή εξωτερικό χώρο.

4.3 Σημαντικοί πίνακες της Laravel

Για τη σωστή λειτουργία μιας Laravel εφαρμογής το framework απαιτεί τη δημιουργία ορισμένων βασικών πινάκων στη βάση δεδομένων. Αυτοί οι πίνακες χρησιμοποιούνται για τη διαχείριση λειτουργιών όπως τα migrations, τα jobs, οι ουρές εργασιών (queues) και άλλες βασικές διαδικασίες.

Όταν εκτελούμε εντολές όπως php artisan migrate, η Laravel δημιουργεί αυτόματα τους απαραίτητους πίνακες για την αποθήκευση πληροφοριών σχετικά με τις εκτελεσμένες migrations. Παρομοίως, αν χρησιμοποιούμε το queue system της Laravel απαιτούνται ειδικοί πίνακες για την αποθήκευση και επεξεργασία των εργασιών που εκτελούνται στο παρασκήνιο.

Η σωστή διαχείριση αυτών των πινάκων είναι κρίσιμη για την εύρυθμη λειτουργία της εφαρμογής διασφαλίζοντας ότι όλες οι διαδικασίες καταγράφονται και εκτελούνται με ασφάλεια και αποδοτικότητα.

4.3.1 Πίνακας migrations

Ο πίνακας migrations στη βάση δεδομένων του Laravel είναι ένας ειδικός πίνακας που χρησιμοποιείται για την παρακολούθηση της κατάστασης των migrations του συστήματος. Τα migrations είναι αρχεία που περιλαμβάνουν τον ορισμό και τις αλλαγές της βάσης δεδομένων. Με τη χρήση των migrations μπορούμε να διαχειριστούμε την αλλαγή της δομής της βάσης δεδομένων (όπως πίνακες, στήλες, ευρετήρια, κ.λπ.) με τρόπο που να είναι εύκολα αναπαραγώγιμος και επαναλαμβανόμενος για την ανάπτυξη του συστήματος ανεξάρτητα από το περιβάλλον.

Ο πίνακας migrations βοηθά τη Laravel να παρακολουθεί ποια migrations έχουν εκτελεστεί και ποια όχι. Όταν εκτελούμε ένα migration μέσω της εντολής php artisan migrate η Laravel προσθέτει μια εγγραφή στον πίνακα migrations για να σημειώσει ότι το συγκεκριμένο migration έχει εκτελεστεί. Όταν εκτελούμε την εντολή php artisan migrate:rollback για να αναιρέσουμε τα τελευταία migrations η Laravel ανατρέχει στον πίνακα migrations για να προσδιορίσει ποιά migrations να επαναφέρει.

Ο πίνακας migrations περιέχει τις εξής στήλες:

- **id**: Ένα μοναδικό αναγνωριστικό για κάθε εγγραφή.
- **migration**: Το όνομα του αρχείου migration που εκτελέστηκε. Κάθε migration έχει ένα μοναδικό όνομα (π.χ., 2025_02_06_123456_create_users_table).
- **batch**: Ο αριθμός της παρτίδας (batch) στην οποία ανήκει το migration. Αυτό επιτρέπει στη Laravel να διαχωρίζει ποια migrations έχουν εκτελεστεί σε κάθε εκτέλεση.

- **created_at**: Η χρονική στιγμή που δημιουργήθηκε η εγγραφή στον πίνακα.

4.3.2 Πίνακας jobs

Ο πίνακας jobs στη βάση δεδομένων της Laravel χρησιμοποιείται για την αποθήκευση των jobs που εκκρεμούν ή περιμένουν να εκτελεστούν μέσω του Laravel Queue System. Το Queue System της Laravel επιτρέπει την καθυστέρηση εκτέλεσης εργασιών σε διάφορα υποσυστήματα ή την εκτέλεσή τους στο παρασκήνιο χωρίς να επηρεάζεται η απόδοση της εφαρμογής. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για την εκτέλεση βαρύτερων διαδικασιών όπως η αποστολή email, η επεξεργασία εικόνας ή η εξαγωγή δεδομένων χωρίς να μπλοκάρουν την εφαρμογή.

Ο πίνακας jobs χρησιμεύει στο να αποθηκεύει τα jobs που βρίσκονται στην ουρά περιμένοντας να εκτελούνται. Όταν ένα job προστίθεται στην ουρά μέσω της εντολής dispatch() το Laravel δημιουργεί μια εγγραφή στον πίνακα jobs. Ο πίνακας αυτός παρακολουθεί την κατάσταση του job (π.χ., αν είναι σε εκκρεμότητα ή αν έχει εκτελεστεί), τον αριθμό των προσπαθειών εκτέλεσης και τις πληροφορίες σχετικά με την καθυστέρηση ή την ώρα εκκίνησης.

Ο πίνακας jobs περιέχει τις εξής στήλες:

- **id**: Ένα μοναδικό αναγνωριστικό για κάθε job.
- **queue**: Το όνομα της ουράς (queue) στην οποία ανήκει το job. Η Laravel υποστηρίζει πολλαπλές ουρές επιτρέποντας την οργάνωση των jobs σε διαφορετικές ομάδες.
- **payload**: Τα δεδομένα του job συνήθως σε μορφή serialized που περιέχουν όλες τις απαραίτητες πληροφορίες για την εκτέλεση του.
- **attempts**: Ο αριθμός των προσπαθειών εκτέλεσης του job. Αν το job αποτύχει αυξάνεται ο αριθμός αυτός και η Laravel θα προσπαθήσει να το εκτελέσει ξανά.
- **reserved_at**: Η χρονική στιγμή που το job καταλαμβάνει μια θέση για εκτέλεση από έναν worker.
- **available_at**: Η χρονική στιγμή που το job είναι διαθέσιμο για εκτέλεση. Αν το job είναι καθυστερημένο (delayed) αυτή η τιμή καθορίζει πότε θα εκτελεστεί.
- **created_at**: Η χρονική στιγμή που το job προστέθηκε στην ουρά.

4.4 Βάση δεδομένων στο Android (Room Database)

Η Βάση Δεδομένων στο Android (Room Database) παρέχει έναν αποτελεσματικό και εύκολο τρόπο αποθήκευσης και διαχείρισης δεδομένων τοπικά στην εφαρμογή. Χρησιμοποιώντας το Room οι προγραμματιστές μπορούν να δημιουργήσουν και να διαχειριστούν πίνακες δεδομένων χωρίς να χρειάζεται να χειρίζονται άμεσα την SQL. Στην εφαρμογή μας οι πίνακες που δημιουργούμε για να αποθηκεύσουμε δεδομένα όπως χρήστες, προϊόντα και κατηγορίες ακολουθούν μια σαφή δομή για την αποτελεσματική ανάκτηση και ενημέρωση των δεδομένων.

