ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και υλοποίηση συστήματος διαχείρισης αποθήκης και πωλήσεων (POS)»

Productive Manager

Τον φοιτητή
Χρήστου Αβράμη
Αρ. Μητρώου: it174918

Επιβλέπων
Βασίλειος Κώστογλου
Καθηγητής

Ημερομηνία 25 ΜΑΪΟΥ 2023

Τίτλος Δ.Ε. Σχεδιασμός και υλοποίηση συστήματος διαχείρισης αποθήκης και πωλήσεων (POS)
Κωδικός Δ.Ε. 22125
Ονοματεπώνυμο φοιτητή Χρήστος Αβράμης
Ονοματεπώνυμο εισηγητή Βασίλειος Κώστογλου
Ημερομηνία ανάληψης Δ.Ε 06-03-2022
Ημερομηνία περάτωσης Δ.Ε 25-05-2023

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.

Πρόλογος

Ο λόγος για τον οποίο επέλεξα να ακολουθήσω αυτό το συγκεκριμένο έργο στον κλάδο του λιανικού εμπορίου ήταν η απασχόληση μου από μικρός σε αντίστοιχο περιβάλλον λόγω των γονιών μου. Καθώς οι απαιτήσεις μεγάλωναν ετσι δημιουργήθηκε και η ανάγκη για την ψηφιοποίηση των προϊόντων και των παραγγελιών. Στόχος της πτυχιακής είναι να εμβαθύνω τις γνώσεις μου στην ανάπτυξη ενός διαδικτυακού συστήματος Point of Sale (POS) χρησιμοποιώντας Spring Boot και Vaadin Flow. Με την ολοκλήρωση αυτού του έργου, στοχεύω να αξιοποιήσω τις δεξιότητες και τις γνώσεις μου που απέκτησα πρόσφατα για να αναπτύξω πιο ισχυρά και αποτελεσματικά συστήματα για τη βιομηχανία της λιανικής χρησιμοποιώντας το Spring Boot και το Vaadin Flow.

Περίληψη

Σκοπός αυτού του έργου είναι ο σχεδιασμός και η εφαρμογή συστήματος διαχείρισης αποθήκης και πωλήσεων (POS) για μια επιχείρηση λιανικής. Το σύστημα προορίζεται να βελτιώσει την αποτελεσματικότητα και την ακρίβεια των συναλλαγών διαχείρισης αποθεμάτων και πωλήσεων. Η εφαρμογή παρέχει λειτουργίες όπως εισαγωγή και μορφοποίηση προϊόντων σε ηλεκτρονική μορφή, αποθήκευση πελατών και προμηθευτών και τέλος, διεκπεραίωση παραγγελιών στο κατάστημα και δημιουργία αναφορών. Η διεπαφή χρήστη έχει σχεδιαστεί για να είναι γρήγορη και εύκολα προσβάσιμη μέσω προγραμμάτων περιήγησης στο web. Το σύστημα διαχείρισης αποθέματος και POS αναπτύχθηκε χρησιμοποιώντας σύγχρονες τεχνολογίες όπως το Spring Boot, το Vaadin Flow και τη βάση δεδομένων H2. Συνολικά, το σύστημα διαχείρισης αποθέματος και POS παρέχει μια ολοκληρωμένη λύση για τις επιχειρήσεις για τον έλεγχο του αποθέματός τους, τις πωλήσεις και τη βελτίωση της απόδοσής τους μέσω αναφορών.

«Design and implementation of an inventory and point of sale (POS) management system»

Christos Avramis

Abstract

The purpose of this project is to design and implement a warehouse and sales management system (POS) for a retail business. The system is intended to improve the efficiency and accuracy of inventory and sales management transactions. The application provides functions such as the input and formatting of products in electronic format, the storage of customers and suppliers, and finally, the processing of orders in the store and the creation of reports. The user interface is designed to be fast and easily accessible through web browsing programs. The inventory and POS management system was developed using modern web technologies such as Spring Boot, Vaadin Flow, and the H2 database. Overall, the inventory and POS management system provides a comprehensive solution for businesses to control their inventory, sales, and improve their performance through monitoring.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Βασίλη Κώστογλου, για την καθοδήγηση που μου προσέφερε και το χρόνο που διέθεσε δίνοντάς μου χρήσιμες συμβουλές και οδηγίες για την ολοκλήρωση της πτυχιακής μου εργασίας. Στο ίδιο πλαίσιο ευγνωμοσύνης, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνές Πανεπιστημίου της Ελλάδος για τη συμβολή τους στην επιστημονική και τεχνολογική μου συγκρότηση στα χρόνια της φοίτησής μου στο Τμήμα. Οφείλω επίσης ένα μεγάλο ευχαριστώ σε όλους εκείνους που συνέβαλαν είτε πρακτικά δοκιμάζοντας την εφαρμογή είτε ψυχικά βοηθώντας στην ολοκλήρωση της εργασίας μου. Τέλος, ένα μεγάλο ευχαριστώ στους γονείς μου Δημήτρη και Ελένη για την υποστήριξη τους, καθώς και τους συγγενείς και τους φίλους για την υποστήριξη σε όλο το διάστημα των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
POS	Point Of Sale
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	Javascript
CRM	Customer Relationship Management
WWW	World Wide Web
SQL	Structured Query Language
DDL	Data Definition Language
DCL	Data Control Language
TCL	Transaction Control Language
ORM	Object Relational Mapping
JDK	Java Development Kit
JVM	Java virtual machine
JRE	Java Runtime Environment

Κεφάλαιο 1ο: Συστήματα διαχείρισης αποθήκης και πωλήσεων

1.1 Ανάγκη και πλεονεκτήματα

Στον κλάδο λιανικής, η διαχείριση αποθεμάτων και η επεξεργασία των συναλλαγών είναι βασικές λειτουργίες που μπορεί να είναι αρκετά προκλητικές χωρίς τα κατάλληλα εργαλεία. Προκειμένου να εξορθολογιστούν αυτές οι λειτουργίες και να μεγιστοποιηθεί η αποτελεσματικότητα, πολλές επιχειρήσεις βασίζονται σε συστήματα διαχείρισης σημείων πώλησης (POS) και αποθεμάτων. Το σύστημα POS είναι μια εφαρμογή λογισμικού που επιτρέπει στις επιχειρήσεις να επεξεργάζονται συναλλαγές και να διαχειρίζονται τις πωλήσεις. Συνήθως περιλαμβάνει λειτουργίες όπως σάρωση γραμμωτού κώδικα, ηλεκτρονική επεξεργασία πληρωμών και παρακολούθηση αποθέματος. Με την αυτοματοποίηση πολλών από τις εργασίες που εμπλέκονται στην επεξεργασία των πωλήσεων, ένα σύστημα POS μπορεί να βοηθήσει τις επιχειρήσεις να εξοικονομήσουν χρόνο και να μειώσουν τα σφάλματα. Ομοίως, τα συστήματα διαχείρισης αποθεμάτων έχουν σχεδιαστεί για να βοηθούν τις επιχειρήσεις να παρακολουθούν τα επίπεδα των αποθεμάτων τους, να παρακολουθούν τις κινήσεις των προϊόντων και να βελτιστοποιούν τις παραγγελίες και την αποθήκευση. Ένα σύστημα διαχείρισης αποθεμάτων μπορεί να βοηθήσει τις επιχειρήσεις να αποφύγουν τα αποθέματα, να μειώσουν τα απόβλητα και να βελτιώσουν την κερδοφορία.

Αυτά τα συστήματα είναι απαραίτητα εργαλεία για κάθε επιχείρηση λιανικής που θέλει να ευδοκιμήσει στη σημερινή ανταγωνιστική αγορά. Επενδύοντας σε ένα σύστημα POS και διαχείρισης αποθεμάτων, οι επιχειρήσεις μπορούν να εξορθολογίσουν τις λιανικές τους δραστηριότητες, να βελτιώσουν την αποτελεσματικότητα και την κερδοφορία και να προσφέρουν καλύτερη εμπειρία στον πελάτη. Μερικές από τις ειδικές ανάγκες που καλύπτουν αυτά τα συστήματα περιλαμβάνουν:

- Ακριβής και σε πραγματικό χρόνο διαχείριση αποθεμάτων
- Ταχύτερη και πιο αποτελεσματική διαδικασία ολοκλήρωσης αγοράς
- Μειωμένα σφάλματα στην τιμολόγηση και τη διαχείριση αποθεμάτων
- Βελτιωμένη εξυπηρέτηση πελατών με καλύτερη διαθεσιμότητα προϊόντων και ταχύτερη εκπλήρωση παραγγελιών
- Αναλύσεις δεδομένων και αναφορά για καλύτερη λήψη αποφάσεων και προβλέψεις
- Ενοποίηση με άλλα επιχειρηματικά συστήματα όπως η λογιστική και η διαχείριση σχέσεων με τους πελάτες.

Συνοπτικά, ένα σύστημα διαχείρισης POS και αποθέματος είναι ένα ουσιαστικό εργαλείο για κάθε επιχείρηση λιανικής που θέλει να βελτιώσει την αποτελεσματικότητά της, την κερδοφορία και την εμπειρία του πελάτη. Τα POS προσφέρουν προηγμένες δυνατότητες, όπως ανάλυση δεδομένων, διαχείριση σχέσεων με πελάτες και δυνατότητες πωλήσεων σε όλους τους τομείς. Αυτές οι δυνατότητες μπορούν να βοηθήσουν τις επιχειρήσεις να παραμείνουν ανταγωνιστικές σε ένα ταχέως εξελισσόμενο τοπίο λιανικής και να προσφέρουν στους πελάτες μια απρόσκοπτη εμπειρία αγορών σε πολλά κανάλια και σημεία επαφής.

Συνοπτικά, ένα σύστημα διαχείρισης POS και αποθέματος είναι ένα ουσιαστικό εργαλείο για κάθε επιχείρηση λιανικής που θέλει να βελτιώσει την αποτελεσματικότητά της, την κερδοφορία και την εμπειρία του πελάτη. Με την αυτοματοποίηση πολλών από τις εργασίες που εμπλέκονται στην επεξεργασία των πωλήσεων και στη διαχείριση του αποθέματος, οι επιχειρήσεις μπορούν να επικεντρωθούν στην παροχή καλύτερης εμπειρίας στους πελάτες τους και να παραμείνουν μπροστά από τον ανταγωνισμό.

Σχήμα 1: POS

1.2 Παρόμοιες Εφαρμογές

Υπάρχουν πολλά διαφορετικά συστήματα POS και διαχείρισης αποθεμάτων διαθέσιμα στην αγορά κάποια από τα πιο διάσημα είναι τα παρακάτω:

Square

Το Square είναι ένα πολύ γνωστό σύστημα POS που προσφέρει μια σειρά λειτουργιών για να βοηθήσει τις επιχειρήσεις να διαχειρίζονται τις δραστηριότητές τους περιλαμβάνουν [13]:

- Ανεπαφές και αναγνώστης chip για αποδοχή πληρωμών
- Προσαρμόσιμος πίνακας εργαλείων για παρακολούθηση πωλήσεων και αποθέματος σε πραγματικό χρόνο
- Εργαλεία διαχείρισης αποθέματος για την παρακολούθηση των επιπέδων αποθεμάτων και τη λήψη ειδοποιήσεων όταν τα στοιχεία εξαντλούνται
- Εργαλεία διαχείρισης πελατειακών σχέσεων (CRM) για την παρακολούθηση των πληροφοριών πελατών και του ιστορικού αγορών
- Εργαλεία διαχείρισης προγραμμάτων επιβράβευσης για τη δημιουργία και τη διαχείριση προγραμμάτων επιβράβευσης
- Η Square προσφέρει επίσης μια σειρά από πρόσθετα και ενσωματώσεις, όπως διαχείριση μισθοδοσίας, μάρκετινγκ ηλεκτρονικού ταχυδρομείου και ενσωμάτωση ηλεκτρονικού εμπορίου, για να βοηθήσει τις επιχειρήσεις να προσαρμόσουν το σύστημά τους στις συγκεκριμένες ανάγκες τους.

Σχήμα 2: Logo POS Square

Lightspeed

Το Lightspeed είναι ένα σύστημα διαχείρισης αποθεμάτων POS και αποθέματος που βασίζεται σε cloud, σχεδιασμένο για επιχειρήσεις λιανικής. Μερικά από τα βασικά χαρακτηριστικά του περιλαμβάνουν[14]:

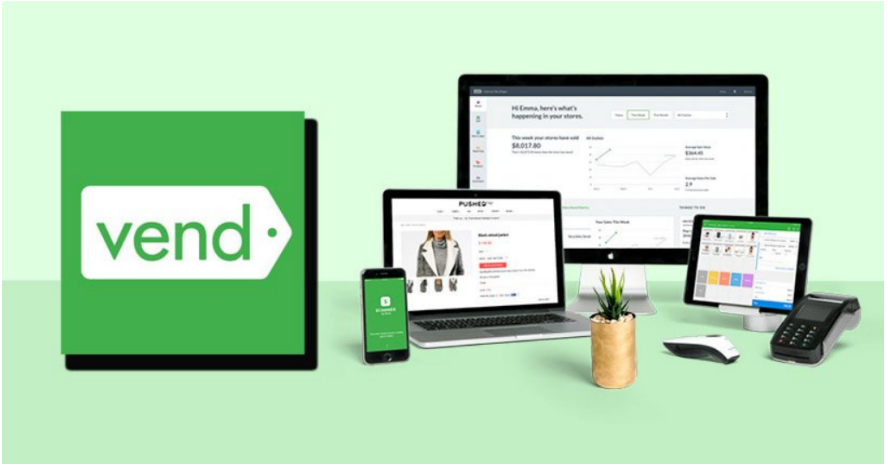
- Προσαρμόσιμος κατάλογος προϊόντων για διαχείριση αποθέματος και πωλήσεων
- Εργαλεία διαχείρισης παραγγελιών για την παρακολούθηση παραγγελιών και τη διαχείριση των επικοινωνιών με τους πελάτες
- Αυτοματοποιημένα εργαλεία αγορών για τη διαχείριση των επιπέδων αποθεμάτων και την αναπλήρωση του αποθέματος
- Εργαλεία διαχείρισης πελατών για παρακολούθηση ιστορικού αγορών και δημιουργία προφίλ πελατών
- Εργαλεία Analytics για την παρακολούθηση δεδομένων πωλήσεων και αποθέματος, καθώς και απόδοσης εργαζομένων
- Το Lightspeed προσφέρει επίσης μια σειρά από πρόσθετα και ενσωματώσεις, όπως ενσωμάτωση ηλεκτρονικού εμπορίου, κράτηση ραντεβού και διαχείριση προγραμμάτων αφοσίωσης, για να βοηθήσει τις επιχειρήσεις να προσαρμόσουν περαιτέρω το σύστημά τους.

Σχήμα 3: Logo POS Lightspeed

Vend

Το Vend είναι ένα σύστημα διαχείρισης αποθεμάτων και POS που βασίζεται σε cloud, σχεδιασμένο για μικρές και μεσαίες επιχειρήσεις. Μερικά από τα βασικά χαρακτηριστικά του περιλαμβάνουν[12]:

- Προσαρμόσιμη οθόνη ταμείου για αποδοχή πληρωμών και διαχείριση πωλήσεων
- Εργαλεία διαχείρισης αποθέματος για την παρακολούθηση των επιπέδων αποθεμάτων, τη λήψη ειδοποιήσεων και τη διαχείριση εντολών αγοράς
- Εργαλεία διαχείρισης πελατών για την παρακολούθηση δεδομένων πελατών και ιστορικού αγορών
- Εργαλεία Analytics για την παρακολούθηση των πωλήσεων, του αποθέματος και των δεδομένων απόδοσης των εργαζομένων
- Πρόσθετα και ενσωματώσεις για διαχείριση προγραμμάτων αφοσίωσης, ενσωμάτωση ηλεκτρονικού εμπορίου και πολλά άλλα
- Εκτός από αυτές τις δυνατότητες, το Vend προσφέρει επίσης δυνατότητες διαχείρισης πολλών καταστημάτων, επιτρέποντας στις επιχειρήσεις να διαχειρίζονται τα δεδομένα αποθέματος και πωλήσεων σε πολλές τοποθεσίες.



Σχήμα 4: Logo POS Vend

Συνολικά, αυτά τα συστήματα προσφέρουν μια σειρά από λειτουργίες που βοηθούν τις επιχειρήσεις να διαχειρίζονται τις δραστηριότητές τους πιο αποτελεσματικά, από την επεξεργασία πληρωμών και τη διαχείριση αποθέματος έως τη διαχείριση σχέσεων με τους πελάτες και τα αναλυτικά στοιχεία. Οι επιχειρήσεις θα πρέπει να εξετάσουν προσεκτικά τις συγκεκριμένες ανάγκες τους και να επιλέξουν ένα σύστημα που μπορεί να καλύψει καλύτερα αυτές τις ανάγκες.

1.3 Productive Manager

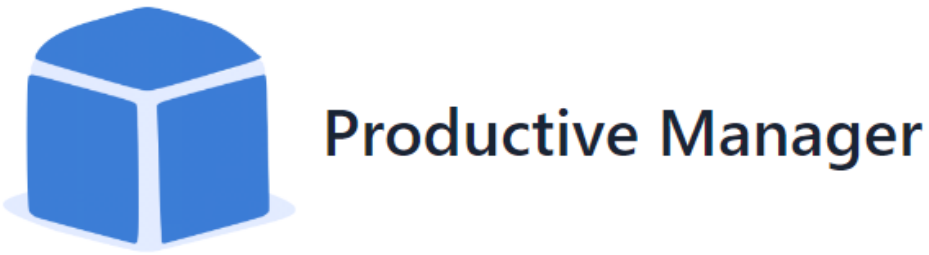
Το Productive Manager είναι ένα σύστημα διαχείρισης αποθήκης και πωλήσεων που παρέχει μια ολοκληρωμένη λύση στις επιχειρήσεις για τη διαχείριση του αποθέματος και των πωλήσεών τους. Σε σύγκριση με άλλα συστήματα, προσφέρει πολλά βασικά χαρακτηριστικά που το ξεχωρίζουν:

Φιλικό προς τον χρήστη διεπαφή: Το Productive Manager διαθέτει ένα γρήγορο και εύκολα προσβάσιμο περιβάλλον χρήστη στο οποίο μπορείτε να έχετε πρόσβαση μέσω προγραμμάτων περιήγησης στο web. Αυτό διευκολύνει τους διαχειριστές να πλοηγούνται και να χειρίζονται αποτελεσματικά το σύστημα.

Διαχείριση αποθέματος: Το σύστημα χρησιμοποιεί σύγχρονες τεχνολογίες ιστού όπως το Spring Boot και το Vaadin Flow για να παρέχει δυνατότητες διαχείρισης αποθέματος όπως εισαγωγή και μορφοποίηση προϊόντων σε ηλεκτρονική μορφή, αποθήκευση πελατών και προμηθευτών και δημιουργία αναφορών. Το σύστημα παρέχει επίσης παρακολούθηση σε πραγματικό χρόνο των επιπέδων αποθεμάτων για να βοηθήσει τις επιχειρήσεις να αποφύγουν τα αποθέματα και να μειώσουν τα απόβλητα.

Διαχείριση πωλήσεων: Το σύστημα POS του Productive Manager έχει σχεδιαστεί για να βελτιώνει την αποτελεσματικότητα και την ακρίβεια των συναλλαγών διαχείρισης πωλήσεων. Παρέχει λειτουργίες όπως επεξεργασία παραγγελιών στο κατάστημα και δημιουργία αναφορών. Με αυτό το σύστημα, οι επιχειρήσεις μπορούν εύκολα να παρακολουθούν τις πωλήσεις τους και να παρακολουθούν την απόδοσή τους με την πάροδο του χρόνου.

Συνολικά, το Productive Manager προσφέρει στις επιχειρήσεις μια ολοκληρωμένη και αποτελεσματική λύση για τη διαχείριση αποθεμάτων και πωλήσεων. Με τη φιλική προς τον χρήστη διεπαφή και τις προηγμένες δυνατότητες, οι επιχειρήσεις μπορούν να βελτιώσουν την απόδοσή τους και να μεγιστοποιήσουν τα κερδη τους.



Σχήμα 5: Logo Εφαρμογής Productive Manager

Κεφάλαιο 2ο: Τεχνολογίες Υλοποίησης

2.1 To backend

Στον κόσμο των υπολογιστών, το "backend" αναφέρεται σε οποιοδήποτε μέρος ενός ιστότοπου ή ενός προγράμματος λογισμικού που οι χρήστες δεν βλέπουν. Στην ορολογία προγραμματισμού, το backend είναι το "επίπεδο πρόσβασης δεδομένων". Οι περισσότεροι σύγχρονοι ιστότοποι είναι δυναμικοί, πράγμα που σημαίνει ότι το περιεχόμενο της ιστοσελίδας δημιουργείται εν κινήσει. Μια δυναμική σελίδα περιέχει ένα ή περισσότερα σενάρια που εκτελούνται στον διακομιστή web κάθε φορά που γίνεται πρόσβαση στη σελίδα. Αυτά τα σενάρια δημιουργούν το περιεχόμενο της σελίδας, το οποίο αποστέλλεται στο πρόγραμμα περιήγησης ιστού του χρήστη. Όλα όσα συμβαίνουν πριν από την εμφάνιση της σελίδας σε ένα πρόγραμμα περιήγησης ιστού είναι μέρος του backend.

2.2 Παραδείγματα backend διεργασιών:

- επεξεργασία ενός εισερχόμενου αιτήματος ιστοσελίδας
- εκτέλεση ενός σεναρίου (JAVA, ASP, JSP, κ.λπ.) για τη δημιουργία HTML
- πρόσβαση σε δεδομένα, όπως ένα άρθρο, από μια βάση δεδομένων χρησιμοποιώντας ερωτήματα SQL
- αποθήκευση ή ενημέρωση εγγραφών σε μια βάση δεδομένων
- χειρισμός μεταφορτώσεων και λήψεων αρχείων

Όλα τα παραπάνω παραδείγματα είναι διεργασίες του backend που εκτελούνται στον διακομιστή ιστού. Για την εργάσια στο backend η υλοποίηση εγινε με την γλώσσα προγραμματισμού Java που θα αναλυθεί παρακάτω.

2.3 Εισαγωγή στη Java

Η Java είναι μια γλώσσα προγραμματισμού γενικής χρήσης υψηλού επιπέδου, βασισμένη σε κλάσεις και αντικείμενα, σχεδιασμένη για να έχει μικρότερες εξαρτήσεις υλοποίησης ωστέ να μπορεί να τρέχει σε οποιοδήποτε λειτουργικό σύστημα. Είναι γρήγορη, ασφαλής και αξιόπιστη καθώς χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών σε φορητούς υπολογιστές, κέντρα δεδομένων, κονσόλες παιχνιδιών, επιστημονικούς υπερυπολογιστές, και κινητά τηλέφωνα.[1]

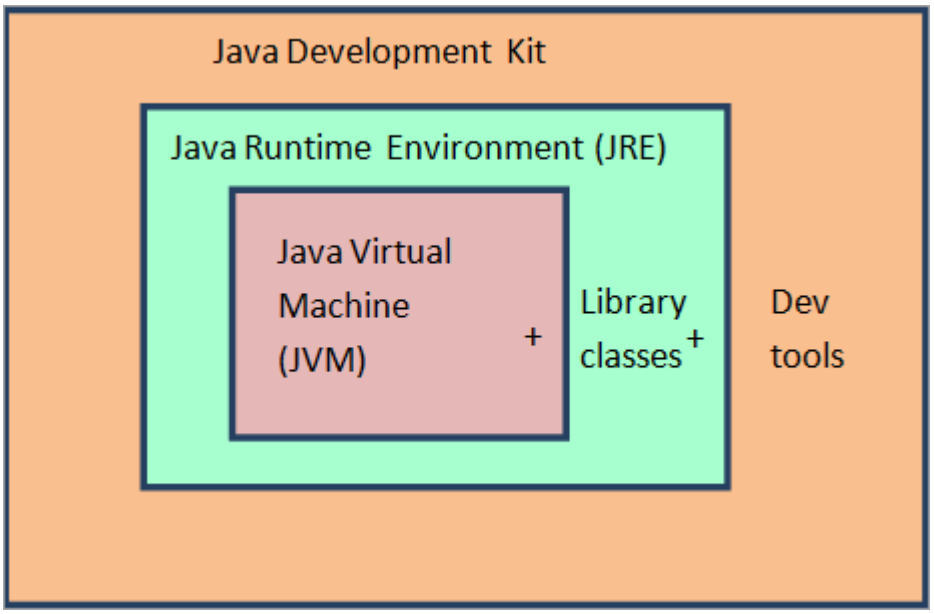
2.4 Μικρή αναφορά στην ιστορία της Java

Η γλώσσα Java αρχικά ονομαζόταν OAK και αναπτύχθηκε για το χειρισμό φορητών συσκευών και αποκωδικοποιητών ωστόσο ήταν μια τεράστια αποτυχία. Το 1995, η Sun άλλαξε το όνομα σε "Java" και τροποποίησε τη γλώσσα για να επωφεληθεί από την αναπτυσσόμενη επιχείρηση ανάπτυξης www.

2.5 Κύρια χαρακτηριστικά της Java

Είναι μια από τις πιο εύχρηστες γλώσσες προγραμματισμού για εκμάθηση. Ανεξάρτητη από την πλατφόρμα με αποτέλεσμα τα προγράμματα που έχουν αναπτυχθεί σε ένα μηχάνημα να μπορούν να εκτελεστούν σε ένα άλλο μηχάνημα. Έχει σχεδιαστεί για τη δημιουργία αντικειμενοστραφών εφαρμογών. Είναι μια γλώσσα πολλαπλών νημάτων με αυτόματη διαχείριση μνήμης. Τέλος, διευκολύνει τον κατανεμημένο υπολογισμό.

2.6 Εσωτερική δομή της γλώσσας προγραμματισμού Java



Σχήμα 6: Δομή Γλώσσας Java

Ένας προγραμματιστής γράφει ένα πρόγραμμα σε μια γλώσσα αναγνώσιμη από τον άνθρωπο που ονομάζεται Πηγαίος Κώδικας. Ωστόσο, η CPU ή τα Chips δεν κατανοούν τον πηγαίο κώδικα που είναι γραμμένος σε πηγαίος κώδικας. Οι υπολογιστές ή τα τσιπ καταλαβαίνουν μόνο ένα πράγμα, το οποίο ονομάζεται γλώσσα ή κώδικας μηχανής. Αυτοί οι κωδικοί μηχανών εκτελούνται σε επίπεδο CPU και είναι διαφορετικοί ανα μοντέλο CPU. Επομένως, θα χρειαζόταν να ασχοληθει και με για τον κώδικα του μηχανήματος μαζί με τον προγραμματισμο του πηγαίου κώδικα.

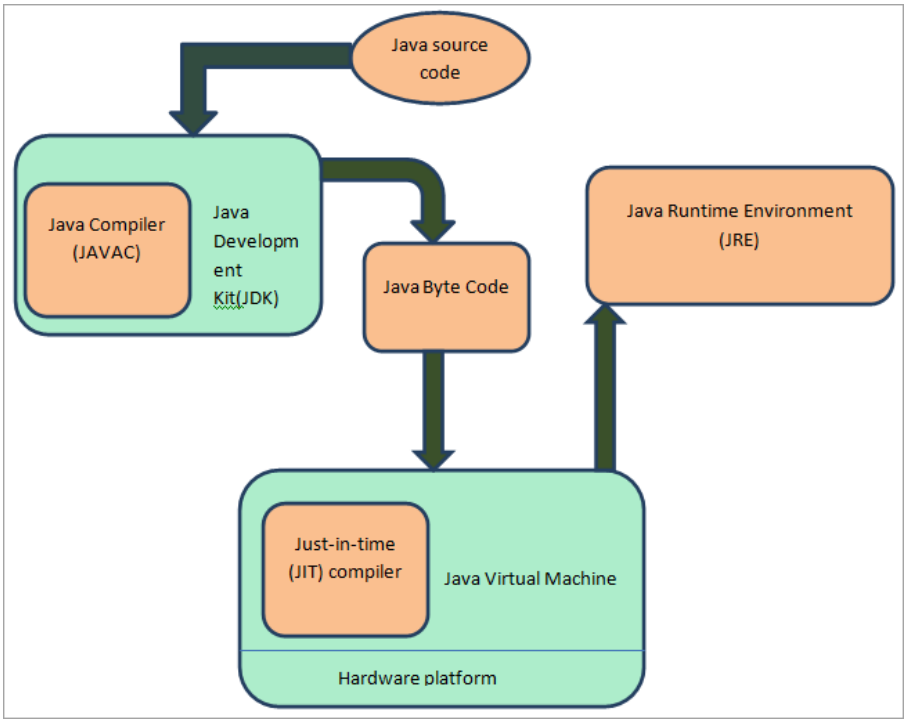
Η Java προσφέρει τα παρακάτω εργαλεία που απαλλάσσουν τον προγραμματιστή απο το παραπάνω πρόβλημα [15]:

- Το Java Development kit (JDK) το οποίο είναι ένα περιβάλλον ανάπτυξης λογισμικού που χρησιμοποιείται για τη δημιουργία μικροεφαρμογών και εφαρμογών Java. Το JDK βοηθάει στη κωδικοποιηση και την εκτέλεση προγραμμάτων Java. Είναι δυνατή η εγκατάσταση περισσότερων από μία εκδόσεων JDK στον ίδιο υπολογιστή και υποστηρίζεται σε Windows, macOS, Solaris και Linux. Οι κύριοι λόγοι για τη χρήση του JDK είναι ότι περιέχει εργαλεία που απαιτούνται για τη σύνταξη προγραμμάτων Java και το JRE για την εκτέλεσή τους. Επίσης, περιλαμβάνει έναν μεταγλωττιστή, έναν αποσφαλμάτωση και ένα πρόγραμμα εκκίνησης εφαρμογών. Ο μεταγλωττιστής μετατρέπει τον κώδικα που είναι γραμμένος σε Java σε κώδικα byte. Ο εκκινητής εφαρμογών ανοίγει ένα JRE, φορτώνει την απαραίτητη κλάση και εκτελεί την κύρια μέθοδο της.
- Το Java Virtual Machine (JVM) το οποιο είναι μέρος του του Java Run Environment (JRE) και αποτελείται από μια μηχανή που παρέχει ένα εικονικό περιβάλλον για την εκτέλεση του κώδικα Java ή των εφαρμογών. Επίσης, ενώ σε άλλες γλώσσες προγραμματισμού, ο μεταγλωττιστής παράγει κώδικα μηχανής για ένα συγκεκριμένο σύστημα ο μεταγλωττιστής της Java παράγει κώδικα για το (JVM). Με αυτόν το τρόπο το JVM παρέχει έναν ανεξάρτητο από πλατφόρμα τρόπο εκτέλεσης πηγαίου κώδικα Java. Επιπλέον, διαθέτει πολυάριθμες

βιβλιοθήκες, εργαλεία και έναν εσωτερικό μεταγλωττιστή JIT (Just-in-Time) που μετατρέπει τον πηγαίο κώδικα σε γλώσσα μηχανής χαμηλού επιπέδου επιτυγχάνοντας την πιο γρήγορη εκτέλεση του προγράμματος.

- Το Java Runtime Environment (JRE) το οποίο είναι είναι ένα κομμάτι λογισμικού που έχει σχεδιαστεί για να τρέχει τα προγράμματα Java. Περιέχει μόνο τα αναγκαία εργαλεια για την εκτέλεση της εφαρμογής οπως τι βιβλιοθήκες κλάσεων, την κλάση του φορτωτή και το JVM. Το οποίο είναι αρκετά χρήσιμο για την χρήση και εκτέλεση τον εφαρμογών στο στάδιο της παραγωγής.

2.7 Πώς παραμένει η Java ανεξάρτητη του λειτουργικού;



Σχήμα 7: Java compilation

Ο μεταγλωττιστής της Java δεν παράγει εγγενή εκτελέσιμο κώδικα για μια συγκεκριμένη μηχανή. Αντίθετα, η παράγει μια μοναδική μορφή που ονομάζεται bytecode και εκτελείται σύμφωνα με τους κανόνες που ορίζονται στην προδιαγραφή της εικονικής μηχανής. Επομένως, η Java είναι μια γλώσσα ανεξάρτητη από πλατφόρμα. Ο Bytecode είναι κατανοητός σε οποιοδήποτε JVM εγκατεστημένο σε οποιοδήποτε λειτουργικό σύστημα το οποίο έχει ως αποτέλεσμα ο πηγαίος κώδικας της java να μπορεί να εκτελεστεί σε όλα τα λειτουργικά συστήματα. [1]

2.8 OOP Java Χαρακτηριστικά για την σύνταξη του πηγαίου κώδικα

- **Αντικείμενα και κλάσεις**

Μια οντότητα που έχει κατάσταση και συμπεριφορά είναι γνωστή ως αντικείμενο, π.χ. καρέκλα, ποδήλατο, μαρκαδόρος, στυλό, τραπέζι, αυτοκίνητο κ.λπ. Μπορεί να είναι φυσικό ή λογικό (απτό και αύλο). Ένα αντικείμενο αποτελείται από τρία χαρακτηριστικά, την κατάσταση, την συμπεριφορά και την ταυτότητα. Η κατάσταση αντιπροσωπεύει τα δεδομένα ενός αντικειμένου. Η συμπεριφορά αντιπροσωπεύει την λειτουργικότητα ενός αντικειμένου όπως παραδείγματος χάρη κάνω μια αγορά ή πίνω νερό. Η ταυτότητα του αντικειμένου συνήθως υλοποιείται μέσω ενός μοναδικού αναγνωριστικού και της δεν είναι ορατή στον εξωτερικό χρήστη. Ωστόσο, χρησιμοποιείται εσωτερικά από το JVM ωστέ να αναγνωρίζει κάθε αντικείμενο μοναδικά. Για παράδειγμα, το στυλό είναι ένα αντικείμενο, το όνομά του είναι bic, το χρώμα είναι λευκό, γνωστό ως κατάσταση του και χρησιμοποιείται για τη γραφή, άρα η γραφή είναι η συμπεριφορά του. Ένα αντικείμενο είναι μια υλοποίηση μιας κλάσης. Μια κλάση είναι ένα πρότυπο ή σχεδιάγραμμα από το οποίο δημιουργούνται αντικείμενα. Άρα, ένα αντικείμενο είναι η παρουσία μιας κλάσης. Γενικότερα μια κλάση είναι η περιγραφή μιας ομάδας αντικειμένων που έχουν κοινές ιδιότητες και λειτουργίες. Περιέχει πεδια, μεθόδους, κατασκευαστές, μπλοκ εντολων και εμφωλευμενες κλασεις και διεπαφες.

- **Ενθυλάκωση**

  Η πρακτική της διατήρησης ιδιωτική πεδίων σε μιας κλάση, και στη συνέχεια η παροχή πρόσβασης σε αυτά τα πεδία μέσω δημόσιων μεθόδων. Η ενθυλάκωση είναι ένα προστατευτικό φράγμα που διατηρεί τα δεδομένα και τον κώδικα ασφαλή μέσα στην ίδια την κλάση. Με αυτόν τον τρόπο, μπορούμε να επαναχρησιμοποιήσουμε αντικείμενα όπως στοιχεία κώδικα ή μεταβλητές χωρίς να επιτρέψουμε την ανοικτή πρόσβαση στα δεδομένα μιας κλάσης σε όλο το σύστημα.

- **Κληρονομικότητα**

  Ένα ειδικό χαρακτηριστικό του Αντικειμενοστραφούς Προγραμματισμού στην Java, η κληρονομικότητα επιτρέπει στους προγραμματιστές να δημιουργούν νέες κλάσεις που μοιράζονται ορισμένα από τα χαρακτηριστικά των υπαρχουσών κλάσεων. Η χρήση της κληρονομικότητας επιτρέπει να βασιστούμε σε προηγούμενη εργασία και να την επεκτείνουμε χωρίς της επανάληψη κώδικα.

- **Πολυμορφισμός**

  Επιτρέπει στους προγραμματιστές να χρησιμοποιούν την ίδια λέξη για να σημαίνουν διαφορετικά πράγματα σε διαφορετικά περιβάλλοντα. Μια μορφή πολυμορφισμού είναι η υπερφόρτωση μεθόδου όπου ο ίδιος ο κώδικας εκτελεί διαφορετικές ενέργειες. Η άλλη μορφή είναι η παράκαμψη μεθόδου όπου ο τύπος και ο αριθμός των παρεχόμενων μεταβλητών εκτελούν διαφορετικές ενέργειες.

- **Αφαίρεση**

  Η χρήση απλών πραγμάτων για την αναπαράσταση της πολυπλοκότητας. Για παράδειγμα γνωρίζουμε πώς να ανοίγουμε την τηλεόραση, αλλά δεν χρειάζεται να ξέρουμε πώς λειτουργεί για να την ανοίξουμε. Στην Java, η αφαίρεση σημαίνει ότι απλά πράγματα όπως αντικείμενα, κλάσεις και μεταβλητές αντιπροσωπεύουν πιο πολύπλοκους υποκείμενους κώδικες και δεδομένα. Αυτό είναι επίσης σημαντικό γιατί επιτρέπει να αποφευγετε η επανάληψη της ίδιας εργασίας πολλές φορές.

2.9 Spring

2.9.1 Τι είναι το Spring

Το Spring Framework παρέχει ένα ολοκληρωμένο μοντέλο προγραμματισμού και διαμόρφωσης για σύγχρονες εφαρμογές που βασίζονται σε Java, σε κάθε είδους πλατφόρμα ανάπτυξης. Ένα βασικό στοιχείο του Spring είναι η υποστήριξη υποδομής σε επίπεδο εφαρμογών: Το Spring εστιάζει στην εύκολη εγκατάσταση και διασύνδεση των εφαρμογών, έτσι ώστε οι ομάδες να μπορούν να επικεντρωθούν στην επιχειρηματική λογική σε επίπεδο εφαρμογής, χωρίς περιττούς δεσμούς με συγκεκριμένα περιβάλλοντα ανάπτυξης. Κυκλοφόρησε τον Ιούνιο του 2003 από τον Rod Johnson με την άδεια Apache 2.0, το Spring Framework φιλοξενείται από το SourceForge. [2]

2.9.2 Γιατί Spring έγινε η επιλογή του Spring

Οι εφαρμογές Java μπορούν να γίνουν πολύπλοκες και να καταλήξουν με αρκετά σύνθετα στοιχεία. Σύνθετα στοιχεία είναι αυτά που εξαρτώνται από το υποκείμενο λειτουργικό σύστημα για την τις ιδιότητές τους. Για παράδειγμα η διασύνδεση στήν βάση δεδομένων ή υλοποίηση ενός διακομιστή ιστοσελίδων διαδικτύου. Για αυτό το Spring θεωρείται ένα ασφαλές, χαμηλού κόστους και ευέλικτο πλαίσιο. Βελτιώνει την αποτελεσματικότητα κωδικοποίησης και μειώνει τον συνολικό χρόνο ανάπτυξης εφαρμογών επειδή είναι ελαφρύ και αποτελεσματικό στη χρήση πόρων συστήματος και έχει αρκετά μεγάλη υποστήριξη. Επιπλέον, υπάρχει αρκετά μεγάλο διαθέσιμο υλικό για εκμάθηση και επίλυση προβλημάτων κατά την ανάπτυξη. Πρακτικά, το Spring διαχειρίζεται την υποδομή, ώστε οι προγραμματιστές να μπορούν να επικεντρωθούν στην εφαρμογή.

2.9.3 Βασικά χαρακτηριστικά του Spring Framework

- Inversion of Control container

  Το Inversion of Control container είναι ο πυρήνας του Spring όπου τα αντικείμενα δημιουργούνται, συνδέονται μεταξύ τους, διαμορφώνονται και διαχειρίζονται σε όλη τη διάρκεια του κύκλου ζωής τους. Χρησιμοποιεί το dependency injection ή τα μοτίβα αναζήτησης εξάρτησης για να παρέχει την αναφορά του επιθυμητού αντικειμένου κατά τη διάρκεια του χρόνου εκτέλεσης. Το Spring παρέχει πακέτα org.springframework.beans και org.springframework.context που μπορούν να χρησιμοποιηθούν για τη διευκόλυνση αυτών των λειτουργιών. Το κοντέινερ αναγνωρίζει ειδικό κωδικά σχολειασμών που απαιτείται για τη διαχείριση της διαμόρφωσης. Αυτός ο κώδικας παρέχεται από τα πακέτα org.springframework.beans και org.springframework.context που μπορούν να χρησιμοποιηθούν για τη διευκόλυνση αυτών των λειτουργιών.

- Υποστήριξη για Aspect Oriented προγραμματισμό

  Ο Aspect Oriented προγραμματισμός στοχεύει στην παροχή μεγαλύτερης λειτουργικότητας στις εργασιες, οι οποίες είναι λειτουργίες που εκτείνονται σε ολόκληρη την εφαρμογή, όπως καταχώρηση αρχείων ενεργειών, διατήρηση δεδομένων σε προσωρινή μνήμη, διαχείριση συναλλαγών και αυθεντικοποίηση. Επιπλέον, συμπληρώνει τον αντικειμενοστραφή

προγραμματισμό παρέχοντας έναν διαφορετικό τρόπο δομής του προγράμματος, σε αντίθεση με τη δομή του OOP που βασίζεται σε κλάσεις. Η λειτουργία του ξεκινάει με την δήλωση ενός Aspect και τις μεθόδους που θα είναι υπεύθυνο να διαχειρίζεται. Συνήθως προτού εκτελεστεί η μέθοδος το Aspect ενεργοποιείται και εκτελεί τις κατάλληλες ενέργειες και στην συνέχεια επιτρέπει ή οχι στην μέθοδο να συνεχίσει την εκτέλεσή της. Για παράδειγμα στην αυθεντικοποίηση πιστοποιει οτι ο χρήστης είναι συνδεδεμένος και έχει τα κατάλληλα δικαιώματα για να συνεχίσει μια ενέργεια αλλαγής ή προβολής δεδομένων.

- Πλαίσιο πρόσβασης δεδομένων

  Το ζήτημα της επικοινωνίας με την βάση δεδομένων είναι ένα από τα κοινά προβλήματα που αντιμετωπίζουν οι προγραμματιστές κατά την ανάπτυξη εφαρμογών. Το Spring απλοποιεί τη διαδικασία επικοινωνίας της βάσης δεδομένων παρέχοντας άμεση υποστήριξη με δημοφιλή πλαίσια πρόσβασης δεδομένων σε Java, όπως το JDBC, το Hibernate, και το Java Persistence API (JPA). Επιπλέον, προσφέρει λειτουργίες όπως διαχείριση πόρων, χειρισμός εξαιρέσεων και αναδίπλωση πόρων για όλα τα υποστηριζόμενα πλαίσια πρόσβασης δεδομένων, απλοποιώντας περαιτέρω τη διαδικασία ανάπτυξης.

- Πλαίσιο διαχείρισης συναλλαγών

  Το Spring προσφέρει έναν μηχανισμό για που επιτρέπει στους προγραμματιστές να εργαστούν με τοπικές, παγκόσμιες και ένθετες συναλλαγές, την αποθήκευση της κατάστασης των δεδομένων σε διάφορα σημεία και την απλοποιημένη διαχείριση συναλλαγών σε όλη την εφαρμογή. Το Πλαίσιο πρόσβασης δεδομένων ενσωματώνεται άμεσα με το Πλαίσιο διαχείρισης συναλλαγών με υποστήριξη για ανταλλαγή μηνυμάτων κατάστασης των δεδομένων και προσωρινή αποθήκευση. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν συστήματα συναλλαγών πλούσια σε χαρακτηριστικά που εκτείνονται σε όλες τις εφαρμογές χωρίς να εξαρτώνται από Java Transaction API.

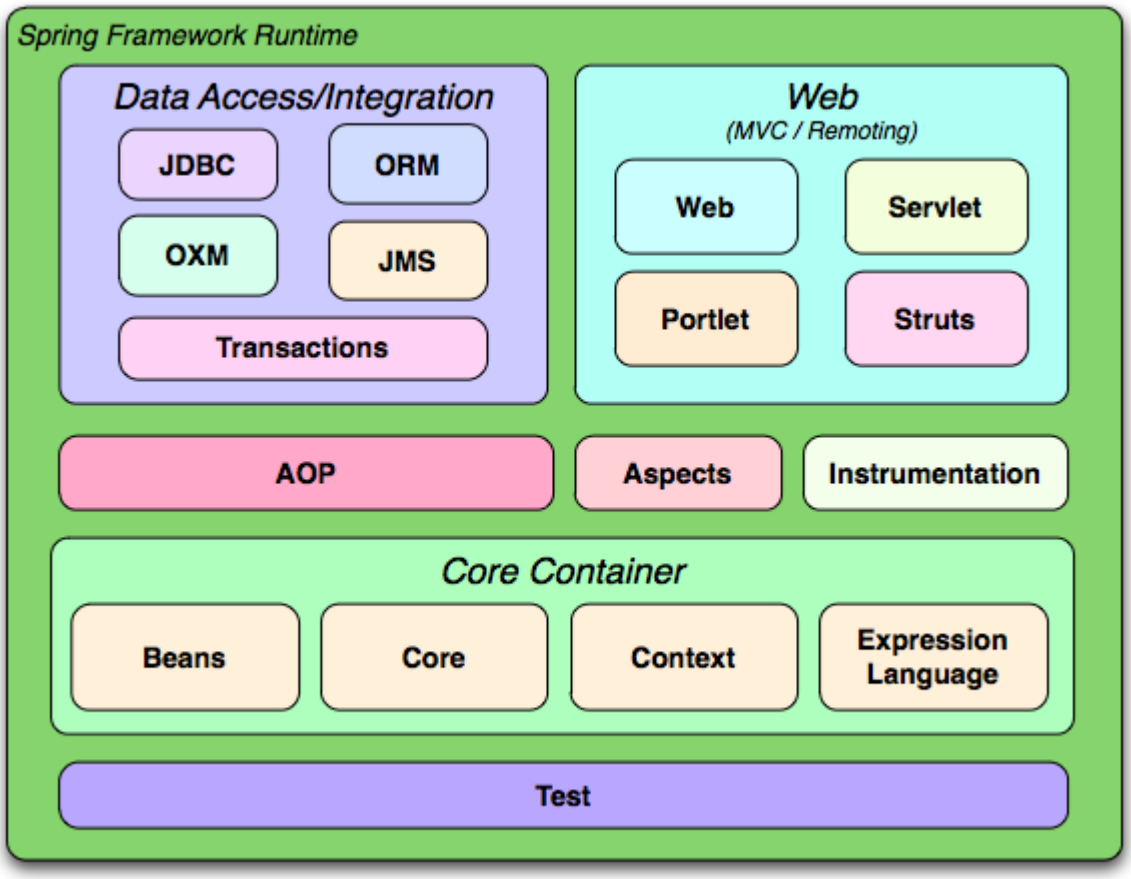
- Υποστήριξη για αρχιτεκτονική MVC

  Το Spring MVC επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές χρησιμοποιώντας το δημοφιλές μοτίβο MVC. Το MVC είναι ένα πλαίσιο βασισμένο σε αιτήματα που επιτρέπει στους προγραμματιστές να δημιουργούν εύκολα προσαρμοσμένες εφαρμογές. Το βασικό συστατικό του Spring MVC είναι η κλάση DispatcherServlet που χειρίζεται αιτήματα χρηστών και στη συνέχεια τα προωθεί στον σωστό ελεγκτή. Αυτό επιτρέπει στον ελεγκτή να επεξεργαστεί το αίτημα, να δημιουργήσει το μοντέλο και στη συνέχεια να παρέχει τις πληροφορίες στον τελικό χρήστη μέσω μιας καθορισμένης προβολής.

