ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΜΕ ΧΡΗΣΗ JAVA SPRING

Της φοιτήτριας Καββαδά Αικατερίνη
Αρ. Μητρώου: 144327

Επιβλέπων
ΙΓΝΑΤΙΟΣ ΔΕΛΗΓΙΑΝΝΗΣ
Καθηγητής

Ημερομηνία 20/01/2023

Τίτλος Δ.Ε.: ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΜΕ ΧΡΗΣΗ JAVA SPRING
Κωδικός Δ.Ε.: 21200
Ονοματεπώνυμο φοιτητή: Καββαδά Αικατερίνη
Ονοματεπώνυμο εισηγητή: Δεληγιάννης Ιγνάτιος
Ημερομηνία ανάληψης Δ.Ε.: 18/03/2021
Ημερομηνία περάτωσης Δ.Ε.: 18/06/2022

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

Η εργασία είναι αφιερωμένη στην οικογένεια μου.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Δεληγιάννη Ιγνάτιο για την ανάθεση της διπλωματικής εργασίας, τη πολύτιμη βοήθεια και σαφή καθοδήγησή κατά την εκπόνησή της. Ακόμη, να ευχαριστήσω τους συμφοιτητές μου από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων (ΔΙΠΑΕ) για τις γνώσεις και τις συμβουλές που μοιράστηκαν κατά τη συγγραφή της εργασίας. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, για την υπομονή και την υποστήριξή κατά τη διάρκεια της ακαδημαϊκής μου σταδιοδρομίας.

Περιεχόμενα

Πρόλογος

Οι βασικοί λόγοι που επιλέχθηκε η διπλωματική αυτή εργασία είναι για να δείξουμε πως τα πληροφοριακά συστήματα αποτελούν για το ανθρώπινο δυναμικό ένα μέσο στη διευκόλυνση διαχείρισης επιχειρήσεων και γενικότερα των επιχειρηματικών διαδικασιών. Συνδυάζουν την επιστήμη των υπολογιστών με τον επιχειρηματικό κόσμο.

«**Ορισμός Πληροφοριακού Συστήματος** στην πιο γενική του έννοια, ένα σύστημα είναι ένα σύνολο συνιστωσών που αλληλοεπιδρούν μεταξύ τους για να επιτύχουν κάποιο σκοπό. Ένας οργανισμός ή επιχείρηση μπορεί να θεωρηθεί ότι αποτελεί ένα σύστημα. Κάθε σύστημα υπάρχει, γιατί έχει ένα σκοπό. Για να επιτύχει τους σκοπούς τους, το σύστημα αλληλοεπιδρά με το περιβάλλον του, δηλαδή με κάθε οντότητα που βρίσκεται έξω από τα όριά του. Μια επιχείρηση μπορεί να θεωρηθεί και αυτή ως ένα σύστημα. Το σύστημα αυτό αποτελείται από κάποια συστατικά μέρη, όπως: πωλήσεις, παραγωγή, λογιστήριο, προσωπικό, μάρκετινγκ κλπ. Οι επιχειρήσεις/οργανισμοί μπορούν να είναι διαφοροποιημένοι σε ενότητες με συγκεκριμένους σκοπούς για να υπηρετούν συγκεκριμένες δραστηριότητες μέσα σε μια εργασιακή ενότητα. Σε γενικές γραμμές, μπορεί κανείς να διακρίνει τρία επίπεδα οργανωτικής δομής: το ανώτερο, το μεσαίο και το κατώτατο. Οι ανήκοντες στο πρώτο σχεδιάζουν στρατηγικές και λαμβάνουν αποφάσεις. Το δεύτερο αφορά τα μεσαία διοικητικά στελέχη που αναλαμβάνουν να εφαρμόσουν τις στρατηγικές της και των αποφάσεων της διοίκησης της επιχείρησης/οργανισμού. Στο ίδιο επίπεδο εντάσσονται και οι δημιουργοί γνώσης όπως π.χ. ερευνητές κ.ά. Τέλος, το τρίτο επίπεδο περιλαμβάνει το κατώτατο διοικητικό προσωπικό και τους εργαζόμενους στην παραγωγή η την παροχή υπηρεσιών ανάλογα με το αντικείμενο ενασχόλησης της επιχείρησης/οργανισμού.»

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός διαδικτυακού πληροφοριακού συστήματος διαχείρισης αποθήκης ειδών ιματισμού. Ο Διαχειριστής κεντρικής αποθήκης θα έχει την δυνατότητα άμεσης εποπτείας αποθεμάτων υλικών σε πραγματικό χρόνο, τα οποία και θα τροποποιούνται ανάλογα με τις παραγγελίες που θα δέχεται από τους διαχειριστές των δευτερευόντων αποθηκών.

Περίληψη

Η παρούσα πτυχιακή, θα παρουσιαστεί μια διαδικτυακή εφαρμογή Διαχείρισης αποθηκών ειδών ιματισμού. Μέσα από την εφαρμογή, ο διαχειριστής της κεντρικής αποθήκης θα δύναται να εποπτεύει σε πραγματικό χρόνο τα διαθέσιμα αποθέματα των ειδών που επιθυμεί, τα οποία τροποποιούνται με την αποστολή των ειδών στις δευτερεύουσες αποθήκες κατά παραγγελία από τις τελευταίες. Το σύστημα κατασκευάστηκε χρησιμοποιώντας την τεχνολογία λογισμικού Spring Boot. Αρχικά, επεξηγείται η αρχιτεκτονική της χρησιμοποιούμενης τεχνολογίας (Spring Boot) και ακολουθεί εγχειρίδιο για την εκτέλεση της εφαρμογής με παραδείγματα χρήσης. Στο τέλος, αναφέρονται μελλοντικές επεκτάσεις της εφαρμογής.

DESIGN AND IMPLEMENTATION OF AN INFORMATION SYSTEM USING JAVA SPRING

Kavvada Aikaterini

Abstract

This dissertation introduces an online warehouse management application for clothing. Using the produced application, the central warehouse manager will be able to monitor, in real time, his item stocks, which are modified by sending the items to the secondary warehouses by order. The objective would be detailing about architecture of the technology used (Spring Boot), walking users through specifics providing them with use cases in the User manual, and providing a roadmap for future updates.

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

ΕΙΣΑΓΩΓΗ

Γενικά, θα εξετάσουμε έννοιες που σχετίζονται με συστήματα βάσεων δεδομένων, πληροφοριακά συστήματα και τις εφαρμογές τους και μοντελοποίηση δεδομένων με ευρείες περιγραφές Μοντέλων Οντοτήτων-Συσχετίσεων και των Σχεσιακών μοντέλων και θα αναλύσουμε τον κύκλο ζωής των συστημάτων βάσεων δεδομένων και πληροφοριακών συστημάτων, με ανάπτυξη και σχεδιασμό σε φάσεις. Θα αναλυθούν επίσης οι απαιτήσεις και οι ανάγκες των πληροφοριακών συστημάτων καθώς και των συστημάτων βάσεων δεδομένων. Θα δώσουμε ιδιαίτερη προσοχή στις πρώτες φάσεις σχεδιασμού πληροφοριακών συστημάτων: διερευνητικές μελέτες και μελέτες σκοπιμότητας. Ακολουθεί το τρίτο στάδιο σχεδιασμού, ανάλυση απαιτήσεων, όπου καθορίζονται οι προδιαγραφές σχεδιασμού για το νέο σύστημα και στη συνέχεια αναπτύσσονται η σχεδιαστική προσέγγιση και οι ιδέες υλοποίησης. Η εργασία περιλαμβάνει σχεδιασμό βάσης δεδομένων για την υποστήριξη ενός νέου πληροφοριακού συστήματος, του οποίου θα γίνει σε τρία στάδια, εννοιολογικό, λογικό και φυσικό σχεδιασμό. Σε τρεις φάσεις σχεδιασμού, θα αναπτυχθεί ένα εννοιολογικό μοντέλο, το οποίο στη συνέχεια θα γίνει λογικό μοντέλο και τελικά τελικό προϊόν για τον φυσικό σχεδιασμό και την υλοποίηση της βάσης δεδομένων. Θα παρουσιαστούν διαγράμματα μοντέλων και ο τρόπος κατασκευής τους μέσω της οργάνωσης των αποθηκευμένων δεδομένων. Θα αναλυθεί επίσης ο σχεδιασμός του νέου συστήματος πληροφορικής, το οποίο είναι η τέταρτη φάση της διαδικασίας σχεδιασμού πριν από την εφαρμογή. Εδώ θα αναπτύξουμε τα απαραίτητα διαγράμματα UML όπως περιπτώσεις χρήσης, διαγράμματα ακολουθίας κ.λπ. Παρακάτω λοιπόν, περιγράφηκε επίσης και αναλύθηκε διεξοδικά η αρχιτεκτονική του συστήματος πληροφορικής, στην οποία χωρίστηκε σε επιμέρους υποσυστήματα. Επιπλέον, εξετάζονται διάφορες ομάδες χρηστών του συστήματος με βάση τους ρόλους τους. Τέλος, παρουσιάζονται οι περιπτώσεις χρήσης του συστήματος από μεμονωμένες ομάδες χρηστών, οι οποίες απεικονίζονται με διαγράμματα ακολουθίας και φυσικά υλοποιείται το σύστημα.

Κεφάλαιο 1ο: SCRUM

Η Scrum είναι μια επαναληπτική και αυξητική μεθοδολογία ανάπτυξης έργων. Χρησιμοποιείται σαν μια μεθοδολογία ανάπτυξης λογισμικού, όσον αφορά την οργάνωση και λειτουργία των ομάδων ανάπτυξης αλλά και στον σχεδιασμό και διαχείριση των ευέλικτων έργων. Η επιτυχία της μεθόδου οφείλεται στον τρόπο με τον οποίο προσεγγίζει και διαχειρίζεται την έννοια της διαδικασίας ανάπτυξης του λογισμικού. Προσδίδοντας ευελιξία στην οργάνωση και διαχείριση της διαδικασίας ανάπτυξης λογισμικού, αυξάνεται η παραγωγικότητα της ομάδας αλλά και η ικανότητα προσαρμογής της σύμφωνα με τις εκάστοτε ανάγκες που προκύπτουν. Δίνεται μεγάλη έμφαση στο πώς θα πρέπει να οργανωθεί η ομάδα ώστε να μπορέσει να επιτύχει το μέγιστο βαθμό αποτελεσματικότητας, σε ένα διαρκώς μεταβαλλόμενο περιβάλλον. Η μέθοδος προβλέπει ένα εντελώς διαφορετικό μοτίβο οργάνωσης και διοίκησης του έργου σε σχέση με τις υπόλοιπες μεθόδους. Πιο συγκεκριμένα, αυτή η μέθοδος στοχεύει να διατηρήσει τον χρόνο του κύκλου ανάπτυξης όσο το δυνατόν συντομότερο και να παραδώσει τα μέρη του κώδικα συστήματος που έχουν συμφωνηθεί για τον κύκλο. Πολύ σημαντικό, όπου χρήζει να αναφερθεί είναι η καθημερινή επαφή όλης της ομάδας ανάπτυξης με κύριο σκοπό την όσο το δυνατόν καλής συνεργασίας και τον σωστό συντονισμό των εργασιών τους.