4.4.1 Πίνακας κατηγοριών (categories)

Ο πίνακας categories αποθηκεύει τις κατηγορίες προϊόντων ή υπηρεσιών που χρησιμοποιούνται στην εφαρμογή. Παρακάτω είναι η δομή του:

- **@Entity(tableName = "categories"):** Αυτή η ανατοποθέτηση καθορίζει ότι η κατηγορία θα αποθηκεύεται στον πίνακα categories της βάσης δεδομένων. Το όνομα του πίνακα είναι καθορισμένο ρητά επιτρέποντας την αποθήκευση των δεδομένων στις αντίστοιχες στήλες του.
- **@PrimaryKey(autoGenerate = true):** Το πεδίο id είναι το πρωτεύον κλειδί (primary key) του πίνακα και εξασφαλίζει την μοναδικότητα κάθε εγγραφής. Η επιλογή autoGenerate = true σημαίνει ότι το Room Database θα δημιουργεί αυτόματα μια μοναδική τιμή για το id κάθε φορά που προστίθεται μια νέα κατηγορία. Δεν χρειάζεται ο χρήστης να καθορίζει χειροκίνητα το id.
- **@ColumnInfo(name = "name"):** Το πεδίο name αντιστοιχεί σε μια στήλη στον πίνακα και χρησιμοποιείται για την αποθήκευση του ονόματος της κατηγορίας. Αν και δεν είναι απαραίτητο να προσδιορίσουμε το όνομα της στήλης, η ανατοποθέτηση @ColumnInfo μας επιτρέπει να το κάνουμε για την καλύτερη αναγνωσιμότητα ή για να το αντιστοιχίσουμε σε υπάρχουσες βάσεις δεδομένων.
- **Κλάση δεδομένων:** Η κλάση Category είναι ένα απλό data class που χρησιμοποιείται για την αποθήκευση και ανάκτηση δεδομένων από τη βάση. Τα δεδομένα της κατηγορίας όπως το id και το name, αντιστοιχούν σε στήλες στον πίνακα categories.

4.4.2 Dao κατηγοριών (CategoryDao)

Το CategoryDao είναι ένα Data Access Object (DAO) που επιτρέπει στην εφαρμογή να διαχειρίζεται τα δεδομένα των κατηγοριών στο Room Database. Οι βασικές λειτουργίες του περιλαμβάνουν:

- Με την **clearCategories()** διαγράφουμε όλα τα δεδομένα ώστε να εισάγονται μόνο οι πιο πρόσφατες κατηγορίες από το API.
- Η **insertCategories(categories: List<Category>)** αποθηκεύει τις νέες κατηγορίες αντικαθιστώντας οποιαδήποτε παλιά δεδομένα με το ίδιο id.
- Η **getAllCategories()** επιστρέφει μια λίστα με όλες τις διαθέσιμες κατηγορίες και ενημερώνεται αυτόματα μέσω LiveData.
- Η **getCategoryById(categoryId: Int)** επιτρέπει την αναζήτηση μιας συγκεκριμένης κατηγορίας με βάση το ID της βοηθώντας στην ακριβή ανάκτηση δεδομένων.

4.4.3 Πίνακας υποκατηγοριών (subcategories)

Ο πίνακας Subcategory αποθηκεύει τις υποκατηγορίες προϊόντων επιτρέποντας τη δημιουργία ιεραρχικής δομής στο σύστημα διαχείρισης καταστημάτων. Κάθε υποκατηγορία συνδέεται με μία κύρια κατηγορία. Παρακάτω είναι η δομή του:

- **@Entity(tableName = "subcategories"):** Ορίζει ότι η κλάση Subcategory αναπαριστά έναν πίνακα με το όνομα subcategories στη βάση δεδομένων.
- **@PrimaryKey(autoGenerate = true):** Το id είναι το πρωτεύον κλειδί και δημιουργείται αυτόματα από τη βάση για κάθε νέα εγγραφή.
- **@ColumnInfo(name = "name"):** Το name είναι το όνομα της υποκατηγορίας και αποθηκεύεται στη στήλη name.
- **@ColumnInfo(name = "category_id"):** Το category_id είναι ένα ξένο κλειδί που συνδέει την υποκατηγορία με την αντίστοιχη κατηγορία.

4.4.4 Dao υποκατηγοριών (SubcategoryDao)

Το SubcategoryDao είναι το interface που επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων παρέχοντας τις βασικές λειτουργίες όπως εισαγωγή, διαγραφή και αναζήτηση δεδομένων. Παρακάτω είναι η ανάλυση των λειτουργιών του DAO:

- **clearSubcategories()**: Διαγράφει όλες τις εγγραφές στον πίνακα subcategories επιτρέποντας την ανανέωση των δεδομένων από το API.
- **insertSubcategories(subcategories: List<Subcategory>)**: Αποθηκεύει μια λίστα υποκατηγοριών αντικαθιστώντας τις υπάρχουσες σε περίπτωση ίδιου id.
- **getAllSubcategories()**: Επιστρέφει όλες τις υποκατηγορίες αποθηκευμένες στη βάση μέσω LiveData για αυτόματη ενημέρωση του UI.
- **getSubcategoriesByCategory(categoryId: Int)**: Επιτρέπει την αναζήτηση όλων των υποκατηγοριών που σχετίζονται με μία συγκεκριμένη κατηγορία.
- **getSubcategoryById(subcategoryId: Int)**: Επιστρέφει μία συγκεκριμένη υποκατηγορία βάσει του id.

4.4.5 Πίνακας προϊόντων (products)

Ο πίνακας Products αποθηκεύει τα προϊόντα που είναι διαθέσιμα στο κατάστημα επιτρέποντας τη διαχείριση των τιμών. Κάθε προϊόν συνδέεται με μια υποκατηγορία επιτρέποντας τη σωστή ταξινόμηση των αντικειμένων. Παρακάτω είναι η δομή του:

- **@Entity(tableName = "products")**: Ορίζει ότι η κλάση Product αναπαριστά έναν πίνακα με το όνομα products στη βάση δεδομένων.
- **@PrimaryKey(autoGenerate = true)**: Το id είναι το πρωτεύον κλειδί και δημιουργείται αυτόματα από τη βάση για κάθε νέα εγγραφή.
- **@ColumnInfo(name = "name")**: Το name είναι το όνομα του προϊόντος και αποθηκεύεται στη στήλη name.
- **@ColumnInfo(name = "price")**: Η τιμή του προϊόντος.
- **@ColumnInfo(name = "subcategory_id")**: Το subcategory_id είναι ένα ξένο κλειδί που συνδέει το προϊόν με την αντίστοιχη υποκατηγορία.

4.4.6 Dao προϊόντων (ProductDao)

Το ProductDao είναι το interface που επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων παρέχοντας τις βασικές λειτουργίες όπως εισαγωγή, διαγραφή και αναζήτηση δεδομένων. Παρακάτω είναι η ανάλυση των λειτουργιών του DAO:

- **clearProducts()**: Διαγράφει όλα τα προϊόντα από τον πίνακα products επιτρέποντας την ανανέωση των δεδομένων από το API.
- **insertProducts(products: List<Product>)**: Αποθηκεύει μια λίστα προϊόντων αντικαθιστώντας τις υπάρχουσες εγγραφές σε περίπτωση ίδιου id.
- **getAllProducts()**: Επιστρέφει όλα τα προϊόντα μέσω LiveData για αυτόματη ενημέρωση του UI.
- **getProductsBySubcategory(subcategoryId: Int)**: Επιτρέπει την αναζήτηση όλων των προϊόντων που σχετίζονται με μία συγκεκριμένη υποκατηγορία.
- **getProductById(productId: Int)**: Επιστρέφει ένα συγκεκριμένο προϊόν βάσει του id.