- Spring web service

  Το Spring Web Service παρέχει έναν βελτιστοποιημένο τρόπο δημιουργίας και διαχείρισης τελικών σημείων υπηρεσίας στην εφαρμογή. Προσφέρει μια πολυεπίπεδη προσέγγιση που μπορεί να διαχειριστεί χρησιμοποιώντας την μορφή δεδομένων XML ή JSON και μπορεί να χρησιμοποιηθεί για την παροχή αντιστοίχισης για αιτήματα ιστού σε ένα συγκεκριμένο αντικείμενο.

2.9.4 Η αρχιτεκτονική του Spring



Σχήμα 8: Spring Framework Runtime

Το παραπάνω διάγραμμα αντιπροσωπεύει τα βασικά στοιχεία της αρχιτεκτονικής του Spring.

Το Core Container περιέχει τις θεμελιώδεις ενότητες που αποτελούν τον κορμό του Spring.

- Το Core που είναι ο πυρήνας του πλαισίου και προσφέρει τις δυνατότητες όπως το inversion of control και το dependency injection.
- Το Beans που παρέχει το Beanfactory, το οποίο είναι μια εξελιγμένη εφαρμογή του εργοστασιακού μοτίβου προγραμματισμού.
- Το Context το οποίο βασίζεται στο πυρήνα και στα Beans και παρέχει ένα μέσο για πρόσβαση σε καθορισμένα αντικείμενα. Η διεπαφή ApplicationContext είναι το βασικό μέρος της λειτουργικής μονάδας Context και το Spring παρέχει υποστήριξη για αλληλεπιδράσεις τρίτων, όπως η προσωρινή αποθήκευση, η αποστολή αλληλογραφίας και οι μηχανές προτύπων.
- Το Expression Language που επιτρέπει στους προγραμματιστές να χρησιμοποιούν τη γλώσσα έκφρασης του Spring για να αλληλεπιδράσουν με τα αντικειμένου κατά τη διάρκεια εκτέλεσης.

Το Data access/integration περιλαμβάνει τις μονάδες που χρησιμοποιούνται για το χειρισμό της πρόσβασης δεδομένων και της επεξεργασίας συναλλαγών σε μια εφαρμογή.

- Το JDBC παρέχει ένα επίπεδο αφαίρεσης που εξαλείφει την ανάγκη διαχωρισμού της κωδικοποίησης SQL και του κώδικα JAVA όταν ασχολούμαστε με βάσεις δεδομένων.
- Το ORM είναι ένα επίπεδο ολοκλήρωσης για δημοφιλείς αντικειμενοσχεσιακά API χαρτογράφησης βάσεων δεδομένων όπως το JPA και το Hibernate.
- Το OXM είναι το επίπεδο αφαίρεσης που υποστηρίζει εφαρμογές αντιστοίχισης αντικειμένων/XML όπως JAXB και XStream.
- Το JMS είναι η λειτουργική μονάδα Java Messaging Service που δημιουργεί και καταναλώνει μηνύματα που ενσωματώνονται απευθείας με τη λειτουργική μονάδα ανταλλαγής μηνυμάτων.
- Το Transaction προσφέρει προγραμματική και δηλωτική διαχείριση συναλλαγών για κλάσεις που περιλαμβάνουν ειδικές διεπαφές και αντικείμενα.

Το επίπεδο Web σχετίζεται με λειτουργίες που βασίζονται στον ιστό του Spring.

- Το WebSocket το οποίο υποστηρίζει την επικοινωνία που βασίζεται στην websocket τεχνολογια για πελάτες και διακομιστές.
- Το Servlet που είναι η λειτουργική μονάδα του Spring WebMVC που περιέχει τις υλοποιήσεις MVC and REST.
- Το Web το οποίο παρέχει όλες τις βασικές λειτουργίες που θα χρειαστει ο ιστός και περιέχει έναν HTTP πελάτη και τμήματα που σχετίζονται με τον ιστό του τηλεχειρισμού Spring.
- Το Portlet το οποίο παρέχει την υλοποίηση MVC που θα χρησιμοποιηθεί σε περιβάλλον portlet.

Άλλες Ενότητες

- Το AOP το οποίο παρέχει μια υλοποίηση προγραμματισμού προσανατολισμένη στα Aspects που μπορεί να χρησιμοποιηθεί κατά τη δημιουργία εφαρμογών.
- Το Aspects τα οποία επιτρέπουν την άμεση ενσωμάτωση με την επέκταση προγραμματισμού AspectJ από το eclipse foundation.
- Το Instrumentation το οποίο υποστηρίζει της κλάσης και περιέχει τις υλοποιήσεις φόρτωσης τους για τους διακομιστές των εφαρμογών.
- Το Messaging παρέχει μια ισχυρή πλατφόρμα για τη διαχείριση των μηνυμάτων στις εφαρμογές.
- Το Test το οποίο είναι η ενότητα που υποστηρίζει δοκιμές μονάδων και ολοκλήρωσης με JUnit και TestNG.

2.9.5 Τρόπος λειτουργίας του Spring για μια εφαρμογή ιστού

Μια εφαρμογή ιστού συνήθως περιλαμβάνει τρία επίπεδα:

- Το επίπεδο προβολής το οποίο είναι το πιο εξωτερικό επίπεδο που χειρίζεται την παρουσίαση του περιεχομένου και την αλληλεπίδραση με τον χρήστη.
- Το επιχειρησιακό επίπεδο λογικής, το κεντρικό επίπεδο που ασχολείται με τη λογική ενός προγράμματος.
- Το επίπεδο πρόσβασης δεδομένων, το βαθύτερο επίπεδο που ασχολείται με την ανάκτηση δεδομένων από διάφορες πηγές όπως βάσεις δεδομένων.

Κάθε επίπεδο εξαρτάται από το άλλο για να λειτουργήσει μια εφαρμογή. Το επίπεδο παρουσίασης συνομιλεί με το επίπεδο επιχειρηματικής λογικής, το οποίο συνομιλεί με το επίπεδο πρόσβασης δεδομένων. Η εξάρτηση είναι αυτό που χρειάζεται κάθε επίπεδο για να εκτελέσει τη λειτουργία του. Μια τυπική εφαρμογή έχει χιλιάδες κλάσεις και πολλές εξαρτήσεις.

Χωρίς το Spring Framework, ο κώδικας εφαρμογής τείνει να είναι στενά συνδεδεμένος, κάτι που δεν θεωρείται καλή πρακτική κωδικοποίησης. Ο χαλαρός σύνδεσμος είναι ιδανικός επειδή τα χαλαρά συζευγμένα εξαρτήματα είναι ανεξάρτητα, που σημαίνει ότι οι αλλαγές σε ένα δεν θα επηρεάσουν τη λειτουργία άλλων.

Η βασική λογική του Spring είναι το Dependency Injection ουσιαστικά η αποδέσμευση της εξάρτησης και η αυτόματη διασύνδεση τους. Το Dependency Injection είναι ένα μοτίβο προγραμματισμού που επιτρέπει στους προγραμματιστές να δημιουργήσουν περισσότερες αποσυνδεδεμένες αρχιτεκτονικές. Αυτό είναι δυνατό γιατί το Spring κατανοεί τους διαφορετικούς σχολιασμούς της Java που βάζει ένας προγραμματιστής στην υπογραφή των κλάσεων. Έτσι, το Spring γνωρίζει ότι ο προγραμματιστής θέλει να δημιουργήσει μια παρουσία μιας κλάσης και ότι αυτό πρέπει να τη διαχειριστεί. Επίσης, κατανοεί και την εξάρτηση ανάμεσα στις κλάσεις και διασφαλίζει ότι όλες οι κλάσεις που δημιουργήθηκαν έχουν σωστά συμπληρωμένες εξαρτήσεις.

Για να δημιουργήσει το Spring Framework αντικείμενα και να συμπληρώσει τις εξαρτήσεις, ένας προγραμματιστής απλώς λέει στο Spring ποια αντικείμενα να διαχειριστεί και ποιες είναι οι εξαρτήσεις για κάθε κλάση. Αυτό γίνεται χρησιμοποιώντας σχολιασμούς όπως:

@component - Ενημερώνει το Spring ποιες κλάσεις να διαχειριστεί και να δημιουργήσει. Επισημαίνει τα Beans τα οποία είναι τα αντικείμενα που είναι να δημιουργηθούν, ως διαχειριζόμενα στοιχεία, με αποτέλεσμα να είναι έτοιμα να διασυνδεδεθούν με τις κλάσεις που τα χρειάζονται.

@autowired - Ενημερώνει το Spring πώς θα χρειαστεί ένα Bean μιας κλάσης, έτσι όταν ξεκινήσει η εφαρμογή αυτό να αρχίζει να αναζητά αυτήν την κλάση μεταξύ των component για να βρει μια κατάλληλη κλάση. Αυτό απαλλάσσει τους προγραμματιστές από την διασύνδεση με κώδικα και επιτρέπει στο Spring να βρει τι πρέπει να εγχυθεί και πού.

2.10 Spring boot

2.10.1 Τι είναι το Spring boot

Το Spring Boot είναι ένα πλαίσιο ανοιχτού κώδικα που βασίζεται στη Java και χρησιμοποιείται για να διευκολύνει τη δημιουργία ενός micro Service. Αναπτύχθηκε από την Pivotal Team για την κατασκευή αυτόνομων και έτοιμων για παραγωγή εφαρμογών Spring. [2]

2.10.2 Micro service

Τα Micro Service είναι μια αρχιτεκτονική που επιτρέπει στους προγραμματιστές να αναπτύσσουν και να ενεργοποιήσουν υπηρεσίες ανεξάρτητα. Κάθε υπηρεσία που εκτελείται έχει τη δική της διαδικασία και με αποτέλεσμα να έχει ένα πιο ελαφρύ μοντέλο υποστήριξης επιχειρηματικών εφαρμογών. Τα πλεονεκτήματα αυτής της αρχιτεκτονικής είναι η εύκολη ανάπτυξη, απλή επεκτασιμότητα, η συμβατότητα με Containers, λιγότερη διαμόρφωση και μικρότερο χρόνο παραγωγής.

2.10.3 Πλεονεκτήματα Spring boot

Τα πλεονεκτήματα που προσφέρει το Spring boot είναι αρκετά ώστε ένας προγραμματιστής να το επιλέξει. Για αρχή παρέχει έναν ευέλικτο τρόπο ρύθμισης των παραμέτρων των Javabeans, των διαμορφώσεων XML και των συναλλαγών βάσης δεδομένων. Επιπλέον, παρέχει βοήθεια στην επεξεργασία batch ενεργειών και την διαχείριση των τελικών σημείων REST. Επίσης στο Spring Boot, τα πάντα ρυθμίζονται από την αρχή αυτόματα χωρίς να χρειάζονται χειροκίνητες ρυθμίσεις από τον

προγραμματιστή. Τέλος, περιλαμβάνει έναν ενσωματωμένο Servlet Container για να τρέξουν οι εφαρμογές.

2.10.4 Πώς λειτουργεί το Spring Boot

Το Spring Boot διαμορφώνει αυτόματα την εφαρμογή με βάση τις εξαρτήσεις που έχουν προστεθεί στην εφαρμογή χρησιμοποιώντας τον σχολιασμό @EnableAutoConfiguration. Για παράδειγμα, εάν μία εφαρμογή χρειάζεται να συνδεθεί με μια βάση δεδομένων MySQL που βρίσκεται μέσα στα αρχεία της εφαρμογής. αλλά δεν έχει διαμορφωθεί καμία βάσης δεδομένων, τότε το Spring Boot θα διαμορφώσει αυτόματα μια βάση δεδομένων στη μνήμη. Το σημείο έναρξης της εφαρμογής spring boot είναι η κλάση που περιέχει τον σχολιασμό @SpringBootApplication και την κύρια μέθοδο. Το Spring Boot σαρώνει αυτόματα όλα τα στοιχεία που περιλαμβάνονται στην εφαρμογή χρησιμοποιώντας τον σχολιασμό @ComponentScan.

2.11 SQL

2.11.1 Τι είναι η SQL

Η Structured query language (SQL) είναι μια γλώσσα προγραμματισμού για την αποθήκευση και την επεξεργασία πληροφοριών σε μια σχεσιακή βάση δεδομένων. Μια σχεσιακή βάση δεδομένων αποθηκεύει πληροφορίες σε μορφή πίνακα, με γραμμές και στήλες που αντιπροσωπεύουν διαφορετικά χαρακτηριστικά δεδομένων και τις διάφορες σχέσεις μεταξύ των δεδομένων. Χρησιμοποιεί ερωτήματα SQL για την αποθήκευση, ενημέρωση, αφαίρεση, αναζήτηση και ανάκτηση πληροφοριών από τη βάση δεδομένων. Επίσης η SQL μπορεί να χρησιμοποιηθεί για την διατηρηση και την βελτιστοποίηση της απόδοσης της σχεσιακής βάσης δεδομένων.[7]

2.11.2 Γιατί χρησιμοποιείται η SQL

Η Structured query language (SQL) είναι μια δημοφιλής γλώσσα ερωτημάτων που χρησιμοποιείται συχνά σε όλους τους τύπους εφαρμογών. Οι αναλυτές δεδομένων και οι προγραμματιστές μαθαίνουν και χρησιμοποιούν την SQL επειδή ενσωματώνεται καλά με διαφορετικές γλώσσες προγραμματισμού. Για παράδειγμα, μπορούν να ενσωματώσουν ερωτήματα SQL με τη γλώσσα προγραμματισμού Java για τη δημιουργία εφαρμογών επεξεργασίας δεδομένων υψηλής απόδοσης με μεγάλα συστήματα βάσεων δεδομένων SQL όπως το Oracle ή το MS SQL Server. Η εκμάθηση της SQL είναι επίσης αρκετά εύκολη καθώς χρησιμοποιεί κοινές αγγλικές λέξεις-κλειδιά στις δηλώσεις της

2.11.3 Ιστορία της SQL

Η γλώσσα προγραμματισμού SQL αναπτύχθηκε τη δεκαετία του 1970 από τους ερευνητές της IBM Raymond Boyce και Donald Chamberlin. Η γλώσσα προγραμματισμού, γνωστή τότε ως SEQUEL, δημιουργήθηκε μετά την εργασία του Edgar Frank Codd, «A Relational Model of Data for Large Shared Data Banks», το 1970. Στην εργασία του, ο Codd πρότεινε όλα τα δεδομένα σε μια βάση δεδομένων να αντιπροσωπεύονται σε σχέσεις. Με βάση αυτή τη θεωρία, οι Boyce και Chamberlin κατέληξαν στην SQL. Στο Oracle Quick Guides (Cornelio Books, 2013), ο συγγραφέας Malcolm Coxall γράφει ότι η αρχική έκδοση της SQL σχεδιάστηκε για να χειρίζεται και να ανακτά δεδομένα που είναι αποθηκευμένα στο αρχικό σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων της IBM, System R. Ωστόσο, μόλις αρκετά χρόνια αργότερα, η γλώσσα SQL έγινε δημόσια διαθέσιμη. Το 1979, μια εταιρεία που ονομάζεται Relational Software, η οποία αργότερα έγινε Oracle, κυκλοφόρησε εμπορικά τη δική της έκδοση της SQL, που ονομάζεται Oracle V2. Έκτοτε, το Αμερικανικό Εθνικό

Ινστιτούτο Προτύπων (ANSI) και ο Διεθνής Οργανισμός Τυποποίησης έχουν θεωρήσει την SQL την τυπική γλώσσα στην επικοινωνία σχεσιακών βάσεων δεδομένων. Ενώ οι μεγάλοι προμηθευτές SQL τροποποιούν τη γλώσσα σύμφωνα με τις επιθυμίες τους, οι περισσότεροι βασίζουν τα SQL προγράμματά τους στην έκδοση από το ANSI.

2.11.4 Ποια είναι τα στοιχεία ενός συστήματος SQL;

Τα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων χρησιμοποιούν τη γλώσσα δομημένης αναζήτησης (SQL) για την αποθήκευση και τη διαχείριση δεδομένων. Το σύστημα αποθηκεύει πολλούς πίνακες βάσεων δεδομένων που σχετίζονται μεταξύ τους. Τα MS SQL Server, MySQL ή MS Access είναι παραδείγματα συστημάτων διαχείρισης σχεσιακών βάσεων δεδομένων. Τα παρακάτω είναι τα στοιχεία ενός τέτοιου συστήματος.

- Πίνακας SQL

  Ένας πίνακας SQL είναι το βασικό στοιχείο μιας σχεσιακής βάσης δεδομένων. Ο πίνακας βάσης δεδομένων SQL αποτελείται από γραμμές και στήλες. Οι μηχανικοί βάσεων δεδομένων δημιουργούν σχέσεις μεταξύ πολλών πινάκων βάσεων δεδομένων για τη βελτιστοποίηση του χώρου αποθήκευσης δεδομένων. Για παράδειγμα, ένας πίνακας SQL για προϊόντα σε ένα κατάστημα θα είχε την ακόλουθη μορφή:

id	Product Name	Color ID
0001	Mattress	Color 1
0002	Pillow	Color 2

Πίνακας 1: Πίνακας SQL Product

Στη συνέχεια, ο πίνακα προϊόντων συνδέεται με τον πίνακα χρωμάτων με το Color ID:

Color ID	Color Name
Color 1	Blue
Color 2	Red

Πίνακας 2: Πίνακας SQL Colour

- SQL ερωτήματα

  Οι δηλώσεις SQL, ή τα ερωτήματα SQL, είναι έγκυρες οδηγίες που κατανοούν τα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Οι προγραμματιστές λογισμικού δημιουργούν δηλώσεις SQL χρησιμοποιώντας διαφορετικά στοιχεία γλώσσας SQL. Τα στοιχεία γλώσσας SQL είναι στοιχεία όπως αναγνωριστικά, μεταβλητές και συνθήκες αναζήτησης που

σχηματίζουν μια σωστή πρόταση SQL. Για παράδειγμα, η ακόλουθη πρόταση SQL χρησιμοποιεί μια εντολή SQL INSERT για να αποθηκεύσει την επωνυμία στρώματος A, με τιμή 499 $, σε έναν πίνακα που ονομάζεται Mattress_table, με ονόματα στηλών brand_name και cost:

INSERT INTO Mattress_table (brand_name, cost) VALUES('A','499');

Επιπλέον, υπάρχει η δυνατότητα να δημιουργηθούν αποθηκευμένες διαδικασίες. Οι οποίες είναι μια συλλογή από μία ή περισσότερες δηλώσεις SQL που είναι αποθηκευμένες στη σχεσιακή βάση δεδομένων. Οι προγραμματιστές λογισμικού χρησιμοποιούν τις αποθηκευμένες διαδικασίες για να βελτιώσουν την αποδοτικότητα και την απόδοση. Για παράδειγμα, μπορούν να δημιουργήσουν μια αποθηκευμένη διαδικασία για την ενημέρωση πινάκων πωλήσεων αντί να γράφουν την ίδια δήλωση SQL σε διαφορετικές εφαρμογές.

2.11.5 Τρόπος λειτουργίας της SQL

Η υλοποίηση της δομημένης γλώσσας ερωτημάτων (SQL) περιλαμβάνει μια μηχανή διακομιστή που επεξεργάζεται τα ερωτήματα της βάσης δεδομένων και επιστρέφει τα αποτελέσματα. Η διαδικασία SQL διέρχεται από διάφορα στοιχεία λογισμικού, συμπεριλαμβανομένων των παρακάτω.

- Parser

  Ο αναλυτής ξεκινά με διακριτικό ή αντικαθιστώντας ορισμένες από τις λέξεις στην πρόταση SQL με ειδικά σύμβολα. Στη συνέχεια ελέγχει τη δήλωση για τα ακόλουθα:

  - Ορθότητα

    Ο αναλυτής επαληθεύει ότι η πρόταση SQL συμμορφώνεται με τη σημασιολογία ή τους κανόνες SQL που διασφαλίζουν την ορθότητα της πρότασης ερωτήματος. Για παράδειγμα, ο αναλυτής ελέγχει εάν η εντολή SQL τελειώνει με άνω και κάτω τελεία. Εάν λείπει το ερωτηματικό, ο αναλυτής επιστρέφει ένα σφάλμα.

  - Εξουσιοδότηση

    Ο αναλυτής επικυρώνει επίσης ότι ο χρήστης που εκτελεί το ερώτημα έχει την απαραίτητη εξουσιοδότηση για να χειριστεί τα αντίστοιχα δεδομένα. Για παράδειγμα, μόνο οι διαχειριστές ενδέχεται να έχουν το δικαίωμα να διαγράψουν δεδομένα.

- Relational engine

  Η σχεσιακή μηχανή, ή ο επεξεργαστής ερωτημάτων, δημιουργεί ένα σχέδιο για την ανάκτηση, εγγραφή ή ενημέρωση των αντίστοιχων δεδομένων με τον πιο αποτελεσματικό τρόπο. Για παράδειγμα, ελέγχει για παρόμοια ερωτήματα, επαναχρησιμοποιεί προηγούμενες μεθόδους χειρισμού δεδομένων ή δημιουργεί μια νέα. Γράφει το σχέδιο σε μια αναπαράσταση ενδιάμεσου επιπέδου της δήλωσης SQL που ονομάζεται κώδικας byte. Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν κώδικα byte για την αποτελεσματική εκτέλεση αναζητήσεων και τροποποιήσεων στη βάση δεδομένων.

- Storage engine

  Η μηχανή αποθήκευσης, ή μηχανή βάσης δεδομένων, είναι το στοιχείο λογισμικού που επεξεργάζεται τον κώδικα byte και εκτελεί την προβλεπόμενη δήλωση SQL. Η μηχανή αυτή

διαβάζει και αποθηκεύει τα δεδομένα στα αρχεία της βάσης δεδομένων σε φυσικό χώρο αποθήκευσης του δίσκου. Με την ολοκλήρωση, η μηχανή αποθήκευσης επιστρέφει το αποτέλεσμα στην αίτηση που το ζήτησε.

2.11.6 Ποιες είναι οι εντολές της SQL

Οι εντολές δομημένης γλώσσας ερωτημάτων (SQL) είναι συγκεκριμένες λέξεις-κλειδιά ή δηλώσεις SQL που χρησιμοποιούν οι προγραμματιστές για να χειριστούν τα δεδομένα που είναι αποθηκευμένα σε μια σχεσιακή βάση δεδομένων και κατηγοριοποιούνται ως εξής:

- Data definition language

  Η γλώσσα ορισμού δεδομένων data definition language (DDL) αναφέρεται σε εντολές SQL που σχεδιάζουν τη δομή της βάσης δεδομένων. Οι μηχανικοί βάσεων δεδομένων χρησιμοποιούν το DDL για να δημιουργήσουν και να τροποποιήσουν αντικείμενα βάσης δεδομένων με βάση τις επιχειρηματικές απαιτήσεις. Για παράδειγμα, ο μηχανικός της βάσης δεδομένων χρησιμοποιεί την εντολή CREATE για να δημιουργήσει αντικείμενα βάσης δεδομένων όπως πίνακες, προβολές και ευρετήρια.

Command Name	Description
CREATE	Make a new table or database
ALTER	Change/Alter the structure of the table
TRUNCATE	Delete records/rows of a table
DROP	Delete a table definition
RENAME	Change name of the table

Σχήμα 9: SQL Data Definition Language

- Data query language

  Η γλώσσα ερωτημάτων δεδομένων data query language (DQL) αποτελείται από οδηγίες για την ανάκτηση δεδομένων που είναι αποθηκευμένα σε σχεσιακές βάσεις δεδομένων. Οι εφαρμογές λογισμικού χρησιμοποιούν την εντολή SELECT για να φιλτράρουν και να επιστρέφουν συγκεκριμένα αποτελέσματα από έναν πίνακα SQL.

Command Name	Description
SELECT	Retrieve data from tables in the database

Σχήμα 10: SQL Data Query Language

- Data manipulation language

  Οι δηλώσεις γλώσσας χειρισμού δεδομένων data manipulation language (DML) γράφουν νέες πληροφορίες ή τροποποιούν υπάρχουσες εγγραφές σε μια σχεσιακή βάση δεδομένων. Για παράδειγμα, μια εφαρμογή χρησιμοποιεί την εντολή INSERT για να αποθηκεύσει μια νέα εγγραφή στη βάση δεδομένων.

Command Name	Description
INSERT	Addition of new rows
UPDATE	Update an existing row
DELETE	Delete a row

Σχήμα 11: SQL Data Manipulation Language

- Data control language

  Οι διαχειριστές βάσεων δεδομένων χρησιμοποιούν τη γλώσσα ελέγχου δεδομένων data control language (DCL) για τη διαχείριση ή την εξουσιοδότηση πρόσβασης στη βάση δεδομένων για άλλους χρήστες. Για παράδειγμα, μπορούν να χρησιμοποιήσουν την εντολή GRANT για να επιτρέψουν σε ορισμένες εφαρμογές να χειριστούν έναν ή περισσότερους πίνακες.

Command Name	Description
GRANT	Give authorization to a user
REVOKE	Take back authorization of a user

Σχήμα 12: SQL Data Control Language

- Transaction control language

  Η σχεσιακή μηχανή χρησιμοποιεί τη γλώσσα ελέγχου συναλλαγών Transaction control language (TCL) για να κάνει αυτόματα αλλαγές στη βάση δεδομένων. Για παράδειγμα, η βάση δεδομένων χρησιμοποιεί την εντολή ROLLBACK για την αναίρεση μιας λανθασμένης συναλλαγής.

Command Name	Description
COMMIT	Make permanent changes
ROLLBACK	Reverse changes
SAVEPOINT	Make temporary changes

Σχήμα 13: SQL Transaction Control Language

2.12 H2 Database

Σχήμα 14: H2 Database Logo

2.12.1 Εισαγωγή στην βάση δεδομένων H2

Η H2 Database είναι μια ελαφριά ενσωματωμένη βάση δεδομένων Java ανοιχτού κώδικα. Μπορεί να ενσωματωθεί σε εφαρμογές Java ή να εκτελεστεί σε λειτουργία πελάτη-διακομιστή. Η βάση δεδομένων H2 μπορεί να ρυθμιστεί ώστε να εκτελείται ως βάση δεδομένων μνήμης, πράγμα που σημαίνει ότι τα δεδομένα θα αποθηκευτούν στη μνήμη και δεν θα παραμείνουν στο δίσκο. Επειδή είναι ενσωματωμένη βάσης δεδομένων συνήθως δεν χρησιμοποιείται κατά την ανάπτυξη της εφαρμογής στην παραγωγή, αλλά χρησιμοποιείται κυρίως για ανάπτυξη και δοκιμή της εφαρμογής.[6]

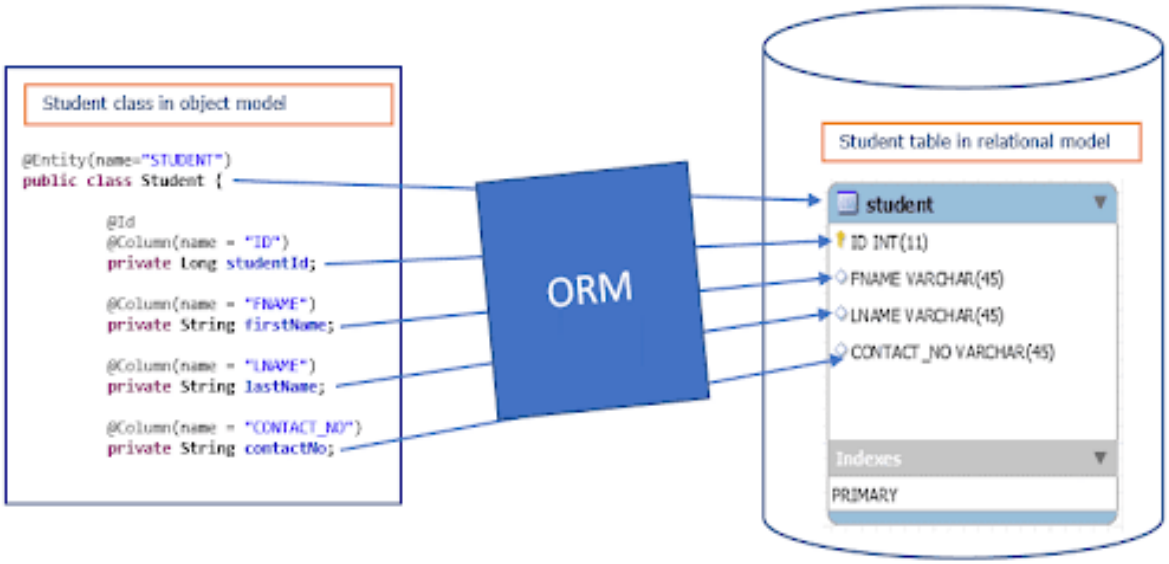
2.12.2 Χαρακτηριστικά της βάσης δεδομένων H2

Είναι μια εξαιρετικά γρήγορη μηχανή βάσης δεδομένων. Είναι ανοιχτού κώδικα και γραμμένη σε Java. Υποστηρίζει τα ερωτήματα SQL και την χρήση του JDBC API. Μπορεί επίσης να χρησιμοποιήσει το πρόγραμμα οδήγησης PostgreSQL ODBC. Έχει ενσωματωμένη και λειτουργία διακομιστή. Επίσης υποστηρίζει ομαδοποίηση και συγχρονισμό πολλών εκδόσεων. Τέλος, διαθέτει ισχυρά χαρακτηριστικά ασφαλείας.

2.13 Object Relational Mapping

2.13.1 Τί είναι το Object Relational Mapping

Το ORM ή το Object Relational Mapping είναι ένα σύστημα που αναλαμβάνει την αντιστοίχισης ενός αντικειμένου με μια οντότητα της σχεσιακής βάσης. Αυτό σημαίνει ότι είναι υπεύθυνο να αποθηκεύει τα δεδομένα του αντικειμένου στο σχεσιακό μοντέλο και να ανακτά τα δεδομένα από το σχεσιακό μοντέλο για να ενημερώσει το αντικείμενο. [3][11]



Σχήμα 15: Object Relational Mapping

2.13.2 Η ανάγκη της χρήσης του Object Relational Mapping

Το αντικειμενοστραφές μοντέλο χρησιμοποιεί κλάσεις ενώ η σχεσιακή βάση δεδομένων χρησιμοποιεί πίνακες με αποτέλεσμα να δημιουργείται ένα κενό. Λόγω της διαφοράς μεταξύ των δύο διαφορετικών μοντέλων, η μεταφορά των δεδομένων και των συσχετίσεων από αντικείμενα σε δομή σχεσιακού πίνακα και αντίστροφα απαιτεί πολύ σύνθετο προγραμματισμό. Η φόρτωση και η αποθήκευση αντικειμένων χρησιμοποιώντας μια σχεσιακή βάση δεδομένων εκθέτει τα παρακάτω 5 προβλήματα αναντιστοιχίας.

- Granularity

  Είναι ο βαθμός λεπτομέρειας στον οποίο ένα σύστημα θα μπορούσε να αναλυθεί σε μικρά μέρη. Για παράδειγμα τα στοιχεία ενος ατομου σε αντικείμενα, θα μπορούσαν να αναλύθουν σε δύο μέρη, το ένα είναι το ίδιο το άτομο και το άλλο η διεύθυνση του έτσι ώστε να πετύχουμε επαναχρησιμοποίηση του κώδικα, αφού και τα άλλα άτομα θα χρησιμοποιούν τις διευθύνσεις. Ωστόσο στην αποθήκευση των στοιχείων του ατόμου στη βάση δεδομένων υπάρχει μόνο ένας πίνακας που ονομάζεται άτομο. Αυτό έχει ως αποτέλεσμα ένα μοντέλο αντικειμένου να έχει περισσότερες κλάσεις από τον αριθμό των αντίστοιχων πινάκων στη βάση δεδομένων. Αυτή είναι η αναντιστοιχία της του βαθμού της λεπτομέρειας μεταξύ του μοντέλου αντικειμένου και του σχεσιακού μοντέλου.

- Inheritance

  Η κληρονομικότητα είναι μια φυσική έννοια στις αντικειμενοστρεφείς γλώσσες προγραμματισμού. Για παράδειγμα, ένα αντικείμενο τύπου τρίγωνου μπορεί να επεκτείνει τις λειτουργίες ενός αντικειμένου τύπου σχήματος. Ωστόσο, τα RDBMS δεν ορίζουν τίποτα παρόμοιο στο σύνολό τους (ορισμένες βάσεις δεδομένων έχουν υποστήριξη υποτύπων, αλλά είναι μη τυποποιημένη). Αυτή είναι η αναντιστοιχία κληρονομικότητας μεταξύ του μοντέλου αντικειμένου και του σχεσιακού μοντέλου.

- Identity

  Ένα RDBMS ορίζει ακριβώς μια έννοια της «ομοιότητας»: το κύριο κλειδί. Η Java, ωστόσο, ορίζει τόσο οτι το αντικείμενο ορίζει την ταυτότητα αντικειμένου a==b όσο και την ισότητα αντικειμένου a.equals(b). Αυτή είναι η αναντιστοιχία ταυτότητας μεταξύ του μοντέλου αντικειμένου και του σχεσιακού μοντέλου.

- Associations

  Οι συσχετίσεις αντιπροσωπεύονται ως αναφορές μονής κατεύθυνσης σε αντικειμενοστρεφείς γλώσσες, ενώ οι συσχετίσεις RDBMS είναι αμφίδρομες χρησιμοποιώντας ξένα κλειδιά. Εάν χρειάζονται αμφίδρομες σχέσεις στην Java, πρέπει να οριστεί η συσχέτιση δύο φορές. Ομοίως, δεν μπορεί να προσδιοριστεί η πολλαπλότητα μιας σχέσης δύο αντικειμένων.

- Data Navigation

  Ο τρόπος με τον οποίο γίνεται η πρόσβαση στα δεδομένα στην Java είναι θεμελιωδώς διαφορετικός από τον τρόπο που το γίνεται σε μια σχεσιακή βάση δεδομένων. Στην Java, η πλοήγηση γίνεται από τη μια συσχέτιση σε μια άλλη, μετακινωντας ετσι στο δίκτυο των αντικειμένων. Ωστοσο, αυτός ο τρόπος δεν είναι αποτελεσματικός για την ανάκτησης δεδομένων από μια σχεσιακή βάση δεδομένων. Γιατί στην SQL ένα τέτοιο ερώτημα θα είχε αρκετά περιττο φόρτο, καθώς θα φορτονόντουσαν παραπάνω οντότητες από όσες ιδανικά χρειάζονται για να επιτευχθεί ο σκοπός του ερωτήματος.

2.13.3 Η επίλυση της αναντιστοιχία μεταξύ αντικειμένου και οντότητας

Ένα εργαλείο σχεσιακής αντιστοίχισης αντικειμένων ORM παρέχει μια απλή διεπαφή για την αποθήκευση και την ανάκτηση αντικειμένων Java απευθείας από και προς τη σχεσιακή βάση δεδομένων. Το ORM είναι τεχνική που επιτρέπει σε μια εφαρμογή γραμμένη σε αντικειμενοστραφή γλώσσα να αντιμετωπίζει τις πληροφορίες ως αντικείμενα, αντί να χρησιμοποιεί συγκεκριμένες έννοιες της βάσης δεδομένων, όπως Γραμμές, Στήλες και Πίνακες που διευκολύνονται από την απεικόνιση Αντικειμένου/Οντότητας.

2.13.4 Πλεονεκτήματα των εργαλείων ORM

Τα εργαλεία σχεσιακής αντιστοίχισης αντικειμένων ORM προσδίδουν στην καλύτερη αρχιτεκτονική του συστήματος, μειώνουν τον χρόνο ανάπτυξης της εφαρμογής και παρέχουν προσωρινή αποθήκευση και διαχείριση συναλλαγών.

- Καλύτερη Αρχιτεκτονική Συστήματος

  Ένα εργαλείο ORM σχεδιασμένο από πολύ έμπειρους αρχιτέκτονες λογισμικού θα εφαρμόσει αποτελεσματικά μοτίβα σχεδιασμού που σχεδόν αναγκάζουν τον προγραμματιστή να χρησιμοποιήσει καλές πρακτικές προγραμματισμού σε μια εφαρμογή. Αυτό μπορεί να βοηθήσει στην υποστήριξη ενός καθαρού διαχωρισμού των χαρακτηριστικων και της ανεξάρτητης ανάπτυξης που επιτρέπει την παράλληλη, ταυτόχρονη ανάπτυξη των επιπέδων εφαρμογής. Εάν δημιουργηθεί μια κλάση πρόσβασης δεδομένων από βιβλιοθήκες του ORM, μπορεί εύκολα να επαναχρησιμοποιηθούν τα αντικείμενα δεδομένων σε μια ποικιλία

εφαρμογών. Με αυτόν τον τρόπο θα αυξηθεί και η επαναχρησιμοποίηση του κώδικα με αποτέλεσμα να υπάρχει καθαρότερος διαχωρισμός του επιπέδου πρόσβασης δεδομένων.

- Μείωση του χρόνου ανάπτυξης

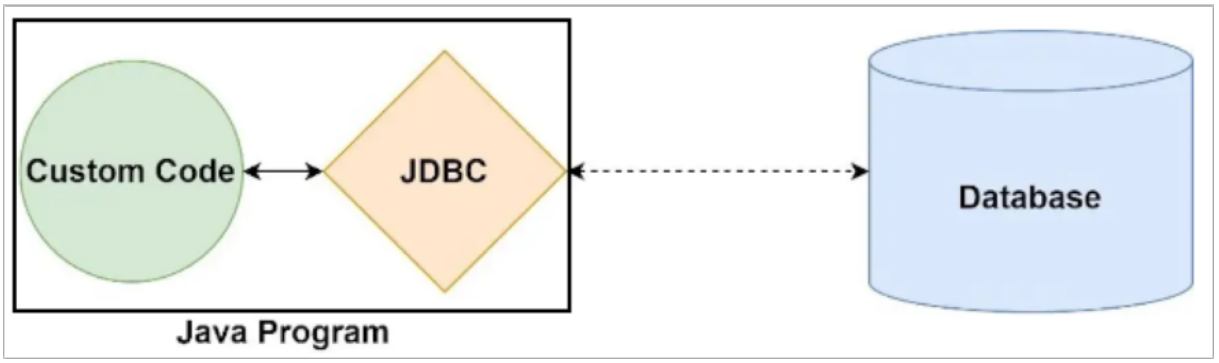
  Τις περισσότερες φορές, ο κώδικας πρόσβασης στη βάση δεδομένων είναι μια απλή εισαγωγή, ενημέρωση ή διαγραφή. Αυτά τα ερωτήματα SQL μερικές φορές μπορεί να γίνουν αρκετά επαναλαμβανόμενα στον κώδικα. Επίσης εάν χρησιμοποιηθεί το JDBC δεν γίνεται να αποθηκευτούν τα αντικείμενα απευθείας στη βάση δεδομένων αλλά πρώτα πρέπει να μετατραπούν σε σχεσιακή μορφή. Το εργαλείο ORM βοηθάει εδώ, δημιουργώντας τα εν κινήσει και εκεί εξοικονομεί πολύ χρόνο.

- Προσωρινή αποθήκευση και διαχείριση συναλλαγών

  Τα περισσότερα εργαλεία ORM, όπως το Hibernate, διαθέτουν λειτουργίες όπως η προσωρινή αποθήκευση και οι συναλλαγές. Η προσωρινή αποθήκευση δίνει τη δυνατότητα να βελτιωθεί την απόδοση των λειτουργιών πρόσβασης δεδομένων. Επιπλέον, παρέχουν υποστήριξη για πολλαπλές συναλλαγές, διαθέτουν μηχανισμό κλειδώματος και έκδοση εκδόσεων, που επιτρέπει σε μια εφαρμογή να έχει πολλαπλές συναλλαγές και δεν επηρεάζει την απόδοση της εφαρμογής.

2.14 JDBC

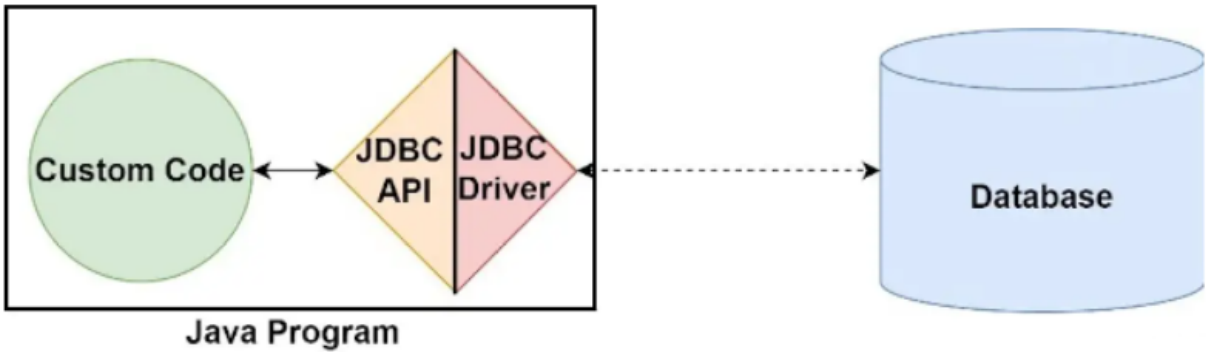
2.14.1 Τι είναι το JDBC



Σχήμα 16: Java Database Connectivity

Το JDBC (Java Database Connectivity) είναι μια διεπαφή της Java που διαχειρίζεται τη σύνδεση σε μια βάση δεδομένων, την δημιουργία ερωτημάτων και εντολών και τον χειρισμό των αποτελεσμάτων που λαμβάνονται από τη βάση δεδομένων. Κυκλοφόρησε ως μέρος του JDK 1.1 το 1997, το JDBC ήταν μια από τις πρώτες βιβλιοθήκες που αναπτύχθηκαν για τη γλώσσα Java. Αρχικά σχεδιάστηκε ως μια διεπαφή από την πλευρά του πελάτη, επιτρέποντας σε έναν πελάτη Java να αλληλεπιδρά με μια πηγή δεδομένων. Αυτό άλλαξε με το JDBC 2.0, το οποίο περιελάμβανε ένα προαιρετικό πακέτο που υποστηρίζει συνδέσεις JDBC από την πλευρά του διακομιστή. Κάθε νέα έκδοση του JDBC έκτοτε περιλαμβάνει ενημερώσεις τόσο για το πακέτο πελάτη (java.sql) όσο και για το πακέτο διακομιστή (javax.sql). [3]

2.14.2 Αρχιτεκτονική του JDBC



Σχήμα 17: Αρχιτεκτονική Java Database Connectivity

Η διεπαφή JDBC αποτελείται από δύο επίπεδα:

- Η διεπαφή JDBC υποστηρίζει την επικοινωνία μεταξύ της εφαρμογής Java και του διαχειριστή JDBC.
- Το πρόγραμμα οδήγησης JDBC υποστηρίζει την επικοινωνία μεταξύ του διαχειριστή JDBC και του προγράμματος οδήγησης της βάσης δεδομένων.

2.14.3 Είδη προγραμμάτων οδήγησης του JDBC

Η διεπαφή του JDBC περιέχει διεπαφές και κλάσεις Java που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για να συνδεθούν με μια βάση δεδομένων και να στείλουν ερωτήματα sql. Ένα πρόγραμμα οδήγησης JDBC υλοποιεί αυτές τις διεπαφές και τις κλάσεις για έναν συγκεκριμένο προμηθευτή DBMS. Ένα πρόγραμμα Java που χρησιμοποιεί την διεπαφή JDBC φορτώνει το καθορισμένο πρόγραμμα οδήγησης για ένα συγκεκριμένο DBMS πριν αυτό συνδεθεί πραγματικά σε μια βάση δεδομένων. Στη συνέχεια, η κλάση JDBC DriverManager στέλνει όλες τις κλήσεις JDBC API στο φορτωμένο πρόγραμμα οδήγησης.

Υπάρχουν τέσσερις τύποι προγραμμάτων οδήγησης JDBC:

- Το JDBC-ODBC bridge και το πρόγραμμα οδήγησης ODBC (Open Database Connectivity), που ονομάζεται επίσης πρόγραμμα οδήγησης τύπου 1. Το οποίο μεταφράζει τις κλήσεις στην διεπαφή του JDBC σε κλήσεις ODBC της Microsoft που στη συνέχεια μεταβιβάζονται στο πρόγραμμα οδήγησης ODBC. Εκεί Ο δυαδικός κώδικας ODBC πρέπει να φορτωθεί σε κάθε υπολογιστή-πελάτη που χρησιμοποιεί αυτόν τον τύπο προγράμματος οδήγησης.
- Το Native-API, εν μέρει πρόγραμμα οδήγησης Java, που ονομάζεται επίσης πρόγραμμα οδήγησης τύπου 2. Το οποίο μετατρέπει τις κλήσεις JDBC API σε κλήσεις API πελάτη για συγκεκριμένα DBMS. Όπως το πρόγραμμα οδήγησης bridge, έτσι και αυτός ο τύπος προγράμματος οδήγησης απαιτεί τη φόρτωση κάποιου δυαδικού κώδικα σε κάθε υπολογιστή-πελάτη.
- Το JDBC-Net, πρόγραμμα οδήγησης καθαρής Java, που ονομάζεται επίσης πρόγραμμα οδήγησης τύπου 3. Το οποίο στέλνει τις κλήσεις της διεπαφής JDBC σε έναν ενδιάμεσο διακομιστή που μεταφράζει τις κλήσεις στο πρωτόκολλο δικτύου του

συγκεκριμένου DBMS. Στη συνέχεια, οι μεταφρασμένες κλήσεις αποστέλλονται σε ένα συγκεκριμένο DBMS.