1.1 Φάσεις μεθοδολογίας

Η μεθοδολογία Scrum αποτελείται από τρεις φάσεις: pregame (συμπεριλαμβανομένων των επιμέρους φάσεων σχεδιασμού και αρχιτεκτονικής του συστήματος), game (συμπεριλαμβανομένης της φάσης ανάπτυξης) και postgame (συμπεριλαμβανομένης της φάσης ολοκλήρωσης). Κατά τη φάση του σχεδιασμού, εκτιμάται ο χρόνος ολοκλήρωσης και το συνολικό κόστος της εφαρμογής. Εάν αναπτύσσεται ένα νέο σύστημα, η φάση σχεδιασμού περιλαμβάνει επίσης μια φάση ανάλυσης. Όσον αφορά την αρχιτεκτονική του συστήματος, η φάση του σχεδιασμού εξετάζει τον τρόπο υλοποίησης όσων μερών παραμένουν ημιτελή. Η φάση αυτή περιλαμβάνει επίσης τροποποιήσεις της αρχιτεκτονικής του συστήματος και του υψηλού επιπέδου σχεδιασμού του συστήματος, εάν είναι απαραίτητο. Κατά την εφαρμογή ενός συστήματος, πολλές μεταβλητές, όπως οι απαιτήσεις, η ποιότητα, το κόστος, ο ανταγωνισμός και ο χρόνος διεκπεραίωσης πρέπει να λαμβάνονται υπόψη και να αντιμετωπίζονται. Η αλληλεπίδραση με αυτές τις μεταβλητές καθορίζει επίσης το τέλος αυτής της φάσης μετά από μια σειρά επαναλήψεων sprint για την εξέλιξη του συστήματος. Τέλος, η φάση ολοκλήρωσης προετοιμάζει το σύστημα για παράδοση, τεκμηρίωση και εκτέλεση δοκιμών. Κάθε ένα από τα παραπάνω περιλαμβάνει τα ακόλουθα βήματα.

Α. Σχεδιασμός.

- Προετοιμάζεται ένας κατανοητός κατάλογος του εναπομένοντος εκκρεμούς συστήματος.
- Καθορίζεται η ημερομηνία παράδοσης και η λειτουργικότητα της εφαρμογής.
- Προσδιορίζεται η ομάδα έργου.
- Προσδιορίζονται οι κατάλληλοι έλεγχοι και αξιολογήσεις κινδύνου.
- Επανεξετάζεται και προσαρμόζεται το ανεκτέλεστο.
- Επιλέγονται τα εργαλεία ανάπτυξης.
- Υπολογίζεται το συνολικό κόστους.
- Επαληθεύεται από τη διοίκηση για έγκριση και χρηματοδότηση

Β. Αρχιτεκτονική

-Ανασκόπηση στο ανεκτέλεστο υπόλοιπο που έχει ανατεθεί.
-Καθορίζονται οι αλλαγές που πρέπει να γίνουν στο backlog.
-Προσδιορίζεται η κατάλληλη αρχιτεκτονική για την υποστήριξη των οδηγιών του συστήματος.
-Εντοπίζονται προβλήματα της εφαρμογής και εφαρμόζονται οι απαραίτητες αλλαγές.
-Σχεδιασμός της συνάντησης ανασκόπησης.

Γ. Υλοποίηση

Η φάση εκτέλεσης είναι ένας επαναληπτικός κύκλος δραστηριοτήτων. Η διοίκηση του οργανισμού καθορίζει την ποιότητα και τη λειτουργικότητα της υλοποίησης και η ομάδα ανάπτυξης επαναλαμβάνει αρκετές επαναλήψεις για να ολοκληρώσει το σύστημα. Τα βήματα σε αυτή τη φάση περιλαμβάνουν:

-Συναντήσεις με άλλες ομάδες για την εξέταση των ολοκληρωμένων τμημάτων της αίτησης.
-Συναντήσεις με άλλα μέλη της ομάδας έργου για να επανεξεταστεί και να συζητηθεί η υλοποίηση του έργου.
-Εκτέλεση των sprints μέχρι το προϊόν να κατασκευαστεί και να είναι έτοιμο για παράδοση.

Κάθε sprint (επανάληψη) ολοκληρώνεται μέσω διαφόρων φάσεων ανάπτυξης. Καθορίζει τις αλλαγές, την ανάλυση, τη δοκιμή και την τεκμηρίωση των αλλαγών που απαιτούνται για την υλοποίηση των ημιτελών απαιτήσεων. Το στάδιο περιτύλιξης δημιουργεί το τμήμα εκτέλεσης της εφαρμογής, το στάδιο της αναθεώρησης όπου όλες οι ομάδες συναντώνται για να συζητήσουν την πρόοδο που έχει επιτευχθεί και να ενημερώσουν την ισορροπία του ημιτελούς συστήματος. Τέλος, το στάδιο της προσαρμογής συγκεντρώνει όλες τις πληροφορίες από τις συναντήσεις, συμπεριλαμβανομένων των αλλαγών και των νέων χαρακτηριστικών που προέκυψαν μετά τη συνάντηση κάθε ομάδας.

Δ. Ολοκλήρωση

Μια εφαρμογή κηρύσσεται "κλειστή" όταν η ομάδα αποφασίσει ότι μεταβλητές όπως ο χρόνος, το κόστος, ο ανταγωνισμός, οι απαιτήσεις και οι προδιαγραφές είναι πλήρεις, πράγμα που σημαίνει ότι έχουν εφαρμοστεί όλες οι λειτουργίες που απαιτούνται για τη δημιουργία μιας νέας έκδοσης λογισμικού. Κατά τη διάρκεια αυτού του σταδίου, το παραδοτέο προϊόν προετοιμάζεται για την τελική κυκλοφορία. Η φάση ολοκλήρωσης περιλαμβάνει τα ακόλουθα:

-Δοκιμή του συστήματος.
-Τεκμηρίωση.
-Προετοιμασία του προϊόντος για εμπορική διάθεση.
-Διανομή.

1.2 Ρόλοι και Ευθύνες

Στην ανάπτυξη έργων που χρησιμοποιούν τη μεθοδολογία Agile SCRUM συναντάμε διαφόρους ρόλους:

- Scrum Manager ή Scrum Specialist (Scrum Master).

Ο SCRUM Manager ευθύνεται για τη σωστή ανάπτυξη ανά πάσα στιγμή, σύμφωνα με τις μεθόδους, τις αξίες και τους κανόνες που ορίζονται κατά τον σχεδιασμό του έργου. Γι' αυτό λοιπόν, συνεργάζεται τόσο με την ομάδα ανάπτυξης όσο και με τον πελάτη κατά την υλοποίηση του έργου. Ακόμη, υπεύθυνος για την εξάλειψη διαφόρων εμποδίων κατά το έργο, ώστε να διασφαλίζεται ότι η ομάδα ανάπτυξης διατηρεί την υψηλότερη δυνατή παραγωγικότητα.

- Ιδιοκτήτης ή Διευθυντής προϊόντος (Product Owner).

Το πρόσωπο που είναι υπεύθυνο για το πρόγραμμα που αφορά τη διαχείριση, τον έλεγχο και την καταγραφή των απαιτήσεων που αποτελούν το ανεκτέλεστο υπόλοιπο του έργου. Ο Ιδιοκτήτης προϊόντος ή ο διαχειριστής επιλέγεται από τον διαχειριστή, τον εμπειρογνώμονα Scrum, τον πελάτη και την ομάδα διαχείρισης. Είναι οι άνθρωποι που, σε συνεργασία με την ομάδα ανάπτυξης, επιλέγουν ποιες απαιτήσεις υλοποιούνται με ποια σειρά για κάθε επανάληψη. Συμμετέχουν επίσης ενεργά στην εκτίμηση της προσπάθειας (χρόνος, εργαλεία κ.λπ.) που χρήζει για την υλοποίηση των απαιτήσεων και την εξεύρεση λύσεων για το χρονικό διάστημα της ανάπτυξης.

- Ομάδα Scrum.

Η ομάδα Scrum είναι υπεύθυνη για την ανάπτυξη του έργου και έχει την εξουσία να καθορίζει και να εφαρμόζει τις σωστές διαδικασίες και εργασίες που απαιτούνται για την υλοποίησή του. Η ομάδα θα πρέπει να είναι σε θέση να αναδιοργανώνει αποτελεσματικά τις λειτουργίες της για την επίτευξη των προκαθορισμένων στόχων κάθε επανάληψης (αυτοδιαχειριζόμενη ομάδα). Η ομάδα ανάπτυξης είναι υπεύθυνη για την εκτίμηση του χρόνου της επανάληψης, των καθυστερημένων τμημάτων και τον έλεγχο του

καταλόγου καθυστερήσεων του έργου. Η ομάδα ανάπτυξης είναι επίσης υπεύθυνη για τον εντοπισμό των εμποδίων που πρέπει να εξαλειφθούν από το έργο προκειμένου να διευκολυνθεί η διαδικασία ανάπτυξης.

- Ο πελάτης (Customer)

Οι πελάτες συμμετέχουν ενεργά σε ολόκληρη τη διαδικασία ανάπτυξης λογισμικού, ιδίως στον καθορισμό των απαιτήσεων, στη διαδικασία επιλογής και ανάπτυξης και στον έλεγχο της καλής λειτουργίας του τελικού προϊόντος. Καθώς οι πελάτες είναι συνήθως και οι χρήστες του λογισμικού, εκτός από τη γνώση που έχουν για το υπό ανάπτυξη προϊόν, μπορούν να εξετάζουν άμεσα τα παραγόμενα αποτελέσματα και να ζητούν τροποποιήσεις, εάν είναι απαραίτητο. Η παραγωγικότητα της ομάδας αυξάνεται επειδή ο πελάτης είναι άμεσα διαθέσιμος για να απαντήσει στις ερωτήσεις της ομάδας.

1.3 Χαρακτηριστικά μεθοδολογίας SCRUM

Χαρακτηριστικά που συνθέτουν την μεθοδολογία είναι τα ακόλουθα:

- Κατάλογος ανεκτέλεστων προϊόντων της παραγγελίας (Product Backlog list)

Αυτή είναι μια λίστα με τις επί του παρόντος γνωστές απαιτήσεις συστήματος. Οι απαιτήσεις μπορεί να προέρχονται από πελάτες, τμήματα πωλήσεων και μάρκετινγκ, υποστήριξη πελατών ή προγραμματιστές λογισμικού. Τα αιτήματα ιεραρχούνται και υπολογίζεται η προσπάθεια που απαιτείται για την εκπλήρωσή τους. Η λίστα των απαιτήσεων ενημερώνεται και αναθεωρείται κατά τη διάρκεια κάθε επανάληψης με νέες, πιο λεπτομερείς πληροφορίες, ακριβείς εκτιμήσεις της απαιτούμενης προσπάθειας και νέες προτεραιότητες.

Εκτίμηση της προσπάθειας (Effort Estimation).

Η εκτίμηση των ανθρωποωρών είναι μια επαναληπτική διαδικασία όπου, καθώς καθίστανται διαθέσιμες περισσότερες πληροφορίες, τα στοιχεία του καταλόγου απαιτήσεων μπορούν να εκτιμηθούν με μεγαλύτερη ακρίβεια. Ο διαχειριστής προϊόντος και η ομάδα ανάπτυξης είναι υπεύθυνοι για τη διαδικασία αυτή.