4.4.7 Πίνακας χαρακτηριστικών (attributes)

Ο πίνακας Attributes αποθηκεύει τα διάφορα χαρακτηριστικά που μπορούν να έχουν τα προϊόντα ή οι κατηγορίες. Κάθε χαρακτηριστικό μπορεί να έχει μια τιμή και να ανήκει σε μια συγκεκριμένη κατηγορία. Παρακάτω είναι η δομή του:

- **@Entity(tableName = "attributes")**: Ορίζει ότι η κλάση Attribute αναπαριστά έναν πίνακα attributes στη βάση δεδομένων.
- **@PrimaryKey(autoGenerate = true):** Το id είναι το πρωτεύον κλειδί το οποίο δημιουργείται αυτόματα από το Room Database.
- **@ColumnInfo(name = "name")**: Το name είναι το όνομα του χαρακτηριστικού και αποθηκεύεται στη στήλη name.
- **@ColumnInfo(name = "price")**: Η προαιρετική τιμή που σχετίζεται με το χαρακτηριστικό.
- **@ColumnInfo(name = "category_id")**: Το category_id είναι ένα προαιρετικό ξένο κλειδί που συνδέει το χαρακτηριστικό με μια κατηγορία.

4.4.8 Dao χαρακτηριστικών (AttributeDao)

Το AttributeDao είναι το interface που επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων, παρέχοντας βασικές λειτουργίες όπως εισαγωγή, αναζήτηση και διαγραφή δεδομένων. Παρακάτω είναι η ανάλυση των λειτουργιών του DAO:

- **clearAttributes()**: Διαγράφει όλα τα χαρακτηριστικά από τον πίνακα attributes.
- **insertAttributes(attributes: List<Attribute>)**: Αποθηκεύει μια λίστα χαρακτηριστικών, αντικαθιστώντας υπάρχουσες εγγραφές αν υπάρχει ίδιο id.
- **getAllAttributes()**: Επιστρέφει όλα τα διαθέσιμα χαρακτηριστικά μέσω LiveData για δυναμική ενημέρωση του UI.
- **getAttributesByCategory(categoryId: Int)**: Επιστρέφει όλα τα χαρακτηριστικά που ανήκουν σε μια συγκεκριμένη κατηγορία.
- **getAttributeById(attributeId: Int)**: Επιστρέφει ένα συγκεκριμένο χαρακτηριστικό βάσει του id.

4.4.9 Πίνακας χρήστη (user)

Ο πίνακας User αποθηκεύει πληροφορίες για το χρήστη της εφαρμογής συμπεριλαμβανομένων των του ρόλου του και πρόσθετων δεδομένων όπως το τελευταίο login. Παρακάτω είναι η δομή του:

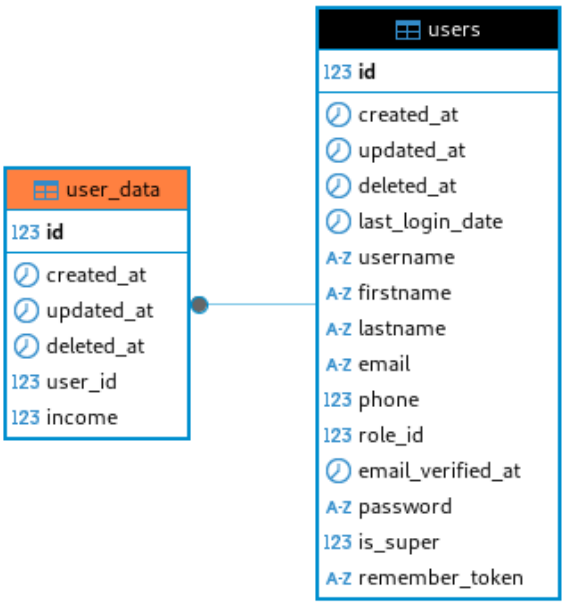
- **@Entity(tableName = "user")**: Δηλώνει ότι η κλάση User αναπαριστά έναν πίνακα user στη βάση δεδομένων.
- **@PrimaryKey(autoGenerate = true)**: Το id είναι το πρωτεύον κλειδί το οποίο δημιουργείται αυτόματα από το Room Database.
- **@ColumnInfo(name = "username")**: Το username είναι μοναδικό όνομα χρήστη.
- **@ColumnInfo(name = "firstname")**: Το όνομα του χρήστη.
- **@ColumnInfo(name = "lastname")**: Το επώνυμο του χρήστη.
- **@ColumnInfo(name = "email")**: Προαιρετικό email επικοινωνίας.
- **@ColumnInfo(name = "role_id")**: Το role_id είναι ξένο κλειδί που συνδέει τον χρήστη με έναν ρόλο.
- **@ColumnInfo(name = "last_login_date")**: Καταγράφει την τελευταία ημερομηνία σύνδεσης.
- **@ColumnInfo(name = "is_super")**: Boolean τιμή που υποδηλώνει αν ο χρήστης είναι υπερ-διαχειριστής.

4.4.10 Dao χρήστη (UserDao)

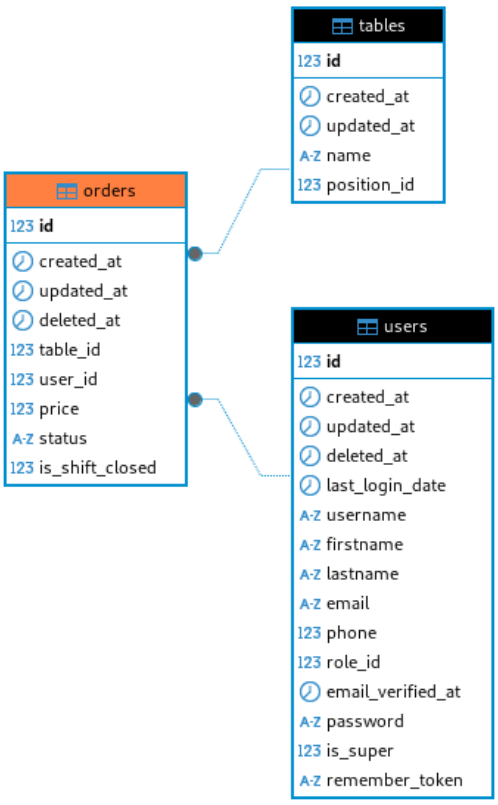
Το UserDao είναι το interface που επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων παρέχοντας βασικές λειτουργίες όπως εισαγωγή, αναζήτηση και διαγραφή δεδομένων. Παρακάτω είναι η δομή του:

- **saveUser(user: User)**: Αντικαθιστά πάντα τον χρήστη που είναι αποθηκευμένος στη βάση.
- **getCurrentUser()**: Επιστρέφει τον τρέχοντα χρήστη (ή null αν δεν υπάρχει).
- **deleteUser()**: Χρησιμοποιείται κατά την αποσύνδεση για να διαγράψει τον χρήστη.