- Το Native-protocol, καθαρής Java, που ονομάζεται επίσης πρόγραμμα οδήγησης τύπου 4. Το οποίο μετατρέπει τις κλήσεις JDBC API απευθείας στο πρωτόκολλο δικτύου ειδικά για το DBMS χωρίς τον ενδιάμεσο διακομιστή. Αυτό το πρόγραμμα οδήγησης επιτρέπει στις εφαρμογές-πελάτες να συνδέονται απευθείας στον διακομιστή της βάσης δεδομένων.

2.15 JPA

2.15.1 Τι είναι το JPA

Το JPA (Java persistence API) είναι μια προδιαγραφή για την πρόσβαση, τη διατήρηση και τη διαχείριση δεδομένων μεταξύ αντικειμένων Java και σχεσιακής βάσης δεδομένων. Το Java Persistence API μετονομάστηκε σε Jakarta Persistence το 2019. Το Java Persistence API παρέχει ένα μοντέλο διατήρησης απλών αντικειμένων Java για αντικειμενική σχέση. Το Java Persistence API αναπτύχθηκε από την ομάδα ειδικών λογισμικού EJB 3.0 ως μέρος του JSR 220, αλλά η χρήση του δεν περιορίζεται στα στοιχεία λογισμικού EJB. Το JPA μπορεί να χρησιμοποιηθεί σε εφαρμογές web, εφαρμογές Java SE απευθείας, ακόμη και το μέρος του EJB που δεν εξαρτάται από κοντέινερ EJB. Τεχνικά το JPA είναι απλώς ένα σύνολο διεπαφών και για να χρησιμοποιηθεί απαιτείται η υλοποίηση τους σε μεταγενέστερες κλάσεις. Χρειάζεται μια υλοποίηση ORM για να μπορέσει να εργαστεί και να διατηρήσει τα αντικείμενα της Java και υπάρχει στο πακέτο javax.persistence.[3][10]

2.15.2 Υλοποίησεις του JPA

Υπάρχουν πολλές υλοποιήσεις για το JPA με τις ακόλουθες να είναι οι πιο δημοφιλείς.

- Hibernate – Open Source
- Top Link – Oracle
- Eclipse Link – Eclipse Persistence Platform
- Open JPA – Apache
- MyBatis – Open Source – Παλαιότερα γνωστό ως iBATIS

2.15.3 JPA and Hibernate

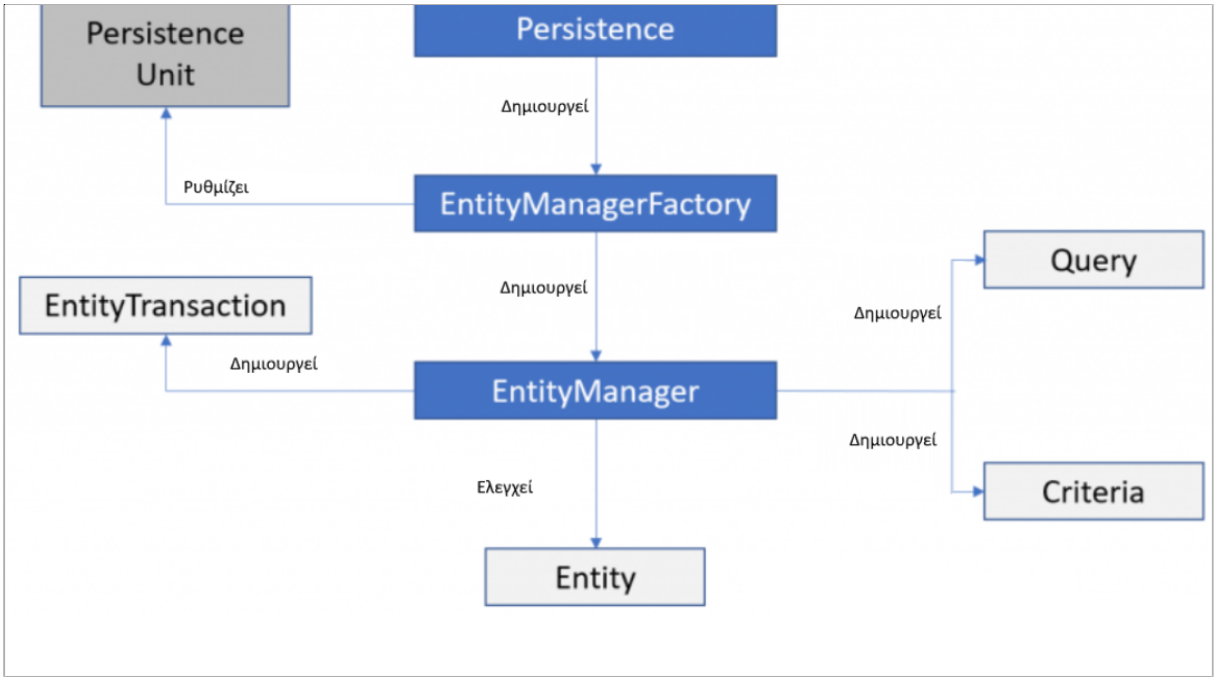
Το Hibernate είναι το δημοφιλές και πιο προηγμένο μεταξύ όλων των εφαρμογών JPA λόγω του Red Hat. Επιπλέον, είναι μια από τις πιο ώριμες υλοποιήσεις JPA και εξακολουθεί να είναι μια δημοφιλής επιλογή για ORM στην Java. Παρέχει τις δικές του τροποποιήσεις και πρόσθετες δυνατότητες που μπορούν να χρησιμοποιηθούν περεταίρω της προδιαγραφής JPA. Μερικές από τις πρόσθετες ιδιόκτητες λειτουργίες είναι η υποστήριξη για πολλαπλές κατοχές, κληρονομικότητα, σύνδεση οντοτήτων σε ερωτήματα, διαχείριση χρονικών σφραγίδων, αναζήτηση αδρανοποίησης, επικύρωση αδρανοποίησης και αδρανοποίηση OGM, η οποία υποστηρίζει τη διατήρηση του μοντέλου τομέα για βάσεις τύπου NoSQL.

2.15.4 Χαρακτηριστικά του JPA

Η προδιαγραφή JPA παρέχει καθαρότερη, ευκολότερη και τυποποιημένη σχέση μεταξύ των αντικειμένων. Υποστηρίζει κληρονομικότητα, πολυμορφισμό και πολυμορφικά ερωτήματα, καθώς και σχολιασμούς μεταδεδομένων/περιγραφες xml για τον καθορισμό της αντιστοίχισης μεταξύ

αντικειμένων Java και σχεσιακής βάσης δεδομένων. Επιπλέον, υποστηρίζει μια πλούσια γλώσσα ερωτημάτων που μοιάζει με SQL τόσο για στατικά όσο και για δυναμικά ερωτήματα. Επίσης παρέχει υποστήριξη για αποθηκευμένες διαδικασίες. Περιέχει την διεπαφή Criteria, η οποία υποστηρίζει τη δημιουργία ερωτημάτων SQL χρησιμοποιώντας απλά αντικείμενα Java, έτσι ώστε να είναι δυνατόν να υπάρχουν ασφαλή ερωτήματα που μπορούν να ελεγχθούν οι τύποι τους κατά τη στιγμή της μεταγλώττισης. Τέλος, παρέχει επικύρωση στα Beans σε όλες τις οντότητες στις οποίες εφαρμόζονται οι σχολιασμοί, προσωρινή αποθήκευση σε 2 είδη κρυφής μνήμης cache πρώτου και δεύτερου επιπέδου και καλύτερη ευθυγράμμιση με τις λειτουργίες της Java 8.

2.15.5 Επισημάνσεις και αρχιτεκτονική του JPA



Σχήμα 18: Αρχιτεκτονική JPA

2.15.5.1 Entity

Ο σχολιασμός Entity χρησιμοποιείται για να ορίζουμε ένα αντικείμενο το οποίο θέλουμε να διατηρήσουμε στη βάση δεδομένων. Κάθε οντότητα αντιπροσωπεύει έναν πίνακα σε μια σχεσιακή βάση δεδομένων και κάθε παρουσία οντότητας αντιστοιχεί σε μια σειρά σε αυτόν τον πίνακα. Το JPA χρησιμοποιεί Annotations ή XML για να αντιστοιχίσει οντότητες σε μια Σχεσιακή βάση δεδομένων. Η μόνιμη κατάσταση των πεδίων ή ιδιοτήτων μιας οντότητας χρησιμοποιούν σχολιασμούς αντιστοίχισης αντικειμένων/σχέσεων για να αντιστοιχίσουν τις οντότητες και τις σχέσεις οντοτήτων στα σχεσιακά δεδομένα στον υποκείμενο χώρο αποθήκευσης δεδομένων. Για παράδειγμα η κλάση του προϊόντος μπορεί να αναπαρασταθεί με τον παρακάτω τρόπο:

ID	NAME	DESCRIPTION	PRICE
0	Coke	a cold drink	1.00
1	Fanta	another cold drink	2.00

Πίνακας 3: Πίνακας Product στη Βάση Δεδομένων

```java
@Entity(name="PRODUCT")
public class Product {

  @Id
  @GeneratedValue(strategy=GenerationType.AUTO, generator="native")
  @GenericGenerator(name = "native", strategy = "native")
  @Column(name = "ID")
  private Long id;

  @Column(name = "NAME")
  private String name;

  @Column(name = "DESCRIPTION")
  private String description;

  @Column(name = "PRICE")
  private float price;

  // getters and setters
```

Σχήμα 19: JPA Representation of Product

2.15.5.2 Persistence Unit

Το Persistence Unit το οποίο είναι μια μονάδα που ορίζει ένα σύνολο από όλες τις κατηγορίες οντοτήτων που διαχειρίζονται οι παρουσίες EntityManager σε μια εφαρμογή. Αυτό το σύνολο κλάσεων οντοτήτων αντιπροσωπεύει τα δεδομένα που περιέχονται σε ένα ενιαίο χώρο αποθήκευσης δεδομένων. Οι μονάδες Persistence ορίζονται από το αρχείο διαμόρφωσης persistence.xml. Το JPA χρησιμοποιεί το αρχείο persistence.xml για να δημιουργήσει τη σύνδεση και να ρυθμίσει το απαιτούμενο περιβάλλον. Παρέχει πληροφορίες που είναι απαραίτητες για την πραγματοποίηση συνδέσεων με τις βάσεις δεδομένων. Για παράδειγμα η διαμόρφωσης μιας μονάδας στο persistence.xml μπορεί να γίνει με το παρακάτω τρόπο:

```xml
<persistence-unit name="jbd-pu">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <properties>
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.password" value="password"/>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/jpa_JBD"/>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />

            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLInnoDBDialect"/>
            <property name="hibernate.show_sql" value="true"/>
        </properties>
</persistence-unit>
```

Σχήμα 20: JPA Representation of Persistence Unit

2.15.5.3 Persistence Class

Το Persistence Class περιέχει στατικές μεθόδους java για τη λήψη των αντικειμένων EntityManagerFactory.

2.15.5.4 EntityManagerFactory

Η κλάση EntityManagerFactory είναι ένα εργοστάσιο για την δημιουργία αντικειμένων τύπου EntityManagers. Κατά τη διάρκεια του χρόνου εκκίνησης της εφαρμογής δημιουργείται ένα EntityManagerFactory με τη βοήθεια του Persistence-Unit. Συνήθως, το EntityManagerFactory δημιουργείται μία φορά, ένα αντικείμενο EntityManagerfactory ανά βάση δεδομένων και διατηρείται ζωντανό για μελλοντική χρήση. Για παράδειγμα η δημιουργία ενός EntityManagerFactory μπορεί να γίνει με τον παρακάτω τρόπο:

```java
// jbd-pu is the persistence unit configured in persistence.xmpl
EntityManagerFactory emf = Persistence.createEntityManagerFactory("jbd-pu");
```

Σχήμα 21: JPA Representation of Entity Manager Factory

2.15.5.5 EntityManager

Το EntityManager είναι μια διεπαφή για την εκτέλεση βασικών αλληλεπιδράσεων μεταξύ της βάσης δεδομένων. Διαχειρίζεται τα αντικείμενα τα οποία θα αποθηκευτούν. Επιπλέον, βρίσκει οντότητες με βάση το πρωτεύον κλειδί της οντότητας και επιτρέπει την εκτέλεση ερωτημάτων σε οντότητες. Τέλος, κάθε παρουσία του EntityManager σχετίζεται με ένα PersistenceContext, το οποίο είναι ένα περιβάλλον αποθήκευσης που χειρίζεται ένα σύνολο οντοτήτων που κρατούν δεδομένα που πρέπει να διατηρηθούν σε κάποιο χώρο αποθήκευσης (π.χ. βάση δεδομένων). Για παράδειγμα η δημιουργία ενός EntityManager μπορεί να γίνει με τον παρακάτω τρόπο:

```java
EntityManagerFactory emf = Persistence.createEntityManagerFactory("jbd-pu");
EntityManager entityManager = emf.createEntityManager();
```

Σχήμα 22: JPA Representation of Entity Manager

2.15.5.6 EntityTransaction

Το EntityTransaction ειναι μια συναλλαγή, ένα σύνολο πράξεων, που είτε αποτυγχάνουν είτε πετυχαίνουν ως μονάδα. Μια συναλλαγή στη βάση των δεδομένων αποτελείται από ένα σύνολο λειτουργιών SQL που δεσμεύονται ή επαναφέρονται ως ενιαία μονάδα Κάθε είδους τροποποιήσεις που ξεκινούν μέσω του αντικειμένου EntityManager τοποθετούνται σε μια συναλλαγή. Ένα αντικείμενο EntityManager βοηθά στη δημιουργία ενός EntityTransaction. Για παράδειγμα η δημιουργία ενός EntityTransaction μπορεί να γίνει με τον παρακάτω τρόπο:

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("jbd-pu");
EntityManager entityManager = emf.createEntityManager();
EntityTransaction transaction = entityManager.getTransaction();
```

Σχήμα 23: JPA Representation of Entity Transaction

Με βάση τα παραπάνω η αποθήκευση ενός προϊόντος με την χρήση του JPA μπορεί να γίνει με τον παρακάτω τρόπο:

```
EntityManagerFactory emf = null;
EntityManager entityManager = null;
EntityTransaction transaction = null;
try {
  emf = Persistence.createEntityManagerFactory("jbd-pu");
  entityManager = emf.createEntityManager();
  transaction = entityManager.getTransaction();
  transaction.begin();
  Product product = new Product();
  product.setName("Coke");
  product.setDescription("a cold drink");
  product.setPrice(1.00);
  entityManager.persist(product);
  transaction.commit();
} catch (Exception e) {
  transaction.rollback();
} finally {
  entityManager.close();
  emf.close();
}
```

Σχήμα 24: JPA Representation of Entity Transaction 2

2.15.5.7 Query

Το Query είναι μια διεπαφή που χρησιμοποιείται για τον έλεγχο της εκτέλεσης των ερωτημάτων. Ένα αντικείμενο EntityManager βοηθά στη δημιουργία ενός αντικειμένου Query και η υλοποίηση εξαρτάται από τον πάροχο επιμονής. Για παράδειγμα η δημιουργία ενός Query μπορεί να γίνει με τον παρακάτω τρόπο:

```
Query query = entityManager.createQuery("SELECT p FROM PRODUCT p");
List<Prduct> p = query.getResultList();
p.forEach(product-> System.out.println(product.getName()));
```

Σχήμα 25: JPA Representation of Query

2.15.5.8 Criteria Api

Η διεπαφή Criteria, υποστηρίζει τη δημιουργία ερωτημάτων SQL με χρήση αντικειμένων java, έτσι ώστε με το να είναι δυνατόν να υπάρχουν ασφαλή ερωτήματα με έλεγχο του τύπου κατά τη στιγμή της μεταγλώττισης.

```
1  CriteriaBuilder cb = entityManager.getCriteriaBuilder();
2  CriteriaQuery<Product> query = cb.createQuery(Product.class);
3  Root<Product> productRoot = query.from(Product.class);
4  query.select(productRoot);
5  TypedQuery<Product> typedQuery = entityManager.createQuery(query);
6  typedQuery.getResultList().forEach(s -> System.out.println(s.getFirstName()));
```

Σχήμα 26: JPA Representation of Criteria API

2.16 Vaadin

2.16.1 Τι είναι το Vaadin

Το Vaadin είναι μια πλατφόρμα ανάπτυξης ιστού ανοιχτού κώδικα για τη δημιουργία διεπαφών χρήστη με Java με ή χωρίς HTML. Το Vaadin είναι ένα API διακομιστή και ξεκινώντας από την έκδοση Vaadin 10 μέχρι τώρα άλλαξαν το GWT σε Web Component Polymer. Με το Vaadin Flow, ένας προγραμματιστής μπορεί να δημιουργήσει την διεπαφή του χρήστη σε Java, για παράδειγμα ένα κουμπί ή ένα πλαίσιο κειμένου.[4] Το Vaadin Flow έχει χαρακτηριστικά:

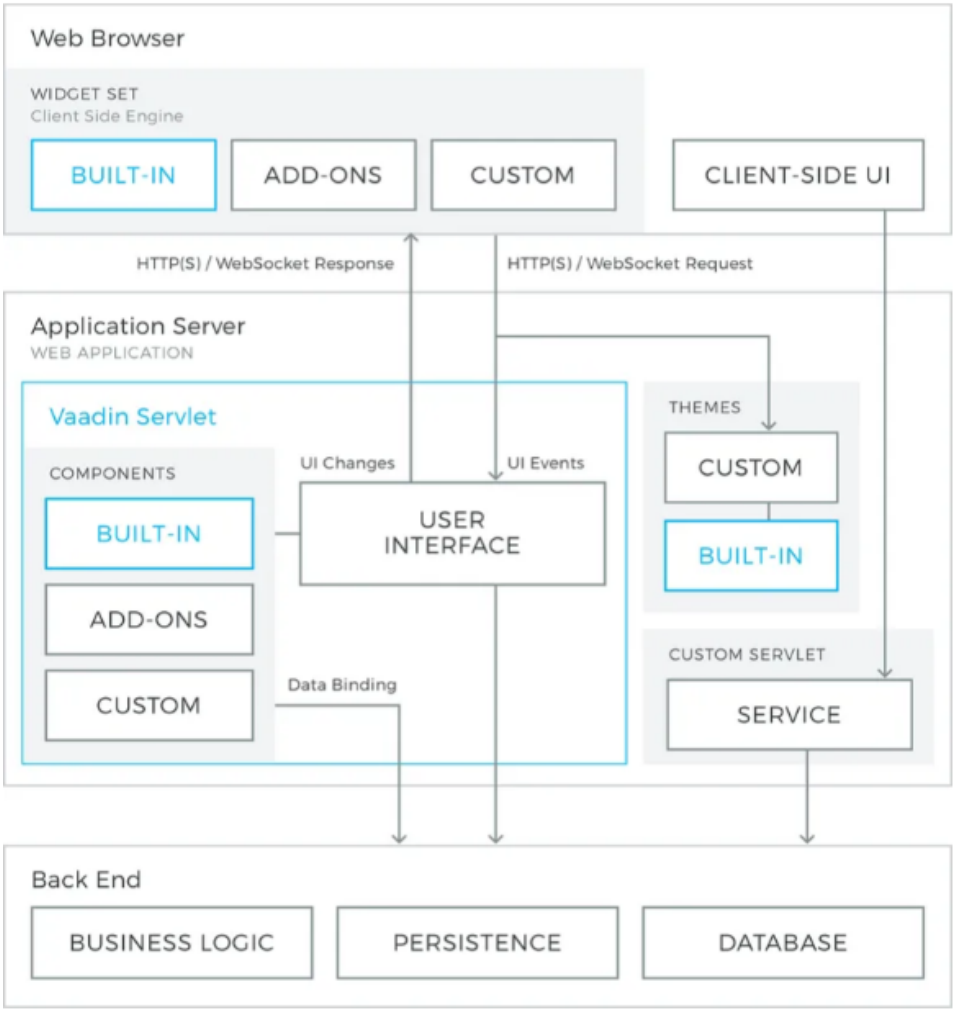
2.16.2 GWT

Η πλευρά του πελάτη του Vaadin Framework βασίζεται στο GWT. Σκοπός του είναι να καταστήσει δυνατή την ανάπτυξη διεπαφών χρήστη Ιστού που εκτελούνται στο πρόγραμμα περιήγησης εύκολα με Java αντί για JavaScript. Οι μονάδες από την πλευρά του πελάτη αναπτύσσονται με Java και μεταγλωτίζονται σε JavaScript με τον μεταγλωττιστή Vaadin, ο οποίος είναι μια επέκταση του μεταγλωττιστή GWT. Το πλαίσιο από την πλευρά του πελάτη κρύβει επίσης μεγάλο μέρος του χειρισμού HTML DOM και επιτρέπει το χειρισμό συμβάντων προγράμματος περιήγησης σε Java.

Το GWT είναι ουσιαστικά μια τεχνολογία από την πλευρά του πελάτη, που χρησιμοποιείται συνήθως για την ανάπτυξη λογικής διεπαφής χρήστη στο πρόγραμμα περιήγησης ιστού. Οι καθαρές λειτουργικές μονάδες από την πλευρά του πελάτη εξακολουθούν να πρέπει να επικοινωνούν με έναν διακομιστή χρησιμοποιώντας κλήσεις RPC και ‘σειριοποιώντας’ τυχόν δεδομένα. Η λειτουργία ανάπτυξης που βασίζεται σε διακομιστή στο πλαίσιο αποκρύπτει αποτελεσματικά όλες τις επικοινωνίες πελάτη-διακομιστή και επιτρέπει το χειρισμό της λογικής αλληλεπίδρασης χρήστη σε μια εφαρμογή από την πλευρά του διακομιστή. Αυτό κάνει την αρχιτεκτονική μιας εφαρμογής web που βασίζεται σε AJAX πολύ πιο απλή. Ωστόσο, το Vaadin Framework επιτρέπει επίσης την ανάπτυξη καθαρών εφαρμογών από την πλευρά του πελάτη, όπως περιγράφεται παρακάτω στα χαρακτηριστικά του Vaadin.

2.16.3 Αρχιτεκτονική του Vaadin

Το Vaadin Framework παρέχει δύο μοντέλα ανάπτυξης για εφαρμογές web: για την πλευρά του πελάτη και για την πλευρά του διακομιστή. Το μοντέλο ανάπτυξης που βασίζεται σε διακομιστή είναι το πιο ισχυρό, που επιτρέπει την ανάπτυξη εφαρμογών αποκλειστικά από την πλευρά του διακομιστή, χρησιμοποιώντας μια μηχανή από την πλευρά πελάτη Vaadin που βασίζεται σε AJAX ωστε να αποδώσει τη διεπαφή του χρήστη στο πρόγραμμα περιήγησης. Το μοντέλο από την πλευρά του πελάτη επιτρέπει την ανάπτυξη γραφικών στοιχείων και εφαρμογών σε Java, τα οποία μεταγλωτίζονται σε JavaScript και εκτελούνται στο πρόγραμμα περιήγησης. Τα δύο μοντέλα μπορούν να μοιράζονται τα γραφικά στοιχεία διεπαφής χρήστη, τα θέματα και τον κώδικα και τις υπηρεσίες back-end τους και μπορούν να αναμειχθούν εύκολα.



Σχήμα 27: Vaadin Framework

Η παραπάνω αρχιτεκτονική του χρόνου εκτέλεσης του Vaadin παρέχει μια βασική απεικόνιση των επικοινωνιών από την πλευρά του πελάτη και του διακομιστή, σε μια κατάσταση λειτουργίας όπου η σελίδα με τον κώδικα από την πλευρά του πελάτη (μηχανή ή εφαρμογή) έχει αρχικά φορτωθεί στο πρόγραμμα περιήγησης.

Το Vaadin Framework αποτελείται από ένα API από την πλευρά του διακομιστή, ένα API από την πλευρά του πελάτη, ένα σύνολο από στοιχεία/γραφικά στοιχεία διεπαφής χρήστη και στις δύο

πλευρές, θέματα για τον έλεγχο της εμφάνισης και ένα μοντέλο δεδομένων που επιτρέπει τη σύνδεση των στοιχείων διακομιστή απευθείας στα δεδομένα. Για την ανάπτυξη από την πλευρά του πελάτη, περιλαμβάνει τον μεταγλωττιστή Vaadin, ο οποίος επιτρέπει τη μεταγλώττιση της Java σε JavaScript.

Στην συνέχεια, μια εφαρμογή Vaadin από την πλευρά του διακομιστή εκτελείται ως servlet σε έναν διακομιστή ιστού Java, εξυπηρετώντας αιτήματα HTTP. Το VaadinServlet χρησιμοποιείται συνήθως ως κλάση servlet. Ο servlet λαμβάνει αιτήματα πελατών και τα ερμηνεύει ως συμβάντα για μια συγκεκριμένη περίοδο λειτουργίας χρήστη. Τα συμβάντα συσχετίζονται με στοιχεία διεπαφής χρήστη και παραδίδονται στους ακροατές συμβάντων που ορίζονται στην εφαρμογή. Εάν η λογική διεπαφής χρήστη κάνει αλλαγές στα στοιχεία της διεπαφής χρήστη από την πλευρά του διακομιστή, το servlet τις αποδίδει στο πρόγραμμα περιήγησης ιστού δημιουργώντας μια απόκριση. Η μηχανή από την πλευρά του πελάτη που λειτουργεί στο πρόγραμμα περιήγησης λαμβάνει τις απαντήσεις και τις χρησιμοποιεί για να κάνει τις απαραίτητες αλλαγές στη σελίδα του προγράμματος περιήγησης.

Τα κύρια μέρη της αρχιτεκτονικής ανάπτυξης που βασίζεται σε διακομιστή και η λειτουργία τους είναι τα εξής:

2.16.3.1 Διεπαφή χρήστη

Οι εφαρμογές Vaadin παρέχουν μια διεπαφή χρήστη για τη διασύνδεση του χρήστη με την επιχειρηματική λογική και τα δεδομένα της εφαρμογής. Σε τεχνικό επίπεδο, το UI υλοποιείται ως κλάση διεπαφής χρήστη που επεκτείνει το com.vaadin.ui.UI. Το κύριο καθήκον του είναι να δημιουργήσει την αρχική διεπαφή χρήστη από στοιχεία διεπαφής χρήστη και να ρυθμίσει προγράμματα ακρόασης συμβάντων για να χειριστεί την εισαγωγή του χρήστη. Στη συνέχεια, η διεπαφή χρήστη μπορεί να φορτωθεί στο πρόγραμμα περιήγησης χρησιμοποιώντας μια διεύθυνση URL ή μπορεί να ενσωματωθεί σε οποιαδήποτε σελίδα HTML.

2.16.3.2 Στοιχεία διεπαφής χρήστη και τα γραφικά στοιχεία

Η διεπαφή χρήστη μιας εφαρμογής Vaadin αποτελείται από στοιχεία που δημιουργούνται και τοποθετούνται από την εφαρμογή. Κάθε στοιχείο από την πλευρά του διακομιστή έχει ένα αντίστοιχο από την πλευρά του πελάτη, ένα γραφικό στοιχείο, με το οποίο αποδίδεται στο πρόγραμμα περιήγησης και με το οποίο αλληλεπιδρά ο χρήστης. Τα γραφικά στοιχεία από την πλευρά του πελάτη μπορούν επίσης να χρησιμοποιηθούν από εφαρμογές από την πλευρά του πελάτη. Τα στοιχεία του διακομιστή αναμεταδίδουν αυτά τα συμβάντα στη λογική της εφαρμογής. Τα στοιχεία πεδίου που έχουν μια τιμή, την οποία ο χρήστης μπορεί να δει ή να επεξεργαστεί, μπορούν να συνδεθούν σε μια πηγή δεδομένων.

2.16.3.3 Client-Side Engine

Το Client-Side Engine διαχειρίζεται την απόδοση της διεπαφής χρήστη στο πρόγραμμα περιήγησης ιστού χρησιμοποιώντας διάφορα widget από την πλευρά του πελάτη, αντίστοιχες των στοιχείων του διακομιστή. Επικοινωνεί την αλληλεπίδραση του χρήστη στην πλευρά του διακομιστή και, στη συνέχεια, αποδίδει ξανά τις αλλαγές στη διεπαφή χρήστη. Οι επικοινωνίες γίνονται χρησιμοποιώντας ασύγχρονα αιτήματα HTTP ή HTTPS.

2.16.3.4 Vaadin Servlet

Η εφαρμογές Vaadin από την πλευρά του διακομιστή χρησιμοποιούν για την λειτουργία τους. Το Vaadin servlet που υλοποιείται στη κλάση VaadinServlet, λαμβάνει αιτήματα από διαφορετικούς πελάτες, καθορίζει σε ποια περίοδο λειτουργίας χρήστη ανήκουν παρακολουθώντας τις συνεδρίες με

cookies και αναθέτει τα αιτήματα στις αντίστοιχες συνεδρίες τους. Τέλος, το Vaadin servlet μπορεί να προσαρμοστεί επεκτείνοντάς το.

2.16.3.5 Θεμάτα

Το Vaadin στην παρουσίαση της σελίδας κάνει έναν διαχωρισμό μεταξύ της εμφάνισης και της δομής στοιχείων της διεπαφής χρήστη. Ενώ η λογική διεπαφής χρήστη αντιμετωπίζεται ως κώδικας Java, η παρουσίαση ορίζεται στα θέματα ως CSS ή Sass. Το Vaadin παρέχει έναν αριθμό προεπιλεγμένων θεμάτων. Τα θέματα χρήστη μπορούν, εκτός από css, να περιλαμβάνουν πρότυπα HTML που ορίζουν προσαρμοσμένες διατάξεις και άλλους πόρους, όπως εικόνες και γραμματοσειρές.

2.16.3.6 Συμβάντα

Η αλληλεπίδραση με τα στοιχεία της διεπαφής χρήστη δημιουργεί συμβάντα, τα οποία πρώτα υποβάλλονται σε επεξεργασία στην πλευρά του πελάτη από τα γραφικά στοιχεία και στη συνέχεια περνούν από τον διακομιστή HTTP, τον διακομιστή Vaadin και τα στοιχεία της διεπαφής χρήστη στα προγράμματα ακρόασης συμβάντων που ορίζονται στην εφαρμογή.

2.16.3.7 Server Push

Εκτός από το μοντέλο προγραμματισμού που βασίζεται σε συμβάντα, το Vaadin υποστηρίζει την προώθηση διακομιστή, όπου οι αλλαγές διεπαφής χρήστη προωθούνται απευθείας από τον διακομιστή στον πελάτη χωρίς αίτημα πελάτη ή συμβάν. Αυτό καθιστά δυνατή την άμεση ενημέρωση των διεπαφών από άλλα νήματα και άλλες διεπαφές, χωρίς να χρειάζεται να περιμένετε για ένα αίτημα.

2.16.3.8 Data Binding

Εκτός από το μοντέλο διεπαφής χρήστη, το Vaadin παρέχει ένα API δέσμευσης δεδομένων για τη συσχέτιση δεδομένων που παρουσιάζονται σε στοιχεία πεδίου, όπως πεδία κειμένου, πλαίσια ελέγχου και στοιχεία επιλογής, με μια πηγή δεδομένων. Χρησιμοποιώντας τη δέσμευση δεδομένων, τα στοιχεία της διεπαφής χρήστη μπορούν να ενημερώσουν τα δεδομένα της εφαρμογής απευθείας, συχνά χωρίς να απαιτείται κωδικός ελέγχου. Για παράδειγμα, μπορεί να συνδεθεί ένα στοιχείο πλέγματος δεδομένων σε μια απάντηση ερωτήματος από την βάση δεδομένων.

2.16.3.9 Εφαρμογές από την πλευρά του πελάτη

Εκτός από τις εφαρμογές ιστού από την πλευρά του διακομιστή, το Vaadin υποστηρίζει λειτουργικές μονάδες εφαρμογών από την πλευρά του πελάτη, οι οποίες εκτελούνται στο πρόγραμμα περιήγησης. Οι λειτουργικές μονάδες από την πλευρά του πελάτη μπορούν να χρησιμοποιούν τα ίδια γραφικά στοιχεία, θέματα και υπηρεσίες υποστήριξης όπως οι εφαρμογές Vaadin από την πλευρά του διακομιστή. Είναι χρήσιμες όταν χρειάζεται μια λογική διεπαφής χρήστη με υψηλή απόκριση, όπως για τα παιχνίδια ή για την εξυπηρέτηση μεγάλου αριθμού πελατών με stateless κωδικα από την πλευρά του διακομιστή, και για διάφορους άλλους σκοπούς, όπως η προσφορά λειτουργίας εκτός σύνδεσης για διακομιστή ή και πλευρικές εφαρμογές.

2.16.3.10 Backend

Το Vaadin προορίζεται για τη δημιουργία διεπαφών χρήστη και συνιστάται τα άλλα επίπεδα εφαρμογών να διατηρούνται χωριστά από τη διεπαφή χρήστη. Η επιχειρησιακή λογική μπορεί να εκτελεστεί στον ίδιο servlet με τον κώδικα διεπαφής χρήστη, που συνήθως διαχωρίζεται τουλάχιστον από ένα Java API ή διανέμεται σε μια απομακρυσμένη υπηρεσία υποστήριξης. Η αποθήκευση δεδομένων συνήθως διανέμεται σε ένα σύστημα διαχείρισης βάσης δεδομένων όπως το JPA.

2.17 Αλλά δημοφιλή Framework της Java

Τα Frameworks ή αλλιώς πλαίσια της Java είναι σώματα προγραμμένου κώδικα και τεχνικών που χρησιμοποιούνται από προγραμματιστές για τη δημιουργία εφαρμογών χρησιμοποιώντας τη γλώσσα προγραμματισμού Java. Ένα πλαίσιο συχνά υπαγορεύει τη δομή μιας εφαρμογής. Ορισμένα πλαίσια παρέχουν αρκετό κώδικα ώστε να μην χρειάζεται μεγάλη προσπάθεια για την δημιουργία ενός χαρακτηριστικού. Αυτό μπορεί να είναι καλό ή κακό, ανάλογα με το πόσο εύκολο είναι στη χρήση. Τα πλαίσια είναι η ουσία του προγραμματισμού και επηρεάζουν την βιωσιμότητα της εφαρμογής.

2.17.1 Struts

Σχήμα 28: Logo Struts

Είναι ένα πλαίσιο εταιρικού επιπέδου, το οποίο διατηρεί το Ίδρυμα Λογισμικού Apache. Ένα πλήρως εξοπλισμένο πλαίσιο εφαρμογής web Java που επιτρέπει στους προγραμματιστές να δημιουργήσουν μια εύκολη στη συντήρηση εφαρμογή Java. Υπάρχουν δύο εκδοχές. Struts 1 και Struts 2. Το Struts 2 είναι ο συνδυασμός του πλαισίου webwork του OpenSymphony και του Struts 1. Αλλά όλες οι εταιρείες προτιμούν να χρησιμοποιούν το Struts 2 επειδή είναι η αναβαθμισμένη έκδοση του Apache Struts. Γενικότερα, το πλαίσιο Struts 2 μπορεί να χρησιμοποιηθεί για την ανάπτυξη μιας διαδικτυακής εφαρμογής που βασίζεται σε MVC. Χρησιμοποιεί και επεκτείνει το Java Servlet API για να βοηθήσει τους προγραμματιστές να υιοθετήσουν την αρχιτεκτονική MVC.

Τελος, τα πλεονεκτήματα του Struts ειναι:

- Η τεκμηρίωση του πλαισίου είναι γραμμένη για ενεργούς προγραμματιστές ιστού και προϋποθέτει μια λειτουργική γνώση σχετικά με τον τρόπο κατασκευής των εφαρμογών ιστού Java
- Μειώνει τον χρόνο ανάπτυξης και διευκολύνει τη διαχείριση της εφαρμογής
- Προσφέρει κεντρική διαμόρφωση, δηλαδή, αντί να κωδικοποιεί πληροφορίες σε προγράμματα Java, πολλές τιμές Struts αντιπροσωπεύονται σε αρχεία XML ή ιδιοτήτων.
- Μπορεί να ενσωματωθεί με άλλα πλαίσια Java για να εκτελεστούν εργασίες που δεν είναι ενσωματωμένες στην πλατφόρμα

2.17.2 JavaServer Faces

Σχήμα 29: Logo Java Server Faces

Το JavaServer Faces (JSF) αναπτύχθηκε από την Oracle για τη δημιουργία διεπαφών χρήστη για εφαρμογές web που βασίζονται σε Java. Είναι ένα σταθερό και επίσημο πρότυπο της πρωτοβουλίας Java Community Process (JCP). Το JSF βασίζεται στο μοτίβο σχεδιασμού λογισμικού MVC και έχει μια αρχιτεκτονική που καθορίζει πλήρως τη διάκριση μεταξύ λογικής εφαρμογής και αναπαράστασης

Τα πλεονεκτήματα του JSF:

- Το JSF είναι αναπόσπαστο μέρος της Java EE
- Παρέχει εξαιρετικά εργαλεία και πλούσιες βιβλιοθήκες
- Επιτρέπει την επέκταση του υπάρχοντος κώδικα Java υποστήριξης με μια διεπαφή ιστού χωρίς να χρειάζεται να αλλάξει η βασική εφαρμογή εισάγοντας ένα νέο πλαίσιο.

2.17.3 Grails

Σχήμα 30: Logo Grails

Το Grails είναι ένα δυναμικό πλαίσιο, που επινοήθηκε από τη γλώσσα προγραμματισμού Groovy JVM. Είναι μια αντικειμενοστραφή γλώσσα για την πλατφόρμα Java που σκοπεύει να βελτιώσει την παραγωγικότητα των προγραμματιστών. Η σύνταξη είναι συμβατή με Java και έχει μεταγλωττιστεί σε bytecode JVM (Java Virtual Machine). Το Grails λειτουργεί με τεχνολογίες Java, συμπεριλαμβανομένων κοντέινερ Java EE, Spring, SiteMesh, Quartz και Hibernate.

Τα πλεονεκτήματα του Grails:

- Είναι πολύ φιλικό προς τους προγραμματιστές καθώς συνοδεύεται από λεπτομερή και ευανάγνωστη τεκμηρίωση

- Υποστηρίζει την δημιουργια προσαρμοσμενων πρόσθετων και την υποστήριξη Grails IDE για διαφορετικές πλατφόρμες.
- Οι περισσότεροι από τους ιστότοπους ηλεκτρονικού εμπορίου χρησιμοποιούν το Grails
- Χρησιμοποιεί το Groovy ως πρότυπο κωδικοποίησης και δεδομένου ότι το Groovy είναι παρόμοιο με την Java, οι προγραμματιστές πιστεύουν ότι είναι ευκολότερο να ξεκινήσουν με το Grails.
- Εύκολο στη χρήση της δυνατότητας αντιστοίχισης αντικειμένων
- Προωθεί την επαναχρησιμοποίηση του κώδικα μεταξύ διαφορετικών εφαρμογών Grails διαμορφώνοντας την εφαρμογή με τη μορφή πρόσθετων
- Παρέχει ευέλικτα προφίλ

2.17.4 Dropwizard

Σχήμα 31: Logo Dropwizard

Το Dropwizard είναι ένα πλαίσιο Java υψηλής απόδοσης για ταχεία ανάπτυξη υπηρεσιών web RESTful. Είναι ιδιαίτερα κατάλληλο για τη δημιουργία microservices. Το πλαίσιο Dropwizard συγκεντρώνει διάφορες καθιερωμένες βιβλιοθήκες Java προκειμένου να παρέχει μια γρήγορη και χωρίς περισπασμούς πλατφόρμα ανάπτυξης. Είναι ένα ξεχωριστό οικοσύστημα που περιέχει όλες τις εξαρτήσεις ομαδοποιημένες σε ένα ενιαίο πακέτο.

Τα πλεονεκτήματα του πλαισίου Dropwizard:

- Επιτρέπει την γρήγορη δημιουργία πρωτοτύπων
- Αναπτύσσει υπηρεσίες ιστού RESTful υψηλής απόδοσης
- Υποστηρίζει επίσης πολλές βιβλιοθήκες ανοιχτού κώδικα και ανεξάρτητες.
- Αύξηση της παραγωγικότητας

2.17.5 Play

Σχήμα 32: Logo Play

Πρόκειται για ένα διαφορετικό πλαίσιο ιστού και κινητών εφαρμογών Java υψηλής κλιμάκωσης. Το Play καθιστά δυνατή την ανάπτυξη ελαφρών και φιλικών στον ιστό εφαρμογών Java και Scala για διεπαφές επιτραπέζιων υπολογιστών και φορητών υπολογιστών. Συχνά συγκρίνεται με ισχυρά πλαίσια άλλων γλωσσών, όπως το Ruby on Rails για τη Ruby ή το Django για την Python. Το Play είναι ένα μοναδικό πλαίσιο Java καθώς δεν βασίζεται στα πρότυπα Java EE. Αντίθετα, σκοπεύει να αφαιρέσει όλες τις ταλαιπωρίες της παραδοσιακής ανάπτυξης ιστού Java, όπως αργούς κύκλους ανάπτυξης, πολλές ρυθμίσεις παραμέτρων και πολλά άλλα. Βασισμένο στο Akka Toolkit, το πλαίσιο Play περικόπτει τη δημιουργία ταυτόχρονων και κατανεμημένων εφαρμογών στην εικονική μηχανή Java.

Πλεονεκτήματα του πλαισίου Play:

- Προσφέρει λειτουργίες όπως επαναφόρτωση κώδικα σε μια αλλαγή, ρύθμιση παραμέτρων μέσω σύμβασης και μηνύματα σφάλματος στο πρόγραμμα περιήγησης.
- Υποστηρίζει I/O χωρίς αποκλεισμό, κάτι που είναι ζωτικής σημασίας για εφαρμογές υψηλής απόδοσης
- Παρέχει πιο ευέλικτα και ανθεκτικά στην αστοχία αποτελέσματα

2.18 Frontend

Η ανάπτυξη web front-end, γνωστή και ως ανάπτυξη από την πλευρά του πελάτη, είναι η πρακτική της παραγωγής HTML, CSS και JavaScript για έναν ιστότοπο ή μια εφαρμογή Ιστού, έτσι ώστε ο χρήστης να μπορεί να τα δει και να αλληλεπιδράσει άμεσα μαζί τους. Η πρόκληση που σχετίζεται με την ανάπτυξη της διεπαφής είναι ότι τα εργαλεία και οι τεχνικές που χρησιμοποιούνται για τη δημιουργία της διεπαφής ενός ιστότοπου αλλάζουν συνεχώς και έτσι ο προγραμματιστής πρέπει να γνωρίζει συνεχώς πώς αναπτύσσεται το πεδίο. Ο στόχος του σχεδιασμού ενός ιστότοπου είναι να διασφαλιστεί ότι όταν οι χρήστες ανοίγουν τον ιστότοπο βλέπουν τις πληροφορίες σε μια μορφή που είναι ευανάγνωστη και σχετική. Αυτό περιπλέκεται περαιτέρω από το γεγονός ότι οι χρήστες χρησιμοποιούν πλέον μια μεγάλη ποικιλία συσκευών με διαφορετικά μεγέθη οθόνης και αναλύσεις, αναγκάζοντας έτσι τον σχεδιαστή να λάβει υπόψη αυτές τις πτυχές κατά το σχεδιασμό του ιστότοπου. Πρέπει να διασφαλίσουν ότι ο ιστότοπός τους εμφανίζεται σωστά σε διαφορετικά προγράμματα περιήγησης, διαφορετικά λειτουργικά συστήματα και διαφορετικές συσκευές, κάτι που απαιτεί προσεκτικό σχεδιασμό από την πλευρά του προγραμματιστή.

2.18.1 HTML

Η HTML σημαίνει HyperText Markup Language, μια γλώσσα σήμανσης για το υπερκείμενο. Είναι ένα σύνολο εντολών που περιγράφει στα προγράμματα περιήγησης ιστού πώς να εμφανίζουν συγκεκριμένα μέρη του ιστότοπου. Οι πρώτες ιστοσελίδες κυκλοφόρησαν το 1990 και χρησιμοποιήθηκαν μόνο για παρουσίαση.[5][9]

Για παράδειγμα:

```
<h1>Αυτή είναι μια κεφαλίδα</h1>
<p>Αυτή είναι μια παράγραφος κειμένου με <a href="#">υπερσύνδεσμο</a>.</p>
<ul><li>Αυτό είναι ένα στοιχείο μιας μη ταξινομημένης λίστας</li><li>Αυτό είναι άλλο</li></ul>
```

Σχήμα 33: HTML παράδειγμα κώδικα

Τα στοιχεία ανάμεσα σε γωνιακές αγκύλες ονομάζονται ετικέτες. Μια ετικέτα είναι μια εντολή που λέει στο πρόγραμμα περιήγησης ότι το ακόλουθο κείμενο πρέπει να εμφανίζεται με συγκεκριμένο τρόπο. Μια ετικέτα μπορεί επίσης να ενημερώσει το πρόγραμμα περιήγησης ότι πρέπει να υπάρχει μια εικόνα ή ένα κουμπί σε ένα συγκεκριμένο μέρος. Αυτές οι ετικέτες μπορούν να αποθηκευτούν σε ένα αρχείο κειμένου το οποίο θα εμφανίσει το πρόγραμμα περιήγησης, το οποίο αποδίδει την ιστοσελίδα χωρίς τις ετικέτες, αλλά προσέχοντας τι σημαίνουν. Τα προγράμματα περιήγησης σε υπολογιστές και κινητά κατανοούν αυτές τις εντολές με τον ίδιο περίπου τρόπο, ώστε να είστε σίγουροι ότι η ιστοσελίδα θα λειτουργεί με τον ίδιο τρόπο στα περισσότερα προγράμματα περιήγησης σε όλο τον κόσμο.

Για παράδειγμα, το παραπάνω κείμενο HTML θα μοιάζει με αυτό:

Αυτή είναι μια κεφαλίδα

Αυτή είναι μια παράγραφος κειμένου με υπερσύνδεσμο.

- Αυτό είναι ένα στοιχείο μιας μη ταξινομημένης λίστας
- Αυτό είναι άλλο

Σχήμα 34: HTML υλοποίηση παραδείγματος κώδικα

Εκτός από την HTML υπάρχουν πολλές γλώσσες σήμανσης. Για παράδειγμα, τόσο το Microsoft Word όσο και το Apple Pages χρησιμοποιούν τις δικές τους ιδιόκτητες γλώσσες σήμανσης για την αποθήκευση πληροφοριών σχετικά με τα έγγραφα κειμένου. Στο MS Word αρκετά δεδομένα σχεδίασης ενσωματωνονται στο έγγραφό με μορφή εντολών σήμανσης, όπως μια νέα σειρά παραγράφου ή ότι ένα κείμενο θα εμφανίζεται με έντονα γράμματα.

2.18.2 Η ανάγκη της HTML

Η HTML είναι η θεμελιώδης γλώσσα στην οποία είναι γραμμένες οι περισσότερες ιστοσελίδες. Κάθε παράγραφος κειμένου στον ιστό είναι πιθανότατα τυλιγμένη σε … και κάθε σύνδεσμος στον περιβάλλεται απο <a>…. Χωρίς την HTML, θα έπρεπε να βρεθεί κάποια άλλη κοινή μορφή για να επιτρέψει την εύκολη ανταλλαγή εγγράφων στον ιστό.

2.18.3 Δομή σελίδας HTML

Μια σελίδα HTML αποτελειται απο τα τρεις βασικές ετικέτες δομής την <html>, το <head> και το <body>. Η ετικέτα <html> αντιπροσωπεύει τη ρίζα ενός εγγράφου και είναι ένα κοντέινερ για όλα τα άλλα στοιχεία της HTML. Το <head> χρησιμοποιείται για τον τίτλο και τις μετα-ετικέτες. Τέλος, μόνο το περιεχόμενο εντός της ενότητας <body> εμφανίζεται από το πρόγραμμα περιήγησης ιστού.

- **HTML Tags**

  Οι ετικέτες HTML είναι ονόματα στοιχείων που περιβάλλονται από αγκύλες και συνήθως σε ζεύγη όπως <table> και </table>.

- **HTML Attributes**

  Τα χαρακτηριστικά HTML χρησιμοποιούνται για την τροποποίηση της τιμής ενός στοιχείου HTML. Τα στοιχεία έχουν συχνά πολλαπλά χαρακτηριστικά. Πχ <tagname attributename=”setting”>περιεχόμενο για το στοιχείο html…</tagname>. Το χαρακτηριστικό <id> ορίζει ένα μοναδικό αναγνωριστικό για ένα στοιχείο HTML. Το αναγνωριστικό πρέπει να είναι μοναδικό στο έγγραφο HTML. Το <id> μπορεί να χρησιμοποιηθεί για να βρεθεί το στοιχείο HTML και, στη συνέχεια, να διαβαστεί ή να επεξεργαστεί με CSS και JavaScript.

- **HTML Block-level Elements**

  Ένα στοιχείο επιπέδου μπλοκ ξεκινά πάντα σε μια νέα γραμμή. Αυτός ο τύπος στοιχείων καταλαμβάνει επίσης πάντα το πλήρες πλάτος που είναι διαθέσιμο στη σελίδα. Αυτό σημαίνει ότι εκτείνεται όσο το δυνατόν περισσότερο προς τα αριστερά και όσο το δυνατόν πιο μακριά προς τα δεξιά.

- **HTML Inline Elements**

  Σε αντίθεση με ένα στοιχείο σε επίπεδο μπλοκ, ένα στοιχείο Inline δεν ξεκινά σε μια νέα γραμμή. Επιπλέον δεν καταλαμβάνει όλο το πλάτος, αλλά όσο πλάτος χρειάζεται.

- **HTML Empty Elements**

  Υπάρχουν επίσης στοιχεία HTML που δεν έχουν περιεχόμενο και ονομάζονται κενά στοιχεία. Ένα κενό στοιχείο που χρησιμοποιείται συχνά είναι το στοιχείο
. Αυτό είναι ένα κενό στοιχείο που δεν έχει περιεχόμενο.

- **HTML Headings**

  Τα στοιχεία HTML <h1>–<h6> αντιπροσωπεύουν έξι ξεχωριστά επίπεδα επικεφαλίδων ενοτήτων. Το υψηλότερο τμήμα είναι το επίπεδο <h1> και το χαμηλότερο τμήμα είναι το <h6>. Συνήθως στο στοιχείο <h1> αποδίδεται η υψηλότερη σημασία ενώ στο <h6> η μικρότερη σημασία. Αυτές οι επικεφαλίδες είναι επίσης σημαντικές όταν πρόκειται για τον τρόπο λειτουργίας των μηχανών αναζήτησης. Για την ευρετηρίαση της δομής και του

περιεχομένου μιας ιστοσελίδας, οι μηχανές αναζήτησης χρησιμοποιούν τις επικεφαλίδες. Οι χρήστες συχνά κάνουν έναν γρήγορο έλεγχο της σελίδας διαβάζοντας απλώς τις επικεφαλίδες οπότε είναι σημαντικό να χρησιμοποιουνται επικεφαλίδες για τη καλύτερη δομή της σελίδας.

- **HTML Paragraphs**

  Το στοιχείο HTML <p> ορίζει μια παράγραφο, η οποία τοποθετεί μια κενή γραμμή πάνω και κάτω από το κείμενο.

- **HTML Links**

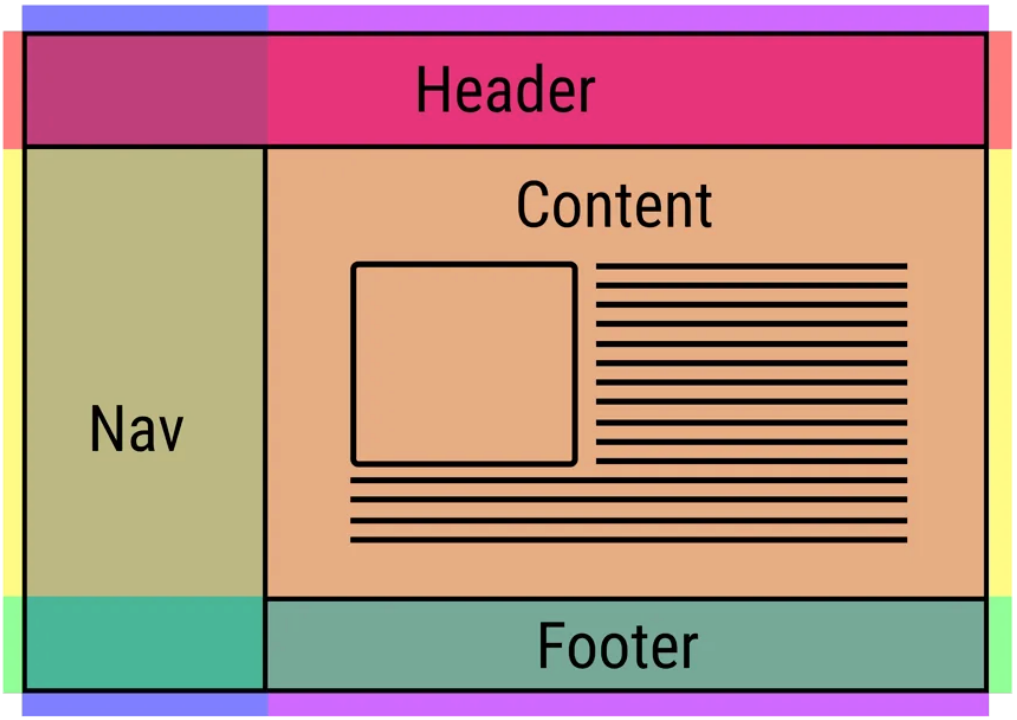
  Οι Σύνδεσμοι HTML ή υπερσύνδεσμοι χρησιμοποιούνται για τη σύνδεση σελίδων μεταξύ τους. Ένας ιστότοπος αποτελείται από πολλές σελίδες και οι σύνδεσμοι διευκολύνουν τους χρήστες να πλοηγούνται μεταξύ των σελίδων, ακόμη και σε άλλους ιστότοπους.

- **HTML Tables**

  Το στοιχείο HTML <table> χρησιμοποιείται για την αναπαράσταση δεδομένων πίνακα. Τα δεδομένα ενος πίνακα είναι πληροφορίες που μπορούν να παρουσιάσουν σε έναν δισδιάστατο πίνακα. Ένας τέτοιος πίνακας αποτελείται από έναν αριθμό σειρών και έναν αριθμό στηλών κελιών που περιέχουν δεδομένα.

- **HTML Layout**

  Όταν αναπτύσσεται μια ιστοσελίδα, είναι πολύ σημαντικη και η διάταξη και πώς θα πρέπει να είναι η σελίδα. Αυτό είναι σημαντικό για ολόκληρη την εμπειρία χρήστη και για τη φιλικότητα προς το χρήστη της σελίδας. Είναι σύνηθες και συνιστάται ιδιαίτερα η χρήση σύγχρονων τεχνολογιών όπως CSS, JavaScript και Angular για τη δημιουργία των σελίδων. Αλλά είναι επίσης δυνατή η δημιουργία σελίδων μόνο με HTML.



Σχήμα 35: HTML Layout

- **HTML Images**

  Στην αρχή του διαδικτύου, υπήρχαν μόνο κείμενα και όχι εικόνες. Αργότερα προστέθηκε η δυνατότητα της ενσωματωσης φωτογραφιων και άλλων τύπων περιεχομένου. Οι εικόνες δεν είναι μόνο όμορφες στην εμφάνιση, αλλά και σημαντικές στη χρήση για την εύκολη περιγραφή πολύπλοκων εννοιών στην ιστοσελίδα.

- **HTML Classes**

  Το χαρακτηριστικό HTML class χρησιμοποιείται για να οριστεί το ίδιο στυλ ή μια μορφοποίηση μεσο κώδικα για ένα ή πολλά στοιχεία στην ιστοσελίδα. Όλα τα στοιχεία HTML με την ίδια ιδιότητα ονόματος κλάσης θα έχουν το ίδιο στυλ και μορφή.

- **HTML Styles**

  Το στοιχείο HTML <style> περιέχει πληροφορίες σχετικά με το στυλ του εγγράφου ή ενός μέρους του εγγράφου. Στο στοιχείο <style>, βρίσκονται τα CSS, το οποία εφαρμόζονται στο περιεχόμενο. Το στοιχείο <style> μπορεί να τοποθετηθεί στο <head> ή στο <body> του εγγράφου, αλλά συνήθως συνιστάται η τοποθέτηση των στυλ στο <head>. Ακόμα καλύτερο είναι η τοποθέτηση των στυλ σε εξωτερικά φύλλα στυλ και η αναφορά τους με στοιχεία <link>.

- **HTML Input Types**

  Στην HTML, ένα από τα πιο ισχυρά και πολύπλοκα στοιχεία είναι το στοιχείο <Input>. Ο λόγος είναι ο μεγάλος αριθμός συνδυασμών τύπων εισόδου και χαρακτηριστικών. Ανεξάρτητα από τον τύπο, όλα τα στοιχεία <input> βασίζονται στη διεπαφή HTMLInputElement και αυτό σημαίνει ότι όλα μοιράζονται το ίδιο ακριβώς σύνολο χαρακτηριστικών. Ωστόσο, δεν λειτουργούν όλες οι ιδιότητες για όλους τους τύπους εισόδου και ορισμένοι από τους τύπους εισόδου υποστηρίζουν μόνο μερικά από αυτά τα χαρακτηριστικά. Το στοιχείο HTML <input> χρησιμοποιείται για τη δημιουργία στοιχείων ελέγχου που είναι διαδραστικά για φόρμες που βασίζονται στον ιστό. Αυτά χρησιμοποιούνται για την αποδοχή δεδομένων από τον χρήστη. Υπάρχουν πολλοί διαφορετικοί τύποι γραφικών στοιχείων εισόδου και ελέγχου για πολλούς διαφορετικούς τύπους συσκευών.

Σχήμα 36: HTML Input types

- **HTML Methods for Rest Services**

  Υπάρχουν πολλές διαθέσιμες μέθοδοι HTTP για την μεταφορά και την λήψη δεδομένων από API, για παράδειγμα η GET, POST, DELETE και PUT. Οι περισσότερες υπηρεσίες REST βασίζονται σε CRUD το οποίο βασίζεται στο πρωτόκολλο HTTP και υποστηρίζει τις μεθόδους POST, GET, PUT και DELETE.

- **HTTP GET Method**

  Τα αιτήματα GET είναι οι πιο ευρέως χρησιμοποιούμενες από τις μεθόδους HTTP και χρησιμοποιείται για την ανάκτηση δεδομένων από έναν διακομιστή. Για παράδειγμα, σε ένα API με τελικό σημείο /cars μια υποβολή αιτήματος GET σε αυτό το τελικό σημείο θα επιστρέψει μια λίστα με όλα τα διαθέσιμα αυτοκίνητα. Δεδομένου ότι ένα αίτημα GET ζητά μόνο δεδομένα και δεν τροποποιεί πόρους, θεωρείται ασφαλής μέθοδος.

- **HTTP POST Method**

  Τα αιτήματα POST χρησιμοποιούνται για την αποστολή δεδομένων στον διακομιστή για τη δημιουργία ενός πόρου. Τα δεδομένα που αποστέλλονται αποθηκεύονται στο σώμα αιτήματος του αιτήματος HTTP. Ένα παράδειγμα είναι μια φόρμα επικοινωνίας σε έναν ιστότοπο που

οταν συμπληρωθει και θα αποσταλει στο διακομιστή με τα δεδομένα τοποθετημενα στο σώμα του αιτήματος. Αυτό μπορεί να είναι JSON, XML ή παράμετροι ερωτήματος (υπάρχουν πολλές άλλες μορφές, αλλά αυτές είναι οι πιο συνηθισμένες).

- **HTTP PUT Method**

  Τα αιτήματα PUT χρησιμοποιούνται για την αποστολή δεδομένων στον διακομιστή για την ενημέρωση ενός πόρου. Η κλήση του ίδιου αιτήματος PUT πολλές φορές θα έχει πάντα το ίδιο αποτέλεσμα. Αντίθετα, η επανειλημμένη κλήση ενός αιτήματος POST θα έχει παρενέργειες από τη δημιουργία του ίδιου πόρου πολλές φορές.

- **HTTP DELETE Method**

  Τα αιτήματα DELETE χρησιμοποιούνται για την διαγραφη ενος πόρου. Για παράδειγμα αν δημιουργηθεί ένα νέο αυτοκίνητο με αίτημα POST προς /cars και μπορεί να ανακτηθεί με αίτημα GET στο /cars/{{id}}, τότε η υποβολή ενός αιτήματος DELETE στο /cars/{{id}} θα διαγράψει αυτό το αμάξι.

2.18.4 CSS

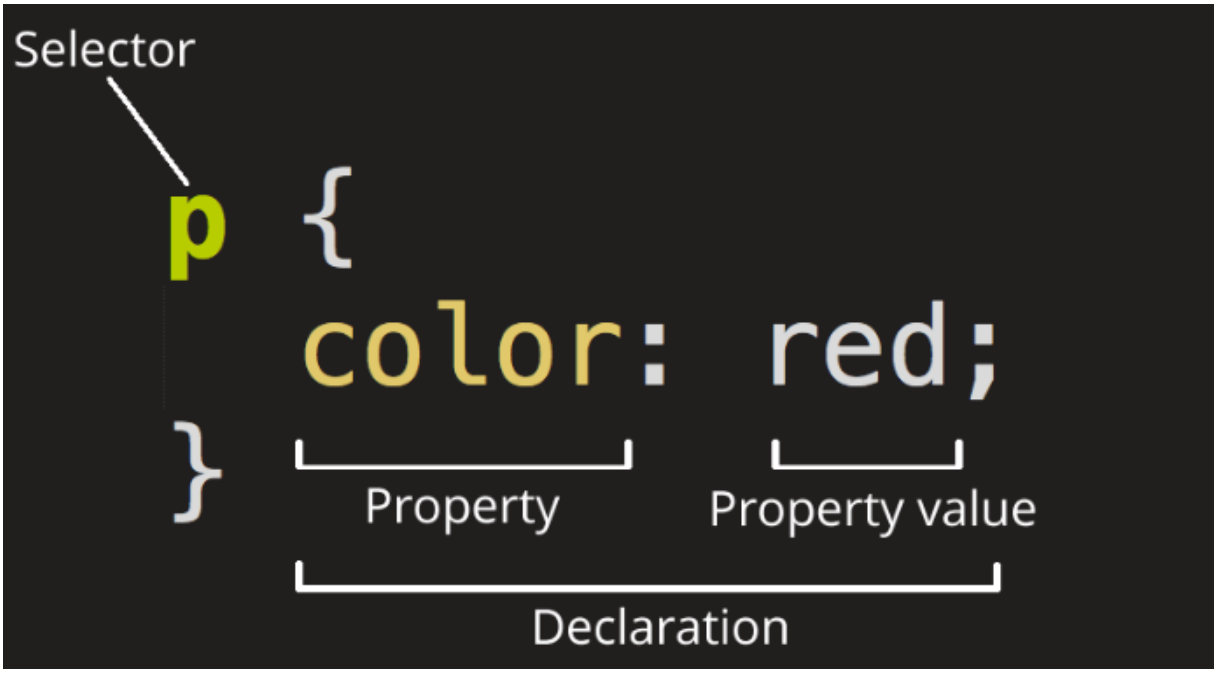
2.18.5 Τι είναι τα CSS

Τα Cascading Style Sheets ή αλλιως CSS είναι μια γλώσσα που χρησιμοποιείται για να περιγράψει την παρουσίαση ενός εγγράφου γραμμένου σε HTML. Το CSS περιγράφει πώς τα στοιχεία πρέπει να αποδίδονται στην οθόνη, στο χαρτί, στην ομιλία ή σε άλλα μέσα. Η CSS είναι μια από τις βασικές γλώσσες του ανοιχτού ιστού και είναι τυποποιημένη σε προγράμματα περιήγησης Ιστού σύμφωνα με τις προδιαγραφές του W3C. Προηγουμένως, η ανάπτυξη διαφόρων τμημάτων των προδιαγραφών CSS γινόταν συγχρονισμένα, γεγονός που επιτρέπει την έκδοση των πιο πρόσφατων συστάσεων. Μετά το CSS 2.1, το εύρος των προδιαγραφών αυξήθηκε σημαντικά και η πρόοδος σε διάφορες μονάδες CSS άρχισε να διαφέρει τόσο πολύ, που έγινε πιο αποτελεσματικό να αναπτύσσονται και να κυκλοφορούν προτάσεις χωριστά ανά ενότητα.[5][9]

```
p {
  color: red;
}
```

Σχήμα 37: CSS παράδειγμα

2.18.6 Ανατομία ενός κανόνα CSS



Σχήμα 38: CSS ανατομία

Ολόκληρη η δομή ονομάζεται σύνολο κανόνων και περιέχει τα επιμέρους τμήματα:

- Selector: Αυτό είναι το όνομα του στοιχείου HTML στην αρχή του συνόλου κανόνων το οποίο καθορίζει τα στοιχεία που θα διαμορφωθούν σε αυτό. Στην παραπάνω εικόνα επιλέγχθηκε να διαμορφωθουν τα HTML στοιχεία <p>.
- Declaration: αυτός είναι ένας κανόνας όπως το “color: red;” και καθορίζει σε ποιες από τις ιδιότητες του στοιχείου θα αλλάξει η εμφάνιση.
- Property: είναι μια ιδιότητα του πού αργότερα της αποδίδεται μια τιμή.
- Property value: είναι η τιμή από ένα σύνολο επιλογών που θέλουμε να δώσουμε σε μια ιδιότητα. Για παράδειγμα η ιδιότητα “color” δέχεται τιμες οπως χρώματα “red” ή “green”.

Τέλος το CSS της παραπάνω εικόνας μεταφράζεται σε “όλες οι παράγραφοι θα εμφανίζουν το κείμενο με κόκκινο χρώμα.”.

2.18.7 Javascript

2.18.8 Τι είναι η Javascript ?

Η JavaScript είναι μια γλώσσα προγραμματισμού που επιτρέπει την εφαρμογή σύνθετων λειτουργιών σε ιστοσελίδες κανοντάς τες πιο ζωντανές. Όπως αναφέρθηκε και παραπάνω τα βασικά μέρη μιας ιστοσελίδας είναι η HTML, το CSS και η Javascript. Η HTML είναι ο σκελετός της ιστοσελίδας, τα CSS καθορίζει πώς θα εμφανίζεται αυτός ο σκελετός και η Javascript μπορει να ενημερώσει και να επηρεάσει δυναμικά τα CSS και την HTML. [5][8]

2.18.9 Scripting γλώσσες προγραμματισμού

Η Javascript μερικές φορές αναφέρεται ως η γλώσσα Scripting. Ένα πρόγραμμα γραμμένο σε μια γλώσσα προγραμματισμού όπως η C πρέπει να μεταγλωττιστεί για να μπορέσει να εκτελεστεί, ενώ η javascript δεν χρειάζεται να μεταγλωττιστεί. Ο κώδικας Javascript περιέχει μια σειρά από εντολές που ερμηνεύονται μία προς μία στο χρόνο εκτέλεσης.

2.18.10 Πώς λειτουργεί η JavaScript;

Η JavaScript είτε είναι ενσωματωμένη σε μια ιστοσελίδα είτε περιλαμβάνεται σε ένα αρχείο .js. Η Javascript έχει πιο νόημα να λειτουργεί μετά τη φόρτωση του HTML και των CSS στη σελίδα, επομένως τοποθετείται στο κλείσιμο της ετικέτας σώματος σε HTML.

Βασικό πρότυπο HTML:

```
<html>
    <head>
       <style>/*I am Style of The Website*/</style>
    </head>
    <body>
       <!--I am body of Website. Everything is inside me-->

    </body>
</html>
```

Σχήμα 39: Πρότυπο HTML

Με javascript:

```
<html>
    <head>
       <style>/*I am Style of The Website*/</style>
    </head>
    <body>
       <!--I am body of Website. Everything is inside me-->

       <!--After Everything-->
       <script type="text/javascript">
           //I am Javascript. I make website funny place.
       </script>