- Επανάληψη (Sprint)

Μια επανάληψη είναι ένας ολοκληρωμένος κύκλος ανάπτυξης που περιέχει τον προγραμματισμό, την ανάλυση και την υλοποίηση συγκεκριμένων απαιτήσεων (sprint backlog). Οι ομάδες ανάπτυξης οργανώνουν τις εργασίες τους για τη δημιουργία μιας νέας, εκτελέσιμης, σταδιακής εφαρμογής σε διάστημα μιας έως τεσσάρων εβδομάδων (30 ημερών). Οι συνεδριάσεις σχεδιασμού του sprint, οι καθημερινές συνεδριάσεις scrum και οι συνεδριάσεις ανασκόπησης του sprint είναι καθοριστικές για τον καθορισμό και την εκτέλεση των στόχων της επανάληψης.

- Συνεδρίαση για το σχεδιασμό της επανάληψης (sprint planning meeting)

Συναντήσεις για τον προγραμματισμό και την υλοποίηση επαναλήψεων που οργανώνονται από ειδικούς του Scrum και διεξάγονται σε δύο φάσεις. Το πρώτο μέρος της σύσκεψης περιλαμβάνει πελάτες, χρήστες, διοίκηση, ιδιοκτήτη προϊόντος και την ομάδα ανάπτυξης για να καθορίσουν τους στόχους και τα χαρακτηριστικά της επόμενης επανάληψης. Στο δεύτερος μέρος, στο οποίο συμμετέχουν μόνο οι ειδικοί του Scrum και η ομάδα ανάπτυξης, πραγματοποιούνται συναντήσεις για να καθοριστεί ο τρόπος με τον οποίο θα πραγματοποιηθεί η σταδιακή ανάπτυξη του προϊόντος κατά τη διάρκεια της επανάληψης.

- Λίστα απαιτήσεων της επανάληψης (sprint backlog)

Αυτή η λίστα απαιτήσεων είναι το σημείο εκκίνησης για κάθε επανάληψη. Η ομάδα ανάπτυξης μαζί με τον

Scrum master και σαφώς τον owner της εφαρμογής, επιλέγουν απαιτήσεις που είναι οι πιο σημαντικές και που τη βοηθούν να επιτύχουν τους στόχους τους για αυτήν την επανάληψη. Αυτές οι απαιτήσεις παραμένουν ίδιες μέχρι να ολοκληρωθεί και όταν πληρούνται όλες οι απαιτήσεις της λίστας, το σύστημα ενημερώνεται και παραδίδεται στον πελάτη.

- Καθημερινή συνεδρίαση Scrum (daily scrum meeting)

Αυτή η σύντομη συνάντηση (διαρκεί περίπου 15 λεπτά) πραγματοποιείται καθημερινά για τον έλεγχο της προόδου και της απόδοσης της ομάδας ανάπτυξης. Η συνάντηση αυτή μπορεί επίσης να περιγραφεί ως συνάντηση προγραμματισμού, καθώς εξετάζει τι έχει γίνει από την τελευταία συνάντηση και τι πρέπει να γίνει μέχρι την επόμενη συνάντηση. Συζητούνται όλα τα προβλήματα, οι ελλείψεις και τα εμπόδια που παρουσιάστηκαν κατά τη διαδικασία ανάπτυξης. Στη συνάντηση συμμετέχουν ο ειδικός Scrum, μέλη της ομάδας ανάπτυξης και η διοίκηση.

- Συνεδρίαση για την επανεξέταση της επανάληψης (sprint review meeting)

Μια άτυπη συνάντηση την τελευταία ημέρα της επανάληψης, κατά την οποία ο ειδικός Scrum και η ομάδα ανάπτυξης παρουσιάζουν τα αποτελέσματα στη διοίκηση, τους πελάτες, τους χρήστες και τους ιδιοκτήτες προϊόντων. Οι συμμετέχοντες αξιολογούν την εξέλιξη του έργου και αποφασίζουν ποια χαρακτηριστικά θα πρέπει να υλοποιηθούν την επόμενη φορά. Η συνάντηση αυτή είναι σημαντική για την πρόοδο του έργου, καθώς μπορεί να σηματοδοτήσει την προσθήκη νέων στοιχείων στο ανεκτέλεστο ή την αλλαγή κατεύθυνσης.

Κεφάλαιο 2ο: Προγραμματιστικό Περιβάλλον

Εισαγωγή

Η εφαρμογή που θα δημιουργήσουμε αποτελεί βασικό στοιχείο της νέας τεχνολογίας και εκτιμάται ότι θα αποτελέσει και το υπόδειγμα για την ανάπτυξη αντίστοιχων εφαρμογών για την χρήση του spring boot στην ανάπτυξη εφαρμογών. Στις επόμενες παραγράφους θα καταγραφούν οι αρχικές ανάγκες οι οποίες θα αποτελέσουν τον κορμό της ανάπτυξης της εφαρμογής και οι οποίες εκτιμάται ότι θα καλύψουν τα απαιτούμενα της εφαρμογής. Θα παρουσιαστούν οι βασικές λειτουργίες της εφαρμογής και οι ανάγκες σε ασφάλεια για την εν λόγω εφαρμογή.

Αξίζει να σημειωθεί ότι για να μπορέσουμε να μιλάμε για μια ολοκληρωμένη εφαρμογή θα πρέπει να επιδιώκουμε την ενσωμάτωση διαφόρων παρεχόμενων εφαρμογών και των δεδομένων τους σε μια κοινή και δια-λειτουργική υπηρεσία όπου διάφορες βάσεις δεδομένων από διάφορα συστήματα και εφαρμογές θα χρησιμοποιούνται από ένα απλό web interface φιλικό προς το χρήστη σε μια υπηρεσία διαδικτύου εύκολης πρόσβασης και χρήσης.

Το πιο σημαντικό στην σχεδίαση ενός προγράμματος είναι τα δεδομένα. Χωρίς την κατάλληλη υποδομή δεδομένων δεν θα είναι εύκολη η υλοποίηση καμιάς εφαρμογής ενώ η απαίτηση για ατομική συλλογή δεδομένων θα ήταν σίγουρα χρονοβόρα.

Το προγραμματιστικό μας περιβάλλον περιέχει JAVA, , Spring Boot Αρχιτεκτονική, Spring Security, Apache Maven.

JAVA

Η Java επιλέχθηκε ως γλώσσα προγραμματισμού λόγω των πλεονεκτημάτων της στην δημιουργία διαδικτυακών εφαρμογών. Ο λόγος είναι ότι μια εφαρμογή java μπορεί να μεταγλωττιστεί μία φορά και στη συνέχεια να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα. Έτσι, μια εφαρμογή Ιστού μπορεί εύκολα να

μεταφερθεί από ένα περιβάλλον διακομιστή σε άλλο.

Πρόκειται για μια γλώσσα προγραμματισμού ανοιχτού κώδικα που παρέχεται από την Oracle. Η γλώσσα αυτή χρησιμοποιείται πλέον ευρέως για την υλοποίηση των περισσοτέρων διαδικτυακών εφαρμογών στο μεγαλύτερο εύρος συσκευών. Υπάρχουν πάρα πολλές βιβλιοθήκες που μπορούν να χρησιμοποιηθούν ελεύθερα ενώ η παροχή βοήθειας με το javadoc κατά την υλοποίηση μιας εφαρμογής είναι ιδιαίτερα επαναστατική. Ο κώδικας εκτελείται μέσω του JVM (Java Virtual Machine). Ο compiler παρέχεται στο πακέτο JDK (Java Development Kit) και παράγει ένα ενδιάμεσο κώδικα, τον Java Bytecode ο οποίος μετασχηματίζεται σε εκτελέσιμη μορφή με τον compiler του JRE (Java Runtime Environment).

Η java αποτελεί μια αντικειμενοστραφή γλώσσα προγραμματισμού όπου όλες οι μεταβλητές της είναι τύπου object στα οποία και γίνονται αναφορές.

Spring Framework

Το Framework είναι ένα τυποποιημένο σύνολο αρχών, πρακτικών και προτύπων για την επίλυση ενός προβλήματος. Στην πράξη, πρόκειται για δομημένα (σε φακέλους) πακέτα κλάσεων που μπορούν να χρησιμοποιηθούν για τη δημιουργία εφαρμογών και χρησιμεύουν για την παροχή μιας κοινής δομής κλάσεων - αρχείων κώδικα σε μια επαναχρησιμοποιήσιμη λογική.
Ως ένα από τα πιο δημοφιλή πλαίσια για την ανάπτυξη εφαρμογών ιστού σε Java, το Spring Framework παρέχει πολλά προγραμματιστικά εργαλεία που απλοποιούν τη δημιουργία κώδικα και δομούν βέλτιστα τις υλοποιήσεις.

Spring Boot αρχιτεκτονική

Μία από τις τεχνολογίες που χρησιμοποιούνται στο Spring Framework είναι η αρχιτεκτονική υλοποίησης του Spring Boot. Χαρακτηρίζεται από τη χρήση του @SpringBootApplication στη μέθοδο main για την αυτόματη διαμόρφωση της εφαρμογής με βάση τις εξαρτήσεις που έχουν καταχωρηθεί στο έργο.
Ο κώδικας χωρίζεται σε τρία μέρη: models, views and controllers .
Ένα view είναι ο κώδικας που παρέχει μια γραφική αναπαράσταση του συστήματος στο χρήστη, δηλαδή μια γραφική επαφή χρήστη.
Ωστόσο, ενώ τα views είναι υπεύθυνα για την οπτική αναπαράσταση μιας εφαρμογής, δεν αποτελούν ανεξάρτητο μέρος του λογισμικού. Για να εμφανιστεί, ένα view απαιτούνται δεδομένα. Τα δεδομένα που χρησιμοποιεί μια προβολή για τη δημιουργία μιας απεικόνισης ονομάζονται models.
Οι controllers είναι κλάσεις Java Servlet που είναι υπεύθυνες για τη διαχείριση των δεδομένων models που ανακτώνται από τη βάση δεδομένων, εισάγονται από τον χρήστη ή δημιουργούνται κατά την επεξεργασία και την εκτέλεση διαφόρων εργασιών του συστήματος. Ένας controller λαμβάνει ένα αίτημα από ένα πρόγραμμα περιήγησης, επεξεργάζεται το αίτημα με βάση τα δεδομένα και τα χαρακτηριστικά του αιτήματος, όπως η διεύθυνση URL του αιτήματος και οι μέθοδοι (π.χ. GET, POST), ανακτά και επεξεργάζεται τα δεδομένα από τη βάση δεδομένων και, τέλος, διαβιβάζει τα επεξεργασμένα δεδομένα models στην view, η οποία παράγει μια γραφική ιστοσελίδα που αποστέλλεται στο πρόγραμμα περιήγησης ως απάντηση στο αίτημα- η view αντιπροσωπεύεται από το πρότυπο Thymeleaf.

Spring Security

Το Spring Security είναι ένα Framework που ήταν υπεύθυνο για την ασφάλεια των δεδομένων σε διαδικτυακά πληροφοριακά συστήματα και ήταν το πιο συχνά χρησιμοποιούμενο Security Framework σε εφαρμογές Java EE. Οι δύο κύριες λειτουργίες που πρέπει να παρέχει ένα ασφαλές λειτουργικό σύστημα είναι η πιστοποίηση (authentication) των πελατών που στέλνουν αιτήσεις στο σύστημα και ο έλεγχος των δικαιωμάτων (authorization) για την εκτέλεση ορισμένων λειτουργιών.