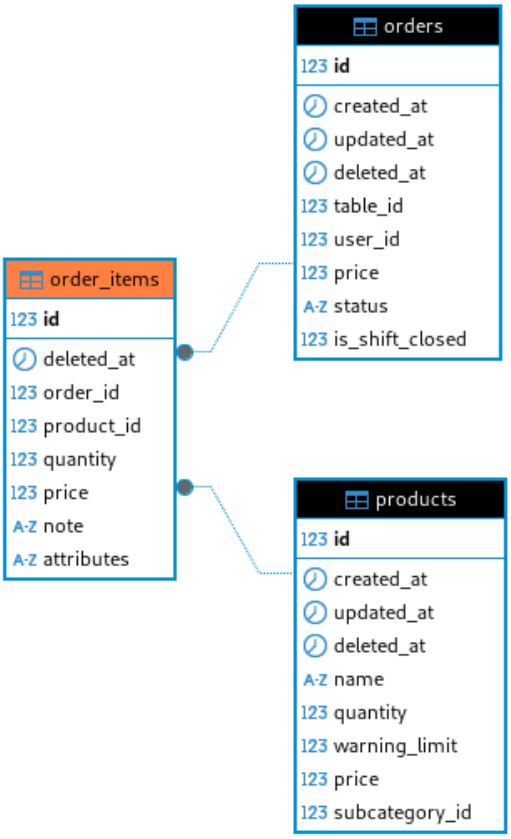
4.5 Σχήματα



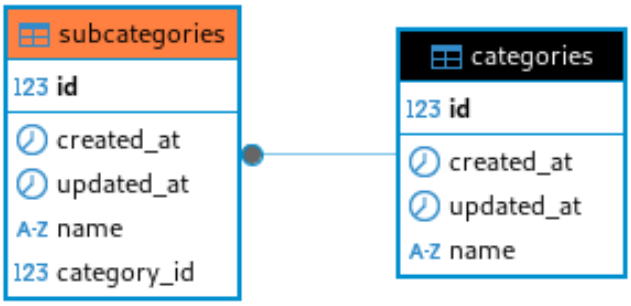
Σχήμα 4.1 Διάγραμμα UML users με user_data



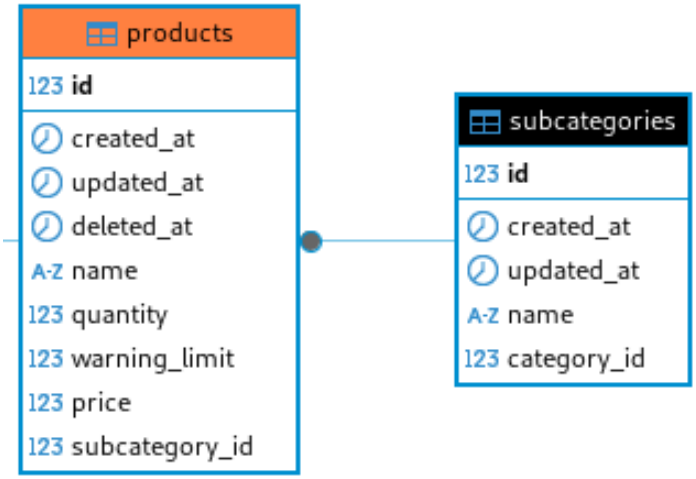
Σχήμα 4.2 *Διάγραμμα UML orders με tables και users*



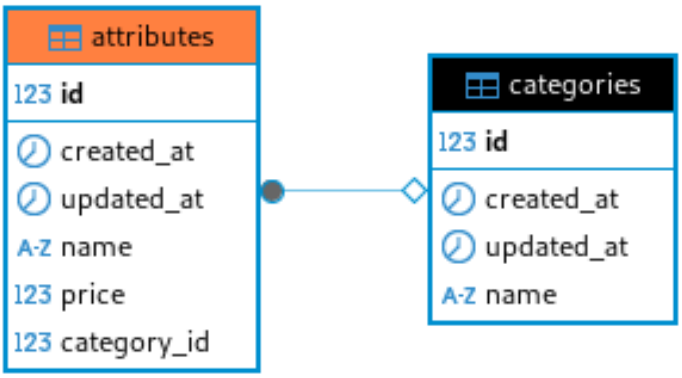
Σχήμα 4.3 *Διάγραμμα UML order_items με products και orders*



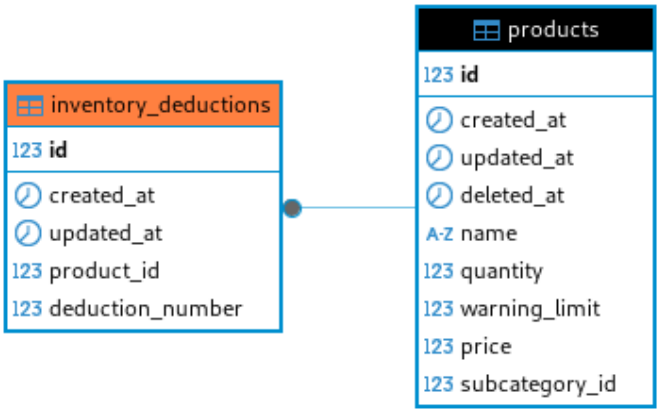
Σχήμα 4.4 *Διάγραμμα UML subcategories με categories*



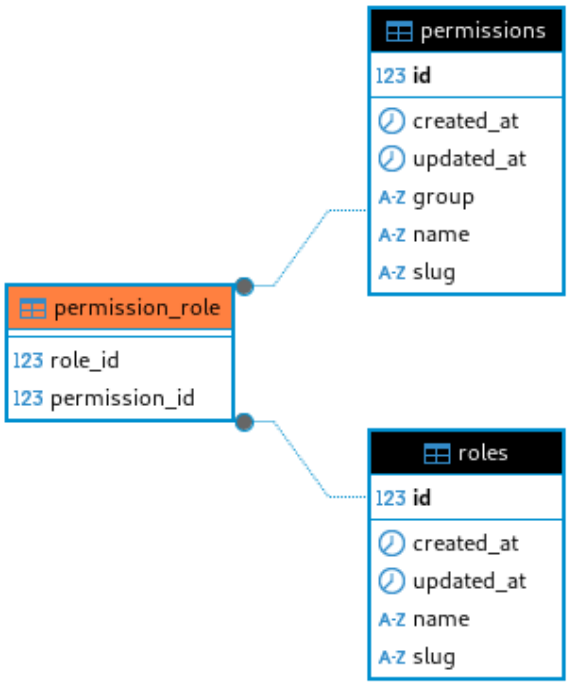
Σχήμα 4.5 *Διάγραμμα UML products με subcategories*