       <!-- OR With External .js File-->
       <script type="text/javascript" src="./myFile/myScript.js">
       </script>
    </body>
</html>
```

Σχήμα 40: Πρότυπο Javascript

2.18.11 Χρήσεις Javascript στον σημερινό κόσμο

Παρόλο που η javascript δημιουργήθηκε για να κάνει τους ιστότοπους διαδραστικούς, αυτή τη στιγμή χρησιμοποιείται για:

- Για την ανάπτυξη εφαρμογών για κινητές συσκευές (React Native, Native Script, Ionic)
- Για την ανάπτυξη παιχνιδιών που βασίζονται σε πρόγραμμα περιήγησης
- Ανάπτυξη εφαρμογών ιστου στο πίσω μέρος της εφαρμογής (Nodejs)

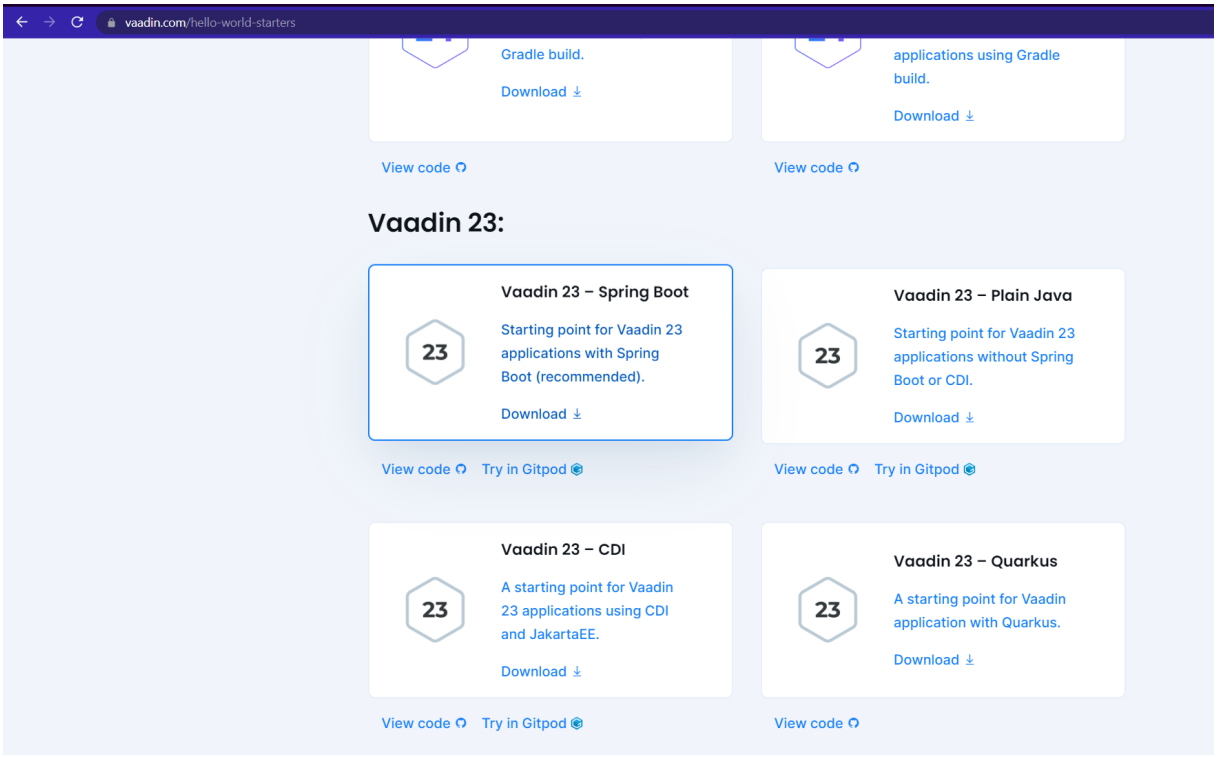
Κεφάλαιο 3ο: Αρχιτεκτονική λογισμικού Productive Manager

3.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί θα παρουσιαστούν οι τεχνικές πτυχές της εργασίας. Συγκεκριμένα, θα αναλυθεί η δημιουργία ένα νεου Spring Boot Vaadin project χρησιμοποιώντας το IDE Intellij και το vaadin starter project. Τέλος, θα δοθεί μια λεπτομερής περιγραφή του σχεσιακού σχήματος των οντοτήτων της βάσης της εργασίας και θα αναφερθούν οι επιμέρους λειτουργίες κάθε τομέα.

3.2 Δημιουργία project

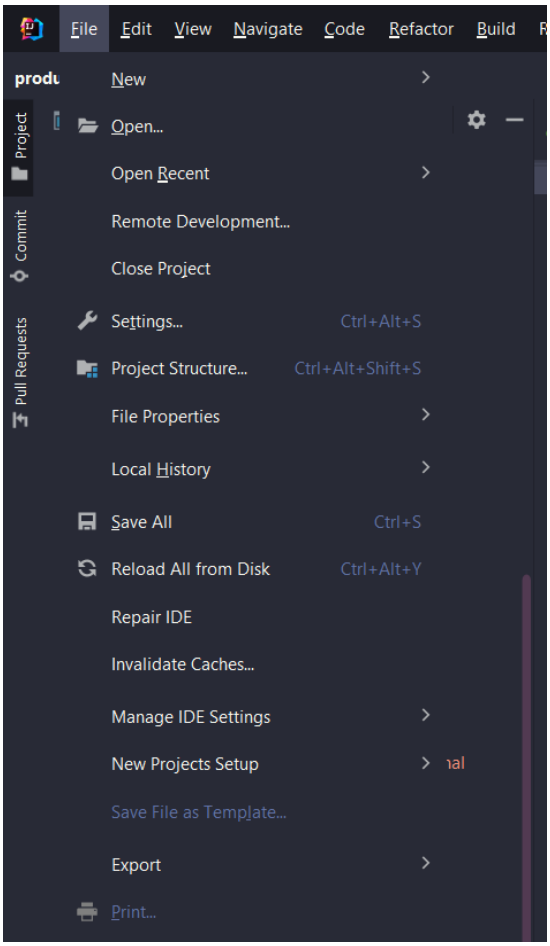
Η δημιουργία ενός Vaadin project μπορεί να γίνει με πολλούς τρόπους. Ο πιο εύκολος και ενδεδειγμένος τρόπος είναι με την χρήση ενός προ φτιαγμένου αρχικού project το οποίο θα περιέχει τις αναγκαίες βιβλιοθήκες και την διασυνδεση με το Spring Boot. Αυτό μπορεί να γίνει με την απο την ιστοσελίδα https://vaadin.com/hello-world-starters η οποία περιέχει αρκέτα προ φτιαγμένα project ανάλογα με τις απαιτησεις της πλατφορμας.



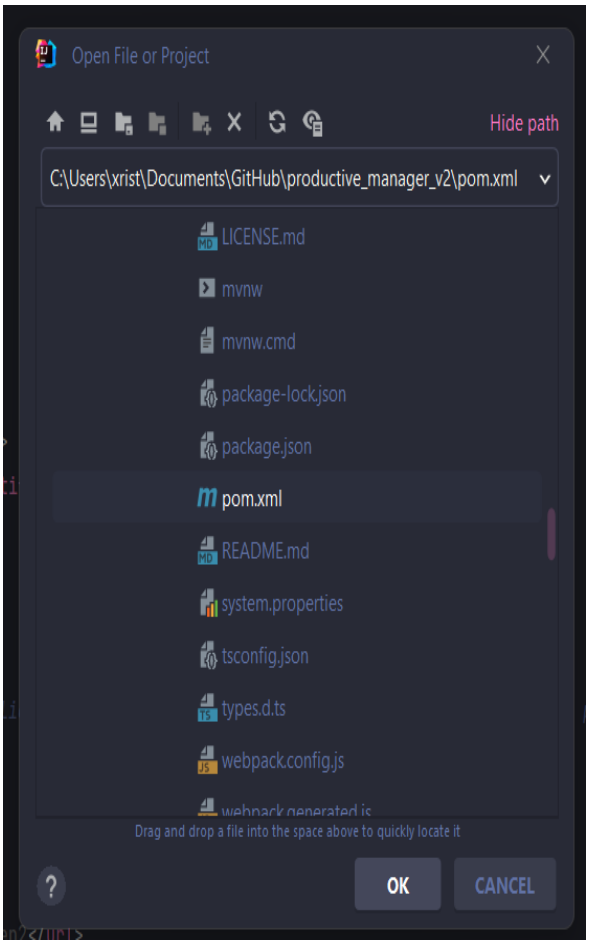
Σχήμα 41: Vaadin - Spring Boot

Απο την παραπάνω ιστοσελίδα έγινε η επιλογή της έκδοσης Vaadin 23 καθώς κατά την ανάπτυξη

ήταν η πιο ενημερωμένη και σταθερή έκδοση που πρόσφερε το Vaadin. Αφού γίνει η λήψη του αρχείου θα πρέπει να ανοιχτεί ως νεο project στο Intellij IDEA βρίσκοντας και ανοιγοντας το αρχείο pom.xml. Στη περίπτωση που δεν υπάρχει το Intellij IDEA θα χρειαστεί ένα διαφορετικό πρόγραμμα ανάπτυξης κώδικα.



Σχήμα 42: IntelliJ Δημιουργία Project 1



Σχήμα 43: IntelliJ Δημιουργία Project 2

Τέλος αφού ανοίξει το project και φορτώσουν τα απαραίτητα αρχεία η εφαρμογή πρέπει να δοκιμαστεί ότι μπορεί να ξεκινήσει χρησιμοποιώντας το κουμπί “Run Application” του Intellij IDEA.

Σχήμα 44: IntelliJ Δημιουργία Project 3

Απο την παραπάνω εικόνα μπορούμε να δούμε ότι η εφαρμογή ξεκίνησε κανονικά την λειτουργία της. Επιπλέον, λόγο ότι είναι μια εφαρμογή διαδικτύου μπορούμε να την δοκιμάσουμε πηγαίνοντας στο στην προκαθορισμένη διεύθυνση http://localhost:8080 μέσω ενός περιηγητή.

3.3 Πακέτα κώδικα της εφαρμογής

Η εφαρμογή Productive Manager αποτελείται από πολλά πακέτα που συνεργάζονται για να παρέχουν μια ολοκληρωμένη λύση για τη διαχείριση δεδομένων και παραγγελιών πελατών. Κάθε πακέτο έχει έναν συγκεκριμένο ρόλο στην εφαρμογή και έχει σχεδιαστεί για να λειτουργεί σε συνδυασμό με τα άλλα για να δημιουργήσει μια αδιάλειπτη εμπειρία για τους χρήστες.

Πακέτα

- **Backend**: Περιέχει τη βασική λογική και τους επιχειρηματικούς κανόνες της εφαρμογής, καθώς και διεπαφές για αλληλεπίδραση με άλλα πακέτα.
- **Data**: Ασχολείται με την αποθήκευση και την ανάκτηση δεδομένων που χρησιμοποιούνται από την εφαρμογή και παρέχει ένα επίπεδο αφαίρεσης μεταξύ της εφαρμογής και της βάσης δεδομένων.
- **Entity**: Περιέχει τις κλάσεις που χρησιμοποιούνται για την αναπαράσταση οντοτήτων στην εφαρμογή, όπως πελάτες και παραγγελίες.
- **Exceptions**: Ασχολείται με εξαιρέσεις που ενδέχεται να προκύψουν κατά την εκτέλεση της εφαρμογής, παρέχοντας έναν τρόπο χειρισμού σφαλμάτων και εμφάνισης σημαντικών μηνυμάτων στους χρήστες.
- **Repository**: Παρέχει ένα σύνολο διεπαφών για αλληλεπίδραση με τη βάση δεδομένων, επιτρέποντας εύκολη αναζήτηση και χειρισμό δεδομένων.
- **Resources**: Περιέχει πόρους που χρησιμοποιούνται από την εφαρμογή, όπως εικόνες και αρχεία κειμένου.
- **Services**: Εφαρμόζει την επιχειρηματική λογική της εφαρμογής, σε στενή συνεργασία με το επίπεδο δεδομένων απο το πακετο Repository για την παροχή λειτουργικότητας στους χρήστες.
- **Util**: Περιέχει κλάσεις και μεθόδους χρησιμότητας που χρησιμοποιούνται σε όλη την εφαρμογή, παρέχοντας κοινή λειτουργικότητα με αρθρωτό και επαναχρησιμοποιήσιμο τρόπο.
- **Security**: Παρέχει λειτουργικότητα που σχετίζεται με την ασφάλεια, όπως έλεγχο ταυτότητας και εξουσιοδότηση.
- **UI**: Περιέχει τα στοιχεία διεπαφής χρήστη της εφαρμογής, επιτρέποντας στους χρήστες να αλληλεπιδρούν με το σύστημα και να προβάλλουν δεδομένα.
- **View**: Παρέχει τις προβολές που χρησιμοποιεί η εφαρμογή, όπως η αρχική οθόνη και η οθόνη λεπτομερειών πελάτη.
- **Forms**: Περιέχει τις φόρμες που χρησιμοποιούνται για τη δημιουργία και την ενημέρωση οντοτήτων στην εφαρμογή.

Στις επόμενες ενότητες, θα ρίξουμε μια πιο προσεκτική ματιά στα βασικά πακέτα της εφαρμογής και τα αντίστοιχα στοιχεία τους, παρέχοντας λεπτομερείς πληροφορίες για το ρόλο τους στην εφαρμογή και πώς συνεργάζονται για να δημιουργήσουν μια απρόσκοπτη εμπειρία στο χρήστη.

3.3.1 Πακέτο Entities

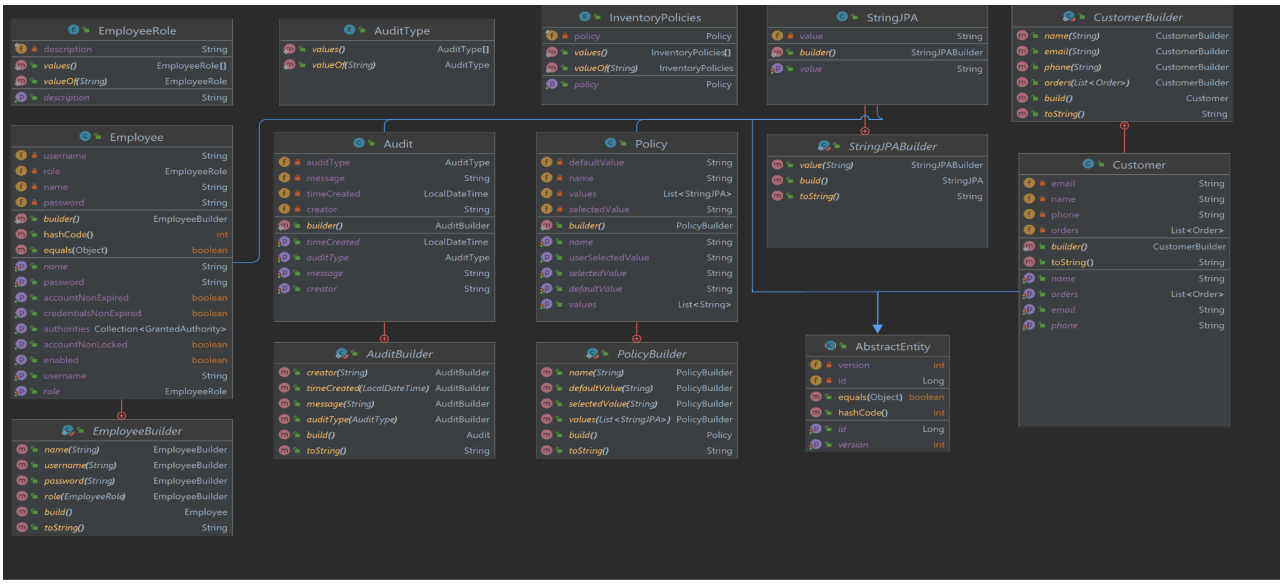
Το πακέτο "backend.data.entity" περιέχει κλάσεις που περιγράφουν τις θεμελιώδεις οντότητες δεδομένων της εφαρμογής και είναι υπεύθυνες για την αποθήκευση και την ανάκτηση πληροφοριών που σχετίζονται με πελάτες, παραγγελίες, πολιτικές και άλλα σημαντικά δεδομένα. Κάθε οντότητα έχει σχεδιαστεί για να αντιπροσωπεύει ένα συγκεκριμένο αντικείμενο δεδομένων και έχει ιδιότητες που αντιστοιχούν στα διαφορετικά πεδία και χαρακτηριστικά αυτού του αντικειμένου. Μέσα σε αυτό το πακέτο, υπάρχουν πολλές κλάσεις που επεκτείνουν την κλάση AbstractEntity, η οποία παρέχει ένα κοινό σύνολο ιδιοτήτων και μεθόδων που μοιράζονται όλες οι οντότητες. Αυτές οι κλάσεις περιλαμβάνουν τις οντότητες όπως ο Πελάτης, μια Παραγγελίας και μια Πολιτική, καθεμία από τις οποίες έχει το δικό της συγκεκριμένο σύνολο ιδιοτήτων και μεθόδων που σχετίζονται με το συγκεκριμένο αντικείμενο δεδομένων. Μαζί, αυτές οι κατηγορίες οντοτήτων παρέχουν τη ραχοκοκαλιά της εφαρμογής Productive Manager, επιτρέποντας την αποτελεσματική αποθήκευση και ανάκτηση δεδομένων που υποστηρίζουν την ομαλή λειτουργία της εφαρμογής. Για την σύνδεση με την βάση δεδομένων το πακέτο Entities, για κάθε κλάση χρησιμοποιεί το πρότυπο JPA που αναφέρθηκε στο προηγούμενο κεφάλαιο.

Παρακάτω θα δούμε την κάθε κλαση αναλυτικά

- **AbstractEntity**: Η κλάση AbstractEntity αποτελεί τη βάση για όλες τις οντότητες που υπάρχουν στο σύστημα διαχείρισης αποθεμάτων. Περιλαμβάνει ένα μοναδικό αναγνωριστικό και έναν αριθμό έκδοσης για κάθε οντότητα. Η χρήση της κλάσης αυτής επιτρέπει την ενοποίηση και την απλοποίηση της υλοποίησης των υπόλοιπων κλάσεων του συστήματος.
- **Audit**: Η κλάση Audit επεκτείνει την κλάση AbstractEntity και καταγράφει τις ενέργειες που γίνονται στο σύστημα διαχείρισης αποθεμάτων. Περιλαμβάνει πληροφορίες όπως ο χρήστης που πραγματοποίησε την ενέργεια, η ώρα που πραγματοποιήθηκε, μια περιγραφή της ενέργειας και το επίπεδο σοβαρότητας του ελέγχου που εκτελέστηκε. Τέλος, χρησιμοποιώντας αυτήν την κλάση το εργαλείο διαχείρισης αποθέματος παρέχει σημαντικές πληροφορίες για την ιστορία του συστήματος και την ανίχνευση σφαλμάτων ή ανωμαλιών στη λειτουργία του.
- **Category**: Αυτή η κλάση επεκτείνει την AbstractEntity και αντιπροσωπεύει μια κατηγορία προϊόντων στο εργαλείο διαχείρισης αποθεμάτων. Περιέχει ένα πεδίο ονόματος που είναι μοναδικό και όχι κενό, καθώς και μια λίστα προϊόντων που ανήκουν στην κατηγορία.
- **Customer**: Αυτή η κατηγορία επεκτείνει το AbstractEntity και αντιπροσωπεύει έναν πελάτη της επιχείρησης. Περιέχει πεδία για το όνομα, το email και τον αριθμό τηλεφώνου του πελάτη, καθώς και μια λίστα με τις παραγγελίες που έχει κάνει ο πελάτης.
- **Employee**: Αυτή η κλάση επεκτείνει την AbstractEntity και υλοποιεί την UserDetails και αντιπροσωπεύει έναν υπάλληλο της επιχείρησης. Περιέχει πεδία για το όνομα, το όνομα χρήστη και τον κωδικό πρόσβασης του υπαλλήλου, καθώς και τον ρόλο του στην επιχείρηση (όπως διευθυντής ή απλός υπάλληλος).
- **Order**: Η κλάση επεκτείνει την AbstractEntity και αντιπροσωπεύει μια παραγγελία που υποβάλλεται για έναν πελάτη. Περιέχει πεδία για τον πελάτη που έκανε την παραγγελία, την ώρα παραγγελίας, την κατάσταση της παραγγελίας (όπως "νέα" ή "πληρωμένη"), τη συνολική τιμή της παραγγελίας και μια λίστα με αντικείμενα τύπου OrderProduct. Με τη χρήση αυτής της κλάσης, το εργαλείο διαχείρισης αποθέματος μπορεί να παρακολουθεί τις παραγγελίες πελατών και τα επίπεδα αποθέματος.
- **OrderProduct**: Η κλάση επεκτείνει την AbstractEntity και αντιπροσωπεύει ένα συγκεκριμένο προϊόν που παραγγέλθηκε ως μέρος μιας παραγγελίας. Περιέχει ένα πεδίο για το αντικείμενο

Product που παραγγέλθηκε, καθώς και πεδία για την ποσότητα που παραγγέλθηκε και την τιμή στην οποία παραγγέλθηκε. Χρησιμοποιώντας αυτήν την κλάση, το εργαλείο διαχείρισης αποθέματος μπορεί να παρακολουθεί ποια προϊόντα παραγγέλθηκαν, πόσα από κάθε προϊόν παραγγέλθηκαν και την τιμή στην οποία πωλήθηκαν.

- **Product**: Η κλάση επεκτείνει το AbstractEntity και αντιπροσωπεύει ένα προϊόν που μπορεί να πωληθεί ή να διαχειριστεί στο σύστημα. Περιέχει πεδία για το όνομα του προϊόντος, την τιμή, το κόστος, την ποσότητα, την κατηγορία, τον προμηθευτή, τον φόρο, τη διεύθυνση URL εικόνας, το barcode και εάν το προϊόν είναι ενεργοποιημένο ή επισημασμένο για διαγραφή. Το πεδίο κατηγορίας συνδέει το προϊόν με μια κατηγορία στην οποία ανήκει, ενώ το πεδίο προμηθευτή το συνδέει με τον προμηθευτή που το παρέχει. Το πεδίο φόρου το συνδέει με έναν φόρο που εφαρμόζεται στο προϊόν. Τέλος, χρησιμοποιώντας αυτήν την κλάση το εργαλείο διαχείρισης αποθέματος μπορεί να παρακολουθεί τα προϊόντα, τις τιμές, τις ποσότητες και άλλες λεπτομέρειες, καθώς και να τα συνδέσει με κατηγορίες, προμηθευτές και φόρους.
- **ProductSupplier**: Η κλάση ProductSupplier επεκτείνει το AbstractEntity και αντιπροσωπεύει έναν προμηθευτή προϊόντων στο ένα σύστημα. Περιέχει πεδία για το όνομα, το email και τον αριθμό τηλεφώνου του προμηθευτή. Χρησιμοποιώντας αυτή την κλάση, το εργαλείο διαχείρισης αποθέματος μπορεί να παρακολουθεί τους προμηθευτές και τα στοιχεία επικοινωνίας τους, καθώς και να συνδέσει προϊόντα με τον προμηθευτή που τους παρέχει.
- **Tax**: Η κλάση επεκτείνει το AbstractEntity και αντιπροσωπεύει έναν φόρο που μπορεί να εφαρμοστεί σε προϊόντα σε ένα σύστημα διαχείρισης αποθεμάτων. Περιέχει πεδία για το όνομα και την αξία φόρου. Χρησιμοποιώντας αυτήν την κατηγορία, το εργαλείο διαχείρισης μπορεί να παρακολουθεί τους φόρους και το ποσοστό με το οποίο εφαρμόζονται στα προϊόντα, επιτρέποντας ακριβείς τιμές και αναφορές.
- **Policy**: Η κλάση Policy επεκτείνει την AbstractEntity και αντιπροσωπεύει μια πολιτική που χρησιμοποιείται στο σύστημα. Περιέχει πεδία για το όνομα της πολιτικής, την προεπιλεγμένη τιμή της και την τιμή που έχει επιλεγεί. Επιπλέον, περιέχει ένα πεδίο για τη λίστα των τιμών που μπορούν να επιλεγούν για αυτήν την πολιτική, και αυτό πραγματοποιείται μέσω μιας λίστας των StringJPA. Χρησιμοποιώντας αυτήν την κλάση, το σύστημα μπορεί να παρακολουθεί τις πολιτικές που χρησιμοποιούνται και ποιες τιμές έχουν επιλεγεί για κάθε μία από αυτές. όπως για παράδειγμα ο τρόπος χειρισμού των διαθέσιμων αποθεμάτων.
- **StringJPA**: Η κλάση StringJPA επεκτείνει την AbstractEntity και αντιπροσωπεύει μια απλή αλφαριθμητική τιμή και περιέχει μόνο ένα πεδίο για την τιμή του αλφαριθμητικού. Χρησιμοποιείται από το Java Persistence API για την αποθήκευση ή τον χειρισμό δεδομένων μια λίστας που περιέχει συμβολοσειρές στο σύστημα διαχείρισης αποθεμάτων.



Σχήμα 45: Πακέτο Entities 1

Σχήμα 46: Πακέτο Entities 2

3.3.2 Πακέτο Services

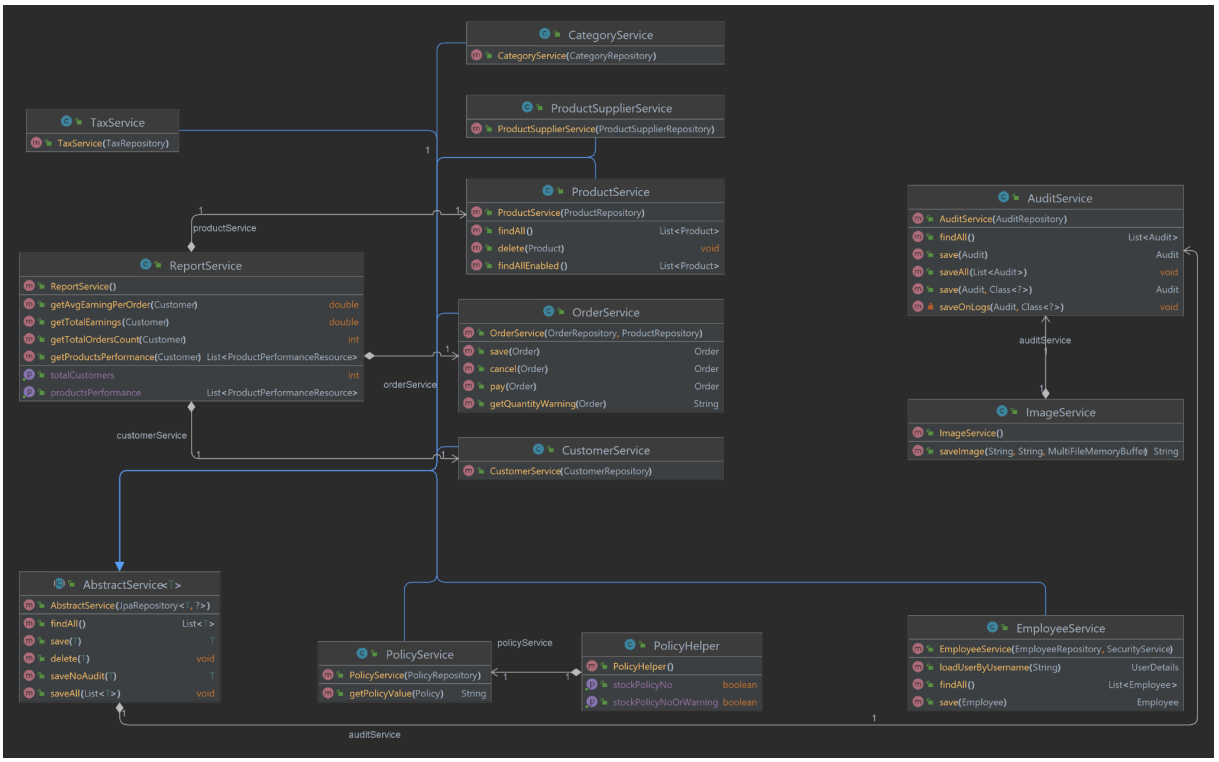
Το πακέτο Services περιέχει κλάσεις που υλοποιούν την βασική επιχειρηματική λογική της εφαρμογής, σε στενή συνεργασία με το επίπεδο δεδομένων που παρέχεται από το πακέτο repository για την παροχή λειτουργικότητας στους χρήστες. Αυτές οι υπηρεσίες είναι υπεύθυνες για την εκτέλεση των κύριων λειτουργιών της εφαρμογής, όπως η διαχείριση δεδομένων προϊόντων, ο χειρισμός παραγγελιών και η δημιουργία αναφορών.

Παρακάτω θα δούμε την κάθε κλαση αναλυτικά:

- **AbstractService:** Αυτή η αφηρημένη κλάση παρέχει μια βασική υλοποίηση λειτουργιών για τις υπόλοιπες κλάσεις υπηρεσιών που την κληρονομούν, ορίζοντας κοινές μεθόδους για την πρόσβαση και την διαχείριση δεδομενων ως προς την βάση.
- **AuditService:** Αυτή η κλάση παρέχει λειτουργικότητα για την παρακολούθηση και την καταγραφή ενεργειών και συμβάντων του χρήστη και την καταγραφή τους στη βάση δεδομένων, για σκοπούς ελέγχου και εντοπισμού σφαλμάτων.
- **CategoryService:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση των κατηγοριών των προϊόντων, όπως η δημιουργία, η ενημέρωση και η διαγραφή εγγραφών κατηγοριών.
- **CustomerService:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση δεδομένων πελατών, όπως τη δημιουργία, την ενημέρωση και τη διαγραφή εγγραφών εγγραφών πελατών.
- **EmployeeService:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση δεδομένων υπαλλήλων, όπως δημιουργία, ενημέρωση και διαγραφή εγγραφών υπαλλήλων.
- **ImageService:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση εικόνων, όπως η αποστολή, η λήψη και η διαγραφή αρχείων εικόνας.
- **OrderService:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση παραγγελιών προϊόντων, όπως η δημιουργία, η ενημέρωση και η διαγραφή εγγραφών παραγγελιών.
- **PolicyHelper:** Αυτή η κλάση παρέχει βοηθητικές μεθόδους για την εφαρμογή πολιτικών ελέγχου της συμπεριφοράς του συστήματος.
- **Policy Service:** Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση πολιτικών ελέγχου συμπεριφοράς, όπως η δημιουργία, η ενημέρωση και η διαγραφή πολιτικών.
- **ProductService**: Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση προϊόντων, όπως τη δημιουργία, την ενημέρωση και τη διαγραφή εγγραφών προϊόντων.
- **ProductSupplierService**: Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση προμηθευτών προϊόντων, όπως τη δημιουργία, την ενημέρωση και τη διαγραφή εγγραφών προμηθευτών και την υποβολή ερωτημάτων σε αυτές.
- **Report Service:** Αυτή η κλάση παρέχει λειτουργικότητα για τη δημιουργία αναφορών πωλήσεων.
- **Tax Service**: Αυτή η κλάση παρέχει λειτουργικότητα για τη διαχείριση φορολογικών συντελεστών, όπως η δημιουργία, η ενημέρωση και η διαγραφή φορολογικών εγγραφών.
- **Security Service**: Αυτή η κλάση παρέχει λειτουργικότητα για την πιστοποίηση των χρηστών της εφαρμογής και τον περιορισμων των διαθεσιμων λειτουργιων αναλογα με το είδος του χρήστη (Διαχειριστής, Υπαλληλος, Ανωνυμος).

Συνολικά, το πακέτο υπηρεσιών διαδραματίζει κρίσιμο ρόλο στην υλοποίηση της βασικής λειτουργικότητας της εφαρμογής και δίνει τη δυνατότητα στους χρήστες να εκτελούν διάφορες εργασίες και λειτουργίες. Σε στενή συνεργασία με το πακέτο Entities αυτές οι υπηρεσίες παρέχουν την διαχείριση επιχειρηματικών διαδικασιών και δεδομένων. Είτε πρόκειται για διαχείριση πληροφοριών πελατών, επεξεργασία παραγγελιών, δημιουργία αναφορών το πακέτο υπηρεσιών τις καλύπτει.

Στην παρακάτω εικόνα μπορούμε να δουμε μια αναπαράσταση απο τις κλάσεις και το πως συνδέονται μεταξύ τους, οι κλάσεις των υπηρεσιων.



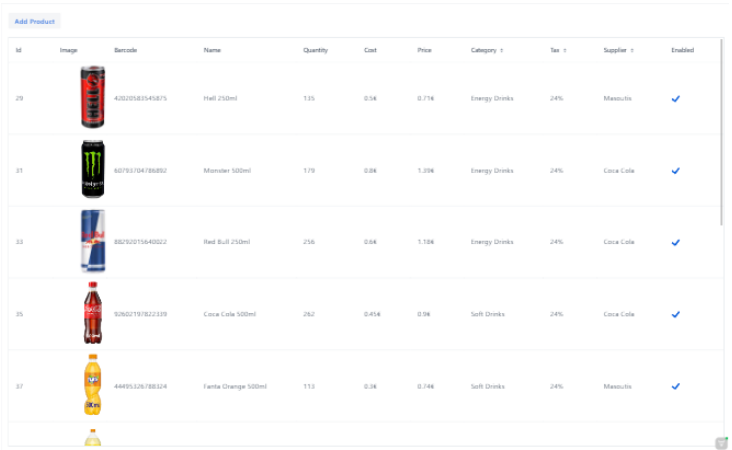
Σχήμα 47: Πακέτο Service

3.3.3 Πακέτο View

Το πακέτο View είναι υπεύθυνο για την δημιουργία της γραφικής διεπαφής του χρήστη της εφαρμογής. Χρησιμεύει ως το κύριο μέσο προς τους χρήστες ώστε να μπορέσουν να αλληλεπιδράσουν με το λογισμικό και να έχουν πρόσβαση στις δυνατότητές του. Το πακέτο View συνδέεται στενά με τα πακέτα Entities και Services ώστε να μπορέσει να εκτελέσει τις λειτουργίες της εφαρμογής. Για τις επιμέρους κλάσεις του πακέτου View γίνεται η χρήση του Vaadin Flow και των Web components του.

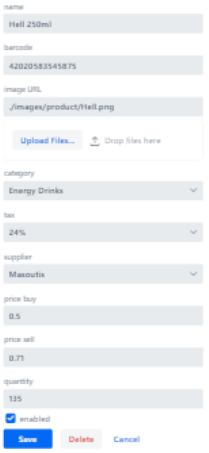
Παρακάτω θα δούμε την κάθε κλαση αναλυτικά:

- **AbstractCrudView:** Αυτή η αφηρημένη κλάση παρέχει μια βασική υλοποίηση λειτουργιών για τις υπόλοιπες κλάσεις αναπαράστασης δεδομένων της γραφικής διεπαφής που την κληρονομούν, ορίζοντας κοινές μεθόδους για την αναπαράσταση και την διαχείριση δεδομενων ως προς τις κλάσεις υπηρεσιών. Η αναπαράσταση των δεδομένων γίνεται με την μορφή πίνακα, όπου οι στήλες αντιστοιχουν στις ιδιότητες και οι σειρές στις εγγραφες. Η παραπάνω κλάση υλοποιείται από τις CategoryView για τις κατηγορίες των προϊόντων, την CustomerView για τους πελάτες, την EmployeeView για τους υπαλλήλους, την OrderView για τις παραγγελιες, την PolicyView για τις παραμετροποιήσεις του συστήματος, την ProductSupplierView για τους προμηθευτές των προϊόντων, την ProductView για τα προϊόντα και τελος, την TaxView για τους φορους.



Σχήμα 48: Abstract Crud View

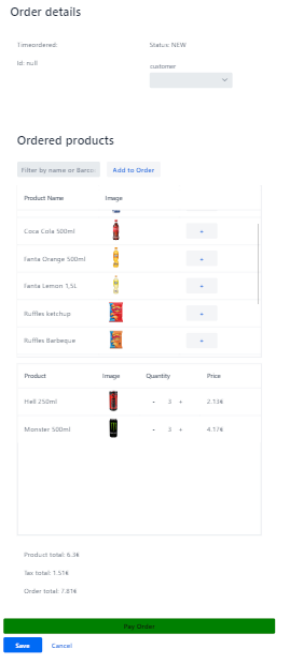
- **AbstractForm:** Αυτή η αφηρημένη κλάση παρέχει μια βασική υλοποίηση λειτουργιών για τις υπόλοιπες κλάσεις με φορμες της γραφικής διεπαφής που την κληρονομούν, ορίζοντας κοινές μεθόδους για την εκχώρηση δεδομενων ως προς την κλάση AbstractCrudView. Η συμπλήρωση των δεδομένων γίνεται με την μορφή πεδίων ομαδοποιημένων σε μια κοινή φορμα. Η παραπάνω κλάση υλοποιείται από τις CategoryForm για τις κατηγορίες των προϊόντων, την CustomerForm για τους πελάτες, την EmployeeForm για τους υπαλλήλους, την OrderForm για τις παραγγελιες, την PolicyForm για τις παραμετροποιήσεις του συστήματος, την ProductSupplierForm για τους προμηθευτές των προϊόντων, την ProductForm για τα προϊόντα και τελος, την TaxForm για τους φορους.



Σχήμα 49: Abstract Form

Επιπλέον υπάρχουν οι παρακάτω κλάσεις οι οποίες δεν υλοποιούν τις AbstractCrudView / AbstractForm για την αναπαράσταση των λειτουργιών τους και τις δομούν ξεχωριστά :

- ProductOrderMaker: περιέχει την γραφική αναπαράσταση για την παραγγελιοληψια και το καλάθι αγορών.

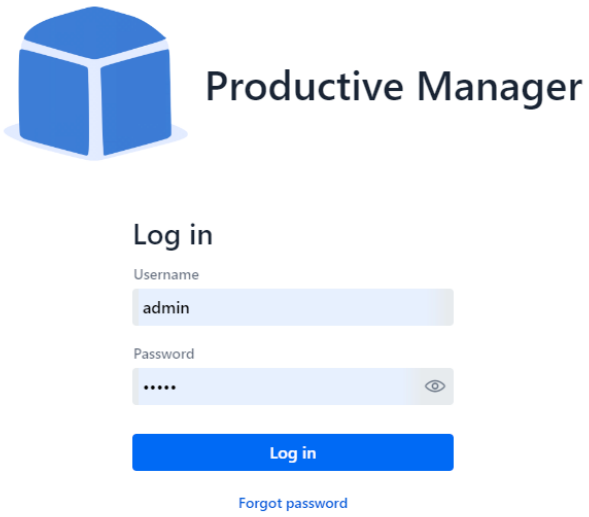


Σχήμα 50: Product Order Maker

- AuditView: προβάλει σε μορφή πίνακα τα δεδομένα για τις ενέργειες που έγιναν στο σύστημα.

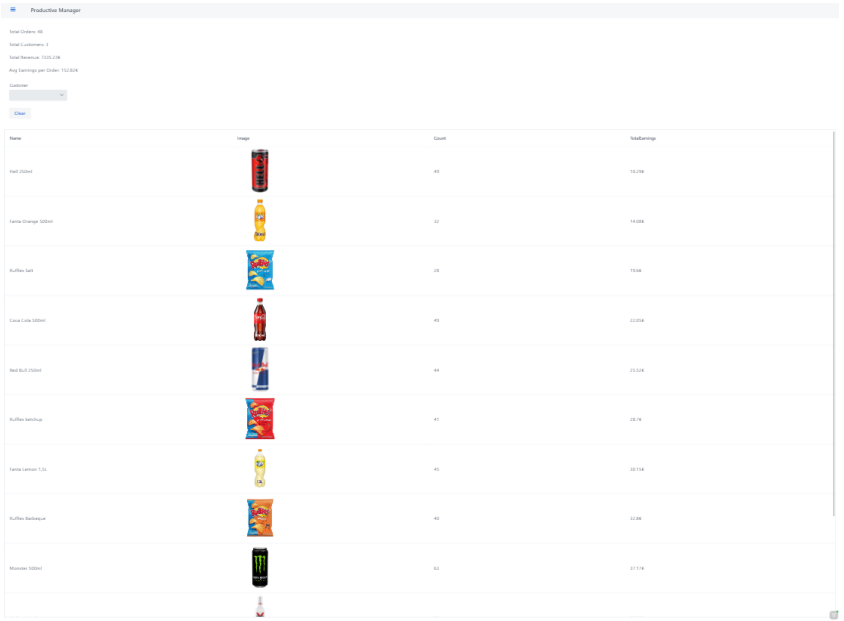
Σχήμα 51: Audit View

- LoginView: εμφανίζει τα πεδια εισαγωγης των στοιχειων σύνδεσης για τους υπαλλήλους και τους διαχειριστές. Η πιστοποίηση των στοιχείων γίνεται μέσω του SecurityService.



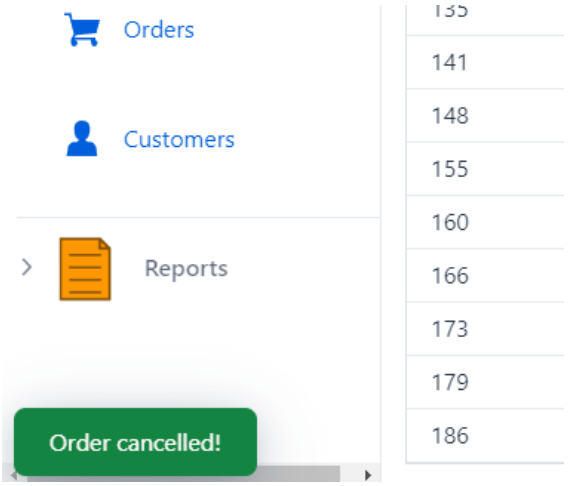
Σχήμα 52: Login View

- ReportProductView: Αναπαριστά στατιστικές αναφορές σχετικά με τις παραγγελίες, τα προιοντα που αγοράστηκε και τους πελάτες.



Σχήμα 53: Report Product View

- NotificationManager: παρέχει την δυνατότητα για την εμφάνιση μηνυμάτων λάθους ή επιτυχίας στον χρήστη της εφαρμογής.

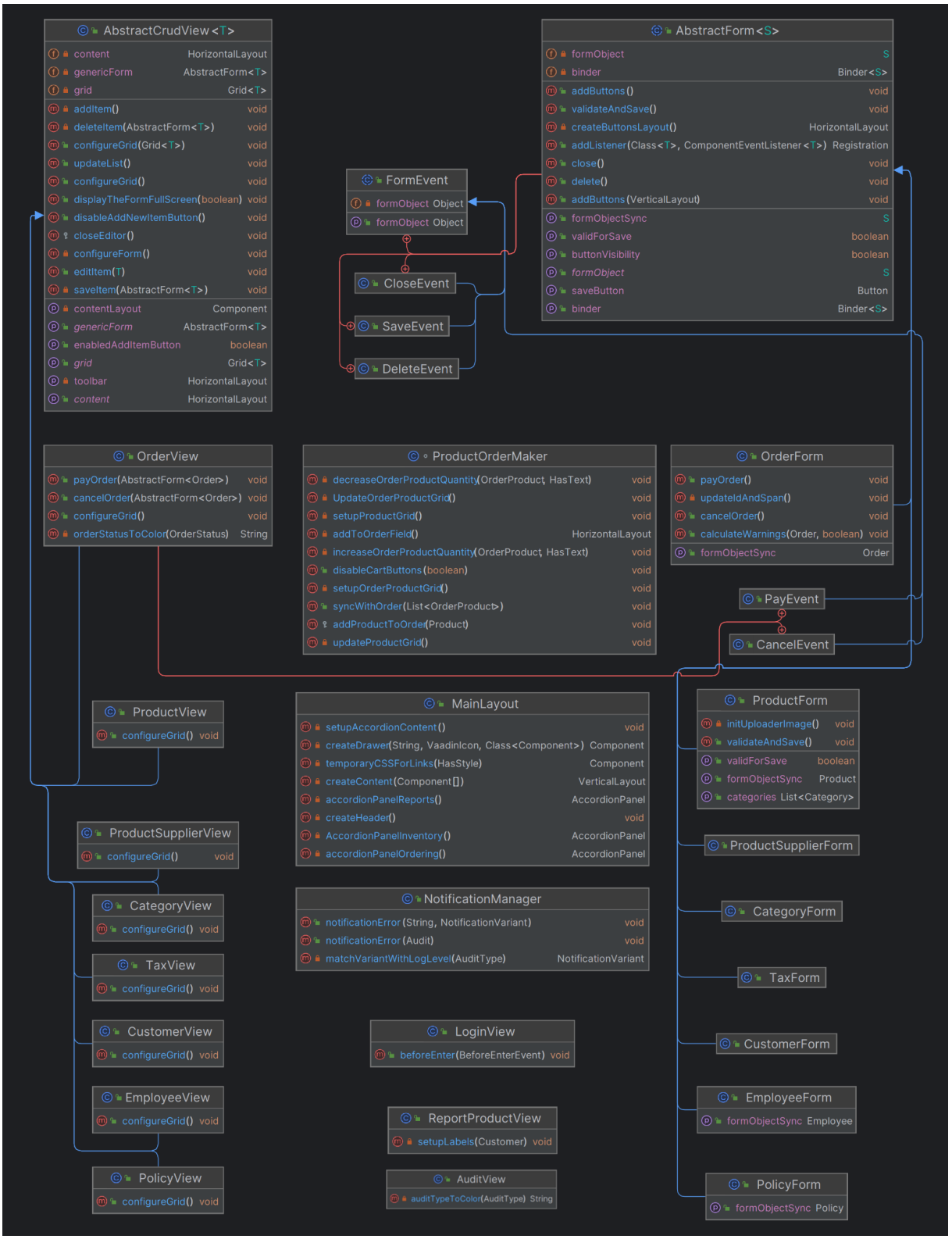


Σχήμα 54: Notification Manager

- MainLayout: είναι η πιο βασική δομη της γραφικης διεπαφης, καθως, περιέχει την στοίχιση των υπολοιπων διεπαφών και το μενού προσβασης τους.

Σχήμα 55: Main Layout

Στην παρακάτω εικόνα μπορούμε να δουμε μια αναπαράσταση απο τις κλάσεις και το πως συνδέονται μεταξύ τους, οι κλάσεις των διεπαφών.



Σχήμα 56: Πακέτο View

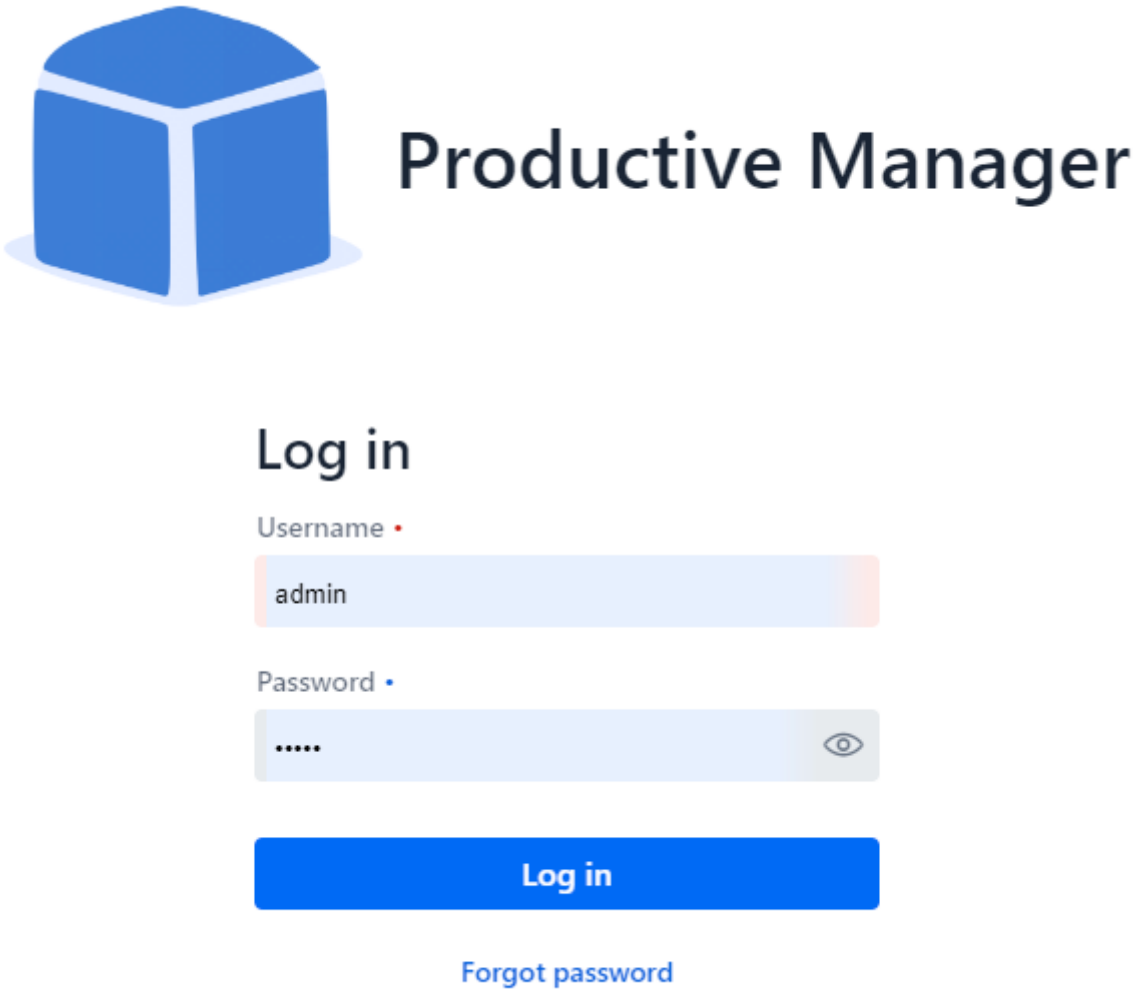
Κεφάλαιο 4ο: Οδηγός χρήσης λογισμικού

4.1 Εισαγωγή

Οπως αναφέρθηκε και σε προηγουμενο κεφαλαιο, το Productive Manager είναι ένα σύστημα διαχείρισης αποθήκης και πωλήσεων που παρέχει μια ολοκληρωμένη λύση στις επιχειρήσεις για τη διαχείριση του αποθέματος και των πωλήσεών τους. Στο επόμενο κεφάλαιο, θα εξηγήσουμε τον τρόπο λειτουργίας της εφαρμογής και πώς μπορεί κανείς να τη χρησιμοποιήσει με παράδειγμα μιας κάβας ποτων.

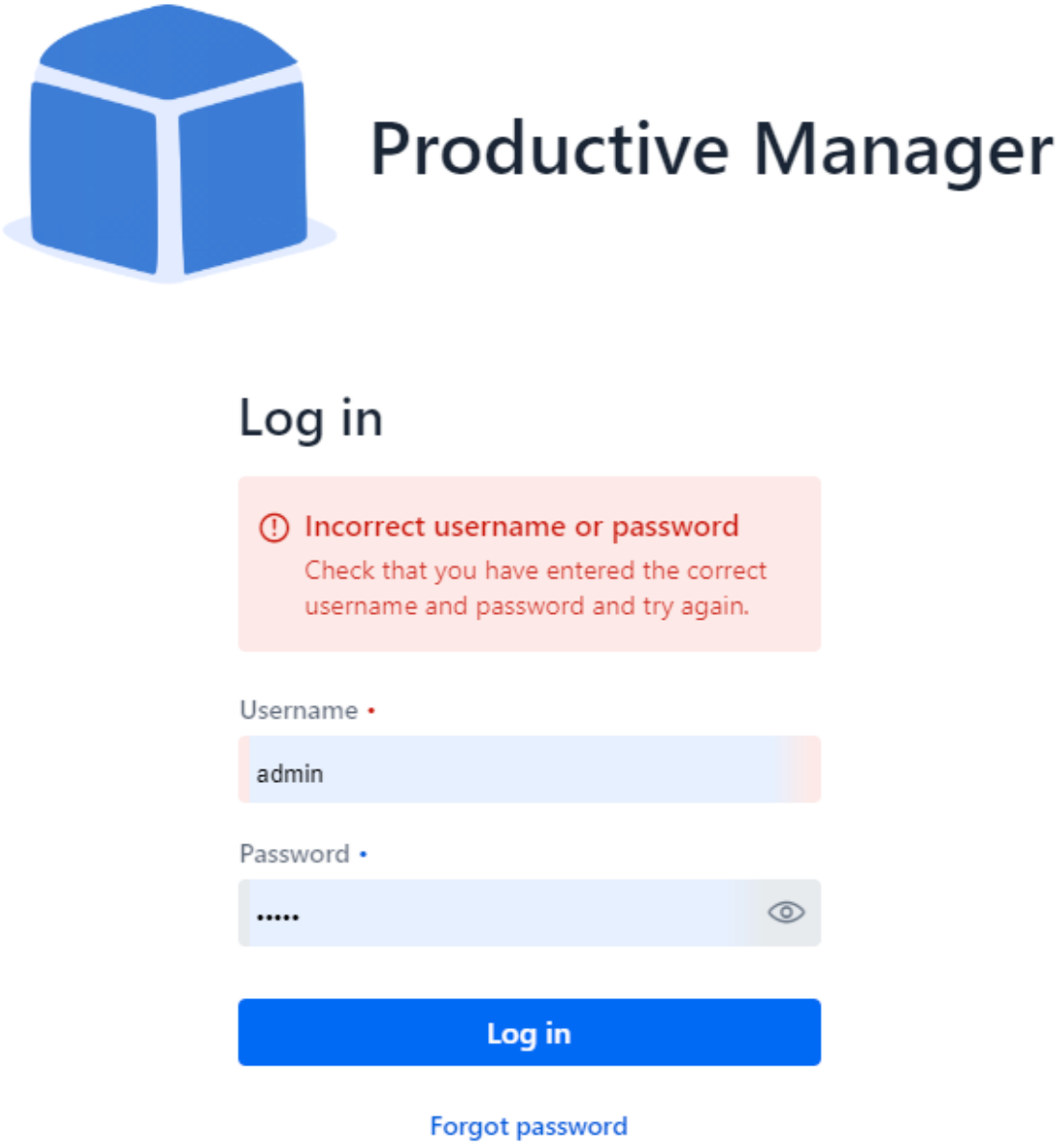
4.1.1 Είσοδος χρήστη

Για να μπορεσει καποιος να χρησιμοποιησει την εφαρμογη θα πρεπει πρωτα να διαθετει εναν λογαριασμο. Κατα την πρωτη εκκινηση του προγραμματος δημιουργειται ο λογαριασμος του διαχειριστη με αρχικοποιημένη τιμες των στοιχειων συνδεσης με username "admin" και password "admin".



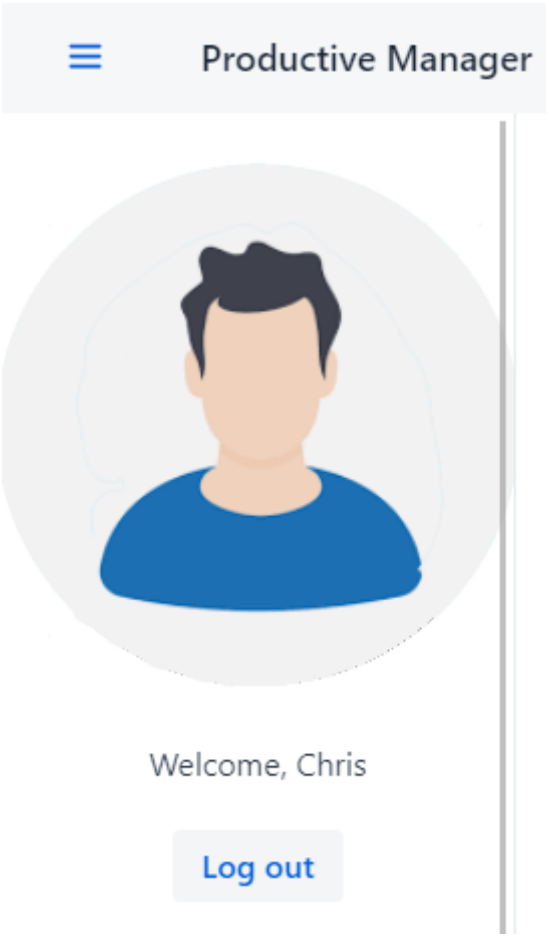
Σχήμα 57: Είσοδος Χρήστη

Στη περίπτωση που ο χρήστης δώσει λάθος στοιχεία τοτε θα εμφανιστεί το παρακάτω μήνυμα λάθους.



Σχήμα 58: Είσοδος Χρήστη Λάθος Στοιχεία

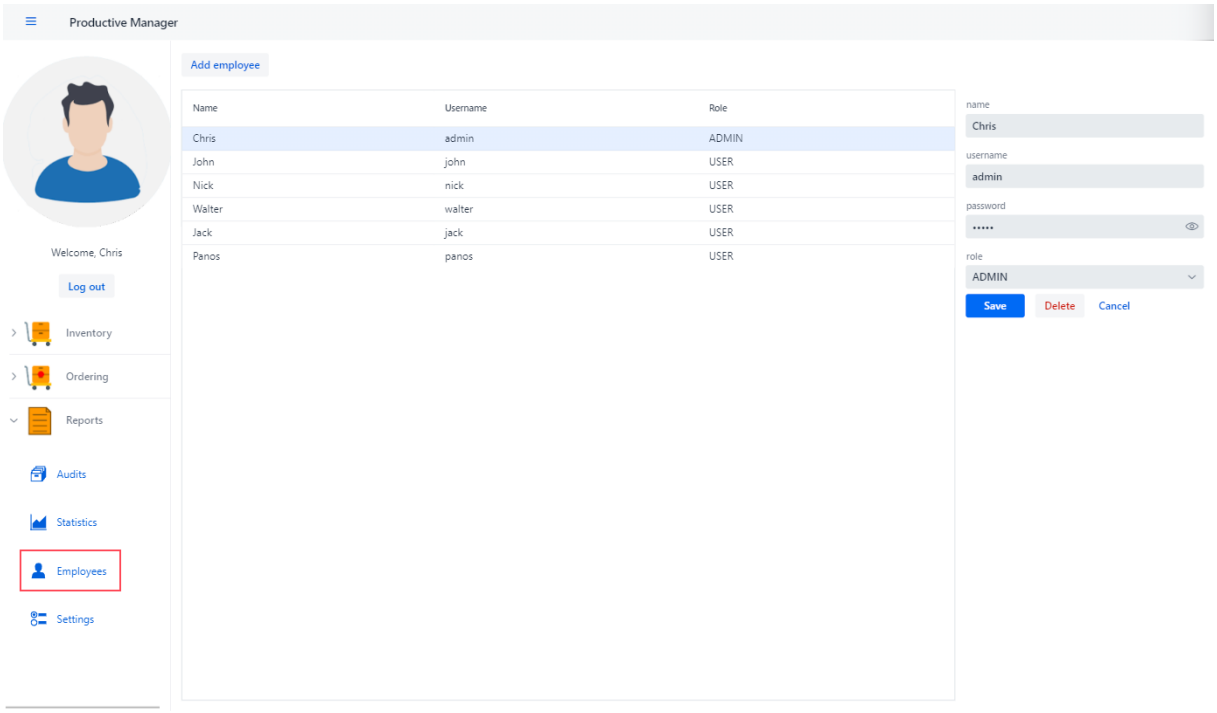
Για να αποσυνδεθεί ένας χρήστης από την εφαρμογή μπορεί να πατησει το κουμπι “Log out”.



Σχήμα 59: Έξοδος Χρήστη

Αφου πραγματοποιηθει η πρωτη συνδεση ο λογαριασμος του διαχειριστή μπορει να δημιουργήσει επιμέρους λογαριασμούς για τους υπαλληλους του καταστήματος. Πατωντας το κουμπι “Add employee”, συμπληρωνοντας τα στοιχεια στην δεξια φορμα και τελος πατωντας το κουμπι “Save”.

Δίνεται επίσης η δυνατότητα της επιλογής ενός χρήστη από την λίστα και στην συνέχεια η ενημέρωση των στοιχείων του. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που ένας χρήστης δεν χρειάζεται μπορεί να γίνει η διαγραφή του με το κουμπί “Delete”.



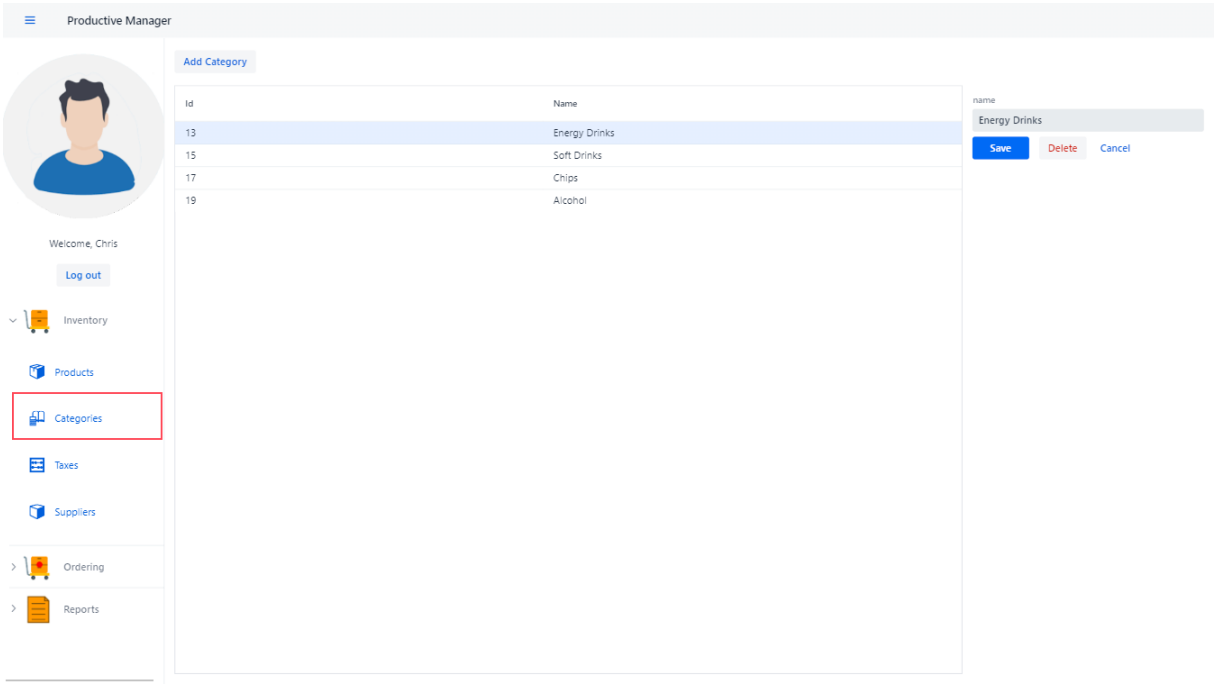
Σχήμα 60: Διαχείριση Χρήστη

4.1.2 Εισαγωγή προϊόντων

Για την εισαγωγή των προϊόντων πρέπει πρώτα να δημιουργηθούν οι κατηγορίες, οι φόροι και οι προμηθευτές.

4.1.2.1 Δημιουργία νέων κατηγοριών

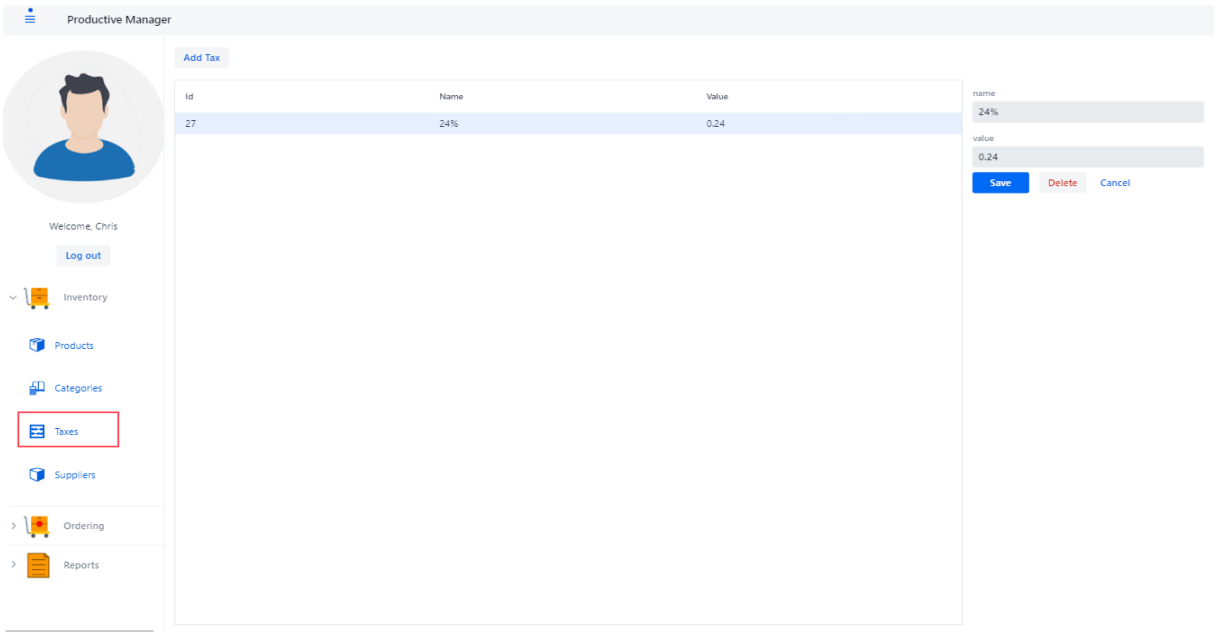
Για την δημιουργία νέων κατηγοριων πρέπει ο χρήστης να περιηγηθεί στη σελίδα με τις κατηγορίες κανοντας κλικ το κουμπί “Categories” και στην συνέχεια να πατήσει το κουμπί “Add Category”, μετά να συμπληρώσει τα στοιχεια στην δεξιά φορμα και να πατήσει “Save”. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που μια κατηγορία δεν χρειάζεται μπορεί να γίνει η διαγραφή της με το κουμπί “Delete”.



Σχήμα 61: Διαχείριση Κατηγοριών

4.1.2.2 Δημιουργία νέων φόρων

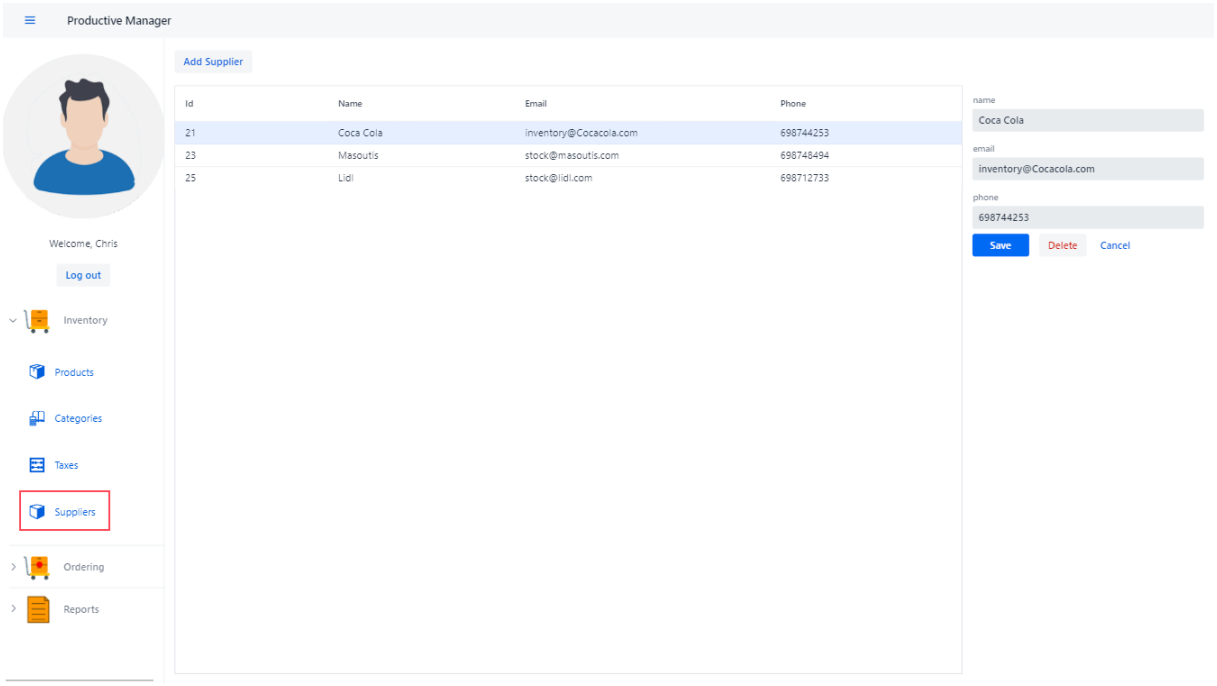
Για την δημιουργια νεων κατηγοριων πρέπει ο χρήστης να περιηγηθεί στη σελίδα με τους φόρους κανοντας κλικ το κουμπί “Taxes” και στην συνέχεια να πατήσει το κουμπί “Add Tax”, μετά να συμπληρώσει τα στοιχεια στην δεξιά φορμα και να πατήσει “Save”. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που ένας φόρος δεν χρειάζεται μπορεί να γίνει η διαγραφή του με το κουμπί “Delete”.



Σχήμα 62: Διαχείριση Φόρων

4.1.2.3 Δημιουργία νέων προμηθευτών

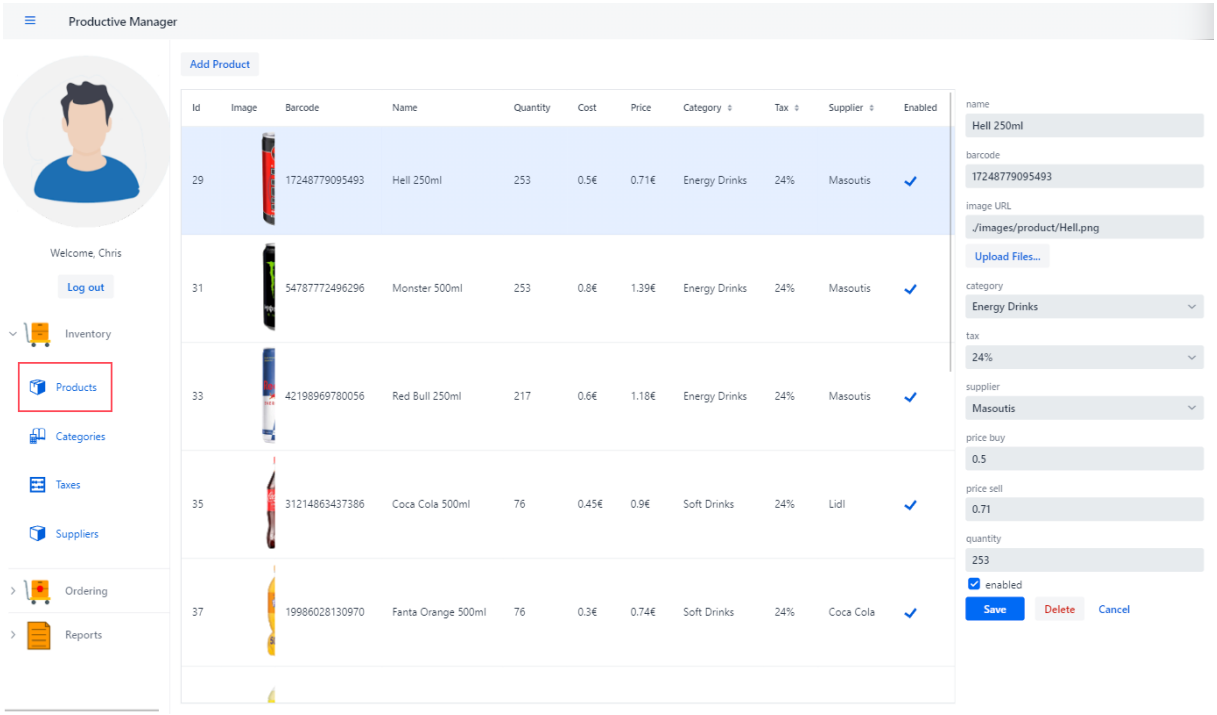
Για την δημιουργια νεων προμηθευτών πρέπει ο χρήστης πρώτα να περιηγηθεί στη σελίδα με τους προμηθευτες κανοντας κλικ το κουμπί “Suppliers” και στην συνέχεια να πατήσει το κουμπί “Add Tax”, μετά να συμπληρώσει τα στοιχεια στην δεξιά φορμα και να πατήσει “Save”. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που ένας προμηθευτής δεν χρειάζεται μπορεί να γίνει η διαγραφή του με το κουμπί “Delete”.



Σχήμα 63: Διαχείριση Προμηθευτών

4.1.3 Δημιουργία νέων προϊόντων

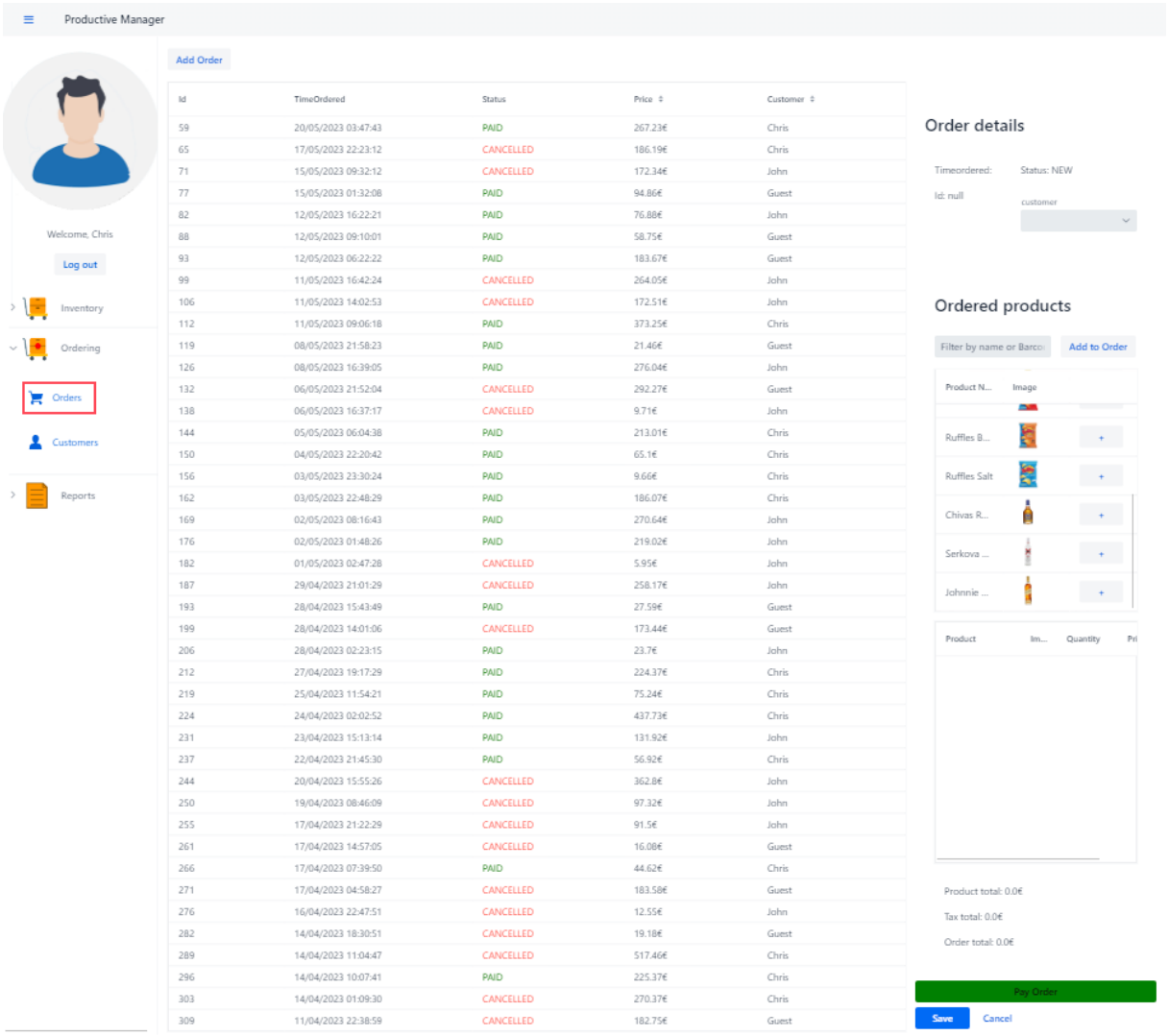
Αφού έχουν δημιουργηθεί οι παραπάνω κατηγοριες, φόροι και προμηθευτές μπορουν να δημιουργηθουν τα προιοντα. Για την δημιουργια νεων προιοντων πρέπει πρώτα ο χρήστης να περιηγηθεί στη σελίδα με τα προϊόντα κανοντας κλικ το κουμπί “Products” και στην συνέχεια να πατήσει το κουμπί “Add Product”, μετά να συπληρώσει τα στοιχεια στην δεξιά φορμα και να πατήσει “Save”. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που ένα προϊόν δεν χρειάζεται μπορεί να γίνει η διαγραφή του με το κουμπί “Delete”.



Σχήμα 64: Διαχείριση Προϊόντων

4.1.4 Παραγγελιοληψία

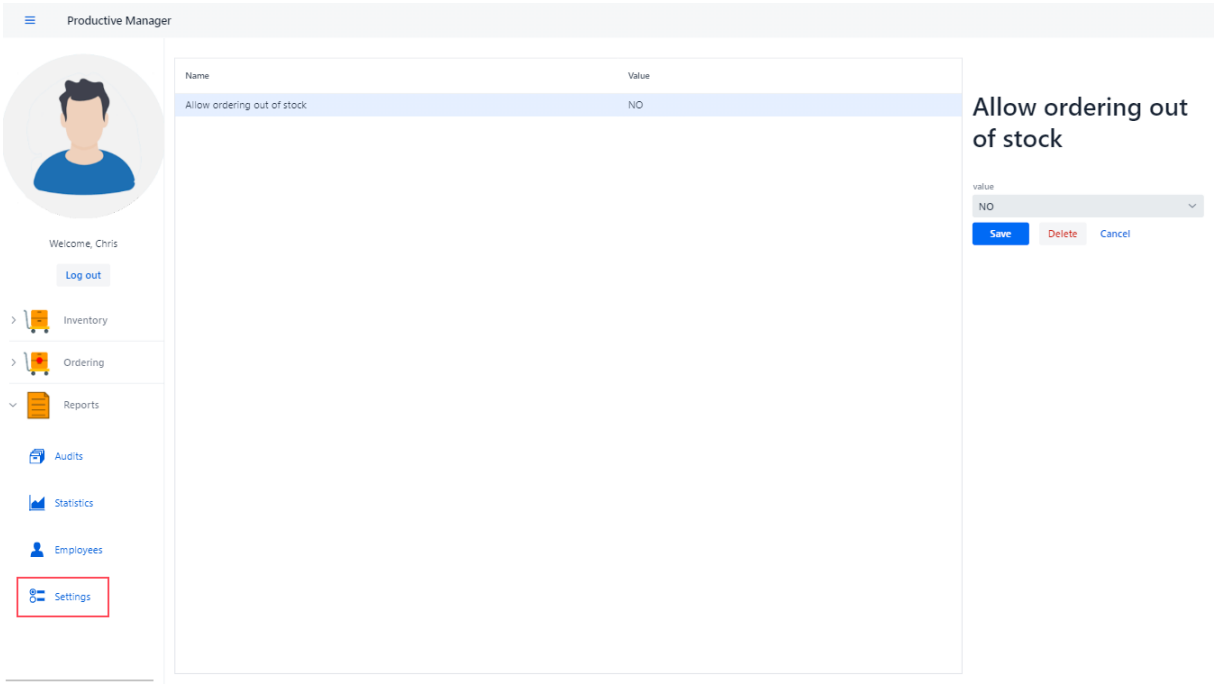
Μετά την εισαγωγή των προιοντων μπορει να ξεκινήσει η παραγγελιοληψία, για την δημιουργία νέων παραγγελιών ο χρήστης πρέπει πρώτα να παει στην σελιδα με τις παραγγελιες κανοντας κλικ το κουμπί “Orders”, στην συνέχεια να κάνει κλικ στο κουμπί “Add Order” και να συμπληρώσει την φόρμα στο αριστερό μερος. Στο πεδιο “customer” ο χρήστης μπορεί να επιλέξει τον πελάτη που έκανε την αγορά, όπου μπορεί να είναι είτε ενας συγκεκριμένος πελάτης ή ενας ανωνυμος πελάτης. Τα προϊόντα μπορούν να βρεθούν είτε στη λιστα με τα προϊόντα και πατωντας το κουμπι “+” είτε με την εισαγωγή του κωδικού τους. Όταν ένα προϊόν έχει μπει στην παραγγελία θα εμφανιστεί στην κάτω λιστα οπου ο χρήστης μπορεί να ρυθμίσει την ποσοτητα του. Αν η ποσότητα φτάσει το 0 το προϊόν θα αφαιρεθει απο την παραγγελια. Για την ολοκλήρωση της παραγγελίας ο χρήστης μπορεί να πατησει το κουμπι “Save” αν θελει να αποθηκευση την παραγγελια για να πληρωθεί αργότερα ή το κουμπι “Pay Order” για να πληρωσει κατευθειαν την παραγγελια. Τέλος αν μια παραγγελια έχει πληρωθεί δεν μπορει να γίνει τροποποιηση στα προιοντα της αλλα μόνο να ακυρωθεί.



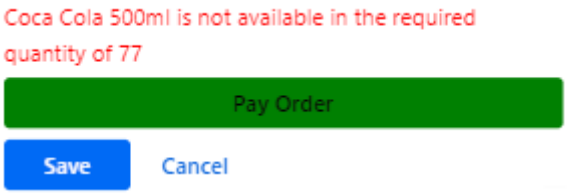
Σχήμα 65: Διαχείριση Παραγγελιών

4.1.4.1 Πολιτικές συστήματος

Η εφαρμογή προσφέρει παραμετροποίηση στη λειτουργια της με την μορφή πολιτικών συστηματος τις οποιες ο χρήστης μπορεί να βρεί και να τροποποιήσει στη σελίδα με τις ρυθμίσεις κανοντας κλικ στο κουμπί “Settings”. Στη τωρινή εκδοση του Productive Manager, υπαρχει η ρυθμιση για το αν θα επιτρέπεται η αγορα προιοντων των οποιον εχει τελειωσει το αποθεμα. Οι τιμές που μπορουν να δοθουν στη ρυθμιση είναι “NO” οπου αν ο αριθμος ενός προϊόντος δεν ειναι διαθεσιμος τοτε ο χρήστης δεν μπορεί να τα παραγγειλει και το κουμπί “Pay Order” θα απενεργοποιηθεί, “Warning” οπου εμφανιζεται ενα προειδοποιητικο μυνημα οτι τα προιοντα δεν ειναι διαθεσιμα αλλα ο χρηστης μπορει να τα παραγγειλει και “YES” οπου το συστημα δεν θα εμφανισει καποια ειδοποιηση και θα επιτρεψει κανονικα την παραγγελία των προιοντων.



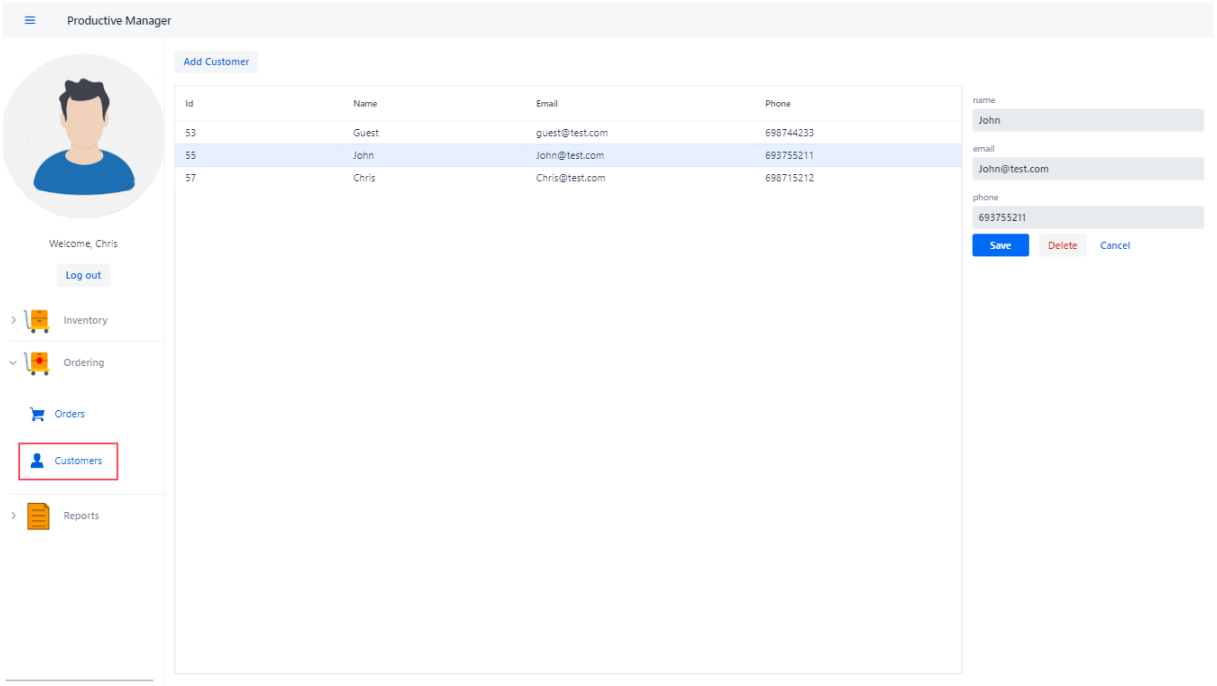
Σχήμα 66: Ρυθμίσεις



Σχήμα 67: Μήνυμα Λάθους

4.1.4.2 Πελατες

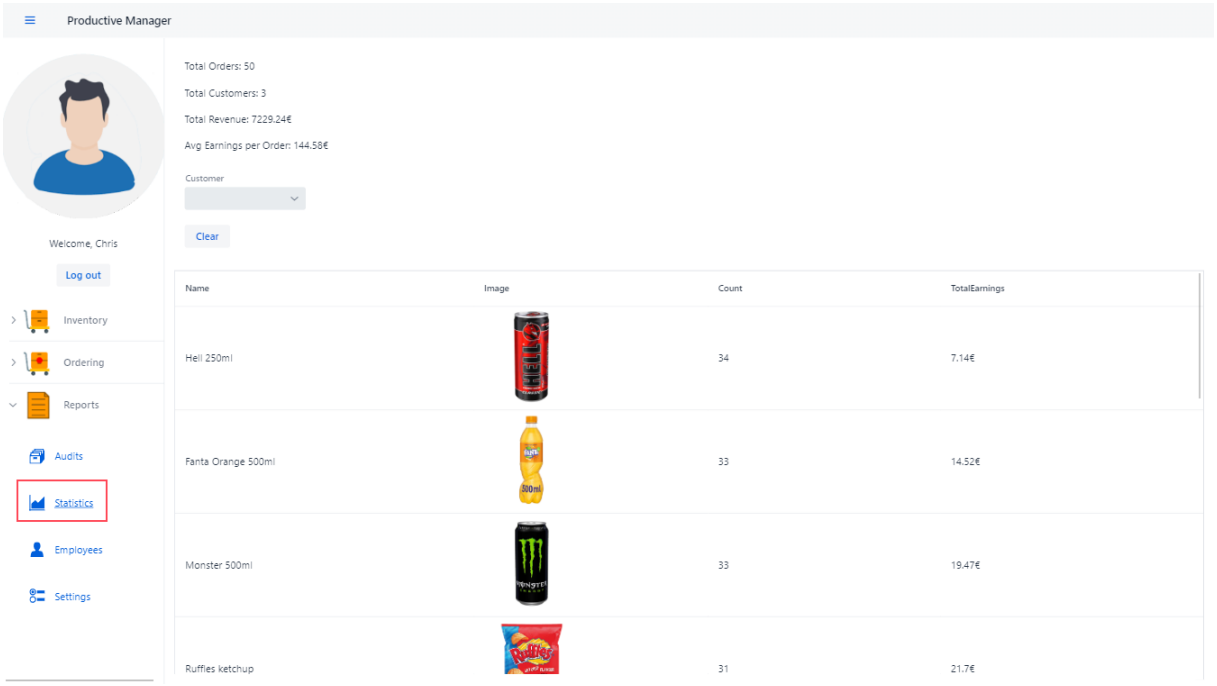
Για την διαχείριση πελατών ο χρήστης πρέπει να περιηγηθει στη σελίδα με τους πελατες κανοντας κλικ το κουμπί “Customers”. Για την δημιουργια ενος πελάτη ο χρήστης πρέπει να πατήσει το κουμπί “Add Customer”, μετά να συμπληρώσει τα στοιχεια στην δεξιά φορμα και να πατήσει το κουμπί “Save”. Αν εχει γινει καποια λάθος εισαγωγή στα στοιχεία ο χρήστης μπορεί να πατησει το κουμπι “Cancel” και να ακυρωσει τις αλλαγές. Τέλος στην περίπτωση που ένας προμηθευτής δεν χρειάζεται μπορεί να γίνει η διαγραφή του με το κουμπί “Delete”.



Σχήμα 68: Διαχείριση Πελατών

4.1.4.3 Αναφορές παραγγελιων

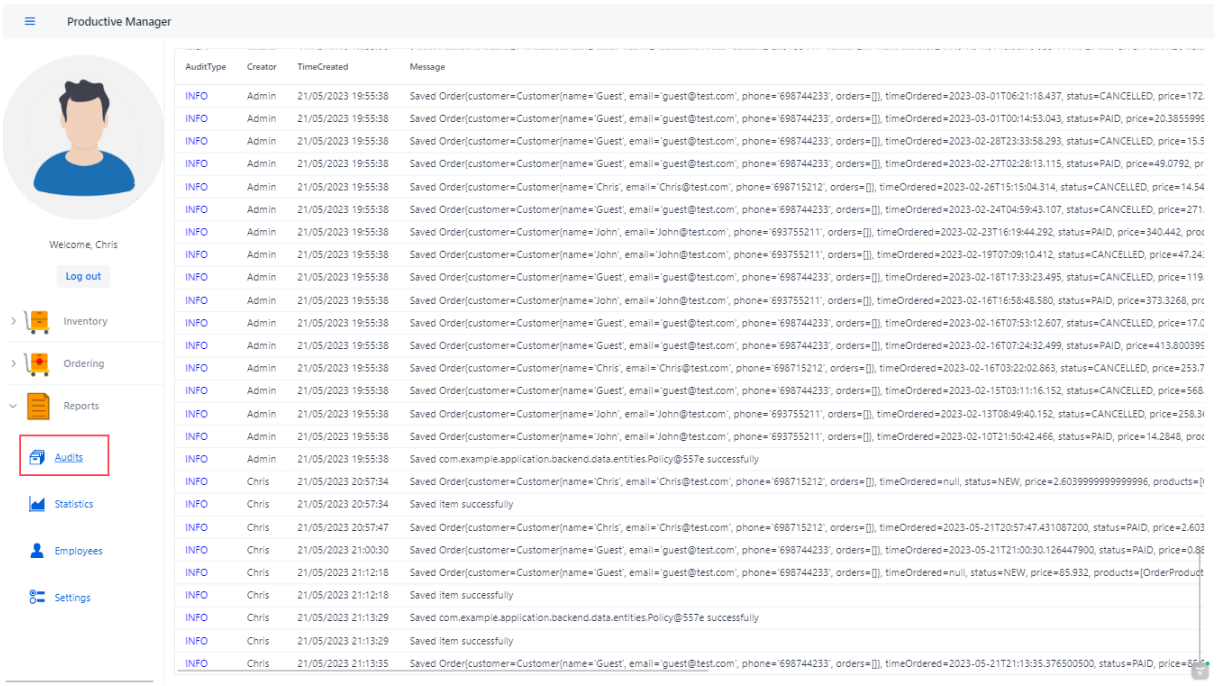
Η εκτός απο το ιστορικο των παραγγελιων υποστηρίζει και την ληψη στατιστικων στοιχειων σχετικα με τις παραγγελιες, τους πελάτες αλλα και τα εκαστοτε προιοντα. Ο χρήστης μπορεί να δει ποσες παραγγελιες εγιναν συνολικα, ποσους πελάτες εχει, ποσο ηταν το συνολικό κέρδος αλλα και το κέρδος ανα παραγγελια. Τα παραπανω επισης μπορουν να προσαρμοστουν ανα πελατη απο το πεδιο “Customer” και για την επιστροφη στις συνολικες τιμες ο χρήστης μπορεί να πατήσει το κουμπί “Clear”.



Σχήμα 69: Στατιστικά

4.1.5 Αναφορές συστήματος

Η εφαρμογή κραταει ιστορικο με ολες τις ενεργειες που εγιναν καθως και ποιος χρήστης τις έχει κάνει, επίσης κραταει ιστορικο και απο τυχον προβληματα που μπορουν να προκυψουν κατα την διαρκεια εκτελεσης της. Για να δει το ιστορικό ο χρήστης πρεπει να περιηγηθεί στην σελίδα με τις αναφορες πατώντας το κουμπί “Audits”.



Σχήμα 70: Αναφορές Συστήματος

Κεφάλαιο 5ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

- Πληρωμές με πιστωτικές κάρτες :

Η εφαρμογή πληρωμών με πιστωτικές κάρτες στο Productive Manager θα επιτρέψει στους πελάτες να πραγματοποιούν εύκολα αγορές χρησιμοποιώντας τις πιστωτικές ή χρεωστικές τους κάρτες. Αυτή η δυνατότητα παρέχει πολλά οφέλη, όπως αυξημένες δυνατότητες πωλήσεων, βελτιωμένη ικανοποίηση πελατών και ταχύτερη διεκπεραίωση συναλλαγών. Με την αποδοχή πληρωμών CC, δίνετε η δυνατότητα στους πελάτες να ολοκληρώνουν τις αγορές τους γρήγορα και με ασφάλεια, το οποίο μπορεί να οδηγήσει σε υψηλότερα ποσοστά μετατροπών και αυξημένα έσοδα για την επιχείρησή.

- Αποδείξεις ηλεκτρονικού ταχυδρομείου - Εκτυπωτές:

Η αποστολή αποδείξεων μέσω email και η υποστήριξη αποδείξεων μέσω εκτυπωτων προσφέρει μια ευέλικτη προσέγγιση για την παροχή αρχείων συναλλαγών στους πελάτες. Οι αποδείξεις ηλεκτρονικού ταχυδρομείου παρέχουν μια βολική και φιλική προς το περιβάλλον εναλλακτική λύση αντί των χάρτινων αποδείξεων, επιτρέποντας στους πελάτες να έχουν εύκολη πρόσβαση και να οργανώνουν τις πληροφορίες αγοράς τους. Επιπλέον, η υποστήριξη εκτυπωτών διασφαλίζει τη συμβατότητα με πελάτες που προτιμούν τα φυσικά αντίγραφα των αποδείξεων τους. Αυτός ο συνδυασμός χαρακτηριστικών ενισχύει την άνεση και την ικανοποίηση των πελατών, ενώ μειώνει τα απορρίμματα χαρτιού, καθιστώντας τον μια φιλική προς το περιβάλλον επιλογή.

- Αναδιαμόρφωση της διάταξης:

Η αναδιαμόρφωση της διάταξης της εφαρμογής για να ενσωματώσει μια πιο αποκριτική και αντιδραστική σχεδίαση βελτιώνοντας τη συνολική εμπειρία του χρήστη. Έτσι, διασφαλίζετε ότι προσαρμόζεται απρόσκοπτα σε διαφορετικά μεγέθη οθόνης και συσκευές. Αυτό σημαίνει ότι είτε οι χρήστες χρησιμοποιούν smartphone, tablet ή επιτραπέζιους υπολογιστές, θα έχουν μια συνεπή και φιλική εμπειρία. Ο αντιδραστικός σχεδιασμός ενισχύει επίσης τη διαδραστικότητα και την ανταπόκριση, επιτρέποντας την ομαλή και διαισθητική πλοήγηση, γρήγορους χρόνους φόρτωσης και δυναμικές ενημερώσεις περιεχομένου. Αυτή η δυνατότητα παρέχει πλεονεκτήματα όπως αυξημένη αφοσίωση χρήστη, μειωμένα ποσοστά εγκατάλειψης και βελτιωμένη ικανοποίηση πελατών παρέχοντας ένα σύγχρονο και απρόσκοπτο περιβάλλον εργασίας χρήστη.

- Απευθείας παραγγελία πελάτη:

Η ενσωμάτωση της δυνατότητας άμεσης παραγγελίας επιτρέπει στους πελάτες να κάνουν παραγγελίες απευθείας μέσω της εφαρμογής, παρακάμπτοντας την ανάγκη για ενδιάμεσα μέσα. Διευκολύνοντας την άμεση επικοινωνία και τις συναλλαγές μεταξύ των ίδιων των πελατών και της επιχείρησής, γίνεται ευκολότερη η διαδικασία παραγγελιών, μειώνονται πιθανά σφάλματα ή λανθασμένες επικοινωνίες και βελτιώνονται η αποτελεσματικότητα. Αυτή η δυνατότητα ενδυναμώνει τους πελάτες παρέχοντάς τους μια επιλογή αυτοεξυπηρέτησης, που οδηγεί σε ταχύτερη τοποθέτηση παραγγελιών και δυνητικά αύξηση της αφοσίωσης των πελατών μέσω μιας απλοποιημένης και άμεσης εμπειρίας παραγγελίας.

- Βελτιωμένη παρακολούθηση αποθέματος:

Η επέκταση των δυνατοτήτων παρακολούθησης του αποθέματος προσθέτοντας περισσότερες μονάδες παρακολούθησης, όπως βάρος, λίτρα ή άλλες σχετικές μετρήσεις, ενισχύει την ικανότητά της εφαρμογής να διαχειρίζει και να παρακολουθεί με ακρίβεια το απόθεμα. Δίνει τη δυνατότητα για μια πιο ολοκληρωμένη κατανόηση των επιπέδων του αποθέματος, επιτρέποντας πιο ενημερωμένες αποφάσεις αγορών, βελτιστοποιημένες στρατηγικές ανεφοδιασμού και την αποφυγή εξαντλήσεων. Διαθέτοντας αναλυτικές μονάδες παρακολούθησης, επιτρέπει την καλύτερη αναλύση στην κίνηση του προϊόντος, καθώς και τις προβλέψεις στη ζήτηση του προϊόντος.

Ενσωματώνοντας αυτές τις δυνατότητες στο Productive Manager, θα βελτιωθεί δραματικά τη συνολική λειτουργικότητα και την εμπειρία του χρήστη της εφαρμογής. Αυτό θα οδηγήσει σε αυξημένη ικανοποίηση των πελατών, βελτιωμένη λειτουργική αποτελεσματικότητα και ανταγωνιστικό πλεονέκτημα στην αγορά, ωφελώντας τελικά τόσο το ιδιο όσο και τους χρήστες της εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] Walter Savitch and Kenrick Mock, *Absolute Java 6th Edition*, Montgomery, Illinois, US, Pearson, May 2015.

[2] Moisés Macero García, *Learn Microservices with Spring Boot: A Practical Approach to RESTful Services Using an Event-Driven Architecture, Cloud-Native Patterns, and Containerization 2nd Edition,* Apress, Nov. 2020

[3] Madhusudhan Konda, *Just Spring Data Access: Covers JDBC, Hibernate, JPA and JDO 1st Edition,* Sebastopol, CA, O'Reilly Media, Jul. 2012

[4] Vaadin Team, *Book of Vaadin: Vaadin 14 Edition Paperback,* Vaadin Ltd, Jul. 2019

[5] Jennifer Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics 5th Edition, Kindle Edition,* O'Reilly Media, May, 2018

[6] Vincent van der Leun, *Introduction to JVM Languages: Get familiar with the world of Java, Scala, Clojure, Kotlin, and Groovy,* Packt Publishing, Jun. 2017

[7] Allen G. Taylor, *SQL For Dummies (For Dummies (Computer/Tech)) 9th Edition,* For Dummies, Dec. 2018

[8] Mark Myers, *A Smarter Way to Learn JavaScript. The new tech-assisted approach that requires half the effort First Edition Used,* Independent Publisher, Mar. 2014

[9] Jon Duckett, *HTML and CSS: Design and Build Websites First Edition,* John Wiley & Sons, Nov. 2011

[10] Mike Keith, *Pro JPA 2 Paperback,* Apress, Jan. 2013

[11] John Kouraklis, *Introducing Delphi ORM: Object Relational Mapping Using TMS Aurelius 1st Edition,* Apress, Aug. 2019

Internet Site

[12] Vend POS ”Everything you need to run the world’s best retail.”, 5/25/2023. [Online]. Available: https://www.vendhq.com/tour

[13] Square POS ”A point of sale for however you sell..”, 5/25/2023. [Online]. Available: https://squareup.com/us/en/point-of-sale

[14] LightSpeed POS ”POS & COMMERCE PLATFORM Be the best in your business”, 5/25/2023. [Online]. Available: https://www.lightspeedhq.com/home/

[15] Java Components: Java Platform, JDK, JRE, & Java Virtual Machine, 5/25/2023. [Online]. Available: https://www.softwaretestinghelp.com/java-components-java-platform-jdk/

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας Λογισμικού

Κλάση AbstractEntity