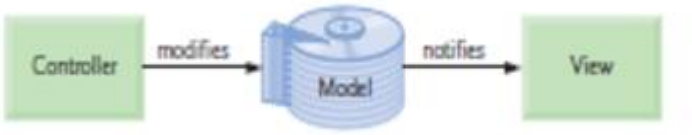
Apache Maven

Το Apache Maven είναι ένα εργαλείο διαχείρισης έργων λογισμικού που βασίζεται στο Μοντέλο Αντικειμένου Έργου (Project Object Model - POM). Το POM είναι ένα αρχείο XML που χρησιμοποιεί το Maven για την κεντρική οργάνωση και διαχείριση της δομής όλων των στοιχείων της ηλεκτρονικής εφαρμογής. Στο αρχείο POM δηλώνονται οι βιβλιοθήκες (όπως τα jars) και οι εκδόσεις τους που πρέπει να συμπεριληφθούν στην εφαρμογή, ποια στοιχεία εφαρμογής θα δημιουργηθούν και ποια όχι, ποιοι τύποι αρχείων κατασκευάστηκαν (π.χ. jar, war), εκδόσεις λογισμικού και γενικά διαφορετικές φάσεις διαχείρισης εφαρμογής. Το κύριο πλεονέκτημα του Maven είναι ότι χρησιμοποιεί κεντρική αποθήκη βιβλιοθηκών (Repository) και μπορεί να χρησιμοποιηθεί για και κάθε βιβλιοθήκη που υπάρχει εκεί μπορεί να παρέχεται αυτόματα σε ποικίλες εφαρμογές όταν απαιτείται μια συγκεκριμένη βιβλιοθήκη, χωρίς ο προγραμματιστής να χρειάζεται να την αναζητήσει ξανά.

2.1 MODEL-VIEW-CONTROLLER

Οι εφαρμογές που χρησιμοποιούν την τεχνολογία Springboot ακολουθούν το μοντέλο Model View Controller (MVC), το οποίο διαχωρίζει τα δεδομένα της εφαρμογής από τα γραφικά παρουσίασης και τη λογική επεξεργασίας, όπως φαίνεται στην εικόνα.

Εικόνα 1 MVC Model [Πηγή Java Programming Deitel 2011]



Ο controller είναι υπεύθυνος για τον έλεγχο και την αλληλεπίδραση μεταξύ των προβολών και των μοντέλων. Ένα μοντέλο περιέχει δεδομένα εφαρμογής, δηλαδή μια βάση δεδομένων, και μια προβολή περιέχει μια αναπαράσταση δεδομένων, δηλαδή μια ιστοσελίδα. Εάν ένας χρήστης χρησιμοποιεί μόνο μια σελίδα, η σελίδα αυτή αλληλοεπιδρά με το μοντέλο για την αποθήκευση και ανάκτηση δεδομένων. Όταν το μοντέλο αλλάζει, η προβολή ενημερώνεται με τα νέα δεδομένα.
Στην εφαρμογή μας είναι σημαντικό να χρησιμοποιήσουμε τις τεχνολογίες session tracking. Όπως μπορείτε να δείτε, το @SessionScoped Bean μας επιτρέπει να διατηρούμε τις ρυθμίσεις διαμόρφωσης του χρήστη ενώ ο χρήστης χρησιμοποιεί την τοποθεσία. Αυτό το φασόλι δημιουργείται στην αρχή της συνεδρίας και υπάρχει για όλη τη διάρκεια της συνεδρίας. Ένα SessionScoped Bean μπορεί να έχει πρόσβαση από όλες τις ιστοσελίδες και διατηρεί τη συνεδρία ο server.

2.2 Java persistence

Το Java Persistence API παρέχει χαρτογράφηση βάσεων δεδομένων με τη δημιουργία αντικειμένων για τη διαχείριση σχεσιακών βάσεων δεδομένων. Το interface αυτό αποτελείται από τις τέσσερις παρακάτω περιοχές

• Το Java Persistence API

• Την Γλώσσα εκτέλεσης των Queries

• Το Java Criteria API

• Μεταδεδομένα χαρτογράφησης αντικειμένων/σχέσεων

2.3 Βάση Δεδομένων

Για την υλοποίηση της εφαρμογής θα χρησιμοποιηθεί βάση ανοιχτού κώδικα. Δεν χρειάζεται να καθορίσουμε εξαρχής το εργαλείο ανάπτυξης της βάσης γιατί μπορεί να διαφέρει αναλόγως τις προτιμήσεις του χρήστη. Δύο πολύ γνωστά περιβάλλοντα ανάπτυξης είναι αυτά της mysql και η postgres. Παρακάτω θα περιγράφουν συνοπτικά τα χαρακτηριστικά και οι δυνατότητες των δυο αυτών σχεδιαστικών εργαλείων προκειμένου ο αναγνώστης να έχει μια ιδέα το τι προσφέρει κάθε εργαλείο.

2.4 MySQL Workbench

Το MySQL Workbench μπορεί να χρησιμοποιηθεί για τον οπτικό σχεδιασμό και τη δημιουργία βάσεων δεδομένων οποιουδήποτε τύπου, συμπεριλαμβανομένων των βάσεων δεδομένων ιστού, OLTP και αποθηκών δεδομένων. Περιλαμβάνει μοντελοποίηση δεδομένων και είναι απαραίτητη για τη δημιουργία σύνθετων μοντέλων ER. Περιλαμβάνει επίσης βασικές λειτουργίες για την εκτέλεση πολύπλοκων διοικητικών αλλαγών και τεκμηρίωσης, οι οποίες συνήθως απαιτούν πολύ χρόνο και προσπάθεια. Το MySQL Workbench είναι διαθέσιμο για Windows, Linux και Mac OS.

2.5 Unified Modeling Language (UML)

Η Unified Modeling Language (UML) είναι πλέον η τυπική γλώσσα μοντελοποίησης για τη μηχανική λογισμικού. Χρησιμοποιείται για τη γραφική αναπαράσταση, την περιγραφή, την δημιουργία και την τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού. Μπορεί να χρησιμοποιηθεί σε διάφορα στάδια της ανάπτυξης, από την ανάλυση απαιτήσεων έως τη δοκιμή τελικών συστημάτων. Διαθέτει συγκεκριμένο σύνολο από όρους, σύμβολα και διαγράμματα που του επιτρέπουν :

- εμφάνιση των ορίων και των βασικών λειτουργιών του, συστήματος, χρησιμοποιώντας «περιπτώσεις χρήσης» (use-cases) και «actors».

- Επεξήγηση με "διαγράμματα αλληλεπίδρασης" για την υλοποίηση των περιπτώσεων χρήσης.

- Αναπαράσταση της στατικής δομής του συστήματος με τη βοήθεια "διαγραμμάτων κλάσεων".

- Μοντελοποίηση της συμπεριφοράς αντικειμένων με τη χρήση "διαγραμμάτων κατάστασης".

- Η χρήση "διαγραμμάτων συστατικών" και "διαγραμμάτων ανάπτυξης" για την αποκάλυψη της υλοποίησης της αρχιτεκτονικής.

- Επέκταση της λειτουργικότητας με τη βοήθεια "στερεοτύπων".

2.6 Τρόπος ανάπτυξης της υπηρεσίας

Για την εφαρμογής έπρεπε να ακολουθηθούν κάποια βήματα για την σωστή ανάπτυξη του λογισμικού. Όπως και για κάθε εφαρμογή έτσι και για αυτή, η διαδικασία είναι αυτής της τεχνολογίας λογισμικού και φαίνεται στο παρακάτω Σχήμα 1.

Εικόνα 2 Διαδικασία Ανάπτυξης

Κεφάλαιο 3ο: ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ

Εισαγωγή

Όπως είπαμε σκοπός μας είναι ο σχεδιασμός ενός πληροφοριακού συστήματος διαχείρισης. Η υπηρεσία αυτή αποσκοπεί στην ενσωμάτωση διαφόρων εφαρμογών και τεχνολογιών συμβάλλοντας έτσι στην δια λειτουργικότητα των συστημάτων και την επίτευξη της ενοποίησης των συστημάτων και εφαρμογών. Έχουμε αναφέρει ότι πλέον η κατεύθυνση προγραμματιστών είναι προς την υλοποίηση εφαρμογών που δίνουν τη δυνατότητα μέσα από ένα φιλικό περιβάλλον ο χρήστης να αναζητεί δεδομένα από διάφορες εφαρμογές και διάφορες βάσεις δεδομένων. Οι υπηρεσίες αυτές μπορούν να επεξεργαστούν όλα τα δεδομένα απομακρυσμένα χωρίς την ανάγκη επιπλέον λογισμικού και γνώσης από το χρήστη. Σημαντικό στοιχείο στις διαδικτυακές υπηρεσίες είναι η επεκτασιμότητα και η προσαρμοστικότητα σε διάφορες τεχνολογίες.

Παροχή Υπηρεσίας

Η υπηρεσία θα σχεδιαστεί σαν δικτυακή εφαρμογή προκειμένου να υλοποιείται μέσω του internet, ώστε ο χρήστης να μπορεί εύκολα και γρήγορα χωρίς προαπαιτούμενα, όπως εγκατάσταση και δημιουργία λογαριασμού, να έχει πρόσβαση στην υπηρεσία. Η υπηρεσία θα υλοποιηθεί σε έναν απομακρυσμένο server ο οποίος και θα υποστηρίζει την υπηρεσία και ο χρήστης με την χρήση του domain name που θα του δοθεί θα καλεί την υπηρεσία. Στόχος είναι η εύκολη πρόσβαση χωρίς την ανάγκη χρήσης username και δημιουργίας λογαριασμών, προκειμένου να χρησιμοποιήσει την υπηρεσία. Έτσι αποφεύγουμε τις διαδικασίες αυθεντικότητας, πιστοποίησης και κρυπτασφάλισης και δίνουμε την ελευθερία σε όλους να χρησιμοποιούν την υπηρεσία χωρίς να δίνουν προσωπικά δεδομένα.

Επιπλέον, η εφαρμογή αυτή είναι ανεξάρτητη υπολογιστή. Η προτεινόμενη γλώσσα προγραμματισμού είναι η Java που είναι ανοιχτού κώδικα και παρέχει ένα μεγάλο πλήθος από βιβλιοθήκες, που μπορούν να χρησιμοποιηθούν, και πολύ καλή υποστήριξη.

Η τεχνολογία σήμερα προχωρά με γρήγορα βήματα και εμείς πρέπει να την ακολουθούμε. Η χρήση των κινητών τηλεφώνων, των smartphone, των tablets και η χρήση γενικά του διαδικτύου αυξάνεται καθημερινά γι' αυτό και πρέπει η υπηρεσία μας να είναι διαθέσιμη σε όλες τις συσκευές. Η δυνατότητα επέκτασης και αναβάθμισης της υπηρεσίας πρέπει να αποτελεί κύριο συστατικό της σχεδίασης.