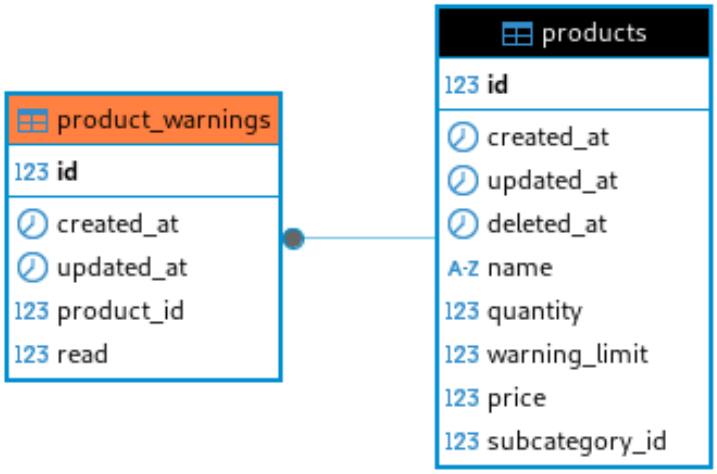
Σχήμα 4.6 *Διάγραμμα UML attributes με categories*



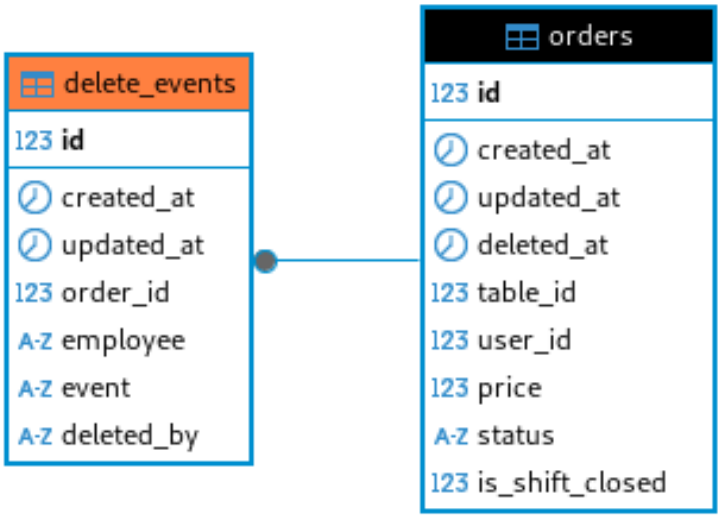
Σχήμα 4.7 *Διάγραμμα UML inventory_deductions με products*



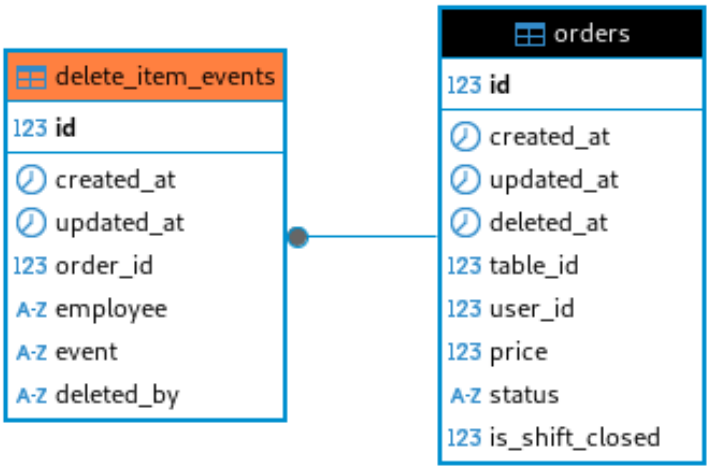
Σχήμα 4.8 *Διάγραμμα UML permission_role με permissions και roles*



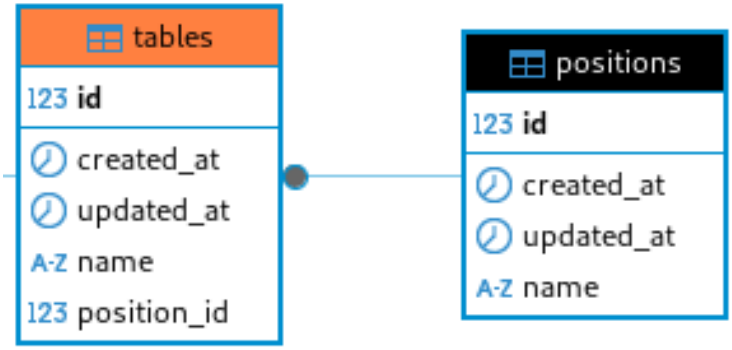
Σχήμα 4.9 *Διάγραμμα UML product_warnings και products*



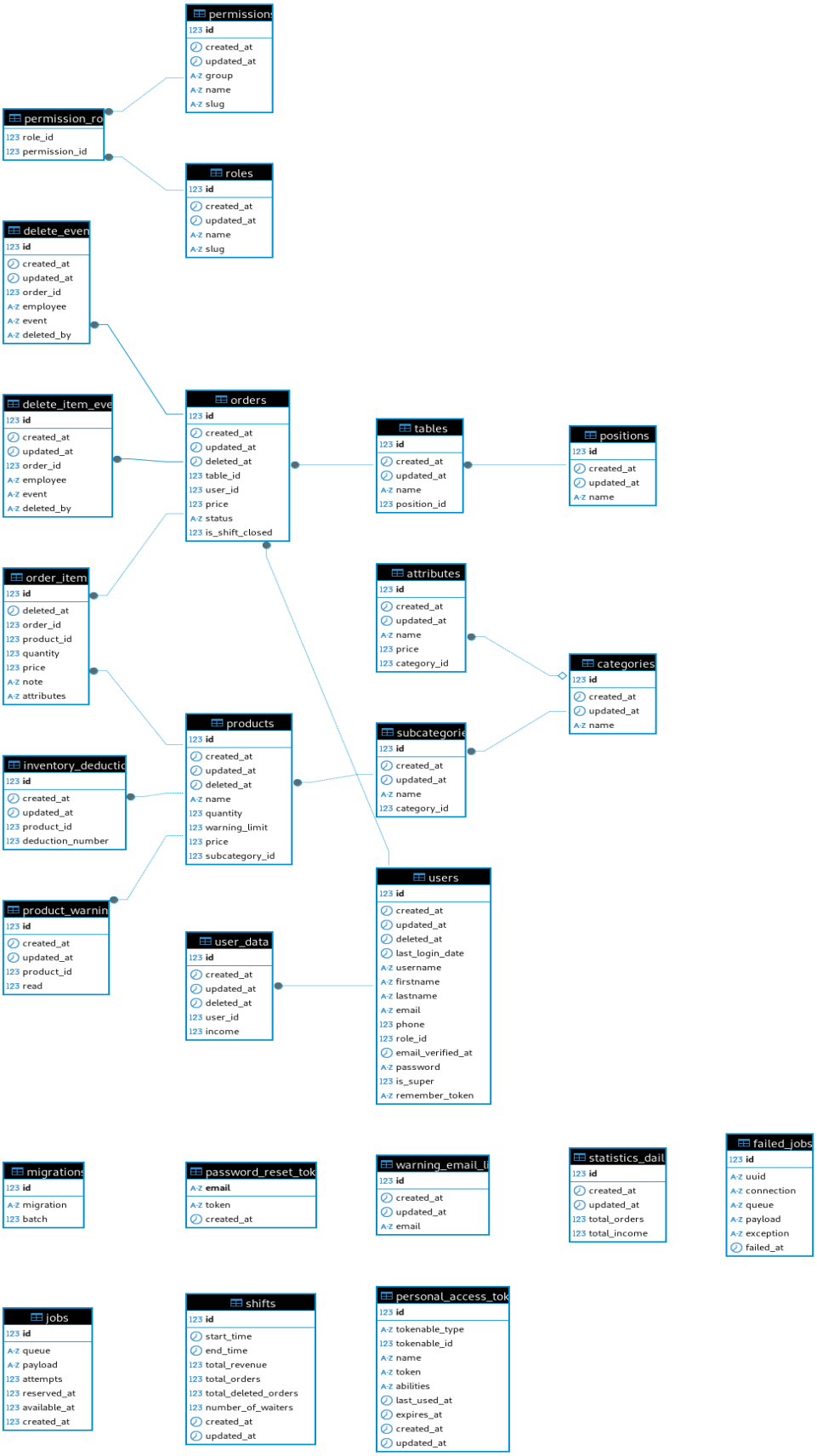
Σχήμα 4.10 *Διάγραμμα UML delete_events και orders*