```
package com.example.application.backend.data.entities;

import java.util.Objects;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.MappedSuperclass;
import javax.persistence.Version;

import lombok.*;

@Setter @Getter
@MappedSuperclass
public abstract class AbstractEntity {

   @Id
   @GeneratedValue
   private Long id;

   @Version
   private int version;

   @Override
   public int hashCode() {
       return Objects.hash(id, version);
   }

   @Override
   public boolean equals(Object o) {
       if (this == o) {
           return true;
       }
       if (o == null || getClass() != o.getClass()) {
           return false;
       }
       AbstractEntity that = (AbstractEntity) o;
       return version == that.version &&
               Objects.equals(id, that.id);
   }
}
```

Κλάση Audit

```
package com.example.application.backend.data.entities;


import com.example.application.backend.data.AuditType;

import lombok.*;

import javax.persistence.Entity;

import javax.persistence.Lob;

import java.time.LocalDateTime;


@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor @Entity

public class Audit extends AbstractEntity {

   private String creator;
```

```
    private LocalDateTime timeCreated;
    @Lob private String message;
    private AuditType auditType;
    public Audit(LocalDateTime timeCreated, String message, AuditType auditType) {
        this.timeCreated = timeCreated;
        this.message = message;
        this.auditType = auditType;
    }
}
```

Κλάση Category

```
package com.example.application.backend.data.entities;

import lombok.*;
import javax.persistence.*;
import javax.validation.constraints.*;
import java.util.ArrayList;
import java.util.List;

@Setter @Getter @Builder @AllArgsConstructor
@Entity
public class Category extends AbstractEntity {

    public Category(){}

    @Column(nullable = false, unique=true)
    @NotEmpty
    private String name;

     @OneToMany(targetEntity = Product.class, cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
    private final List<Product> products = new ArrayList<>();

    public void addProduct(Product product){
        products.add(product);
        product.setCategory(this);
    }

    @Override
```

```
    public String toString() {
        return "Category{" +
                "name='" + name + '\'' +
                '}';
    }
}
```

Κλάση Customer

```
package com.example.application.backend.data.entities;

import lombok.*;

import javax.persistence.CascadeType;
import javax.persistence.FetchType;
import javax.persistence.OneToMany;
import javax.persistence.Entity;
import java.util.ArrayList;
import java.util.List;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class Customer extends AbstractEntity {

    private String name;

    private String email;

    private String phone;

    @OneToMany(targetEntity = Order.class, cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
    private List<Order> orders = new ArrayList<Order>();

    @Override
    public String toString() {
        return "Customer{" +
                "name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
```

```
                ", orders=" + orders +
                '}';
    }
}
```

Κλάση Employee

```
package com.example.application.backend.data.entities;

import com.example.application.backend.data.EmployeeRole;
import lombok.*;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.Entity;
import java.util.Collection;
import java.util.List;
import java.util.Objects;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class Employee extends AbstractEntity implements UserDetails {

    private String name;
    private String username;
    private String password;

    private EmployeeRole role;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return List.of(new SimpleGrantedAuthority(role.getDescription()));
    }

    @Override
    public String getPassword() {
        return password;
    }
```

```
    @Override
    public String getUsername() {
        return username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Employee employee = (Employee) o;
                                return     username.equals(employee.username)     &&
password.equals(employee.password) && role == employee.role;
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), username, password, role);
```

```
    }
}
```

Κλάση Order

```
package com.example.application.backend.data.entities;

import com.example.application.backend.data.OrderStatus;
import lombok.*;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity @Table(name = "order_info")
public class Order extends AbstractEntity {

    @ManyToOne(cascade = CascadeType.MERGE, optional = false)
    @JoinColumn
    private Customer customer;

    private LocalDateTime timeOrdered;

    private OrderStatus status = OrderStatus.NEW;

    private double price;

     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, orphanRemoval =
true)
    @JoinColumn
    private List<OrderProduct> products = new ArrayList<>();

    @PrePersist
    @PreUpdate
    public void calcPrice() {
        products.forEach(OrderProduct::updatePrice);
        price = products.stream().mapToDouble(OrderProduct::getPrice).sum();
    }
```

```
    public void close(){
        if (timeOrdered == null){
            timeOrdered = LocalDateTime.now();
        }
    }

    @Override
    public String toString() {
        return "Order{" +
                "customer=" + customer +
                ", timeOrdered=" + timeOrdered +
                ", status=" + status +
                ", price=" + price +
                ", products=" + products +
                '}';
    }
}
```

Κλάση OrderProduct

```
package com.example.application.backend.data.entities;
import lombok.*;
import javax.persistence.*;
import java.util.Objects;
@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class OrderProduct extends AbstractEntity {

    @ManyToOne(cascade = CascadeType.DETACH)
    private Product product;

    private int quantity = 1;

    private double price;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        OrderProduct that = (OrderProduct) o;
          return quantity == that.quantity && Double.compare(that.price, price) == 0
&& product.equals(that.product);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), product, quantity, price);
    }

    public OrderProduct(Product product, int quantity) {
        this.product = product;
        this.quantity = quantity;
        this.price = quantity * product.getPrice();
    }

    public OrderProduct(OrderProduct orderProduct) {
                              this(orderProduct.getProduct(),    orderProduct.quantity,
orderProduct.getPrice());
    }

    public OrderProduct(Product product) {
        this(product, 1);
    }

    @PreUpdate
    @PrePersist
    public void updatePrice() {
        price = product.getPrice() * quantity;
        price *= product.getTax().getValue() + 1;
    }

    public double getPriceWithoutTax() {
        return product.getPrice() * quantity;
    }
```

```
    public double getPriceWithTax() {
        return product.getPrice() * quantity * (1 + product.getTax().getValue());
    }

    public double getOnlyTaxPrice() {
        return getPriceWithoutTax() * product.getTax().getValue();
    }

    public void incrementQuantity() {
        quantity += 1;
        price = quantity * product.getPrice();
    }

    public boolean decrementQuantityOrRemove() {
        if (quantity -1 >= 0) {
            quantity -= 1;
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "OrderProduct{" +
                "product=" + product.getName() +
                ", quantity=" + quantity +
                ", price=" + price +
                '}';
    }
}
```

Κλάση Policy

```
package com.example.application.backend.data.entities;
import lombok.*;
import javax.persistence.*;
import java.util.List;
import java.util.stream.Collectors;
@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
```

```
@Entity
public class Policy extends AbstractEntity {
   private String name;
   private String defaultValue;
   private String selectedValue;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, orphanRemoval =
true)
   @JoinColumn
   private List<StringJPA> values;

   public String getUserSelectedValue() {
       if (selectedValue == null) {
           return defaultValue;
       }

       return selectedValue;
   }

   public List<String> getValues(){
                                                                          return
values.stream().map(StringJPA::getValue).collect(Collectors.toList());
   }
}
```

Κλάση Product

```
package com.example.application.backend.data.entities;

import lombok.*;
import javax.persistence.*;
import javax.validation.constraints.*;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class Product extends AbstractEntity {

//    @NotBlank(message = "Product name is required")
   @Size(max = 255)
   @Column(unique = true)
```

```
    private String name;

//    @Min(value = 0, message = "Minimum price is 0")
    private double price;

//    @Min(value = 0, message = "Minimum price is 0")
    private double cost;

//    @Min(value = 0, message = "Minimum quantity is 0")
    private int quantity;

    private boolean enabled;

    @ManyToOne(cascade = CascadeType.MERGE, optional = false)
    @JoinColumn
    private Category category;

    @ManyToOne(cascade = CascadeType.MERGE, optional = false)
    @JoinColumn
    private ProductSupplier supplier;

    @ManyToOne(cascade = CascadeType.MERGE, optional = false)
    private Tax tax;

    @Lob
    @Basic(fetch = FetchType.LAZY)
    private String imageUrl;

    private boolean markedForDelete = false;

    private String barcode;

    public Product copy() {
        Product product = Product.builder()
                .name(name)
                .price(price)
                .cost(cost)
                .quantity(quantity)
```

```
                .category(category)
                .tax(tax)
                .imageUrl(imageUrl)
                .barcode(barcode)
                .markedForDelete(markedForDelete)
                .build();
        product.setId(getId());
        product.setVersion(getVersion());
        return product;
    }

    @Override
    public String toString() {
        return "Product{" +
                "name='" + name + '\'' +
                ", price=" + price +
                ", cost=" + cost +
                ", quantity=" + quantity +
                ", enabled=" + enabled +
                ", category=" + category +
                ", supplier=" + supplier +
                ", tax=" + tax +
                ", imageUrl='" + imageUrl + '\'' +
                ", markedForDelete=" + markedForDelete +
                ", barcode='" + barcode + '\'' +
                '}';
    }
}
```

Κλάση ProductSupplier

```
package com.example.application.backend.data.entities;

import lombok.*;

import javax.persistence.Entity;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor @Entity
public class ProductSupplier extends AbstractEntity {
```

```
    private String name;
    private String email;
    private String phone;

    @Override
    public String toString() {
        return "ProductSupplier{" +
                "name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
                '}';
    }
}
```

Κλάση StringJPA

```
package com.example.application.backend.data.entities;

import lombok.*;

import javax.persistence.Entity;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class StringJPA extends AbstractEntity {
    private String value;
}
```

Κλάση Tax

```
package com.example.application.backend.data.entities;

import lombok.*;

import javax.persistence.*;

@Setter @Getter @Builder @AllArgsConstructor @NoArgsConstructor
@Entity
public class Tax extends AbstractEntity {

```

```
    @Column(nullable = false, unique = true)
    private String name;

    private double value;

    @Override
    public String toString() {
        return "Tax{" +
                "name='" + name + '\'' +
                ", value=" + value +
                '}';
    }
}
```

Κλάση AuditType

```
package com.example.application.backend.data;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public enum AuditType {
    TRACE, DEBUG, INFO, WARN, ERROR
}
```

Κλάση EmployeeRole

```
package com.example.application.backend.data;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public enum EmployeeRole {
    ADMIN("Admin"), USER("User");
    private final String description;
}
```

Κλάση InventoryPolicies

```
package com.example.application.backend.data;

import com.example.application.backend.data.entities.Policy;
import com.example.application.backend.data.entities.StringJPA;
import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.List;

@Getter
@AllArgsConstructor
public enum InventoryPolicies {
   STOCK_POLICY(new Policy("Allow ordering out of stock",
           "NO", null,
           List.of(new StringJPA("NO"),
                   new StringJPA("WARNING"),
                   new StringJPA("YES"))));

   private final Policy policy;

}
```

Κλάση OrderStatus

```
package com.example.application.backend.data;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public enum OrderStatus {
   CANCELED, NEW, PAID;
}
```

Κλάση DuplicateFieldException

```
package com.example.application.backend.exceptions;
```

```
import org.springframework.dao.DataIntegrityViolationException;

public class DuplicateFieldException extends DataIntegrityViolationException {

   public DuplicateFieldException(String s) {
       super(s);
   }
}
```

Κλάση ReferentialIntegrityException

```
package com.example.application.backend.exceptions;

import org.springframework.dao.DataIntegrityViolationException;

public class ReferentialIntegrityException extends DataIntegrityViolationException
{

   public ReferentialIntegrityException(String s) {
       super(s);
   }
}
```

Κλάση AuditRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Audit;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface AuditRepository extends JpaRepository<Audit, Long> {
   @Override
   List<Audit> findAll();

   @Override
   <S extends Audit> S save(S entity);
```

```
    @Override
    <S extends Audit> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Audit entity);

}
```

Κλάση CategoryRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Category;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface CategoryRepository extends JpaRepository<Category, Long> {
    @Override
    List<Category> findAll();

    @Override
    <S extends Category> S save(S entity);

    @Override
    <S extends Category> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Category entity);


    Optional<Category> findCategoryByName(String category);
}
```

Κλαση CustomerRepository

```java
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CustomerRepository extends JpaRepository<Customer, Long> {
    @Override
    List<Customer> findAll();

    @Override
    <S extends Customer> S save(S entity);

    @Override
    <S extends Customer> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Customer entity);

}
```

Κλάση EmployeeRepository

```java
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    @Override
    List<Employee> findAll();
```

```
    @Override
    <S extends Employee> S save(S entity);

    @Override
    <S extends Employee> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Employee entity);

    Optional<Employee> findEmployeeByUsername(String username);
}
```

Κλάση OrderRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface OrderRepository extends JpaRepository<Order, Long> {
    @Override
    List<Order> findAll();

    @Override
    <S extends Order> S save(S entity);

    @Override
    <S extends Order> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Order entity);

}
```

Κλάση PolicyRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Policy;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface PolicyRepository extends JpaRepository<Policy, Long> {
    @Override
    List<Policy> findAll();

    @Override
    <S extends Policy> S save(S entity);

    @Override
    <S extends Policy> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(Policy entity);

}
```

Κλάση ProductRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
    @Override
    List<Product> findAll();
```

```
    @Override
    <S extends Product> S save(S entity);

    @Override
    void delete(Product entity);
}
```

Κλάση ProductSupplierRepository

```
package com.example.application.backend.repository;

import com.example.application.backend.data.entities.ProductSupplier;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface ProductSupplierRepository extends JpaRepository<ProductSupplier,
Long> {
    @Override
    List<ProductSupplier> findAll();

    @Override
    <S extends ProductSupplier> S save(S entity);

    @Override
    <S extends ProductSupplier> List<S> saveAll(Iterable<S> entities);

    @Override
    void delete(ProductSupplier entity);


    Optional<ProductSupplier> findCategoryByName(String supplier);
}
```

Κλάση TaxRepository

```
package com.example.application.backend.repository;
```

```
import com.example.application.backend.data.entities.Tax;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface TaxRepository extends JpaRepository<Tax, Long> {
   @Override
   List<Tax> findAll();

   @Override
   <S extends Tax> S save(S entity);

   @Override
   <S extends Tax> List<S> saveAll(Iterable<S> entities);

   @Override
   void delete(Tax entity);


   Optional<Tax> findTaxByName(String category);
}
```

Κλάση ProductPerformanceResource

```
package com.example.application.backend.resources;

import com.example.application.backend.data.entities.Product;
import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class ProductPerformanceResource {
   private Product product;
   private int count;
```

```
    private double totalEarnings; // cost - price


}
```