3.1 Παρατίθενται οι λειτουργικές απαιτήσεις του συστήματος:

- ✓ Υπάρχει μια κεντρική αποθήκη που κατέχει και διαχειρίζεται το σύνολο των ενδυμάτων.
- ✓ Κάθε ένδυμα το χαρακτηρίζει
    - ο μοναδικός κωδικός του,
    - το μέγεθος του (πχ S, L)
    - και το απόθεμα του στην αποθήκη.
- ✓ Υπάρχουν αρκετές δευτερεύουσες αποθήκες με δικό τους (τοπικό) διαχειριστή οι οποίες αιτούνται κατόπιν παραγγελίας, ενδύματα από την κεντρική αποθήκη.
- ✓ Ο κεντρικός διαχειριστής μπορεί να εγκρίνει ή και να απορρίψει μια παραγγελία.
- ✓ Κάθε παραγγελία χαρακτηρίζεται από:
    - τον μοναδικό κωδικό της,
    - τον χρήστη που τη υπέβαλλε,
    - την αποθήκη για την οποία προορίζεται,
    - η αιτούμενη ποσότητα ενδύματος,
    - το είδος ενδύματος,
    - το μέγεθος ενδύματος,
    - η ποσότητα που πουλήθηκε (εφόσον εγκριθεί),
    - η κατάσταση της (υπό έγκριση από διαχειριστή =0 , απορριφθείσα = 1, εγκεκριμένη =2)
- ✓ Κάθε διαχειριστής δευτερεύουσας αποθήκης οφείλει να ενημερώνει πόσα τεμάχια από τα παραληφθέντα έχουν πουληθεί ώστε να ενημερώνεται και το τρέχον απόθεμα του αντίστοιχου είδους στην αποθήκη του.
- ✓ Είναι δυνατή η προβολή , των αποθεμάτων, σε πραγματικό χρόνο τόσο στην κεντρική αποθήκη, όσο και στις δευτερεύουσες. Η πληροφορία αυτή εξάγεται και σε αρχείο excel.

3.2 Product Backlog list

Σε συνέχεια του ορισμού του Product Backlog list (§3.3) , τα στοιχεία του Product Backlog έχουν ως κύριες ιδιότητες: περιγραφή και σειρά προτεραιότητας και μπορούν να πάρουν την μορφή ιστοριών χρηστών είτε από τη σκοπιά του χρήστη (μερικός διαχειριστής) είτε από τη σκοπιά του διαχειριστή.

Ιστορία χρήστη	**Προτεραιότητα**
Σαν διαχειριστής θα ήθελα να καταχωρώ αποθήκες	1
Υπάρχει μια κεντρική αποθήκη που κατέχει και διαχειρίζεται το σύνολο των δευτερευουσών αποθηκών	
Σαν διαχειριστής θα ήθελα να καταχωρώ ενδύματα	2
Υπάρχει μια κεντρική αποθήκη που κατέχει και διαχειρίζεται το σύνολο των ενδυμάτων.	
Κάθε ένδυμα το χαρακτηρίζει	
· ο μοναδικός κωδικός του,	
· και το απόθεμα του στην αποθήκη.	
Σαν διαχειριστής θα ήθελα να αντιστοιχώ τα ενδύματα σε μεγέθη	3
Κάθε ένδυμα το χαρακτηρίζει	
· το μέγεθος του (πχ M ,S, L , XL, XXL)	
Σας διαχειριστής θα ήθελα να καταχωρώ νέους χρήστες ως μερικούς διαχειριστές	4
Υπάρχουν αρκετές δευτερεύουσες αποθήκες με δικό τους (τοπικό) διαχειριστή	
Σαν διαχειριστής θα ήθελα να έχω εποπτεία των αποθεμάτων ενδυμάτων	5
Κάθε διαχειριστής δευτερεύουσας αποθήκης οφείλει να ενημερώνει πόσα τεμάχια από τα παραληφθέντα έχουν πουληθεί ώστε να ενημερώνεται και το τρέχον απόθεμα του αντίστοιχου είδους στην αποθήκη του και να εποπτεύεται από τον γενικό διαχειριστή	
Σαν μερικός διαχειριστής θα ήθελα να κάνω νέες παραγγελίες	6
Στις δευτερεύουσες αποθήκες, οι (τοπικοί) διαχειριστές αιτούνται κατόπιν παραγγελίας, ενδύματα από την κεντρική αποθήκη.	
Σαν μερικός διαχείρισης θα ήθελα να ελέγχω τα αποθέματα μου	7

Κάθε διαχειριστής δευτερεύουσας αποθήκης οφείλει να ενημερώνει πόσα τεμάχια από τα παραληφθέντα έχουν πουληθεί ώστε να ενημερώνεται και το τρέχον απόθεμα του αντίστοιχου είδους στην αποθήκη του.	
Σαν διαχειριστής θα ήθελα να διαχειρίζομαι νέες παραγγελίες	8
Ο κεντρικός διαχειριστής μπορεί να εγκρίνει ή και να απορρίψει μια παραγγελία.	
Κάθε παραγγελία χαρακτηρίζεται από	
· τον μοναδικό κωδικό της,	
· τον χρήστη που τη υπέβαλλε,	
· την αποθήκη για την οποία προορίζεται,	
· η αιτούμενη ποσότητα ενδύματος,	
· το είδος ενδύματος,	
· το μέγεθος ενδύματος,	
· η ποσότητα που πουλήθηκε (εφόσον εγκριθεί),	
· η κατάσταση της (υπό έγκριση από διαχειριστή =0 , απορριφθείσα = 1, εγκεκριμένη =2)	
Σαν διαχειριστής θα ήθελα να κάνω διαχείριση κωδίκων χρηστών	9
Σαν διαχειριστής θα ήθελα να εκτυπώνω τα αποθέματά	10
Είναι δυνατή η προβολή , των αποθεμάτων, σε πραγματικό χρόνο τόσο στην κεντρική αποθήκη, όσο και στις δευτερεύουσες. Η πληροφορία αυτή εξάγεται και σε αρχείο Excel.	

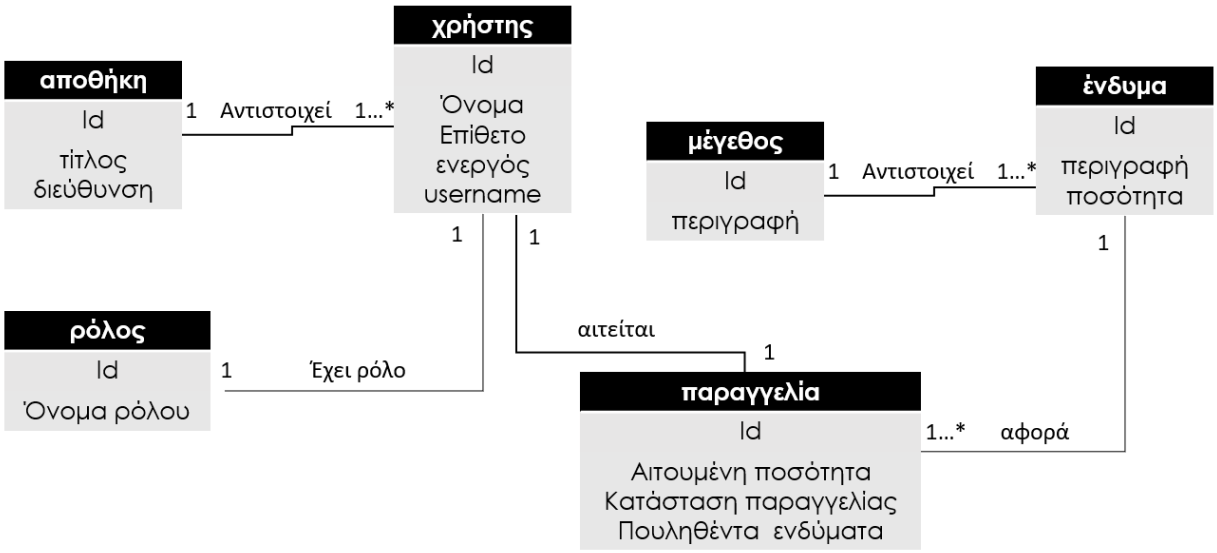
Επιπλέον, οι μη λειτουργικές απαιτήσεις περιγράφουν πώς (ή σε ποιο βαθμό) το λογισμικό υποστηρίζει τις λειτουργικές απαιτήσεις. Η λειτουργική ορθότητα σημαίνει ότι το σύστημα λειτουργεί σύμφωνα με τις τεκμηριωμένες λειτουργικές απαιτήσεις, δεν αποτυγχάνει ή δεν ανακτά λανθασμένα δεδομένα λόγω λανθασμένου χειρισμού από τον χρήστη, παρέχει καλή συμβατότητα και φιλικότητα προς τον χρήστη, επιτρέπει την εύκολη ανάπτυξη του συστήματος καθώς οι απαιτήσεις αλλάζουν, επιτρέπει στο λογισμικό να λειτουργεί σε διαφορετικά περιβάλλοντα, να λειτουργεί και, τελικά, να εξασφαλίζει την ασφάλεια των δεδομένων.

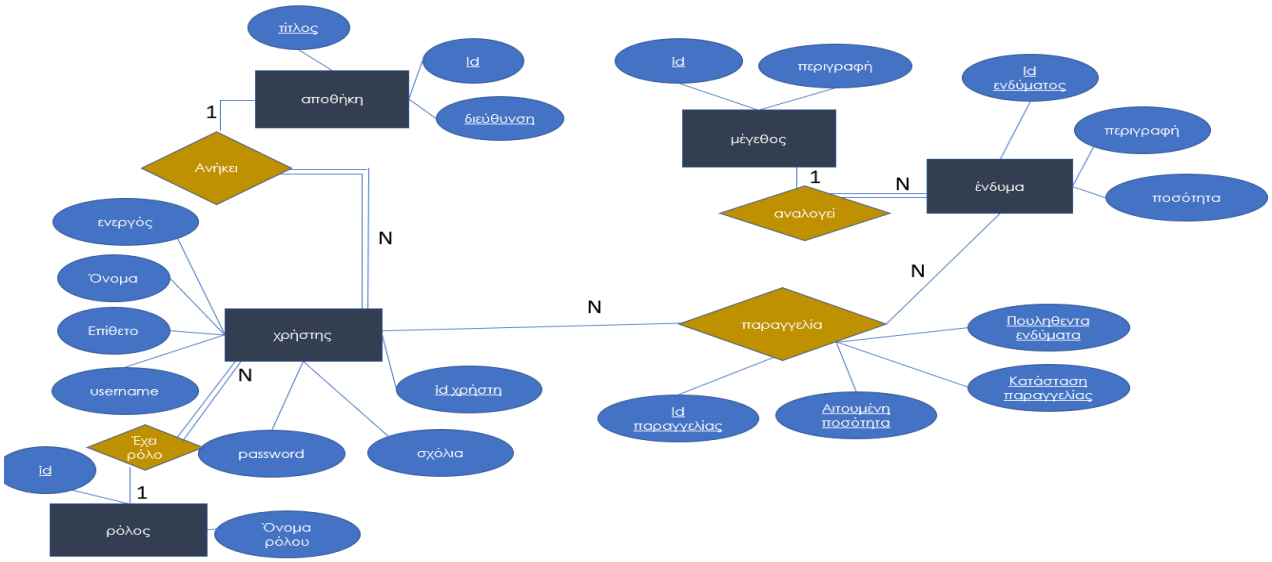
Ως εκ τούτου, ορισμένες μη λειτουργικές απαιτήσεις είναι :

1. Μόνο η γενική διαχείριση της κεντρικής αποθήκης θα πρέπει να έχει πρόσβαση στις λειτουργικές αλλαγές της εφαρμογής.
2. Ο χρόνος απόκρισης του συστήματος πρέπει να είναι εντός 3 δευτερολέπτων (οι φυσικοί πόροι πρέπει να έχουν άριστες προδιαγραφές).
3. Ελέγξτε ότι η εισαγωγή δεδομένων είναι σωστή και δεχτείτε μόνο έγκυρα δεδομένα.
4. Πρέπει να είναι σε θέση να ανακτά το τρέχον απόθεμα και να το εξάγει για εκτύπωση ή αποθήκευση.
5. Το σύστημα χρήζει να είναι φιλικό προς το χρήστη.