Σχήμα 4.11 *Διάγραμμα UML delete_item_events και orders*



Σχήμα 4.12 *Διάγραμμα UML tables και positions*



Σχήμα 4.13 Διάγραμμα UML όλης της βάσης

4.5 Επίλογος

Το κεφάλαιο αυτό περιγράφει τη δομή της βάσης δεδομένων σε κάποιες από τις πιο σημαντικές λειτουργίες του συστήματος. Αναλύονται οι πίνακες που αφορούν τη διαχείριση χρηστών, παραγγελιών, αποθήκης και δικαιωμάτων. Επιπλέον παρουσιάζεται η σχεδίαση και υλοποίηση της βάσης δεδομένων στην Android εφαρμογή η οποία αποθηκεύει κρίσιμα δεδομένα τοπικά στη συσκευή μέσω του Room Database.

Για τη διαχείριση χρηστών ο πίνακας "Users" αποθηκεύει βασικά στοιχεία όπως όνομα χρήστη, κωδικό πρόσβασης, email και ρόλο. Ο πίνακας "user_data" συμπληρώνει τον "Users" με επιπλέον δεδομένα για τους σερβιτόρους, επιτρέποντας την παρακολούθηση της απόδοσής τους.

Οι παραγγελίες καταγράφονται στον πίνακα "orders" ο οποίος συνδέει κάθε παραγγελία με τον σερβιτόρο που την καταχώρησε και το τραπέζι στο οποίο αντιστοιχεί. Επιπλέον αποθηκεύονται πληροφορίες όπως η τιμή, η κατάσταση και η ημερομηνία. Τα επιμέρους προϊόντα κάθε παραγγελίας καταγράφονται στον πίνακα "order_items" μαζί με ποσότητα, τιμή, σημειώσεις και χαρακτηριστικά.

Η διαχείριση της αποθήκης γίνεται μέσω των πινάκων "categories" και "subcategories" οι οποίοι κατηγοριοποιούν τα προϊόντα ιεραρχικά. Ο πίνακας "products" περιέχει πληροφορίες για κάθε προϊόν, ενώ ο πίνακας "attributes" αποθηκεύει χαρακτηριστικά των προϊόντων και τυχόν επιπλέον κόστος. Οι αφαιρέσεις από το απόθεμα καταγράφονται στον πίνακα "inventory_deductions".

Επίσης η διαχείριση δικαιωμάτων γίνεται μέσω των πινάκων "roles", "permissions" και "permission_role". Ο "roles" ορίζει τους ρόλους των χρηστών (π.χ., διαχειριστής, σερβιτόρος), ο "permissions" τα διαθέσιμα δικαιώματα (π.χ., δημιουργία παραγγελίας) και ο "permission_role" συνδέει ρόλους με δικαιώματα, ελέγχοντας την πρόσβαση των χρηστών στις λειτουργίες του συστήματος.

Στην Android εφαρμογή η βάση δεδομένων υλοποιείται μέσω του Room Database επιτρέποντας την τοπική αποθήκευση δεδομένων που είναι κρίσιμα για τη λειτουργία του συστήματος όπως οι κατηγορίες προϊόντων, οι υποκατηγορίες και τα χαρακτηριστικά των προϊόντων. Ιδιαίτερη έμφαση δίνεται στη διαχείριση του συνδεδεμένου χρήστη, ώστε να αποθηκεύονται μόνο τα δικά του δεδομένα στη συσκευή. Μέσω των DAOs (Data Access Objects) η εφαρμογή προσφέρει λειτουργίες για την αποθήκευση ανάκτηση και ενημέρωση δεδομένων βελτιώνοντας την απόδοση και την εμπειρία χρήστη ακόμη και σε συνθήκες περιορισμένης συνδεσιμότητας.

5. Συμπεράσματα και προτάσεις βελτίωσης

5.1 Συμπεράσματα

Η ανάπτυξη ενός ολοκληρωμένου συστήματος παραγγελιοληψίας και διαχείρισης αποθήκης για καταστήματα εστίασης από το backend μέχρι το frontend και την mobile εφαρμογή αποτελεί μια πολύπλευρη διαδικασία που οδηγεί σε πολύτιμα συμπεράσματα.

Καταρχάς η επιλογή του Laravel για το backend προσφέρει υψηλή ευελιξία και επεκτασιμότητα καθιστώντας το ιδανικό για την ανάπτυξη σύγχρονων και ασφαλών web εφαρμογών. Το framework αυτό παρέχει δομημένη αρχιτεκτονική προηγμένες δυνατότητες διαχείρισης δεδομένων και ένα ισχυρό σύστημα διαχείρισης εξουσιοδοτήσεων επιτρέποντας στο σύστημα να διαχειριστεί μεγάλο όγκο δεδομένων με αποδοτικό τρόπο. Παράλληλα, το Vue 3 επιλέχθηκε για το frontend, προσφέροντας μια σύγχρονη και δυναμική διεπαφή χρήστη. Η χρήση του Composition API σε συνδυασμό με τη βελτιωμένη απόδοση και την επεκτασιμότητα επιτρέπει τη δημιουργία ευέλικτων και διαδραστικών UI στοιχείων βελτιώνοντας την εμπειρία του χρήστη.

Επιπλέον η επιλογή της Kotlin για την ανάπτυξη μιας native Android εφαρμογής διασφαλίζει υψηλή απόδοση και πλήρη αξιοποίηση των δυνατοτήτων των κινητών συσκευών. Σε αντίθεση με λύσεις όπως το WebView ή τις υβριδικές εφαρμογές η χρήση native τεχνολογιών παρέχει πιο ομαλή αλληλεπίδραση με το λειτουργικό σύστημα καλύτερη διαχείριση πόρων και πιο αποδοτική χρήση της CPU και της μνήμης της συσκευής. Αυτό σημαίνει ότι η εφαρμογή μπορεί να λειτουργεί πιο γρήγορα, να υποστηρίζει πιο απαιτητικές διεργασίες και να προσφέρει μια βελτιστοποιημένη εμπειρία χρήσης ανεξαρτήτως της συσκευής που χρησιμοποιείται.