Κλάση AbstractService

```
package com.example.application.backend.service;


import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.exceptions.DuplicateFieldException;
import com.example.application.backend.exceptions.ReferentialIntegrityException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.transaction.annotation.Transactional;

import java.time.LocalDateTime;
import java.util.List;

public abstract class AbstractService<T>{
    private final JpaRepository<T, ?> jpaRepository;
    private @Autowired AuditService auditService;

    public AbstractService(JpaRepository<T, ?> jpaRepository) {
        this.jpaRepository = jpaRepository;
    }

    @Transactional
    public T save(T object) throws DuplicateFieldException {
        T objectToBesaved = null;
        try {
            objectToBesaved = jpaRepository.save(object);
                        auditService.save(new  Audit(LocalDateTime.now(),  "Saved  "  +
object.toString() +" successfully", AuditType.INFO));
        } catch (DataIntegrityViolationException exception) {
            throw new DuplicateFieldException("Item already exist");
        }
        return objectToBesaved;
    }
```

```
    public List<T> findAll() {
        return jpaRepository.findAll();
    }

    public void delete(T object) {
        try {
            jpaRepository.delete(object);
        } catch (DataIntegrityViolationException exception) {
                throw new ReferentialIntegrityException("Please delete all references
before deleting this");
        }
    }

    public void saveAll(List<T> objects) {
        objects.forEach(this::save);
    }

    @Transactional
    public T saveNoAudit(T object) throws DuplicateFieldException {
        T objectToBesaved = null;
        try {
            objectToBesaved = jpaRepository.save(object);
        } catch (DataIntegrityViolationException exception) {}
        return objectToBesaved;
    }
}
```

Κλάση AuditService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.exceptions.DuplicateFieldException;
import com.example.application.backend.repository.AuditRepository;
import com.example.application.backend.util.StringUtil;
import com.example.application.security.SecurityService;
import com.example.application.ui.views.NotificationManager;
import org.slf4j.Logger;
```

```java
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class AuditService {
    private @Autowired NotificationManager notificationManager;
    private @Autowired SecurityService securityService;
    private AuditRepository auditRepository;
    public AuditService(AuditRepository repository) {
        this.auditRepository = repository;
    }

    public List<Audit> findAll() {
        return auditRepository.findAll();
    }

    public Audit save(Audit audit, Class<?> clazz) throws DuplicateFieldException {
        if (StringUtil.isEmptyOrNull(audit.getCreator())) {
            Employee employee = securityService.getAuthenticatedUser();
            audit.setCreator(employee != null ? employee.getName() : "Admin");
        }
        saveOnLogs(audit, clazz);
        notificationManager.notificationError(audit);
        return save(audit);
    }

    public Audit save(Audit audit) {
        if (StringUtil.isEmptyOrNull(audit.getCreator())) {
            Employee employee = securityService.getAuthenticatedUser();
            audit.setCreator(employee != null ? employee.getName() : "Admin");
        }
        return auditRepository.save(audit);
    }

    public void saveAll(List<Audit> objects) {
```

```
        objects.forEach(this::save);
    }

    private void saveOnLogs(Audit audit, Class<?> clazz) {
        Logger logger = LoggerFactory.getLogger(clazz);
        switch (audit.getAuditType()) {
            case INFO:
                logger.info(audit.getMessage());
                break;
            case WARN:
                logger.warn(audit.getMessage());
                break;
            case DEBUG:
                logger.debug(audit.getMessage());
                break;
            case ERROR:
                logger.error(audit.getMessage());
                break;
            case TRACE:
                logger.trace(audit.getMessage());
                break;
        }
    }
}
```

Κλάση AuditService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.exceptions.DuplicateFieldException;
import com.example.application.backend.repository.AuditRepository;
import com.example.application.backend.util.StringUtil;
import com.example.application.security.SecurityService;
import com.example.application.ui.views.NotificationManager;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class AuditService {
   private @Autowired NotificationManager notificationManager;
   private @Autowired SecurityService securityService;
   private Audit Repository audit Repository;
   public AuditService(AuditRepository repository) {
       this.auditRepository = repository;
   }

   public List<Audit> findAll() {
       return auditRepository.findAll();
   }

   public Audit save(Audit audit, Class<?> clazz) throws DuplicateFieldException {
       if (StringUtil.isEmptyOrNull(audit.getCreator())) {
           Employee employee = securityService.getAuthenticatedUser();
           audit.setCreator(employee != null ? employee.getName() : "Admin");
       }
       saveOnLogs(audit, clazz);
       notificationManager.notificationError(audit);
       return save(audit);
   }

   public Audit save(Audit audit) {
       if (StringUtil.isEmptyOrNull(audit.getCreator())) {
           Employee employee = securityService.getAuthenticatedUser();
           audit.setCreator(employee != null ? employee.getName() : "Admin");
       }
       return auditRepository.save(audit);
   }

   public void saveAll(List<Audit> objects) {
       objects.forEach(this::save);
   }
```

```
    private void saveOnLogs(Audit audit, Class<?> clazz) {
        Logger logger = LoggerFactory.getLogger(clazz);
        switch (audit.getAuditType()) {
            case INFO:
                logger.info(audit.getMessage());
                break;
            case WARN:
                logger.warn(audit.getMessage());
                break;
            case DEBUG:
                logger.debug(audit.getMessage());
                break;
            case ERROR:
                logger.error(audit.getMessage());
                break;
            case TRACE:
                logger.trace(audit.getMessage());
                break;
        }
    }
}
```

Κλαση Category Service

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Category;
import com.example.application.backend.repository.CategoryRepository;
import org.springframework.stereotype.Service;

@Service
public class CategoryService extends AbstractService<Category> {

    public CategoryService(CategoryRepository categoryRepository) {
        super(categoryRepository);
    }

}
```

Κλάση Customer Service

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Customer;
import com.example.application.backend.repository.CustomerRepository;
import org.springframework.stereotype.Service;

@Service
public class CustomerService extends AbstractService<Customer> {

    public CustomerService(CustomerRepository repository) {
        super(repository);
    }

}
```

Κλάση EmployeeService

```
package com.example.application.backend.service;

import com.example.application.backend.data.EmployeeRole;
import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.repository.EmployeeRepository;
import com.example.application.security.SecurityService;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class EmployeeService extends AbstractService<Employee> implements
UserDetailsService {
    private EmployeeRepository employeeRepository;
    private SecurityService securityService;
```

```
        public EmployeeService(EmployeeRepository repository, SecurityService
securityService) {
        super(repository);
        this.employeeRepository = repository;
        this.securityService = securityService;
    }

    @Override
    public Employee save(Employee employee) {
        return super.save(employee);
    }

    @Override
            public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        Employee employee = employeeRepository.findEmployeeByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException("Could not find
user"));
        return employee;
    }

    @Override
    public List<Employee> findAll() {
        Employee employee = securityService.getAuthenticatedUser();
        if (employee == null) {
            return List.of();
        }
        List<Employee> employees = super.findAll();

        if (EmployeeRole.USER.equals(employee.getRole())) {
            employees = employees.stream()
                    .filter(savedEmployee -> savedEmployee.equals(employee))
                    .collect(Collectors.toList());
        }

        return employees;
    }

}
```

Κλάση EmployeeService

```
package com.example.application.backend.service;

import com.example.application.backend.data.EmployeeRole;
import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.repository.EmployeeRepository;
import com.example.application.security.SecurityService;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class EmployeeService extends AbstractService<Employee> implements
UserDetailsService {
   private EmployeeRepository employeeRepository;
   private SecurityService securityService;

        public EmployeeService(EmployeeRepository repository, SecurityService
securityService) {
       super(repository);
       this.employeeRepository = repository;
       this.securityService = securityService;
   }

   @Override
   public Employee save(Employee employee) {
       return super.save(employee);
   }

   @Override
           public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
       Employee employee = employeeRepository.findEmployeeByUsername(username)
                   .orElseThrow(() -> new UsernameNotFoundException("Could not find
user"));
```

```
        return employee;
    }

    @Override
    public List<Employee> findAll() {
        Employee employee = securityService.getAuthenticatedUser();
        if (employee == null) {
            return List.of();
        }
        List<Employee> employees = super.findAll();

        if (EmployeeRole.USER.equals(employee.getRole())) {
            employees = employees.stream()
                    .filter(savedEmployee -> savedEmployee.equals(employee))
                    .collect(Collectors.toList());
        }

        return employees;
    }

}
```

Κλάση ImageService

```
package com.example.application.backend.service;

import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.vaadin.flow.component.upload.receivers.MultiFileMemoryBuffer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.time.LocalDateTime;

@Service
```

```
public class ImageService {

   private @Autowired AuditService auditService;

   public final static String IMAGE_FOLDER_PATH_SAVE = "./static/images/";

   public final static String IMAGE_FOLDER_PATH_DISPLAY = "./images/";


   public final static String IMAGE_FOLDER_PATH_DISPLAY_GEN = "./images/product/";


     public String saveImage(String name, String saveName, MultiFileMemoryBuffer
buffer) {

       String location = "";

       try {

           location = IMAGE_FOLDER_PATH_DISPLAY + saveName;

           File outputFile = new File(IMAGE_FOLDER_PATH_SAVE + saveName);

           BufferedImage inputImage = ImageIO.read(buffer.getInputStream(name));

           ImageIO.write(inputImage, "png", outputFile);

             auditService.save(new Audit(LocalDateTime.now(),"created new image " +
name, AuditType.INFO), this.getClass());

       } catch (IOException e) {

                   auditService.save(new  Audit(LocalDateTime.now(),  e.getMessage(),
AuditType.ERROR), this.getClass());

       }

       return location;

   }

}
```

Κλάση OrderService

```
package com.example.application.backend.service;


import com.example.application.backend.data.OrderStatus;

import com.example.application.backend.data.entities.Order;

import com.example.application.backend.data.entities.OrderProduct;

import com.example.application.backend.data.entities.Product;

import com.example.application.backend.repository.OrderRepository;

import com.example.application.backend.repository.ProductRepository;

import org.springframework.stereotype.Service;


@Service

public class OrderService extends AbstractService<Order> {
```

```
    private ProductRepository productRepository;

            public    OrderService(OrderRepository    repository,    ProductRepository
productRepository) {
        super(repository);
        this.productRepository = productRepository;
    }


    public Order save(Order order){
        order.calcPrice();


        if (OrderStatus.PAID.equals(order.getStatus())) {
            order.getProducts().forEach(orderProduct ->

productRepository.findById(orderProduct.getProduct().getId()).ifPresent(product   ->
{
                                            product.setQuantity(product.getQuantity()  -
orderProduct.getQuantity());
                        productRepository.save(product);
                    })
            );
        }
        return super.save(order);
    }


    public  Order pay(Order order){
        order.close();
        order.setStatus(OrderStatus.PAID);
        order.getProducts().forEach(orderProduct ->

productRepository.findById(orderProduct.getProduct().getId()).ifPresent(product   ->
{
                                            product.setQuantity(product.getQuantity()  -
orderProduct.getQuantity());
                    productRepository.save(product);
                })
        );
        return super.save(order);
    }


    public Order cancel(Order order){
        order.setStatus(OrderStatus.CANCELLED);
```

```
        order.getProducts().forEach(orderProduct ->

productRepository.findById(orderProduct.getProduct().getId()).ifPresent(product    ->
{
                                            product.setQuantity(product.getQuantity()   +
orderProduct.getQuantity());
                    productRepository.save(product);
                })
        );
        return super.save(order);
    }


    public String getQuantityWarning(Order order) {
                                if     (OrderStatus.PAID.equals(order.getStatus())     ||
OrderStatus.CANCELLED.equals(order.getStatus())) {
            return null;
        }


        for (OrderProduct orderProduct : order.getProducts()) {
                                                                Product      product      =
productRepository.findById(orderProduct.getProduct().getId()).orElse(null);
                                if   (product   !=   null   &&   product.getQuantity()   <
orderProduct.getQuantity()) {
                    return product.getName() + " is not available in the required
quantity of " + orderProduct.getQuantity();
            }
        }


        return null;
    }


}
```

Κλάση PolicyHelper

```
package com.example.application.backend.service;


import com.example.application.backend.data.InventoryPolicies;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;


@Component
```

```
public class PolicyHelper {
    @Autowired
    PolicyService policyService;

    public boolean isStockPolicyNoOrWarning() {
                                                        String        value        =
policyService.getPolicyValue(InventoryPolicies.STOCK_POLICY.getPolicy());
        return value.equals("NO") || value.equals("WARNING");
    }

    public boolean isStockPolicyNo() {
                                                        String        value        =
policyService.getPolicyValue(InventoryPolicies.STOCK_POLICY.getPolicy());
        return value.equals("NO");
    }
}
```

Κλάση PolicyService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Policy;
import com.example.application.backend.repository.PolicyRepository;
import org.springframework.stereotype.Service;


@Service
public class PolicyService extends  AbstractService<Policy> {
    private PolicyRepository policyRepository;
    public PolicyService(PolicyRepository repository) {
        super(repository);
        this.policyRepository = repository;
    }

    public String getPolicyValue(Policy policy) {
        return policyRepository.findAll()
                .stream()
                .filter(policy2 -> policy2.getName().equals(policy.getName()))
                .findFirst()
                .map(Policy::getUserSelectedValue)
```

```
                .orElse(null);
    }
}
```

Κλάση ProductService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Product;
import com.example.application.backend.repository.ProductRepository;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class ProductService extends AbstractService<Product> {

    public ProductService(ProductRepository productRepository) {
        super(productRepository);
    }

    @Override
    public void delete(Product object) {
        object.setMarkedForDelete(true);
        super.saveNoAudit(object);
    }

    @Override
    public List<Product> findAll() {
                                return    super.findAll().stream().filter(product    ->
!product.isMarkedForDelete()).collect(Collectors.toList());
    }

    public List<Product> findAllEnabled() {
                                                                                return
super.findAll().stream().filter(Product::isEnabled).collect(Collectors.toList());
    }
}
```

Κλάση ProductSupplierService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.ProductSupplier;
import com.example.application.backend.repository.ProductSupplierRepository;
import org.springframework.stereotype.Service;

@Service
public class ProductSupplierService extends AbstractService<ProductSupplier> {

    public ProductSupplierService(ProductSupplierRepository repository) {
        super(repository);
    }

}
```

Κλάση ReportService

```
package com.example.application.backend.service;

import com.example.application.backend.data.OrderStatus;
import com.example.application.backend.data.entities.Customer;
import com.example.application.backend.data.entities.Order;
import com.example.application.backend.data.entities.Product;
import com.example.application.backend.resources.ProductPerformanceResource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

@Service
public class ReportService {
    private @Autowired OrderService orderService;
    private @Autowired CustomerService customerService;
    private @Autowired ProductService productService;
```

```
        public   List<ProductPerformanceResource>   getProductsPerformance(Customer
customer) {
            List<ProductPerformanceResource>  productPerformanceResourceList  =  new
ArrayList<>();
       HashMap<Product, Integer> productCounter = new HashMap<>();
       HashMap<Product, Double> earningsPerProductCounter = new HashMap<>();
       orderService.findAll().stream().filter(order -> {
           OrderStatus orderStatus = order.getStatus();
           return orderStatus.equals(OrderStatus.PAID);
       }).filter(order -> order.getCustomer().equals(customer))
               .forEach(order -> {
                   order.getProducts().forEach(
                           orderProduct -> {
                               Product product = orderProduct.getProduct();
                               int count = 0;
                               double earnings = 0;
                               if (productCounter.containsKey(product)) {
                                   count = productCounter.get(product);
                                                                                   earnings  =
earningsPerProductCounter.get(product);
                               }
                               count += orderProduct.getQuantity();
                                                 earnings  +=  +  (product.getPrice()  -
product.getCost()) * orderProduct.getQuantity();
                               productCounter.put(product, count);
                               earningsPerProductCounter.put(product, earnings);
                           }
                   );
               }
       );
       for (Product product:productCounter.keySet()) {
                                             productPerformanceResourceList.add(new
ProductPerformanceResource(product,                    productCounter.get(product),
earningsPerProductCounter.get(product)));
       }
                                                                                   return
productPerformanceResourceList.stream().sorted(Comparator.comparingDouble(ProductPe
rformanceResource::getTotalEarnings)).collect(Collectors.toList());
   }
```

```
   public List<ProductPerformanceResource> getProductsPerformance() {
            List<ProductPerformanceResource>  productPerformanceResourceList  =  new
ArrayList<>();
       HashMap<Product, Integer> productCounter = new HashMap<>();
       HashMap<Product, Double> earningsPerProductCounter = new HashMap<>();
       orderService.findAll().stream().filter(order -> {
           OrderStatus orderStatus = order.getStatus();
           return orderStatus.equals(OrderStatus.PAID);
       }).forEach(
               order -> {
                   order.getProducts().forEach(
                           orderProduct -> {
                               Product product = orderProduct.getProduct();
                               int count = 0;
                               double earnings = 0;
                               if (productCounter.containsKey(product)) {
                                   count = productCounter.get(product);
                                                                          earnings  =
earningsPerProductCounter.get(product);
                               }
                               count += orderProduct.getQuantity();
                                                 earnings  +=  +  (product.getPrice()  -
product.getCost()) * orderProduct.getQuantity();
                               productCounter.put(product, count);
                               earningsPerProductCounter.put(product, earnings);
                           }
                   );
               }
       );
       for (Product product:productCounter.keySet()) {
                                              productPerformanceResourceList.add(new
ProductPerformanceResource(product,                    productCounter.get(product),
earningsPerProductCounter.get(product)));
       }
                                                                          return
productPerformanceResourceList.stream().sorted(Comparator.comparingDouble(ProductPe
rformanceResource::getTotalEarnings)).collect(Collectors.toList());
   }

   public int getTotalOrdersCount(Customer customer) {
```

```
        return (int) orderService.findAll().stream()
                                        .filter(order -> customer == null ||
order.getCustomer().equals(customer))
                .filter(order -> !OrderStatus.CANCELLED.equals(order.getStatus()) &&
!OrderStatus.NEW.equals(order.getStatus()))
                .count();
    }

    public double getTotalEarnings(Customer customer) {
        return orderService.findAll().stream()
                                        .filter(order -> customer == null ||
order.getCustomer().equals(customer))
                .filter(order -> !OrderStatus.CANCELLED.equals(order.getStatus()) &&
!OrderStatus.NEW.equals(order.getStatus()))
                .mapToDouble(Order::getPrice).sum();
    }

    public int getTotalCustomers() {
        return customerService.findAll().size();
    }

    public double getAvgEarningPerOrder(Customer customer) {
        return getTotalEarnings(customer) / getTotalOrdersCount(customer);
    }
}
```

Κλάση TaxService

```
package com.example.application.backend.service;

import com.example.application.backend.data.entities.Tax;
import com.example.application.backend.repository.TaxRepository;
import org.springframework.stereotype.Service;

@Service
public class TaxService extends AbstractService<Tax> {

    public TaxService(TaxRepository taxRepository) {
        super(taxRepository);
    }
```

```
}
```

Κλάση StringUtil

```
package com.example.application.backend.util;


import java.math.BigDecimal;
import java.math.RoundingMode;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;


public class StringUtil {
                private      static      DateTimeFormatter      dateTimeFormatter      =
DateTimeFormatter.ofPattern("dd/MM/uuuu HH:mm:ss");
   public static final String euroSymbol = "€";
   public static String FirstNotNullOrEmpty(String ...strings) {
       String output = null;


       for (String string:strings) {
           if (!isEmptyOrNull(string))
               output = string;
       }
       return output;
   }


   public static boolean isEmptyOrNull(String string) {
       return string == null || string.isEmpty();
   }


   public static double round(double value, int places) {
       if (places < 0) throw new IllegalArgumentException();


       BigDecimal bd = BigDecimal.valueOf(value);
       bd = bd.setScale(places, RoundingMode.HALF_UP);
       return bd.doubleValue();
   }


   public static String formatPrice(double value) {
       return round(value,2) + euroSymbol;
```

```
    }



    public static String formatDate(LocalDateTime localDate) {
        if (localDate == null)
            return "";
        return dateTimeFormatter.format(localDate);
    }

    public static int getRandomNumber(int min, int max) {
        return (int) ((Math.random() * (max - min)) + min);
    }

    public static Object getRandomOfList(List<?> objectList) {
        return objectList.get(getRandomNumber(0, objectList.size()));
    }


}
```

Κλάση WebConfig

```
package com.example.application.backend.configuration;

import com.example.application.backend.service.ImageService;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        try {
            Files.createDirectories(Paths.get(ImageService.IMAGE_FOLDER_PATH_SAVE));
```

```
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        registry.addResourceHandler(ImageService.IMAGE_FOLDER_PATH_DISPLAY + "**")
                                .addResourceLocations("file:resources/",  "file:"  +
ImageService.IMAGE_FOLDER_PATH_SAVE);
    }
}
```

Κλάση DataGenerator

```
package com.example.application.app.generator;

import com.example.application.backend.data.EmployeeRole;
import com.example.application.backend.data.InventoryPolicies;
import com.example.application.backend.data.OrderStatus;
import com.example.application.backend.data.entities.*;
import com.example.application.backend.service.*;
import com.example.application.backend.util.StringUtil;
import com.example.application.security.SecurityService;
import com.vaadin.flow.spring.annotation.SpringComponent;

import java.time.*;
import java.util.*;
import java.util.concurrent.ThreadLocalRandom;
import java.util.concurrent.TimeUnit;

import org.apache.commons.lang3.RandomStringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Bean;

import                                                                  static
com.example.application.backend.service.ImageService.IMAGE_FOLDER_PATH_DISPLAY_GEN;

@SpringComponent
public class DataGenerator {

    @Bean
```

```
          public    CommandLineRunner    loadData(CategoryService    categoryService,
ProductService productService,
                                              TaxService taxService, CustomerService
customerService,
                                                          OrderService orderService,
ProductSupplierService productSupplierService,
                                                    EmployeeService employeeService,
SecurityService securityService,
                                      PolicyService policyService) {


       return args -> {


           if (securityService.getAuthenticatedUser() != null) {
               securityService.logout();
           }


           List<Employee> employees = List.of(

Employee.builder().name("Chris").username("admin").password("admin").role(EmployeeR
ole.ADMIN).build(),

Employee.builder().name("John").username("john").password("john").role(EmployeeRole
.USER).build(),

Employee.builder().name("Nick").username("nick").password("nick").role(EmployeeRole
.USER).build(),

Employee.builder().name("Walter").username("walter").password("walter").role(Employ
eeRole.USER).build(),

Employee.builder().name("Jack").username("jack").password("jack").role(EmployeeRole
.USER).build(),

Employee.builder().name("Panos").username("panos").password("panos").role(EmployeeR
ole.USER).build()
           );


           employeeService.saveAll(employees);



           Logger logger = LoggerFactory.getLogger(getClass());


           logger.info("Generating demo data");
```

```
            List<Category> categories = List.of(
                    Category.builder().name("Energy Drinks").build(),
                    Category.builder().name("Soft Drinks").build(),
                    Category.builder().name("Chips").build(),
                    Category.builder().name("Alcohol").build()
            );
            categoryService.saveAll(categories);

            List<ProductSupplier> productSuppliers = List.of(

ProductSupplier.builder().email("inventory@Cocacola.com").name("Coca
Cola").phone("698744253").build(),

ProductSupplier.builder().email("stock@masoutis.com").name("Masoutis").phone("69874
8494").build(),

ProductSupplier.builder().email("stock@lidl.com").name("Lidl").phone("698712733").b
uild()
            );
            productSupplierService.saveAll(productSuppliers);



            List<Tax> taxes = List.of(
                    Tax.builder().name("24%").value(0.24).build()
            );
            taxService.saveAll(taxes);


            List<Product> products = List.of(
                    //Energy Drinks
                                                        Product.builder().name("Hell
250ml").category(categories.get(0)).price(0.71).cost(0.50)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                    .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "Hell" +
".png")
                                .barcode(RandomStringUtils.randomNumeric(14))
                                                         .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                                .build(),
                                                      Product.builder().name("Monster
500ml").category(categories.get(0)).price(1.39).cost(0.80)
```

```
.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "Monster" +
".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                        .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                    Product.builder().name("Red   Bull
250ml").category(categories.get(0)).price(1.18).cost(0.60)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "RedBull" +
".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                        .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                  Product.builder().name("Coca   Cola
500ml").category(categories.get(1)).price(0.90).cost(0.45)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                               .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "CocaCola" +
".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                        .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                Product.builder().name("Fanta   Orange
500ml").category(categories.get(1)).price(0.74).cost(0.30)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                              .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "FantaOrange"
+ ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                        .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                 Product.builder().name("Fanta   Lemon
1,5L").category(categories.get(1)).price(1.47).cost(0.80)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                              .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "FantaLemon" +
".png")
```

```
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                    .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                Product.builder().name("Ruffles
ketchup").category(categories.get(2)).price(1.40).cost(0.70)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                        .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN +
"ruffles_ketchap" + ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                    .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                Product.builder().name("Ruffles
Barbeque").category(categories.get(2)).price(1.70).cost(0.88)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                        .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN +
"ruffles_barbeque" + ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                    .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                Product.builder().name("Ruffles
Salt").category(categories.get(2)).price(1.60).cost(0.90)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                             .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "ruffles_salt"
+ ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                    .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                            Product.builder().name("Chivas  Regal
Scotch").category(categories.get(3)).price(69.80).cost(35.80)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                        .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN +
"ChivasRegalScotch" + ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                    .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
```

```
                                                    Product.builder().name("Serkova
Votka").category(categories.get(3)).price(14.25).cost(8.25)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                            .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN + "SerkovaVotka"
+ ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                            .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build(),
                                                Product.builder().name("Johnnie  Walker
Scotch").category(categories.get(3)).price(43.20).cost(23.20)

.tax(taxes.get(0)).enabled(true).quantity(StringUtil.getRandomNumber(100, 300))
                                            .imageUrl(IMAGE_FOLDER_PATH_DISPLAY_GEN +
"JohnnieWalker" + ".png")
                            .barcode(RandomStringUtils.randomNumeric(14))
                                                            .supplier((ProductSupplier)
StringUtil.getRandomOfList(productSuppliers))
                            .build()
            );
            productService.saveAll(products);


            List<Customer> customers = List.of(

Customer.builder().name("Guest").phone("698744233").email("guest@test.com").orders(
new ArrayList<>()).build(),

Customer.builder().name("John").phone("693755211").email("John@test.com").orders(ne
w ArrayList<>()).build(),

Customer.builder().name("Chris").phone("698715212").email("Chris@test.com").orders(
new ArrayList<>()).build()
            );
            customerService.saveAll(customers);


                    List<Order>  orderList  =  generateRandomOrders(products,  customers,
List.of(OrderStatus.CANCELLED, OrderStatus.PAID));
            orderList.sort(Comparator.comparing(Order::getTimeOrdered).reversed());
            orderService.saveAll(orderList);


                                                    List<Policy>     policies     =
List.of(InventoryPolicies.STOCK_POLICY.getPolicy());
            policyService.saveAll(policies);
```

```
            logger.info("Generated demo data");
        };
    }


    private List<Order> generateRandomOrders(List<Product> products, List<Customer>
customers, List<OrderStatus> statuses){
        int numberOfOrders = 100;
        List<Order> orders = new ArrayList<>();
        long aDay = TimeUnit.DAYS.toMillis(1);
        long now = new Date().getTime();
        Date tendaysago = new Date(now - aDay *100);
        for (int i = 0; i < numberOfOrders; i++) {
            orders.add(Order.builder()
                    .customer((Customer) StringUtil.getRandomOfList(customers))
                                                    .timeOrdered(between(tendaysago,
convertToDateViaInstant(LocalDateTime.now())))
                    .products(getRandomProducts(products))
                                                                .status((OrderStatus)
StringUtil.getRandomOfList(statuses)).build());
        }
        return orders;
    }


    private List<OrderProduct> getRandomProducts(List<Product> products) {
        List<OrderProduct> orderProducts = new ArrayList<>();
        for (int i = 0; i < 5; i++) {
            Product newProduct = (Product) StringUtil.getRandomOfList(products);
              boolean alreadyAdded = orderProducts.stream().anyMatch(orderProduct ->
orderProduct.getProduct().equals(newProduct));
            if (!alreadyAdded) {

orderProducts.add(OrderProduct.builder().product(newProduct).quantity(StringUtil.ge
tRandomNumber(1, 5)).build());
            }
        }
        return orderProducts;
    }


    public static LocalDateTime between(Date startInclusive, Date endExclusive) {
```

```
        long startMillis = startInclusive.getTime();
        long endMillis = endExclusive.getTime();
        long randomMillisSinceEpoch = ThreadLocalRandom
                .current()
                .nextLong(startMillis, endMillis);

        return convertToLocalDateViaInstant(new Date(randomMillisSinceEpoch));
    }

    public static LocalDateTime convertToLocalDateViaInstant(Date dateToConvert) {
        return dateToConvert.toInstant()
                .atZone(ZoneId.systemDefault())
                .toLocalDateTime();
    }

    static Date convertToDateViaInstant(LocalDateTime dateToConvert) {
        return java.util.Date
                .from(dateToConvert.atZone(ZoneId.systemDefault())
                        .toInstant());
    }
}
```

Κλάση SecurityConfiguration

```
package com.example.application.security;

import com.example.application.backend.service.EmployeeService;
import com.example.application.ui.views.LoginView;
import com.vaadin.flow.spring.security.VaadinWebSecurityConfigurerAdapter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authenticati
onManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
```

```
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@EnableWebSecurity
@Configuration
public class SecurityConfiguration extends VaadinWebSecurityConfigurerAdapter {
   private @Autowired EmployeeService employeeService;

   @Override
   protected void configure(HttpSecurity http) throws Exception {

       super.configure(http);

       setLoginView(http, LoginView.class);
   }

   @Override
   public void configure(WebSecurity web) throws Exception {
       web.ignoring().antMatchers(
               "/images/**"
       );

       super.configure(web);
   }


   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {
       auth.authenticationProvider(authenticationProvider());
   }


   @Bean
   public BCryptPasswordEncoder passwordEncoder() {
```

```
        return new PlainEncoder();
    }

    @Bean
    @Override
    public UserDetailsService userDetailsService() {
        return  employeeService;
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(passwordEncoder());

        return authProvider;
    }

    private class PlainEncoder extends BCryptPasswordEncoder {
        @Override
        public String encode(CharSequence rawPassword) {
            return rawPassword.toString();
        }

        @Override
        public boolean matches(CharSequence rawPassword, String encodedPassword) {
            return rawPassword.toString().equals(encodedPassword);
        }
    }
}
```

Κλάση SecurityService

```
package com.example.application.security;

import com.example.application.backend.data.EmployeeRole;
import com.example.application.backend.data.entities.Employee;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.server.VaadinServletRequest;
```

```
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler
;
import org.springframework.stereotype.Component;


import java.util.Objects;
import java.util.Optional;


@Component
public class SecurityService {
   private static final String LOGOUT_SUCCESS_URL = "/";


   public Employee getAuthenticatedUser() {
                                                                  return         (Employee)
Optional.ofNullable(SecurityContextHolder.getContext().getAuthentication())
               .filter(Objects::nonNull)
               .map(Authentication::getPrincipal)
               .filter(o -> o instanceof Employee).orElse(null);
   }


   public void logout() {
       UI.getCurrent().getPage().setLocation(LOGOUT_SUCCESS_URL);
                           SecurityContextLogoutHandler    logoutHandler    =    new
SecurityContextLogoutHandler();

logoutHandler.logout(VaadinServletRequest.getCurrent().getHttpServletRequest(),
null,
               null);
   }


   public boolean isAdmin() {
       return EmployeeRole.ADMIN.equals(getAuthenticatedUser().getRole());
   }
}
```

Κλάση AbstractCrudView

```
package com.example.application.ui.crud;
```

```java
import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.exceptions.DuplicateFieldException;
import com.example.application.backend.exceptions.ReferentialIntegrityException;
import com.example.application.backend.service.AbstractService;
import com.example.application.backend.service.AuditService;
import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import lombok.Getter;
import org.springframework.beans.factory.annotation.Autowired;

import java.time.LocalDateTime;
import java.util.function.Supplier;

public class AbstractCrudView<T> extends VerticalLayout {
   @Getter
   private final AbstractForm<T> genericForm;
   private @Getter Grid<T> grid = new Grid<>();
   private final String toolbarName;
   private final transient AbstractService<T> abstractService;
   private final transient Supplier<T> supplier;
   private final @Getter HorizontalLayout content = new HorizontalLayout();
   @Autowired
   private transient AuditService auditService;
   private Button addNewItemButton;

       public AbstractCrudView(String toolbarName, AbstractForm<T> genericForm,
AbstractService<T> abstractService, Supplier<T> supplier) {
       this.toolbarName = toolbarName;
       this.genericForm = genericForm;
       this.abstractService = abstractService;
       this.supplier = supplier;

       addClassName("list-view");
       setSizeFull();
       configureGrid();
```

```
        configureForm();
        add(getToolbar(), getContentLayout());
        updateList();
        closeEditor();
    }

    public void disableAddNewItemButton(){
        addNewItemButton.setVisible(false);
    }

    private HorizontalLayout getToolbar() {
        addNewItemButton = new Button("Add "+toolbarName);
        addNewItemButton.addClickListener(click -> addItem());

        HorizontalLayout toolbar = new HorizontalLayout(addNewItemButton);
        toolbar.addClassName("toolbar");
        return toolbar;
    }

    public void setEnabledAddItemButton(boolean enabled) {
        addNewItemButton.setEnabled(enabled);
    }

    private void configureForm() {
                        genericForm.addListener(AbstractForm.SaveEvent.class,   e   ->
saveItem(e.getSource()));
                    genericForm.addListener(AbstractForm.DeleteEvent.class,   e   ->
deleteItem(e.getSource()));
        genericForm.addListener(AbstractForm.CloseEvent.class, e -> closeEditor());
    }

    private Component getContentLayout() {
        content.add(grid, genericForm);
        content.addClassNames("content");
        content.setSizeFull();
        return content;
    }

    public void updateList() {
```

```
        grid.setItems(abstractService.findAll());
    }

    public void configureGrid() {
        grid.setSizeFull();
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
                                grid.asSingleSelect().addValueChangeListener(event    ->
editItem(event.getValue()));
    }

    public void configureGrid(Grid<T> newGrid) {
        grid = newGrid;
        grid.setSizeFull();
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
                                grid.asSingleSelect().addValueChangeListener(event    ->
editItem(event.getValue()));
    }

    private void addItem() {
        grid.asSingleSelect().clear();
        editItem(supplier.get());
    }

    public void editItem(T item) {
        if (item == null) {
            closeEditor();
        } else {
            genericForm.setFormObjectSync(item);
            displayTheFormFullScreen(true);
            genericForm.setVisible(true);
            addClassName("editing");
        }
    }

    private void saveItem(AbstractForm<T> event) {
        try {
            abstractService.save(event.getFormObject());
            updateList();
//              genericForm.updateChildItemsOnStateChange();
```

```
                    auditService.save(new Audit(LocalDateTime.now(), "Saved item
successfully", AuditType.INFO), this.getClass());
            closeEditor();
        } catch (DuplicateFieldException e) {
                    auditService.save(new Audit(LocalDateTime.now(), e.getMessage(),
AuditType.ERROR), this.getClass());
        }

    }

    private void deleteItem(AbstractForm<T> event) {
        try {
            abstractService.delete(event.getFormObject());
            updateList();
//            genericForm.updateChildItemsOnStateChange();
                    auditService.save(new Audit(LocalDateTime.now(), "deleted item
successfully", AuditType.ERROR), this.getClass());
            closeEditor();
        } catch (ReferentialIntegrityException e) {
                    auditService.save(new Audit(LocalDateTime.now(), e.getMessage(),
AuditType.ERROR), this.getClass());
        }
    }

    protected void closeEditor() {
        genericForm.setFormObjectSync(null);
        genericForm.setVisible(false);
        removeClassName("editing");
        displayTheFormFullScreen(false);
    }

    public void displayTheFormFullScreen(boolean showForm){
        /* nothing */
    }

}
```

Κλάση AbstractForm

```
package com.example.application.ui.crud;
```

```java
import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.service.AuditService;
import com.vaadin.flow.component.ComponentEvent;
import com.vaadin.flow.component.ComponentEventListener;
import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.formlayout.FormLayout;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.data.binder.BeanValidationBinder;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.data.binder.ValidationException;
import com.vaadin.flow.shared.Registration;
import lombok.Getter;
import lombok.Setter;
import lombok.SneakyThrows;
import org.springframework.beans.factory.annotation.Autowired;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public abstract class AbstractForm<S> extends FormLayout {
    @Getter @Setter
    private S formObject;
    private Binder<S> binder;

    private Button save = new Button("Save");
    private Button delete = new Button("Delete");
    private Button close = new Button("Cancel");
//    private List<SyncField<?>> syncFields = new ArrayList<>();
//    public void addSyncField(SyncField syncField) {
//        syncFields.add(syncField);
//    }
    public @Autowired AuditService auditService;
    private boolean disableDelete;
```

```
@SneakyThrows
public AbstractForm(Class<S> classUsed, boolean disableDelete) {
    this(classUsed);
    this.disableDelete = disableDelete;
}

@SneakyThrows
public AbstractForm(Class<S> classUsed) {
    binder = new BeanValidationBinder<>(classUsed);
    formObject = classUsed.getDeclaredConstructor().newInstance();
    addClassName(getClass().getSimpleName());
    setWidth("30em");
}

public void addButtons(){
    add(createButtonsLayout());
}
public void addButtons(VerticalLayout verticalLayout){
    verticalLayout.add(createButtonsLayout());
}
public void setButtonVisibility(boolean shouldDisplay) {
    save.setEnabled(shouldDisplay);
    close.setEnabled(shouldDisplay);
}



public Button getSaveButton() {
    return save;
}

public Binder<S> getBinder() {return binder;}
public void setFormObjectSync(S formObject) {
    this.formObject = formObject;
    binder.readBean(formObject);
}

public void validateAndSave() {
```

```
        if (!isValidForSave())
            return;
        try {
            binder.writeBean(formObject);
            fireEvent(new SaveEvent(this, formObject));
        } catch (ValidationException e) {
                    auditService.save(new Audit(LocalDateTime.now(), e.getMessage(),
AuditType.ERROR), this.getClass());
        }
    }

    public boolean isValidForSave() {
        return true;
    }

    public void delete(){
        fireEvent(new DeleteEvent(this, formObject));
    }

    public void close(){
        fireEvent(new CloseEvent(this));
    }

//    public void updateChildItemsOnStateChange() {
//        syncFields.forEach(SyncField::run);
//    };

    private HorizontalLayout createButtonsLayout() {
        save.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        delete.addThemeVariants(ButtonVariant.LUMO_ERROR);
        close.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

        save.addClickShortcut(Key.ENTER);
        close.addClickShortcut(Key.ESCAPE);

        save.addClickListener(event -> validateAndSave());
        delete.addClickListener(event -> delete());
        close.addClickListener(event -> close());
```

```
        binder.addStatusChangeListener(e -> save.setEnabled(binder.isValid()));
        if (disableDelete) {
            delete.setVisible(false);
        }
        return new HorizontalLayout(save, delete, close);
    }

    public static abstract class FormEvent extends ComponentEvent<AbstractForm> {
        private final Object formObject;

        public Object getFormObject() {
            return formObject;
        }

        protected FormEvent(AbstractForm source, Object formObject) {
            super(source, false);
            this.formObject = formObject;
        }

    }

    public static class SaveEvent extends FormEvent {
        SaveEvent(AbstractForm source, Object object) {
            super(source, object);
        }
    }

    public static class DeleteEvent extends FormEvent {
        DeleteEvent(AbstractForm source, Object object) {
            super(source, object);
        }

    }

    public static class CloseEvent extends FormEvent {
        CloseEvent(AbstractForm source) {
            super(source, null);
        }
```

```
    }

        public <T extends ComponentEvent<?>> Registration addListener(Class<T>
eventType,

ComponentEventListener<T> listener) {
        return getEventBus().addListener(eventType, listener);
    }
}
```

Κλάση CategoryForm

```
package com.example.application.ui.views.forms;

import com.example.application.backend.data.entities.Category;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.textfield.TextField;

public class CategoryForm extends AbstractForm<Category> {
 TextField name = new TextField("name");

 public CategoryForm() {
    super(Category.class);
    getBinder().bindInstanceFields(this);
    add(name);
    addButtons();
 }

}
```

Κλάση CustomerForm

```
package com.example.application.ui.views.forms;

import com.example.application.backend.data.entities.Customer;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.textfield.TextField;

public class CustomerForm extends AbstractForm<Customer> {
 TextField name = new TextField("name");
 TextField email = new TextField("email");
```

```
 TextField phone = new TextField("phone");


 public CustomerForm() {
   super(Customer.class);
   getBinder().bindInstanceFields(this);
   add(name, email, phone);;
   addButtons();
 }


}
```

Κλάση EmployeeForm

```
package com.example.application.ui.views.forms;


import com.example.application.backend.data.EmployeeRole;
import com.example.application.backend.data.entities.Employee;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.textfield.PasswordField;
import com.vaadin.flow.component.textfield.TextField;


import java.util.List;


public class EmployeeForm extends AbstractForm<Employee> {
 TextField name = new TextField("name");
 TextField username = new TextField("username");
 private final ComboBox<EmployeeRole> role = new ComboBox("role");
 PasswordField password = new PasswordField("password");



 public EmployeeForm() {
   super(Employee.class);
   getBinder().bindInstanceFields(this);
   role.setItems(List.of(EmployeeRole.values()));
   add(name, username, password, role);


   addButtons();
 }
```

```
 @Override
 public void setFormObjectSync(Employee formObject) {
   super.setFormObjectSync(formObject);
 }


}
```

Κλάση OrderForm

```
package com.example.application.ui.views.forms;


import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.OrderStatus;
import com.example.application.backend.service.AuditService;
import com.example.application.backend.service.PolicyHelper;
import com.example.application.backend.data.entities.*;
import com.example.application.backend.service.OrderService;
import com.example.application.backend.util.StringUtil;
import com.example.application.ui.crud.AbstractForm;
import com.example.application.ui.views.OrderView;
import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.html.*;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.notification.NotificationVariant;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.data.binder.ValidationException;



import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.function.Supplier;

public class OrderForm extends AbstractForm<Order> {
//  private final ComboBox<OrderStatus> status = new ComboBox("status");
```

```
 private final ComboBox<Customer> customer = new ComboBox<>("customer");
 private final Supplier<List<Product>> productSupplier;
 private final Supplier<List<Customer>> customerSupplier;
 private final Span orderStatus = new Span();
 private final Span timeOrderedSpan = new Span();
 private final Span idSpan = new Span();
 private final Span quantityWarningSpan = new Span();
 private ProductOrderMaker productOrderMaker;

 private final Button payButton;
 private boolean isDisabled = false;
 private OrderService orderService;

 private final Button cancelButton;
 private PolicyHelper policyHelper;
 private final AuditService auditService;
 public OrderForm(Supplier<List<Product>> productSupplier, Supplier<List<Customer>>
customerSupplier,
                  OrderService orderService, PolicyHelper policyHelper,
                  AuditService auditService) {
   super(Order.class, true);
   this.setWidth(30, Unit.PERCENTAGE);
   this.customerSupplier = customerSupplier;
   this.productSupplier = productSupplier;
   this.orderService = orderService;
   this.policyHelper = policyHelper;
   this.auditService = auditService;

   getBinder().bindInstanceFields(this);
   customer.setItems(customerSupplier.get());
   customer.setItemLabelGenerator(Customer::getName);

   HorizontalLayout details = new HorizontalLayout();
   details.setWidth(100, Unit.PERCENTAGE);
            details.add(new    VerticalLayout(timeOrderedSpan,    idSpan),    new
VerticalLayout(orderStatus, customer));
     VerticalLayout  detailsWrapper  =  new  VerticalLayout(new  H2("Order  details"),
details);
   detailsWrapper.setSizeFull();
```

```
            OrderStatus savedOrderStatus = getFormObject() != null ?
getFormObject().getStatus() : OrderStatus.NEW;


    boolean isNew = OrderStatus.NEW.equals(savedOrderStatus);
    payButton = new Button("Pay Order", e -> payOrder());
    payButton.getStyle().set("background-color", "green").set("color", "black");
    payButton.setVisible(isNew);


    boolean isPaid = OrderStatus.PAID.equals(savedOrderStatus);
    cancelButton = new Button("Cancel Order", e -> cancelOrder());
    cancelButton.addThemeVariants(ButtonVariant.LUMO_ERROR);
    cancelButton.setVisible(isPaid);
    setButtonVisibility(!(isNew || isPaid));


    productOrderMaker = new ProductOrderMaker(productSupplier, newQuantity -> {
        calculateWarnings(getFormObject(), isPaid);
    });
    quantityWarningSpan.getStyle().set("color", "red");
         add(detailsWrapper, productOrderMaker, quantityWarningSpan, payButton,
cancelButton);
    addButtons();
 }


 private void updateIdAndSpan(){
     if (getFormObject() != null) {
                                      timeOrderedSpan.setText("Timeordered:     "     +
StringUtil.formatDate(getFormObject().getTimeOrdered()));
          idSpan.setText("Id: " + getFormObject().getId());
     }
 }


 @Override
 public void setFormObjectSync(Order formObject) {
    super.setFormObjectSync(formObject);
    if (formObject != null) {
        updateIdAndSpan();
        OrderStatus orderStatus = formObject.getStatus();
        boolean isPaid = OrderStatus.PAID.equals(orderStatus);
```

```
        isDisabled =  isPaid || OrderStatus.CANCELLED.equals(orderStatus);
        productOrderMaker.disableCartButtons(isDisabled);
        payButton.setEnabled(!isDisabled);
        customer.setEnabled(!isDisabled);

        this.orderStatus.setText("Status: " + orderStatus.toString());
        payButton.setVisible(OrderStatus.NEW.equals(orderStatus));
        cancelButton.setVisible(isPaid);
        setButtonVisibility(!isDisabled);
        calculateWarnings(formObject, OrderStatus.PAID.equals(orderStatus));

    } else {
        setButtonVisibility(true);
        quantityWarningSpan.setText("");
    }
    productOrderMaker.syncWithOrder(formObject != null ? formObject.getProducts() :
new ArrayList<>());
 }

 public void payOrder() {
     try {
         getBinder().writeBean(getFormObject());
     } catch (ValidationException e) {
                 auditService.save(new  Audit(LocalDateTime.now(),  e.getMessage(),
AuditType.ERROR), this.getClass());
     }
                                                          Notification.show("Order
paid!").addThemeVariants(NotificationVariant.LUMO_SUCCESS);
     fireEvent(new OrderView.PayEvent(this, getFormObject()));
 }

 public void cancelOrder() {
     try {
         getBinder().writeBean(getFormObject());
     } catch (ValidationException e) {
         throw new RuntimeException(e);
     }
                                                          Notification.show("Order
cancelled!").addThemeVariants(NotificationVariant.LUMO_SUCCESS);
     fireEvent(new OrderView.CancelEvent(this, getFormObject()));
```