Εννοιολογικό Μοντέλο:

Το διάγραμμα οντοτήτων-συσχετίσεων της βάσης δεδομένων που προκύπτει από τις παραπάνω απαιτήσεις είναι το εξής:





Sprint 1 – Αυθεντικοποίηση

Αρχικός στόχος αποτελεί η ολοκλήρωση του συστήματος αυθεντικοποίησης στην εφαρμογή.

- ✓ Σύνδεση με τα στοιχεία ταυτοποίησης του διαχειριστή

- ✓ Εγγραφή νέου χρήστη (διαχειριστή δευτερεύουσας αποθήκης)

Ο κεντρικός διαχειριστής καταχωρεί τα στοιχεία του .

Sprint 2 – Καταχώρηση Ειδών στην Εφαρμογή

- ✓ Καταχώρηση ειδών ιματισμού

Ο κεντρικός διαχειριστής καταχωρεί στοιχεία ειδών που βρίσκονται στην αποθήκη του.

- ✓ Καταχώρηση αποθηκών

Ο κεντρικός διαχειριστής καταχωρεί στοιχεία νέας δευτερεύουσας αποθήκης

Sprint 3 – Καταχώρηση Παραγγελιών και προβολή αποθεμάτων

- ✓ Παραγγελία στον κεντρικό διαχειριστή

Μόνο οι (τοπικοί) διαχειριστές δευτερευόντων αποθηκών μπορούν να υποβάλλουν παραγγελία στον κεντρικό διαχειριστή.

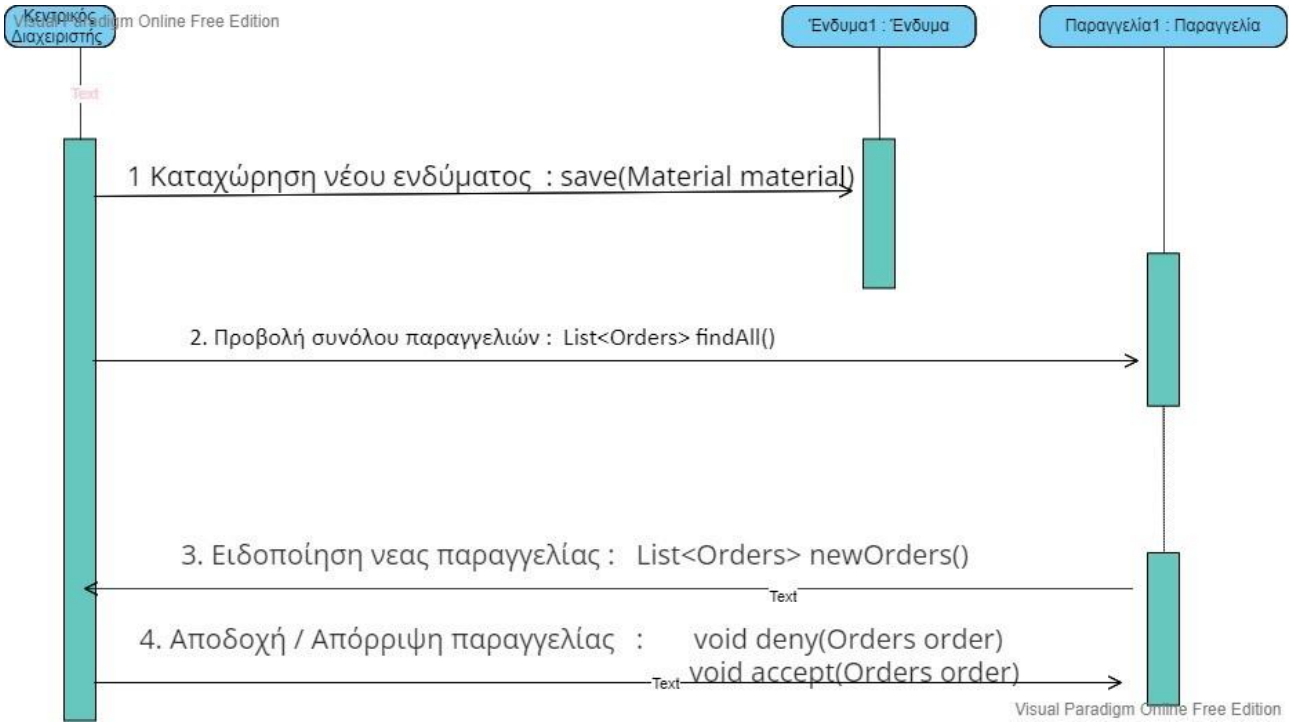
- ✓ Έλεγχος αποθέματος.

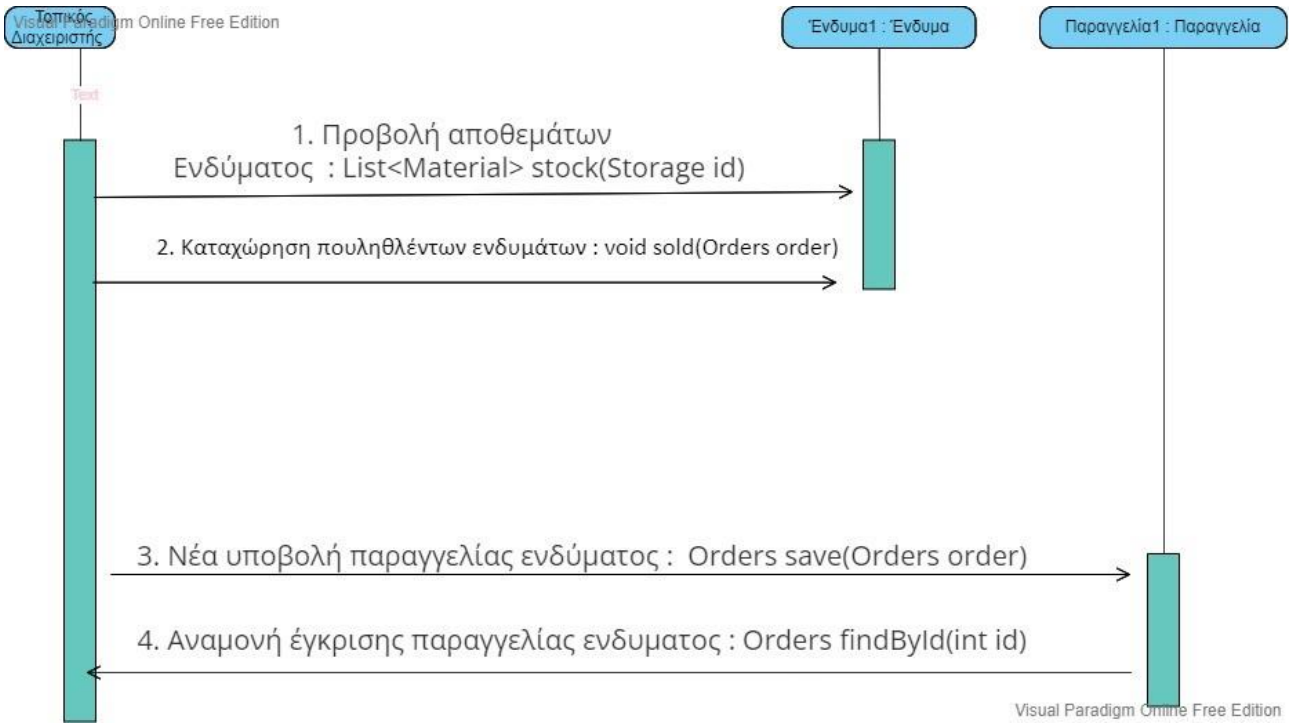
Οι διαχειριστές εποπτεύουν το τρέχον απόθεμα προς εξασφάλισή επάρκειας.

Για την υλοποίηση της «**Υποβολής νέας παραγγελίας**» πρέπει πρώτα να έχει γίνει η «καταχώρηση ενδυμάτων» (include).

Ένα διάγραμμα ακολουθίας είναι μια δισδιάστατη αναπαράσταση της αλληλεπίδρασης μεταξύ αντικειμένων. Με άλλα λόγια, πρόκειται για μια οπτική αναπαράσταση ενός σεναρίου συνεργασίας μεταξύ αντικειμένων. Ο κάθετος άξονας αντιστοιχεί στον άξονα του χρόνου και ο οριζόντιος άξονας

αντιπροσωπεύει ανεξάρτητα αντικείμενα. Ένας τρόπος για να ορίσετε ένα αντικείμενο είναι να προσθέσετε το όνομα της κλάσης στην οποία ανήκει το αντικείμενο μετά από μια άνω και κάτω τελεία (:).
Κάθε αντικείμενο αντιστοιχεί σε μια κάθετη γραμμή που ονομάζεται γραμμή ζωής (lifeline).
Τα αντικείμενα ανταλλάσσουν μηνύματα, όπως κλήση συναρτήσεων άλλων αντικειμένων, αποστολή ασύγχρονων μηνυμάτων σε άλλα αντικείμενα, επιστροφή τιμών κλήσης, μηνύματα αληθούς/ψευδούς υπό όρους κ.λπ.



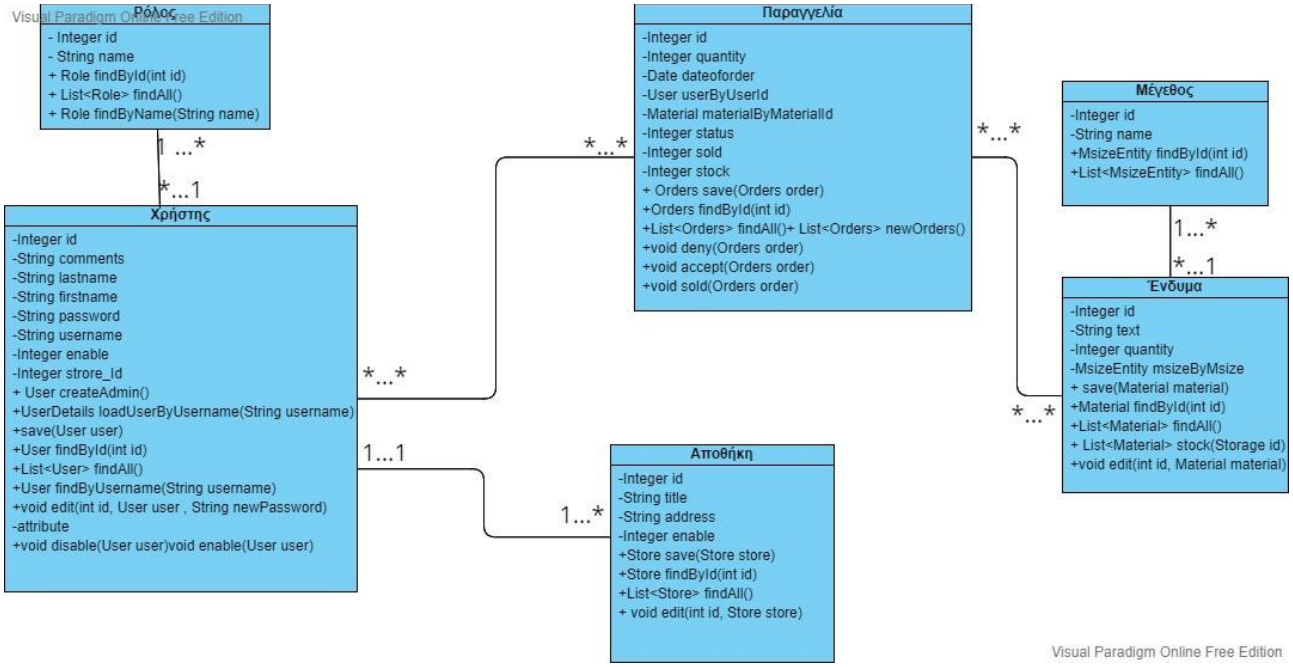


Σε ένα αντικειμενοστραφές σύστημα, οι κλάσεις και οι σχέσεις μεταξύ κλάσεων είναι τα δομικά στοιχεία που επιτρέπουν στα αντικείμενα που δημιουργούνται ως περιπτώσεις μιας κλάσης να συνεργάζονται μεταξύ

τους. Ένα διάγραμμα κλάσεων ενός συστήματος περιέχει τις κλάσεις και τις σχέσεις τους αλλά επίσης περιγράφει τη στατική δομή του.