Η δημιουργία της βάσης δεδομένων από την αρχή αποτελεί κρίσιμο σημείο για την επιτυχία του συστήματος. Η σωστή σχεδίαση της βάσης με καθορισμό των οντοτήτων και των σχέσεων μεταξύ τους εξασφαλίζει την αποτελεσματική αποθήκευση και ανάκτηση των δεδομένων. Αυτό με τη σειρά του οδηγεί σε βελτιστοποίηση της απόδοσης του συστήματος και στην αποφυγή σφαλμάτων.

Πέρα από τα τεχνικά συμπεράσματα η ανάπτυξη αυτού του συστήματος αναδεικνύει και σημαντικά επιχειρηματικά οφέλη. Η αυτοματοποίηση διαδικασιών όπως η παραγγελιοληψία και η διαχείριση αποθήκης οδηγεί σε αύξηση της αποδοτικότητας και μείωση του λειτουργικού κόστους. Παράλληλα η ψηφιοποίηση των διαδικασιών ελαχιστοποιεί τα λάθη και επιτρέπει την ακριβή παρακολούθηση των αποθεμάτων σε πραγματικό χρόνο.

Τέλος η συλλογή και ανάλυση δεδομένων μέσω του συστήματος παρέχει πολύτιμες πληροφορίες για τη λειτουργία του καταστήματος. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν για τη λήψη στρατηγικών αποφάσεων σχετικά με το μενού, τις προμήθειες και τη γενικότερη διαχείριση του καταστήματος.

5.2 Προτάσεις βελτίωσης

Η βελτίωση της cache στην εφαρμογή μπορεί να οδηγήσει σε σημαντική αύξηση της απόδοσης και μία από τις πιο αποδοτικές λύσεις για την αποθήκευση δεδομένων είναι η Redis cache. Στην τρέχουσα υλοποίηση χρησιμοποιούμε file cache το οποίο παρέχει ορισμένα πλεονεκτήματα αλλά η μετάβαση σε Redis μπορεί να προσφέρει ακόμα μεγαλύτερη βελτίωση.

Η Redis είναι μια in-memory βάση δεδομένων που επιτρέπει ταχύτερη πρόσβαση και αναζήτηση δεδομένων σε σύγκριση με τη file cache. Είναι ιδανική για δεδομένα που δεν αλλάζουν συχνά όπως κατηγορίες και υποκατηγορίες προϊόντων τα ίδια τα προϊόντα και τα χαρακτηριστικά τους καθώς και οι ρόλοι και τα δικαιώματα των χρηστών. Τα δεδομένα αυτά μπορούν να αποθηκευτούν στην Redis χωρίς ημερομηνία λήξης και να ενημερώνονται μόνο όταν υπάρχει κάποια αλλαγή.

Ορισμένα από τα βασικά πλεονεκτήματα της Redis σε σχέση με το file cache περιλαμβάνουν:

- **Ταχύτητα πρόσβασης:** Η Redis λειτουργεί εξ ολοκλήρου στη μνήμη γεγονός που επιτρέπει εξαιρετικά γρήγορη ανάκτηση δεδομένων σε αντίθεση με την file cache η οποία απαιτεί πρόσβαση στο σύστημα αρχείων.
- **Αποτελεσματικότητα στην κλιμάκωση:** Η Redis υποστηρίζει κατανεμημένες αρχιτεκτονικές και μπορεί να κλιμακωθεί εύκολα σε περίπτωση αύξησης του φόρτου ή της ανάγκης για μεγαλύτερη αποθηκευτική ικανότητα προσφέροντας καλύτερη ευχέρεια και επεκτασιμότητα από την file cache.
- **Αναγνώριση αλλαγών:** Η Redis μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων με πολιτικές λήξης ή χωρίς (χωρίς ημερομηνία λήξης) εξασφαλίζοντας ότι τα δεδομένα αποθηκεύονται για όσο διάστημα χρειάζεται χωρίς περιττές ανακτήσεις από τη βάση δεδομένων.
- **Μείωση φορτίου στη βάση δεδομένων:** Επειδή τα δεδομένα που δεν αλλάζουν συχνά (όπως οι κατηγορίες, τα προϊόντα και τα δικαιώματα χρηστών) θα αποθηκεύονται στην Redis δεν θα χρειάζεται να γίνονται επαναλαμβανόμενα queries στη βάση δεδομένων για τα ίδια δεδομένα μειώνοντας έτσι την επιβάρυνση και αυξάνοντας την απόδοση της εφαρμογής.

Μια ακόμα σημαντική βελτίωση στην εφαρμογή θα είναι η ενσωμάτωσή της με την ταμειακή μηχανή προκειμένου να αυτοματοποιηθεί η διαδικασία εκτύπωσης απόδειξης όταν αποστέλλεται μια παραγγελία. Με την ολοκλήρωση της παραγγελίας η απόδειξη θα εκτυπώνεται άμεσα προσφέροντας έναν πιο γρήγορο και αξιόπιστο τρόπο εξυπηρέτησης του πελάτη. Παράλληλα τα δεδομένα της παραγγελίας θα αποστέλλονται αυτόματα στο myData, εξασφαλίζοντας ότι οι πληροφορίες καταχωρούνται ακριβώς τη στιγμή που δημιουργούνται χωρίς περιττές καθυστερήσεις.

Επιπλέον η εφαρμογή μπορεί να συνδεθεί με το POS σύστημα κάποιου τραπεζικού φορέα για την υποστήριξη πληρωμών μέσω κάρτας. Με αυτή την ενσωμάτωση οι υπάλληλοι θα έχουν τη δυνατότητα να επεξεργάζονται πληρωμές απευθείας από την εφαρμογή παρέχοντας μια ολοκληρωμένη λύση που συνδυάζει την καταγραφή παραγγελιών και τις πληρωμές με κάρτα σε μια ενιαία διαδικασία. Αυτό θα μειώσει τον χρόνο που απαιτείται για τη διαχείριση των πληρωμών και θα ελαχιστοποιήσει τα σφάλματα προσφέροντας στους πελάτες μια πιο ομαλή και ευχάριστη εμπειρία αγορών.

Η χρήση τεχνολογιών όπως WebSockets ή Server-Sent Events (SSE) επιτρέπει την άμεση και δυναμική ενημέρωση της εφαρμογής σε πραγματικό χρόνο εξασφαλίζοντας ότι οι χρήστες έχουν πάντα τις πιο πρόσφατες πληροφορίες. Αυτές οι τεχνολογίες μπορούν να ενημερώνουν άμεσα για αλλαγές στη διαθεσιμότητα των προϊόντων όπως η αλλαγή της κατάστασης ενός προϊόντος ή η ενημέρωση σχετικά με την απογραφή. Με αυτόν τον τρόπο η εφαρμογή διασφαλίζει ότι οι χρήστες δεν χρειάζεται να ανανεώνουν χειροκίνητα τη σελίδα ή να περιμένουν για τη συγχρονισμένη ενημέρωση των δεδομένων.