```
 }

 public void calculateWarnings(Order order, boolean isPaid) {
     String warnings = null;
     boolean shouldCalcWarnings = policyHelper.isStockPolicyNoOrWarning();
     boolean shouldBlockPayment = policyHelper.isStockPolicyNo();
     if (shouldCalcWarnings) {
         warnings = orderService.getQuantityWarning(order);
     }
     boolean warningsExist = warnings != null;

     payButton.setEnabled(!isPaid && !(shouldBlockPayment && warningsExist));
     quantityWarningSpan.setText(warnings);
 }
}
```

Κλάση PolicyForm

```
package com.example.application.ui.views.forms;



import com.example.application.backend.data.entities.Policy;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.html.H1;

public class PolicyForm extends AbstractForm<Policy> {
 H1 name = new H1("name");
 private final ComboBox<String> selectedValue = new ComboBox<>("value");
 public PolicyForm() {
   super(Policy.class);
   add(name, selectedValue);
   addButtons();
 }

 @Override
```

```
 public void setFormObjectSync(Policy formObject) {
   super.setFormObjectSync(formObject);
   if (formObject!= null) {
     name.setText(formObject.getName());
     selectedValue.setItems(formObject.getValues());
     selectedValue.setValue(formObject.getUserSelectedValue());
              getBinder().forField(selectedValue).bind(Policy::getUserSelectedValue,
Policy::setSelectedValue);
   }
 }

}
```

Κλάση ProductForm

```
package com.example.application.ui.views.forms;

import com.example.application.backend.data.entities.ProductSupplier;
import com.example.application.backend.util.StringUtil;
import com.example.application.backend.data.entities.Category;
import com.example.application.backend.data.entities.Product;
import com.example.application.backend.data.entities.Tax;
import com.example.application.backend.service.ImageService;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.checkbox.Checkbox;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.textfield.NumberField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.component.upload.Upload;
import com.vaadin.flow.component.upload.receivers.MultiFileMemoryBuffer;
import com.vaadin.flow.data.converter.StringToIntegerConverter;

import java.util.List;
import java.util.function.Supplier;

public class ProductForm extends AbstractForm<Product> {
 TextField name = new TextField("name");
 TextField barcode = new TextField("barcode");
 NumberField cost = new NumberField("price buy");
 NumberField price = new NumberField("price sell");
```

```
 TextField quantity = new TextField("quantity");
 TextField imageUrl = new TextField("image URL");
 ComboBox<Category> category = new ComboBox<>("category");
 ComboBox<Tax> tax = new ComboBox<>("tax");
 ComboBox<ProductSupplier> supplier = new ComboBox<>("supplier");
 Checkbox enabled = new Checkbox("enabled");
 private Upload upload;
 private ImageService imageService;

 public void setCategories(List<Category> categories) {
   category.setItems(categories);
 }

 public ProductForm(Supplier<List<Category>> categories, Supplier<List<Tax>> taxes,
ImageService imageService, Supplier<List<ProductSupplier>> suppliers) {
   super(Product.class);
   this.imageService = imageService;

   getBinder().forField(quantity)
           .withConverter(new StringToIntegerConverter("Not a number"))
           .bind(Product::getQuantity, Product::setQuantity);
   getBinder().bindInstanceFields(this);
   name.setRequired(true);
   name.setErrorMessage("name must be filled in!");

   category.setItems(categories.get());
   category.setErrorMessage("category must be filled in!");
   category.setItemLabelGenerator(Category::getName);
   category.setRequired(true);

   tax.setItems(taxes.get());
   tax.setErrorMessage("tax must be filled in!");
   tax.setRequired(true);
   tax.setItemLabelGenerator(Tax::getName);

   supplier.setItems(suppliers.get());
   supplier.setErrorMessage("tax must be filled in!");
   supplier.setRequired(false);
   supplier.setItemLabelGenerator(ProductSupplier::getName);
```

```
    initUploaderImage();
      add(name, barcode, imageUrl, upload, category, tax, supplier, cost, price,
quantity, enabled);
    addButtons();
  }



  @Override
  public boolean isValidForSave() {
    return !name.getValue().isEmpty()
            && tax.getValue() != null
            && category.getValue() != null;
  }


  private void initUploaderImage() {
    MultiFileMemoryBuffer buffer = new MultiFileMemoryBuffer();
    upload = new Upload(buffer);
    upload.setAcceptedFileTypes("image/jpeg","image/jpg", "image/png", "image/gif");
    upload.addSucceededListener(event -> imageUrl.setValue(
            imageService.saveImage(event.getFileName(),
                                   StringUtil.FirstNotNullOrEmpty(name.getValue(),
event.getFileName()),
                    buffer))
    );
    add(upload);
  }


  @Override
  public void validateAndSave() {
    super.validateAndSave();


  }


  @Override
  public void setFormObjectSync(Product formObject) {
    super.setFormObjectSync(formObject);
    upload.clearFileList();
  }
```

```
}
```

Κλάση ProductOrderMaker

```
package com.example.application.ui.views.forms;


import com.example.application.backend.data.entities.OrderProduct;
import com.example.application.backend.data.entities.Product;
import com.example.application.backend.util.StringUtil;
import com.vaadin.flow.component.HasText;
import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.*;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.notification.NotificationVariant;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.value.ValueChangeMode;

import java.util.List;
import java.util.function.Consumer;
import java.util.function.Supplier;

class ProductOrderMaker extends VerticalLayout {
   private final Grid<OrderProduct> orderProductGrid = new Grid<>();
   private List<OrderProduct> orderProducts;
   private final Grid<Product> productGrid = new Grid<>();
   private Div productTotal = new Div();
   private Div taxTotal = new Div();
   private Div orderTotal = new Div();
   private Supplier<List<Product>> productSupplier;
   private Button addToOrderButton;
   private Button productAddToOrderButton = new Button("+");
   private boolean isDisabled = false;
   private TextField productSearch = new TextField();
```

```
    private Consumer<Integer> quantityChangeConsumer;



    ProductOrderMaker(Supplier<List<Product>> productSupplier, Consumer<Integer>
quantityChangeConsumer) {
        this.quantityChangeConsumer = quantityChangeConsumer;
        this.productSupplier = productSupplier;
        setupProductGrid();
        setupOrderProductGrid();


        VerticalLayout orderingLayout = new VerticalLayout();
        orderingLayout.setSizeFull();
                orderingLayout.add(new H2("Ordered products"), addToOrderField(),
productGrid, orderProductGrid, new VerticalLayout(productTotal, taxTotal,
orderTotal));
        add(orderingLayout);
//          getStyle().set("background-color", "red");
    }


    private void setupProductGrid() {
        productGrid.addColumn(Product::getName).setHeader("Product Name");
        productGrid.addComponentColumn(product -> {
            String url = product.getImageUrl();
              Image image = new Image(url != null? url : "" , product.getName() + "
image is missing");
            image.setWidth(50, Unit.PIXELS);
            image.setHeight(50, Unit.PIXELS);
            return image;
        }).setHeader("Image");
        productGrid.addComponentColumn(product -> {
            Button productAddToOrderButton = new Button("+");
            productAddToOrderButton.setEnabled(!isDisabled);
                        productAddToOrderButton.addClickListener(buttonClickEvent ->
addProductToOrder(product));
            return productAddToOrderButton;
        });
        updateProductGrid();
    }


    private void setupOrderProductGrid() {
```

```
                                    orderProductGrid.addColumn(orderProduct ->
orderProduct.getProduct().getName()).setHeader("Product").setWidth("140px");

        orderProductGrid.addComponentColumn(orderProduct -> {

            String url = orderProduct.getProduct().getImageUrl();

                        Image image = new Image(url != null? url : "" ,
orderProduct.getProduct().getName() + " image is missing");

            image.setWidth(50, Unit.PIXELS);

            image.setHeight(50, Unit.PIXELS);

            return image;

        }).setHeader("Image").setWidth("60px");

        orderProductGrid.addComponentColumn(orderProduct -> {

            Span quantity = new Span(orderProduct.getQuantity()+"");

            Div addButton = new Div();

            addButton.setEnabled(!isDisabled);

            addButton.setText("+");

            addButton.setWidth(20, Unit.PIXELS);

            addButton.setHeight(20, Unit.PIXELS);

                                    addButton.addClickListener(buttonClickEvent ->
increaseOrderProductQuantity(orderProduct, quantity));

            Div minusButton = new Div();

            minusButton.setText("-");

            minusButton.setEnabled(!isDisabled);

            minusButton.setWidth(20, Unit.PIXELS);

            minusButton.setHeight(20, Unit.PIXELS);

            minusButton.getStyle().set("font-weight", "bold");

                                minusButton.addClickListener( buttonClickEvent ->
decreaseOrderProductQuantity(orderProduct, quantity));

                HorizontalLayout addMinusLayout = new HorizontalLayout(minusButton,
quantity, addButton);


addMinusLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.CENTER);

            return addMinusLayout;

        }).setHeader("Quantity");

                                    orderProductGrid.addColumn(orderProduct ->
StringUtil.formatPrice(orderProduct.getPriceWithoutTax())).setHeader("Price");

    }


    protected void addProductToOrder(Product product) {

                            if (orderProducts.stream().noneMatch(orderProduct ->
orderProduct.getProduct().equals(product))) {

            orderProducts.add(new OrderProduct(product));
```

```
            orderProductGrid.getDataProvider().refreshAll();
            UpdateOrderProductGrid();
            if (quantityChangeConsumer != null) {
                quantityChangeConsumer.accept(1);
            }
        }

    }

    private void increaseOrderProductQuantity(OrderProduct orderProduct, HasText
hasText) {
        orderProduct.incrementQuantity();
        hasText.setText(orderProduct.getQuantity()+"");
        UpdateOrderProductGrid();
        if (quantityChangeConsumer != null) {
            quantityChangeConsumer.accept(orderProduct.getQuantity());
        }
    }

    private void decreaseOrderProductQuantity(OrderProduct orderProduct, HasText
hasText) {
        if (orderProduct.decrementQuantityOrRemove()) {
            orderProducts.remove(orderProduct);
        } else {
            hasText.setText(orderProduct.getQuantity()+"");
            // to update the price
        }
        UpdateOrderProductGrid();
        if (quantityChangeConsumer != null) {
            quantityChangeConsumer.accept(orderProduct.getQuantity());
        }
    }

    private HorizontalLayout addToOrderField() {
        TextField productSearch = new TextField();
        productSearch.setPlaceholder("Filter by name or Barcode...");
        productSearch.setClearButtonVisible(true);
        productSearch.setValueChangeMode(ValueChangeMode.LAZY);
```

```
        addToOrderButton = new Button("Add to Order");
        addToOrderButton.addClickListener(click -> {
            Product product = productSupplier.get().stream()
                    .filter(productStored -> productStored.getBarcode() != null
                                                                                &&
productStored.getBarcode().equals(productSearch.getValue()))
                    .findAny().orElse(null);
            if (product != null)
                addProductToOrder(product);
            else
                                                    Notification.show("product   not
found").addThemeVariants(NotificationVariant.LUMO_ERROR);
        });

                HorizontalLayout   toolbar   =   new   HorizontalLayout(productSearch,
addToOrderButton);
        toolbar.addClassName("toolbar");
        return toolbar;
    }

    public void syncWithOrder(List<OrderProduct> orderProducts) {
        this.orderProducts = orderProducts;
        UpdateOrderProductGrid();
        updateProductGrid();
    }

    public void disableCartButtons(boolean shouldDisable) {
        addToOrderButton.setEnabled(!shouldDisable);
        isDisabled = shouldDisable;
        productAddToOrderButton.setEnabled(!shouldDisable);
        productSearch.setEnabled(!shouldDisable);
    }

    private void UpdateOrderProductGrid() {
        orderProductGrid.setItems(orderProducts);
                                    productTotal.setText("Product     total:     "     +
StringUtil.formatPrice(orderProducts.stream().mapToDouble(OrderProduct::getPriceWit
houtTax).sum()));
                                        taxTotal.setText("Tax      total:      "      +
StringUtil.formatPrice(orderProducts.stream().mapToDouble(OrderProduct::getOnlyTaxP
rice).sum()));
```

```
                                    orderTotal.setText("Order     total:     "     +
StringUtil.formatPrice(orderProducts.stream().mapToDouble(OrderProduct::getPriceWit
hTax).sum()));


    }


    private void updateProductGrid() {
        productGrid.setItems(productSupplier.get());
    }
}
```

Κλάση ProductSupplierForm

```
package com.example.application.ui.views.forms;



import com.example.application.backend.data.entities.ProductSupplier;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.textfield.TextField;

public class ProductSupplierForm extends AbstractForm<ProductSupplier> {
 TextField name = new TextField("name");
 TextField email = new TextField("email");
 TextField phone = new TextField("phone");

 public ProductSupplierForm() {
    super(ProductSupplier.class);
    getBinder().bindInstanceFields(this);
    name.setRequired(true);
    name.setErrorMessage("name must be filled in!");
    add(name, email, phone);
    addButtons();
 }

}
```

Κλάση TaxForm

```
package com.example.application.ui.views.forms;
```

```
import com.example.application.backend.data.entities.Tax;
import com.example.application.ui.crud.AbstractForm;
import com.vaadin.flow.component.textfield.NumberField;
import com.vaadin.flow.component.textfield.TextField;

public class TaxForm extends AbstractForm<Tax> {
 TextField name = new TextField("name");
 NumberField value = new NumberField("value");

 public TaxForm() {
   super(Tax.class);
   getBinder().bindInstanceFields(this);
   name.setRequired(true);
   name.setErrorMessage("name must be filled in!");
   value.setRequiredIndicatorVisible(true);
   value.setErrorMessage("tax must be filled in!");
   add(name, value);
   addButtons();
 }

}
```

Κλάση AuditView

```
package com.example.application.ui.views;




import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.example.application.backend.service.AuditService;
import com.example.application.backend.util.StringUtil;
import com.example.application.ui.MainLayout;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
```

```
import javax.annotation.security.PermitAll;



@Route(value = "reports/audit", layout = MainLayout.class)
@PageTitle("Audit | Productive Manager")
@PermitAll
public class AuditView extends VerticalLayout {

   public AuditView(AuditService auditService) {
       setSizeFull();
       Grid<Audit> grid = new Grid<>(Audit.class);
       grid.addClassNames("product-grid");
       grid.setColumns();
       grid.addComponentColumn(product -> {
           Span colorSpan = new Span(product.getAuditType().name());
                                                   colorSpan.getStyle().set("color",
auditTypeToColor(product.getAuditType()));
           return colorSpan;
       }).setHeader("AuditType");
       grid.addColumn(Audit::getCreator).setHeader("Creator").setWidth("70px");
                                                       grid.addColumn(audit        ->
StringUtil.formatDate(audit.getTimeCreated())).setHeader("TimeCreated");
       grid.addColumn(Audit::getMessage).setHeader("Message");
       grid.getColumns().forEach(col -> col.setAutoWidth(true));
       grid.setSizeFull();
       grid.setItems(auditService.findAll());
       add(grid);
   }

   private String auditTypeToColor(AuditType auditType) {
       String color = "white";
       switch (auditType) {

           case WARN:
           case DEBUG:
               color = "yellow";
               break;
           case ERROR:
               color = "red";
```

```
                break;
            case INFO:
            case TRACE:
                color = "blue";
                break;
        }
        return color;
    }

}
```

Κλάση CategoryView

```
package com.example.application.ui.views;

import com.example.application.backend.data.entities.Category;
import com.example.application.backend.service.CategoryService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.CategoryForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;



@Route(value = "inventory/categories", layout = MainLayout.class)
@PageTitle("Categories | Productive Manager")
@PermitAll
public class CategoryView extends AbstractCrudView<Category> {

    public CategoryView(CategoryService categoryService) {
        super("Category", new CategoryForm(), categoryService, Category::new);
    }

    @Override
    public void configureGrid() {
        Grid<Category> grid = new Grid<>(Category.class);
```

```
        grid.addClassNames("product-grid");
        grid.setSizeFull();
        grid.setColumns();
        grid.addColumn(Category::getId).setHeader("Id");
        grid.addColumn(Category::getName).setHeader("Name");
        super.configureGrid(grid);
    }
}
```

Κλάση CustomerView

```
package com.example.application.ui.views;

import com.example.application.backend.data.entities.Customer;
import com.example.application.backend.service.CustomerService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.CustomerForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;



@Route(value = "ordering/customer", layout = MainLayout.class)
@PageTitle("Customer | Productive Manager")
@PermitAll
public class CustomerView extends AbstractCrudView<Customer> {

    public CustomerView(CustomerService customerService) {
        super("Customer", new CustomerForm(), customerService, Customer::new);
    }

    @Override
    public void configureGrid() {
        Grid<Customer> grid = new Grid<>(Customer.class);
        grid.addClassNames("product-grid");
        grid.setSizeFull();
```

```
        grid.setColumns();
        grid.addColumn(Customer::getId).setHeader("Id");
        grid.addColumn(Customer::getName).setHeader("Name");
        grid.addColumn(Customer::getEmail).setHeader("Email");
        grid.addColumn(Customer::getPhone).setHeader("Phone");
//          grid.addColumn(customer -> customer.getOrders().size()).setHeader("Total
Orders");
        super.configureGrid(grid);
    }
}
```

Κλάση EmployeeView

```
package com.example.application.ui.views;




import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.service.EmployeeService;
import com.example.application.security.SecurityService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.EmployeeForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;



@Route(value = "reports/employee", layout = MainLayout.class)
@PageTitle("Employee | Productive Manager")
@PermitAll
public class EmployeeView extends AbstractCrudView<Employee> {

          public    EmployeeView(EmployeeService    employeeService,    SecurityService
securityService) {
        super("employee", new EmployeeForm(), employeeService, Employee::new);
        setEnabledAddItemButton(securityService.isAdmin());
    }
```

```
    @Override
    public void configureGrid() {
        Grid<Employee> grid = new Grid<>(Employee.class);
        grid.addClassNames("product-grid");
        grid.setColumns();
        grid.addColumn(Employee::getName).setHeader("Name");
        grid.addColumn(Employee::getUsername).setHeader("Username");
        grid.addColumn(Employee::getRole).setHeader("Role");
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
        grid.setSizeFull();
        super.configureGrid(grid);
    }
}
```

Κλάση EmployeeView

```
package com.example.application.ui.views;




import com.example.application.backend.data.entities.Employee;
import com.example.application.backend.service.EmployeeService;
import com.example.application.security.SecurityService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.EmployeeForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;



@Route(value = "reports/employee", layout = MainLayout.class)
@PageTitle("Employee | Productive Manager")
@PermitAll
public class EmployeeView extends AbstractCrudView<Employee> {

```

```
        public EmployeeView(EmployeeService employeeService, SecurityService
securityService) {
        super("employee", new EmployeeForm(), employeeService, Employee::new);
        setEnabledAddItemButton(securityService.isAdmin());
    }

    @Override
    public void configureGrid() {
        Grid<Employee> grid = new Grid<>(Employee.class);
        grid.addClassNames("product-grid");
        grid.setColumns();
        grid.addColumn(Employee::getName).setHeader("Name");
        grid.addColumn(Employee::getUsername).setHeader("Username");
        grid.addColumn(Employee::getRole).setHeader("Role");
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
        grid.setSizeFull();
        super.configureGrid(grid);
    }
}
```

Κλάση LoginView

```
package com.example.application.ui.views;

import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.login.LoginForm;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.BeforeEnterEvent;
import com.vaadin.flow.router.BeforeEnterObserver;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.server.auth.AnonymousAllowed;

@Route("login")
@PageTitle("Login")
@AnonymousAllowed
public class LoginView extends VerticalLayout implements BeforeEnterObserver {
```

```
    private LoginForm login = new LoginForm();

    public LoginView() {
        addClassName("login-view");
        setSizeFull();

        setJustifyContentMode(JustifyContentMode.CENTER);
        setAlignItems(Alignment.CENTER);

        login.setAction("login");
        Image image = new Image("./images/imm.png" ,"logo image is missing");
        image.setHeight(150, Unit.PIXELS);
        add(new HorizontalLayout(image, new H1("Productive Manager")), login);
    }

    @Override
    public void beforeEnter(BeforeEnterEvent beforeEnterEvent) {
        if(beforeEnterEvent.getLocation()
            .getQueryParameters()
            .getParameters()
            .containsKey("error")) {
            login.setError(true);
        }
    }
}
```

Κλάση NotificationManager

```
package com.example.application.ui.views;

import com.example.application.backend.data.AuditType;
import com.example.application.backend.data.entities.Audit;
import com.vaadin.flow.component.Text;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.notification.Notification;
```

```
import com.vaadin.flow.component.notification.NotificationVariant;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import org.springframework.stereotype.Component;

@Component
public class NotificationManager {

   public void notificationError(Audit audit) {
                                                notificationError(audit.getMessage(),
matchVariantWithLogLevel(audit.getAuditType()));
   }

          public    void    notificationError(String    string,    NotificationVariant
notificationVariant){
       Text notificationText = new Text(string);
       Notification notificationError = new Notification();
       notificationError.addThemeVariants(notificationVariant);
       Div text = new Div(notificationText);
       Button closeButton = new Button(new Icon("lumo", "cross"));
       closeButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY_INLINE);
       closeButton.getElement().setAttribute("aria-label", "Close");
       closeButton.addClickListener(event -> {
           notificationError.close();
       });

       HorizontalLayout layout = new HorizontalLayout(text, closeButton);
       layout.setAlignItems(FlexComponent.Alignment.CENTER);

       notificationError.add(layout);
       notificationError.setDuration(4000);
       notificationError.open();
   }

   private NotificationVariant matchVariantWithLogLevel(AuditType auditType) {
       NotificationVariant notificationVariant;
           switch (auditType) {
               case INFO:
                   notificationVariant = NotificationVariant.LUMO_PRIMARY;
```

```
                    break;
                case WARN:
                case DEBUG:
                case ERROR:
                    notificationVariant = NotificationVariant.LUMO_ERROR;
                    break;
                case TRACE:
                default:
                    notificationVariant = NotificationVariant.LUMO_CONTRAST;
            }
            return notificationVariant;
    }

}
```

Κλαση OrderView

```
package com.example.application.ui.views;

import com.example.application.backend.data.OrderStatus;
import com.example.application.backend.service.*;
import com.example.application.backend.data.entities.Order;
import com.example.application.backend.util.StringUtil;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.crud.AbstractForm;
import com.example.application.ui.views.forms.OrderForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.data.provider.QuerySortOrder;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;
import java.util.stream.Stream;



@Route(value = "inventory/orders", layout = MainLayout.class)
@PageTitle("Orders | Productive Manager")
```

```
@PermitAll
public class OrderView extends AbstractCrudView<Order> {
   private final transient OrderService orderService;
      public OrderView(OrderService orderService, ProductService productService,
CustomerService customerService, PolicyHelper policyHelper, AuditService
auditService) {
                    super("Order", new OrderForm(productService::findAllEnabled,
customerService::findAll, orderService, policyHelper, auditService), orderService,
Order::new);
       this.orderService = orderService;
   }

   @Override
   public void configureGrid() {
       Grid<Order> grid = new Grid<>(Order.class);
       grid.addClassNames("product-grid");
       grid.setSizeFull();
       grid.setColumns();
       grid.addColumn(Order::getId).setHeader("Id");
                                                    grid.addColumn(order ->
StringUtil.formatDate(order.getTimeOrdered())).setHeader("TimeOrdered");
       grid.addComponentColumn(product -> {
           OrderStatus orderStatus = product.getStatus();
           Span colorSpan = new Span(orderStatus.name());
           colorSpan.getStyle().set("color", orderStatusToColor(orderStatus));
           return colorSpan;
       }).setHeader("Status");
                                                    grid.addColumn(order ->
StringUtil.formatPrice(order.getPrice())).setHeader("Price").setSortOrderProvider(p
rice -> Stream.of(new QuerySortOrder("price", price)));
                                                    grid.addColumn(order ->
order.getCustomer().getName()).setHeader("Customer").setSortOrderProvider(customer
-> Stream.of(new QuerySortOrder("name", customer)));
       super.configureGrid(grid);

                      super.getGenericForm().addListener(PayEvent.class, e ->
payOrder(e.getSource()));
                    super.getGenericForm().addListener(CancelEvent.class, e ->
cancelOrder(e.getSource()));
   }

   private String orderStatusToColor(OrderStatus orderStatus) {
```

```
        String color = "black";
        switch (orderStatus) {
            case PAID:
                color = "Green";
                break;
            case CANCELED:
                color = "Red";
                break;
            case NEW:
                color = "Yellow";
                break;
        }
        return color;
    }

    public void payOrder(AbstractForm<Order> event) {
        Order updatedOrder = orderService.pay(event.getFormObject());
        super.getGenericForm().setFormObjectSync(updatedOrder);
        super.closeEditor();
        updateList();
    }

    public void cancelOrder(AbstractForm<Order> event) {
        Order updatedOrder = orderService.cancel(event.getFormObject());
        super.getGenericForm().setFormObjectSync(updatedOrder);
        super.closeEditor();
        updateList();
    }

    public static class PayEvent extends AbstractForm.FormEvent {
        public PayEvent(AbstractForm source, Object object) {
            super(source, object);
        }
    }

    public static class CancelEvent extends AbstractForm.FormEvent {
        public CancelEvent(AbstractForm source, Object object) {
            super(source, object);
```

```
        }
    }


}
```

Κλάση PolicyView

```
package com.example.application.ui.views;

import com.example.application.backend.data.entities.Policy;
import com.example.application.backend.service.PolicyService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.PolicyForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;



@Route(value = "reports/setting", layout = MainLayout.class)
@PageTitle("Setting | Productive Manager")
@PermitAll
public class PolicyView extends AbstractCrudView<Policy> {

    public PolicyView(PolicyService PolicyService) {
        super("Policy", new PolicyForm(), PolicyService, Policy::new);
        disableAddNewItemButton();
    }

    @Override
    public void configureGrid() {
        setSizeFull();
        Grid<Policy> grid = new Grid<>(Policy.class);
        grid.addClassNames("product-grid");
        grid.setSizeFull();
        grid.setColumns();
        grid.addColumn(Policy::getName).setHeader("Name");
```

```
        grid.addColumn(Policy::getUserSelectedValue).setHeader("Value");
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
        super.configureGrid(grid);
    }
}
```

Κλάση ProductSupplierView

```
package com.example.application.ui.views;

import com.example.application.backend.data.entities.ProductSupplier;
import com.example.application.backend.service.ProductSupplierService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.ProductSupplierForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;

@Route(value = "inventory/suppliers", layout = MainLayout.class)
@PageTitle("Suppliers | Productive Manager")
@PermitAll
public class ProductSupplierView extends AbstractCrudView<ProductSupplier> {
    public ProductSupplierView(ProductSupplierService productSupplierService) {
            super("Supplier", new ProductSupplierForm(), productSupplierService,
ProductSupplier::new);
    }

    @Override
    public void configureGrid() {
        Grid<ProductSupplier> grid = new Grid<>(ProductSupplier.class);
        grid.addClassNames("product-grid");
        grid.setSizeFull();
        grid.setColumns();
        grid.addColumn(ProductSupplier::getId).setHeader("Id");
        grid.addColumn(ProductSupplier::getName).setHeader("Name");
        grid.addColumn(ProductSupplier::getEmail).setHeader("Email");
        grid.addColumn(ProductSupplier::getPhone).setHeader("Phone");
```

```
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
        super.configureGrid(grid);
    }

}
```

Κλάση ProductView

```
package com.example.application.ui.views;

import com.example.application.backend.data.entities.Product;
import com.example.application.backend.service.*;
import com.example.application.backend.util.StringUtil;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.ProductForm;
import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.data.provider.QuerySortOrder;
import com.vaadin.flow.data.renderer.LitRenderer;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;
import java.util.stream.Stream;

@Route(value = "", layout = MainLayout.class)
@PageTitle("Products | Productive Manager")
@PermitAll
public class ProductView extends AbstractCrudView<Product> {

      public ProductView(CategoryService categoryService, TaxService taxService,
ProductService productService, ImageService imageService, ProductSupplierService
productSupplierService) {
                    super("Product", new ProductForm(categoryService::findAll,
taxService::findAll, imageService, productSupplierService::findAll),
productService, Product::new);
    }

    @Override
```

```
   public void configureGrid() {

       Grid<Product> grid = new Grid<>(Product.class);

       grid.addClassNames("product-grid");

       grid.setSizeFull();

       grid.setColumns();

       grid.addColumn(Product::getId).setHeader("Id");

       grid.addComponentColumn(product -> {

           String url = product.getImageUrl();

             Image image = new Image(url != null? url : "" , product.getName() + "
image is missing");

//             image.setWidth(90, Unit.PIXELS);

           image.setHeight(150, Unit.PIXELS);

           return image;

       }).setHeader("Image");

       grid.addColumn(Product::getBarcode).setHeader("Barcode");

       grid.addColumn(Product::getName).setHeader("Name");

       grid.addColumn(Product::getQuantity).setHeader("Quantity");

                                                   grid.addColumn(product        ->
StringUtil.formatPrice(product.getCost())).setHeader("Cost");

                                                   grid.addColumn(product        ->
StringUtil.formatPrice(product.getPrice())).setHeader("Price");

                   grid.addColumn(category   ->   category.getCategory()   !=   null   ?
category.getCategory().getName() : "")

               .setHeader("Category")

                                       .setSortOrderProvider(category   ->   Stream.of(new
QuerySortOrder("name", category)));

                       grid.addColumn(product    ->    product.getTax()    !=    null    ?
product.getTax().getName() : "")

               .setHeader("Tax")

                                       .setSortOrderProvider(product   ->   Stream.of(new
QuerySortOrder("name", product)));

                    grid.addColumn(product   ->   product.getSupplier()   !=   null   ?
product.getSupplier().getName() : "")

               .setHeader("Supplier")

                                       .setSortOrderProvider(product   ->   Stream.of(new
QuerySortOrder("name", product)));

       LitRenderer<Product> enabledRenderer = LitRenderer.<Product>of(

                               "<vaadin-icon icon='vaadin:${item.icon}' style='width:
var(--lumo-icon-size-s);        height:        var(--lumo-icon-size-s);        color:
${item.color};'></vaadin-icon>")

                    .withProperty("icon", enabled -> enabled.isEnabled() ? "check" :
"minus").withProperty("color",
```

```
                        enabled -> enabled.isEnabled()
                                ? "var(--lumo-primary-text-color)"
                                : "var(--lumo-disabled-text-color)");

        grid.addColumn(enabledRenderer).setHeader("Enabled").setAutoWidth(true);
        super.configureGrid(grid);
    }
}
```

Κλάση ReportProductView

```
package com.example.application.ui.views;




import com.example.application.backend.data.entities.Customer;
import com.example.application.backend.data.entities.Product;
import com.example.application.backend.resources.ProductPerformanceResource;
import com.example.application.backend.service.CustomerService;
import com.example.application.backend.service.ReportService;
import com.example.application.backend.util.StringUtil;
import com.example.application.ui.MainLayout;
import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.combobox.ComboBox;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;

@Route(value = "reports/product", layout = MainLayout.class)
@PageTitle("Statistics | Productive Manager")
@PermitAll
public class ReportProductView extends VerticalLayout {
    private ComboBox<Customer> selectCustomer = new ComboBox<>("Customer");
```

```
   private Button clearSelectCustomer = new Button("Clear");

               private     Grid<ProductPerformanceResource>     grid     =     new
Grid<>(ProductPerformanceResource.class);


   private Span totalOrderSpan = new Span();

   private Span totalRevenueSpan = new Span();

   private Span totalCustomersSpan = new Span();

   private Span averageEarningSpan = new Span();

   private ReportService reportService;

         public   ReportProductView(ReportService   reportService,   CustomerService
customerService) {

       this.reportService = reportService;

       setSizeFull();

       grid.addClassNames("product-grid");

       grid.setColumns();

                                    grid.addColumn(productPerformanceResource     ->
productPerformanceResource.getProduct().getName()).setHeader("Name");

       grid.addComponentColumn(productPerformanceResource -> {

           Product product = productPerformanceResource.getProduct();

           String url = product.getImageUrl();

             Image image = new Image(url != null? url : "" , product.getName() + "
image              is missing");

//            image.setWidth(90, Unit.PIXELS);

           image.setHeight(150, Unit.PIXELS);

           return image;

       }).setHeader("Image");

       grid.addColumn(ProductPerformanceResource::getCount).setHeader("Count");

                                    grid.addColumn(productPerformanceResource     ->
StringUtil.formatPrice(productPerformanceResource.getTotalEarnings())).setHeader("T
otalEarnings");

       grid.getColumns().forEach(col -> col.setAutoWidth(true));

       grid.setWidth(100, Unit.PERCENTAGE);

       grid.setHeight(50, Unit.PERCENTAGE);

       grid.setItems(reportService.getProductsPerformance());

       VerticalLayout statistics = new VerticalLayout();

       statistics.add(totalOrderSpan);

       statistics.add(totalCustomersSpan);

       statistics.add(totalRevenueSpan);

       statistics.add(averageEarningSpan);

       setupLabels(null);
```

```
        statistics.add(selectCustomer);
        statistics.add(clearSelectCustomer);
        add(statistics);
        add(grid);

        selectCustomer.setItems(customerService.findAll());
        selectCustomer.setItemLabelGenerator(Customer::getName);
        selectCustomer.addValueChangeListener(event -> {
            Customer customer = event.getValue();
            if (customer != null) {
                grid.setItems(reportService.getProductsPerformance(customer));
                setupLabels(customer);
            } else {
                grid.setItems(reportService.getProductsPerformance());
            }
        });
        clearSelectCustomer.addClickListener(e -> {
            selectCustomer.clear();
            grid.setItems(reportService.getProductsPerformance());
            setupLabels(null);
        });
    }

    private void setupLabels(Customer customer) {
                            totalCustomersSpan.setText("Total    Customers:    "    +
reportService.getTotalCustomers());
                                totalOrderSpan.setText("Total    Orders:    "    +
reportService.getTotalOrdersCount(customer));
                               totalRevenueSpan.setText("Total    Revenue:    "    +
StringUtil.formatPrice(reportService.getTotalEarnings(customer)));
                     averageEarningSpan.setText("Avg   Earnings   per   Order:   "   +
StringUtil.formatPrice(reportService.getAvgEarningPerOrder(customer)));
    }
}
```

Κλάση TaxView

```
package com.example.application.ui.views;
```

```
import com.example.application.backend.data.entities.Tax;
import com.example.application.backend.service.TaxService;
import com.example.application.ui.MainLayout;
import com.example.application.ui.crud.AbstractCrudView;
import com.example.application.ui.views.forms.TaxForm;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;

import javax.annotation.security.PermitAll;

@Route(value = "inventory/taxes", layout = MainLayout.class)
@PageTitle("Taxes | Productive Manager")
@PermitAll
public class TaxView extends AbstractCrudView<Tax> {
   public TaxView(TaxService taxService) {
       super("Tax", new TaxForm(), taxService, Tax::new);
   }

   @Override
   public void configureGrid() {
       Grid<Tax> grid = new Grid<>(Tax.class);
       grid.addClassNames("product-grid");
       grid.setSizeFull();
       grid.setColumns();
       grid.addColumn(Tax::getId).setHeader("Id");
       grid.addColumn(Tax::getName).setHeader("Name");
       grid.addColumn(Tax::getValue).setHeader("Value");
       grid.getColumns().forEach(col -> col.setAutoWidth(true));
       super.configureGrid(grid);
   }

}
```

Κλάση MainLayout

```
package com.example.application.ui;

import com.example.application.security.SecurityService;
```

```
import com.example.application.ui.views.*;


import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.HasStyle;
import com.vaadin.flow.component.Text;
import com.vaadin.flow.component.Unit;
import com.vaadin.flow.component.accordion.Accordion;
import com.vaadin.flow.component.accordion.AccordionPanel;
import com.vaadin.flow.component.applayout.AppLayout;
import com.vaadin.flow.component.applayout.DrawerToggle;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.icon.VaadinIcon;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.HighlightConditions;
import com.vaadin.flow.router.RouterLink;
import org.springframework.beans.factory.annotation.Autowired;


public class MainLayout extends AppLayout {


   private SecurityService securityService;
   public MainLayout(@Autowired SecurityService securityService) {
       this.securityService = securityService;
       createHeader();
       VerticalLayout profileWrapper = new VerticalLayout();


       Image profileImage = new Image("images/avatar.png", "alt");
           String employeeName = securityService.getAuthenticatedUser() != null ?
securityService.getAuthenticatedUser().getName() : "none";


       Text text = new Text("Welcome, " + employeeName);
             profileWrapper.add(profileImage, text, new Button("Log out", e ->
securityService.logout()));
       profileWrapper.setAlignItems(FlexComponent.Alignment.CENTER);
```

```
        addToDrawer(profileWrapper);
        setupAccordionContent();
    }

    private void createHeader() {
        H1 logo = new H1("Productive Manager");
        logo.getStyle().set("white-space", "nowrap");
        logo.addClassNames("text-l", "m-m");
        HorizontalLayout headerLogo = new HorizontalLayout();
        headerLogo.setWidth("100%");
        headerLogo.setJustifyContentMode(FlexComponent.JustifyContentMode.END);

          HorizontalLayout header = new HorizontalLayout(new DrawerToggle(), logo,
headerLogo);

        header.setDefaultVerticalComponentAlignment(FlexComponent.Alignment.CENTER);
        header.setWidth("100%");
        header.addClassNames("py-0", "px-m");

        addToNavbar(header);

    }

    private Component createDrawer(String viewName, VaadinIcon viewIcon, Class<?
extends Component> viewClass) {
        Icon icon = viewIcon.create();
        RouterLink link = new RouterLink();
            HorizontalLayout horizontalLayout = new HorizontalLayout(icon, new
Span(viewName));
horizontalLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.BETWEEN);
        horizontalLayout.setAlignItems(FlexComponent.Alignment.CENTER);
        link.add(temporaryCSSForLinks(horizontalLayout));
        link.setRoute(viewClass);
        link.setHighlightCondition(HighlightConditions.sameLocation());
        return new VerticalLayout(link);
    }

    private AccordionPanel AccordionPanelInventory() {
        Image image = new Image("images/inventory.png", "inventory");
```

```
        image.setHeight(50, Unit.PIXELS);
        image.setWidth(50, Unit.PIXELS);

                    HorizontalLayout logoName = new HorizontalLayout(image, new
Span("Inventory"));
        logoName.setAlignItems(FlexComponent.Alignment.CENTER);

        return new AccordionPanel(temporaryCSSForLinks(logoName), createContent(
                createDrawer("Products", VaadinIcon.PACKAGE, ProductView.class),
                                    createDrawer("Categories", VaadinIcon.CALC_BOOK,
CategoryView.class),
                createDrawer("Taxes", VaadinIcon.ABACUS, TaxView.class),
                                        createDrawer("Suppliers", VaadinIcon.CUBE,
ProductSupplierView.class))
        );
    }

    private AccordionPanel accordionPanelOrdering() {
        Image image = new Image("images/ordering.png", "Orders");
        image.setHeight(50, Unit.PIXELS);
        image.setWidth(50, Unit.PIXELS);

                    HorizontalLayout logoName = new HorizontalLayout(image, new
Span("Ordering"));
        logoName.setAlignItems(FlexComponent.Alignment.CENTER);

        return new AccordionPanel(temporaryCSSForLinks(logoName), createContent(
                createDrawer("Orders", VaadinIcon.CART, OrderView.class),
                createDrawer("Customers", VaadinIcon.USER, CustomerView.class)
        ));
    }

    private AccordionPanel accordionPanelReports() {
        Image image = new Image("images/reports.png", "Audits");
        image.setHeight(50, Unit.PIXELS);
        image.setWidth(50, Unit.PIXELS);

                    HorizontalLayout logoName = new HorizontalLayout(image, new
Span("Reports"));
        logoName.setAlignItems(FlexComponent.Alignment.CENTER);
```

```
        return new AccordionPanel(temporaryCSSForLinks(logoName), createContent(
                createDrawer("Audits", VaadinIcon.RECORDS, AuditView.class),
                                        createDrawer("Statistics",   VaadinIcon.CHART,
ReportProductView.class),
                createDrawer("Employees", VaadinIcon.USER, EmployeeView.class),
                createDrawer("Settings", VaadinIcon.OPTIONS, PolicyView.class)
        ));
    }

    private void setupAccordionContent() {
        Accordion accordion = new Accordion();
        accordion.getStyle().set("padding-left", "10px");
        accordion.add(AccordionPanelInventory());
        accordion.add(accordionPanelOrdering());
        if (securityService.isAdmin()) {
            accordion.add(accordionPanelReports());
        }
        addToDrawer(accordion);
    }

    private VerticalLayout createContent(Component ...components) {
        VerticalLayout content = new VerticalLayout();
        content.add(components);
        return content;
    }

    private Component temporaryCSSForLinks(HasStyle component) {
        component.getStyle().set("user-select", "none");
        component.getStyle().set("user-drag", "none");
        component.getStyle().set("cursor", "pointer");
        return (Component) component;
        //To be removed and refactored into a css class in the themes CSS
    }

}
```

Κλάση Application

```
package com.example.application;
```

```
import com.vaadin.flow.component.dependency.NpmPackage;
import com.vaadin.flow.component.page.AppShellConfigurator;
import com.vaadin.flow.server.PWA;
import com.vaadin.flow.theme.Theme;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
@Theme(value = "flowcrmtutorial")
@PWA(name = "Flow CRM Tutorial", shortName = "Flow CRM Tutorial", offlineResources
= {})
@NpmPackage(value = "line-awesome", version = "1.3.0")
public class Application extends SpringBootServletInitializer implements
AppShellConfigurator {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

Dependencies - pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
                             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <!-- Project from https://start.vaadin.com/ -->
    <groupId>com.example.application</groupId>
    <artifactId>flowcrmtutorial</artifactId>
    <name>Flow CRM Tutorial</name>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <java.version>11</java.version>
        <vaadin.version>23.0.8</vaadin.version>
```

```
    </properties>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.7</version>
    </parent>

    <repositories>
        <!-- The order of definitions matters. Explicitly defining central here to
make sure it has the highest priority. -->

        <!-- Main Maven repository -->
        <repository>
            <id>central</id>
            <url>https://repo.maven.apache.org/maven2</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>vaadin-prereleases</id>
            <url>
                https://maven.vaadin.com/vaadin-prereleases/
            </url>
        </repository>
        <!-- Repository used by many Vaadin add-ons -->
        <repository>
            <id>Vaadin Directory</id>
            <url>https://maven.vaadin.com/vaadin-addons</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

    <pluginRepositories>
        <!-- The order of definitions matters. Explicitly defining central here to
make sure it has the highest priority. -->
```

```
    <pluginRepository>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
    <pluginRepository>
        <id>vaadin-prereleases</id>
        <url>
            https://maven.vaadin.com/vaadin-prereleases/
        </url>
    </pluginRepository>
</pluginRepositories>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-bom</artifactId>
            <version>${vaadin.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>com.vaadin</groupId>
        <!-- Replace artifactId with vaadin-core to use only free components -->
        <artifactId>vaadin</artifactId>
    </dependency>
    <dependency>
        <groupId>com.vaadin</groupId>
        <artifactId>vaadin-spring-boot-starter</artifactId>
    </dependency>
    <dependency>
```

```
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>

    <dependency>
        <groupId>com.vaadin</groupId>
        <artifactId>exampledata</artifactId>
        <version>4.1.4</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.vaadin</groupId>
        <artifactId>vaadin-testbench</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
```

```
        <version>2.2</version>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.24</version>
        <scope>provided</scope>
    </dependency>
    <!-- Include JUnit 4 support for TestBench and others -->
    <dependency>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.hamcrest</groupId>
                <artifactId>hamcrest-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>5.0.3</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.google.code.findbugs</groupId>
        <artifactId>jsr305</artifactId>
        <version>3.0.2</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
```

```
            <version>5.6.1</version>
        </dependency>
    </dependencies>

    <build>
        <defaultGoal>spring-boot:run</defaultGoal>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                    <!-- Clean build and startup time for Vaadin apps sometimes may
exceed
                     the default Spring Boot's 30sec timeout.  -->
                <configuration>
                    <wait>500</wait>
                    <maxAttempts>240</maxAttempts>
                </configuration>
            </plugin>

            <!--
                Take care of synchronizing java dependencies and imports in
                package.json and main.js files.
                It also creates webpack.config.js if not exists yet.
            -->
            <plugin>
                <groupId>com.vaadin</groupId>
                <artifactId>vaadin-maven-plugin</artifactId>
                <version>${vaadin.version}</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>prepare-frontend</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

```
<profiles>
    <profile>
        <!-- Production mode is activated using -Pproduction -->
        <id>production</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>com.vaadin</groupId>
                    <artifactId>vaadin-maven-plugin</artifactId>
                    <version>${vaadin.version}</version>
                    <executions>
                        <execution>
                            <goals>
                                <goal>build-frontend</goal>
                            </goals>
                            <phase>compile</phase>
                        </execution>
                    </executions>
                    <configuration>
                        <productionMode>true</productionMode>
                    </configuration>
                </plugin>
            </plugins>
        </build>
    </profile>

    <profile>
        <id>it</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                    <executions>
                        <execution>
                            <id>start-spring-boot</id>
                            <phase>pre-integration-test</phase>
                            <goals>
```

```
                                <goal>start</goal>
                            </goals>
                        </execution>
                        <execution>
                            <id>stop-spring-boot</id>
                            <phase>post-integration-test</phase>
                            <goals>
                                <goal>stop</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>

                    <!-- Runs the integration tests (*IT) after the server is
started -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-failsafe-plugin</artifactId>
                    <executions>
                        <execution>
                            <goals>
                                <goal>integration-test</goal>
                                <goal>verify</goal>
                            </goals>
                        </execution>
                    </executions>
                    <configuration>
                        <trimStackTrace>false</trimStackTrace>
                        <enableAssertions>true</enableAssertions>
                    </configuration>
                </plugin>
            </plugins>
        </build>
    </profile>

  </profiles>
</project>
```

Springboot - application.properties

```
server.port=${PORT:8080}

logging.level.org.atmosphere = warn

spring.mustache.check-template-location = false

vaadin.launch-browser=true

vaadin.whitelisted-packages                                                    =
com.vaadin,org.vaadin,dev.hilla,com.example.application

application.title= ProductiveManager

application.version= Alpha_1
```