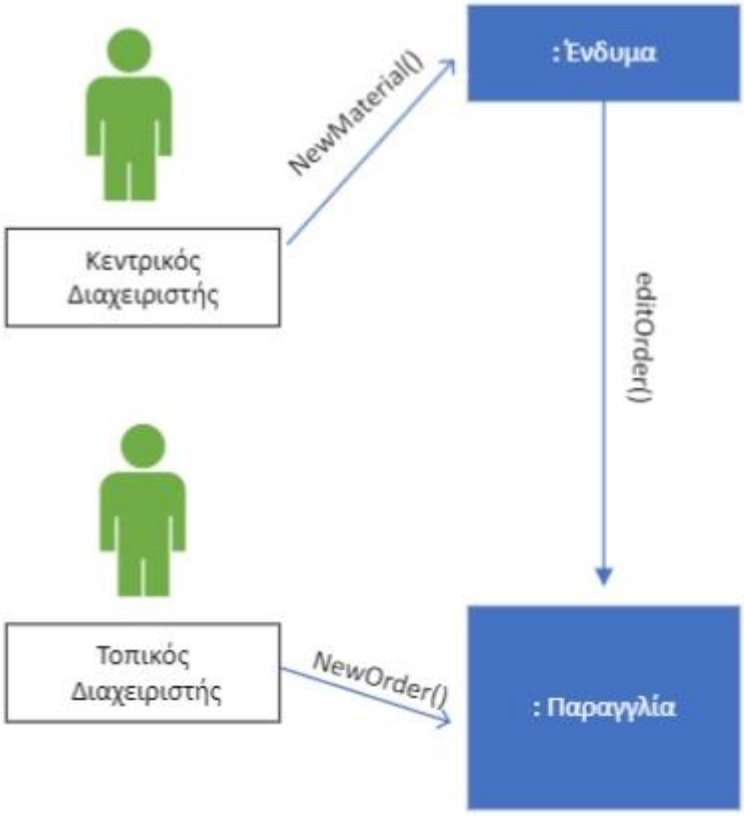
Μια κλάση περιέχει δεδομένα και συναρτήσεις που λειτουργούν σε αυτά τα δεδομένα. Τα βασικά στατιστικά στοιχεία αυτού του διαγράμματος έχουν ως εξής.
- Η ορατότητα αντιπροσωπεύεται από τα σύμβολα "-", "+", "#" και "~" που δηλώνουν την ιδιωτική, τη δημόσια, την ασφαλή και την πρόσβαση πακέτων αντίστοιχα.
- Η πολλαπλότητα αναφέρεται στο ένα άκρο μιας συσχέτισης και παριστάνεται ως εξής.

Ο αριθμός των αντικειμένων που μπορούν να συσχετιστούν με αντικείμενα άλλων κλάσεων.



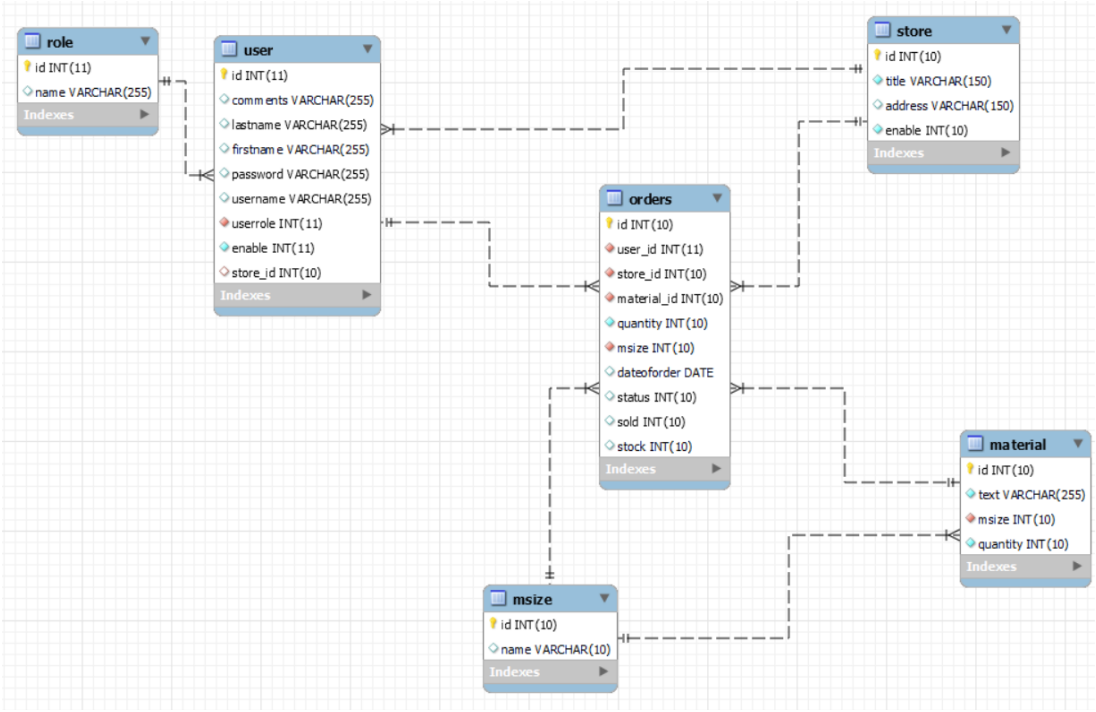
Τα διαγράμματα συνεργασίας δείχνουν τα αντικείμενα που αλληλοεπιδρούν και τις σχέσεις μεταξύ τους. Από την άλλη όμως, τα διαγράμματα ακολουθίας δείχνουν πρωτίστως ποια είναι η ροή μηνυμάτων σε ένα σενάριο περίπτωσης χρήσης, τα διαγράμματα αλληλεπίδρασης τα χρησιμοποιούμε ώστε να δείξουμε τις σχέσεις αντικειμένων που υπάρχουν μεταξύ τους. Δίπλα στις συνδέσεις, τα μηνύματα που έχουν σταλεί με μικρές ακμές. Η αρίθμηση των μηνυμάτων χρησιμοποιείται για να υποδείξει με ποια σειρά είναι απαραίτητο να ανταλλάσσονται τα μηνύματα. Τα διαγράμματα ακολουθίας και τα διαγράμματα συνεργασίας στην ουσία συμπληρώνουν το ένα το άλλο, επειδή αποτελούνται από τις όμοιες πληροφορίες, όμως το καθένα παρέχει μια άλλη οπτική των πραγμάτων.

Αν και είναι εύκολο να προσδιοριστεί από τέτοια διαγράμματα η ύπαρξη ομάδων αλληλοεπιδρώντων οντοτήτων και συνδέσμων μεταξύ οντοτήτων, είναι δύσκολο να απεικονιστεί η ροή των μηνυμάτων, όμως παρουσιάζεται ευκολότερα σε διαγράμματα ακολουθίας.



Κεφάλαιο 4ο: Διάγραμμα πινάκων βάσης

Ως εκ τούτου, οι πίνακες που δημιουργούνται στη βάση δεδομένων ως αποτέλεσμα του εννοιολογικού και λογικού σχεδιασμού μπορούν να συνοψιστούν όπως φαίνεται στο παρακάτω διάγραμμα:



Το ακόλουθο σχεσιακό μοντέλο έχει δημιουργηθεί για τον έλεγχο ταυτότητας και την εξουσιοδότηση των χρηστών που έχουν πρόσβαση στην εφαρμογή. Σε μια βάση δεδομένων θα αποθηκευτούν οι χρήστες. Οι κωδικοί πρόσβασης κρυπτογραφούνται (με τη χρήση του BCryptPasswordEncoder) ενώ το πεδίο ID του πίνακα role ταιριάζει με τα δικαιώματα πρόσβασης της εφαρμογής (δικαιώματα κεντρικού ή τοπικού διαχειριστή), με δυνατότητα πιστοποίησης ταυτότητας των χρηστών.

Ο φυσικός σχεδιασμός της βάσης δεδομένων χρησιμοποίησε τις δυνατότητες απεικόνισης (Hibernate) που παρέχεται από την java persistence.

Κεφάλαιο 5ο: Spring Boot Αρχιτεκτονική

Συστατικά

Τα κύρια στοιχεία του κώδικα της εφαρμογής είναι ο controller, τα Services και τα repositories. Κατά την υλοποίηση των παραπάνω συστατικών, το Spring Framework χρησιμοποιεί ειδικές επισημειώσεις (annotations : @) για να προσδιορίσει συγκεκριμένα τον τύπο και τη λειτουργία κάθε συστατικού: μια κλάση με @Controller υποδεικνύει στο Spring ότι η κλάση αυτή είναι ένας controller. Ως εκ τούτου, οι @Service και @Repository υποδεικνύουν στο Spring ότι πρόκειται για Services και repositories αντίστοιχα. Επομένως, όταν εκτελείται η Spring εφαρμογή και αρχικοποιεί τα συστατικά, όλες οι κλάσεις που έχουν δηλωθεί με ένα από αυτά τα annotations μπορούν και τα αναγνωρίζουν.