Η προσθήκη ενός συστήματος ζωντανών ειδοποιήσεων μπορεί να προσφέρει μια ακόμη διάσταση στην οργάνωση και την εξυπηρέτηση. Αυτές οι ειδοποιήσεις μπορούν να ενημερώνουν σε πραγματικό χρόνο τους σερβιτόρους για σημαντικά γεγονότα όπως το ότι ένα τραπέζι είναι έτοιμο για παραγγελία ή ότι

κάποιος πελάτης απαιτεί άμεση εξυπηρέτηση. Αυτού του είδους η αυτοματοποίηση όχι μόνο βοηθά στην αποφυγή καθυστερήσεων αλλά ενισχύει την αποδοτικότητα της ομάδας διευκολύνοντας την καλύτερη διαχείριση του χρόνου και επιταχύνοντας την ανταπόκριση στις ανάγκες των πελατών. Έτσι οι σερβιτόροι είναι πάντα ενήμεροι και μπορούν να παρέχουν άμεση και εξαιρετική εξυπηρέτηση ενώ η επιχείρηση λειτουργεί πιο ομαλά και αποτελεσματικά.

Η δημιουργία ενός ενιαίου backoffice για τη διαχείριση όλων των πλατφορμών παραγγελιοληψίας αποτελεί ένα σημαντικό βήμα προς την ολοκληρωμένη και αποδοτική διαχείριση της επιχείρησης. Μέσω αυτού του backoffice θα υπάρχει η δυνατότητα να παρακολουθείται σε ζωντανό χρόνο η κατάσταση όλων των συστημάτων παραγγελιών εντοπίζοντας άμεσα οποιαδήποτε προβλήματα ή καθυστερήσεις. Αυτό θα επιτρέπει την ταχεία αντίδραση και την άμεση επίλυση οποιωνδήποτε τεχνικών ή λειτουργικών ζητημάτων. Επιπλέον το ενιαίο backoffice θα παρέχει τη δυνατότητα περιορισμού της πρόσβασης στην εφαρμογή ή σε συγκεκριμένες λειτουργίες ανάλογα με το αν η εφαρμογή είναι πληρωμένη ή αν αφορά σε συνδρομητικό μοντέλο. Οι διαχειριστές θα μπορούν να ορίζουν δικαιώματα πρόσβασης με βάση το επίπεδο συνδρομής ή την εξόφληση των λογαριασμών διασφαλίζοντας ότι οι χρήστες έχουν πρόσβαση μόνο στις δυνατότητες που αντιστοιχούν στην πληρωμή ή την συνδρομή τους.

Για την αποτελεσματική επίλυση προβλημάτων και την υποστήριξη των χρηστών, θα ενσωματωθεί και ένα ticketing system, το οποίο θα επιτρέπει την υποβολή αιτημάτων υποστήριξης από τους χρήστες. Μέσω αυτού του συστήματος, οι υπάλληλοι και οι διαχειριστές θα μπορούν να παρακολουθούν και να διαχειρίζονται τα προβλήματα σε πραγματικό χρόνο, παρέχοντας λύσεις ή ενημερώσεις σχετικά με την πρόοδο της επίλυσης. Αυτό το σύστημα θα βελτιώσει τη συνεργασία μεταξύ των πελατών και της υποστήριξης προσφέροντας έναν οργανωμένο τρόπο επικοινωνίας και εξασφαλίζοντας την ταχύτερη δυνατή επίλυση των ζητημάτων που προκύπτουν.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Site

[1] Laravel official documentation: Authentication
[Online]. Available: https://laravel.com/docs/10.x/authentication#authenticating-users

[2] Laravel official documentation: HTTP Session.

[Online]. Available: https://laravel.com/docs/10.x/session

[3] Laravel official documentation: Laravel Sanctum.

[Online]. Available: https://laravel.com/docs/10.x/session

[4] Laravel docs: Eloquent: Getting Started.

[Online]. Available: https://laravel-docs.readthedocs.io/en/latest/eloquent/

[5] Laravel api: AuthServiceProvider.

[Online]. Available: https://laravel.com/api/10.x/Illuminate/Auth/AuthServiceProvider.html

[6] Laravel official documentation: Queues.

[Online]. Available: https://laravel.com/docs/11.x/queues

[7] wpwebinfotech.com: How to Setup a Laravel Queue System.

[Online]. Available: https://wpwebinfotech.com/blog/laravel-queue-system/

[8] Laravel official documentation: Controllers introduction.

[Online]. Available: https://laravel.com/docs/10.x/controllers

[9] Laravel official documentation: Queues introduction.

[Online]. Available: https://laravel.com/docs/11.x/queues

[10] Laravel official documentation: Middleware introduction.

[Online]. Available: https://laravel.com/docs/10.x/middleware#introduction

[11] Vue.js Ofiicial documentation: introduction

[Online]. Available: https://vuejs.org/guide/introduction.html#the-progressive-framework

[12] pinia.vuejs,org: introduction

[Online]. Available: https://pinia.vuejs.org/introduction.html#Introduction

[13] Vitejs.github.io: introduction

[Online]. Available: https://vitejs.github.io/vite-plugin-react-pages#-vite-plugin-react-pages

[14] Vue.js Ofiicial documentation: Language blocks

[Online]. Available: https://vuejs.org/api/sfc-spec.html#language-blocks

[15] Axios Ofiicial documentation: Getting Started

[Online]. Available: https://axios-http.com/docs/intro

[16] geeksforgeeks.or: Learn Java For App Development – A Complete Guide

[Online]. Available: https://www.geeksforgeeks.org/learn-java-for-android-app-development-a-complete-guide/

[17] developer.android.com: Save simple data with SharedPreferences

[Online]. Available: https://developer.android.com/training/data-storage/shared-preferences

[18] Oracle.com: What Is MySQL?

[Online]. Available: https://www.oracle.com/mysql/what-is-mysql/#what-is-mysql

[19] pixinvent.com: features

[Online]. Available: https://pixinvent.com/vuexy-bootstrap-html-admin-template/#framework-features

[20] nanosoft.gr: εφαρμογές

[Online]. Available: https://nanosoft.gr/%CE%B1%CF%83%CF%8D%CF%81%CE%BC%CE%B1%CF%84%CE%B7-%CF%80%CE%B1%CF%81%CE%B1%CE%B3%CE%B3%CE%B5%CE%BB%CE%B9%CE%BF%CE%BB%CE%B7%CF%88%CE%AF%CE%B1-kalypso-restaurant/

[21] ordersoft.gr: Πλήρες Σύστημα Ασύρματης Παραγγελιοληψίας

[Online]. Available: https://ordersoft.gr/asyrmati-paraggeliolipsia/

[22] dionserve.gr: Παρουσίαση ασύρματης παραγγελιοληψίας Dionserve

[Online]. Available: https://www.dionserve.gr/created-with-passion/

[23] square.github.io: Retrofit - Introduction

[Online]. Available: https://square.github.io/retrofit/

[24] github.com: Gson

[Online]. Available: https://github.com/google/gson?tab=readme-ov-file#gson

[25] twinsoft.gr: Orexsys

[Online]. Available: https://twinsoft.gr/orexsys/

[26] developer.android.com: Glossary of terms

[Online]. Available: https://developer.android.com/reference/androidx/recyclerview

[27] developer.android.com: Create dynamic lists with RecyclerView

[Online]. Available: https://developer.android.com/develop/ui/views/layout/recyclerview#steps-for-implementing

[28] developer.android.com: Save data in a local database using Room

[Online]. Available: https://developer.android.com/training/data-storage/room#components