Controllers

Σκοπός κάθε controller είναι να λαμβάνει και να επεξεργάζεται τα αιτήματα που λαμβάνει η εφαρμογή. Δεδομένου ότι κάθε πρόγραμμα μπορεί να εκτελεί εκατοντάδες ή χιλιάδες ξεχωριστές λειτουργίες, αυτές οι

λειτουργίες ομαδοποιούνται με βάση κοινά χαρακτηριστικά και κάθε ομάδα αυτών των λειτουργιών ανατίθεται σε έναν controller που είναι υπεύθυνος για την εκτέλεσή τους. Ένας controller μπορεί να περιλαμβάνει πολλές μεθόδους, κάθε μία από τις οποίες είναι υπεύθυνη για την εκτέλεση ενός συγκεκριμένου αιτήματος. Όταν μια εφαρμογή λαμβάνει ένα αίτημα, αναζητά και βρίσκει έναν controller που θα περάσει το αίτημα. Η ταυτότητα του controller καθορίζεται από τη διεύθυνση URL του αιτήματος και την επισήμανση @GetMapping στη δήλωση της κλάσης controller. Η χρήση του σχολίου @GetMapping (value = "/viewAllOrders") στη δήλωση της κλάσης ελεγκτή διασφαλίζει ότι ο ελεγκτής λαμβάνει όλες τις αιτήσεις των οποίων η διεύθυνση URL αρχίζει με "http://localhost/storage/viewAllOrders" (όπου "http://localhost/storage/" θεωρείται η βασική διεύθυνση της εφαρμογής και θεωρείται ότι παραμένει σταθερή και κοινή για όλες τις αιτήσεις που λαμβάνονται). Μετά τη λήψη του αιτήματος, ο controller εξετάζει τις μεθόδους για να βρει μια κατάλληλη μέθοδο στην οποία μπορεί να δώσει εντολή να εκτελέσει το αίτημα. Κάθε μέθοδος χρησιμοποιεί το @GetMapping για να υποδείξει ποια αίτηση θα πρέπει να δεχτεί και να εκτελέσει. Εάν ο controller που αναφέρθηκε προηγουμένως έχει δύο μεθόδους και η πρώτη μέθοδος είναι @GetMapping (value="users") και η δεύτερη μέθοδος είναι @GetMapping (value="/users/change"), τότε η διεύθυνση URL της αίτησης είναι " http://. localhost/storage/users", αυτό σημαίνει ότι το @GetMapping θα παραληφθεί από αυτόν τον ελεγκτή και θα ανατεθεί στην πρώτη μέθοδο που ταιριάζει.

```
@GetMapping(value = {"/materials/edit/{id}"})
public String edit(@PathVariable("id") String id, Model model) {    1

    Authentication loggedInUser = SecurityContextHolder.getContext().getAuthentication();
    String role = loggedInUser.getAuthorities().toString();  // ρολος χρηστη

    if (role.equals("[SuperAdmin]")) { // "SuperAdmin"    2

        Material materialForm = null;
        try {
            materialForm = materialService.findById(Integer.parseInt(id));
        } catch (NumberFormatException e) {
            return "redirect:/materials";
        }

        if (materialForm != null) {

            model.addAttribute( attributeName: "materialForm", materialForm);
            model.addAttribute( attributeName: "allMsizes",  getAllMsizes());
            model.addAttribute( attributeName: "user", User.getCurrentUser());
            return "editMaterial";    3

        } else {
            return "redirect:/materials";
        }
    }
    return "redirect:/stores";
}
```

```
@PostMapping(value = "/stores/edit/{id}") 4
public String edit(@PathVariable("id") int id, @ModelAttribute("storeForm") Store storeForm,
                   Model model) {



    if (storeForm.getTitle().equals("")){
        return "redirect:/stores";
    }

    try {

        storeService.edit(id, storeForm);
        return "redirect:/stores";
    } catch (Exception e) {
       // e.printStackTrace(System.err);
        return "redirect:/stores";
    }

}
```

Ο παραπάνω κώδικας δείχνει τη λειτουργία επεξεργασίας πληροφοριών αποθεμάτων. Αυτή η διαδικασία είναι δυνατή μόνο όταν ο χρήστης έχει δικαιώματα διαχειριστή (2). Η μέθοδος controller ξεκινά την επεξεργασία της αίτησης (@GetMapping(value = {"/stores/edit/{id}"}) όταν τη λαμβάνει. Αυτή η μέθοδος έχει δεδομένα (ID αποθήκης) τα οποία αποστέλλονται μαζί με την αίτηση. Ως εκ τούτου, η μέθοδος λαμβάνει αυτά τα δεδομένα ως παράμετρο: @PathVariable υποδηλώνει ότι η τιμή αυτής της μεταβλητής περιλαμβάνεται στην αίτηση που επεξεργάζεται η μέθοδος (1). Η μέθοδος που χρησιμοποιεί την υπηρεσία (π.χ. "storeService") επεξεργάζεται στη συνέχεια την αίτηση, αποθηκεύει το αποτέλεσμα σε ένα αντικείμενο και το "στέλνει" στον χρήστη της αίτησης, το οποίο στη συνέχεια αποθηκεύεται στο model. Στο τέλος της εκτέλεσης αυτής της μεθόδου, ορίζεται ένα συστατικό view που θα αποσταλεί ως response στο αίτημα του πελάτη. Η εντολή στο τέλος της παραπάνω μεθόδου είναι : **return "editStore**;. Κατόπιν, το view που θα προβληθεί, περιέχει μέθοδο αποστολής πληροφοριών με post. (τροποποίηση δεδομένων) . Συνεπώς η μέθοδος που θα κληθεί με το submit του χρήστη είναι η @PostMapping(value = **"/stores/edit/{id}"**) (4).

Services

Όταν μια μέθοδος του controller λαμβάνει ένα αίτημα, χρησιμοποιεί μια υπηρεσία για την επεξεργασία του αιτήματος. Η διεπαφή της υπηρεσίας 'StoreService' παρουσιάζεται παρακάτω.

```java
package com.project.storage.storage.service;

import com.project.storage.storage.entity.Store;

import java.util.List;

public interface StoreService {

    Store save(Store store);


    Store findById(int id);

    List<Store> findAll();

    void edit (int id , Store store ) ;
}
```

Στην υλοποίηση της διεπαφής 'StoreService' της κλάσης StoreServiceImlp, κάθε μέθοδος εκτελεί μια συγκεκριμένη λειτουργία και μπορεί να χρησιμοποιήσει το repository αν χρειαστεί να έχει πρόσβαση στη βάση δεδομένων. Τέλος, όταν η μέθοδος ολοκληρώσει την επεξεργασία της, επιστρέφει το αποτέλεσμα στον controller που την καλεί.

```java
@Service
public class StoreServiceImpl implements StoreService {

    @Autowired
    private StroreRepository stroreRepository;


    public StoreServiceImpl(StroreRepository stroreRepository) { this.stroreRepository = stroreRepository; }


    @Override
    public Store save(Store store) { return stroreRepository.save(store); }

    @Override
    public Store findById(int id) {
      return   stroreRepository.findById(id);
    }

    @Override
    public List<Store> findAll() { return stroreRepository.findAll(); }

    @Override
    public void edit(int id, Store store) {
        Store found = stroreRepository.findById(id);

        found.setTitle(store.getTitle());
        found.setAddress(store.getAddress());

        stroreRepository.save(found);
    }
}
```

Repositories

Το repository παρέχει τη σύνδεση με τη βάση δεδομένων που απαιτείται στα Services της εφαρμογής. Το repository παρέχει τις εξής μεθόδους που επιτρέπουν στις υπηρεσίες να τις καλούν για να εκτελούν διάφορες λειτουργίες στη βάση δεδομένων. Το παρακάτω σχήμα δείχνει την Αποθήκη για την οντότητα Αποθήκη. Οι μέθοδοι που παρουσιάζονται και υλοποιούνται στο εσωτερικό αφορούν την αναζήτηση όλων των αντικειμένων (findAll()) και την αναζήτηση με βάση κριτήρια (findById).

```
@Repository
public interface StroreRepository extends JpaRepository<Store, Integer> {

    Store findById(int id);

    List<Store> findAll();

}
```

Model

Ακολουθεί ένα μοντέλο της οντότητας Αποθήκη. Η σχέση των πεδίων της με άλλες οντότητες χρησιμεύει ως άμεση αναπαράσταση της βάσης δεδομένων.

```java
@Entity
@Table(name = "store", schema = "storage")
public class Store {
    private Integer id;
    private String title;
    private String address;
    private Integer enable;
    private Collection<Orders> ordersById;
    private Collection<User> usersById;

    @Id
    @Column(name = "id")
    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    @Basic
    @Column(name = "title")
    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    @Basic
    @Column(name = "address")
    public String getAddress() { return address; }

    public void setAddress(String address) { this.address = address; }

    @Basic
    @Column(name = "enable")
    public Integer getEnable() { return enable; }

    public void setEnable(Integer enable) { this.enable = enable; }

    @OneToMany(mappedBy = "storeByStoreId" ,fetch = FetchType.LAZY)
    public Collection<Orders> getOrdersById() { return ordersById; }

    public void setOrdersById(Collection<Orders> ordersById) { this.ordersById = ordersById; }
```

Ομοίως για κάθε οντότητα, υπάρχει και διαφορετικό model .

Interfaces

Κάθε λειτουργία που υλοποιείται στο σύστημα πρέπει να πραγματοποιείται από τρία άλλα στοιχεία: Controller, Repository και Service . Η επικοινωνία μεταξύ των τριών στοιχείων μπορεί να πραγματοποιηθεί μέσω διεπαφών. Ο ορισμός διεπαφών για την επικοινωνία μεταξύ των τριών στοιχείων επιτρέπει μεγαλύτερη ευελιξία στο σύστημα: εάν ένα από τα τρία στοιχεία πρέπει να αλλάξει μονομερώς, η αλλαγή αυτή δεν θα επηρεάσει τα άλλα δύο στοιχεία, εφόσον η επικοινωνία τους ορίζεται σε μια αμετάβλητη διεπαφή. Δεν επηρεάζει τα άλλα δύο στοιχεία. Αυτό λύνει επίσης το πρόβλημα πολλών controllers που χρησιμοποιούν τις ίδιες υπηρεσίες, αν θέλετε να εκτελούν τις ίδιες λειτουργίες από διαφορετικά περιβάλλοντα χρηστών.

Σύνοψη

Ο παραπάνω Controller "λαμβάνει" μόνο αιτήσεις από τον Container και δεν επεξεργάζεται ή εκτελεί άμεσα την αίτηση. Αντίθετα, χρησιμοποιεί μία ή περισσότερες υπηρεσίες που αναλαμβάνουν όλη την επεξεργασία της αίτησης. Ομοίως, εάν μια υπηρεσία πρέπει να επικοινωνήσει με μια βάση δεδομένων, δεν επικοινωνεί άμεσα αλλά έμμεσα χρησιμοποιώντας ένα ή περισσότερα Repositories. Ένα Repository είναι ένα κομμάτι κώδικα που είναι υπεύθυνο για την αλληλεπίδραση με μια βάση δεδομένων. Τα αποτελέσματα του Repository επιστρέφονται στην καλούμενη υπηρεσία. Όταν η υπηρεσία ολοκληρώσει την εργασία, τα αποτελέσματα επιστρέφονται στον controller, ο οποίος στη συνέχεια καλεί το κατάλληλο view (εάν απαιτείται).

Κεφάλαιο 6ο: Εκτέλεση εφαρμογής

Το IntelliJ IDEA 2019.1.4 και το MySQL Workbench 8.0 χρησιμοποιήθηκαν ως περιβάλλον εργασίας για τη δημιουργία μιας εφαρμογής που υλοποιεί τη διεπαφή χρήστη με τη βάση δεδομένων.
Τα στοιχεία για την βάση δεδομένων «**storage**» βρίσκονται στο αρχείο «application.properties» .
Στη συνέχεια, δημιουργείται μια βάση δεδομένων με όνομα «**storage**» στο ΣΔΒΒ και εκεί χρήζει η ένταξη των πινάκων και των δεδομένων (όπως στο αντίστοιχο αρχείο storage.sql).

Τα διαπιστευτήρια για τη βάση δεδομένων "storage" βρίσκονται στο αρχείο "application.properties".
Στη συνέχεια πρέπει να δημιουργήσουμε μια βάση δεδομένων με όνομα "storage" στο SDBB και να συγχωνεύσουμε τους πίνακες και τα δεδομένα (σύμφωνα με την εφαρμογή storage.sql).

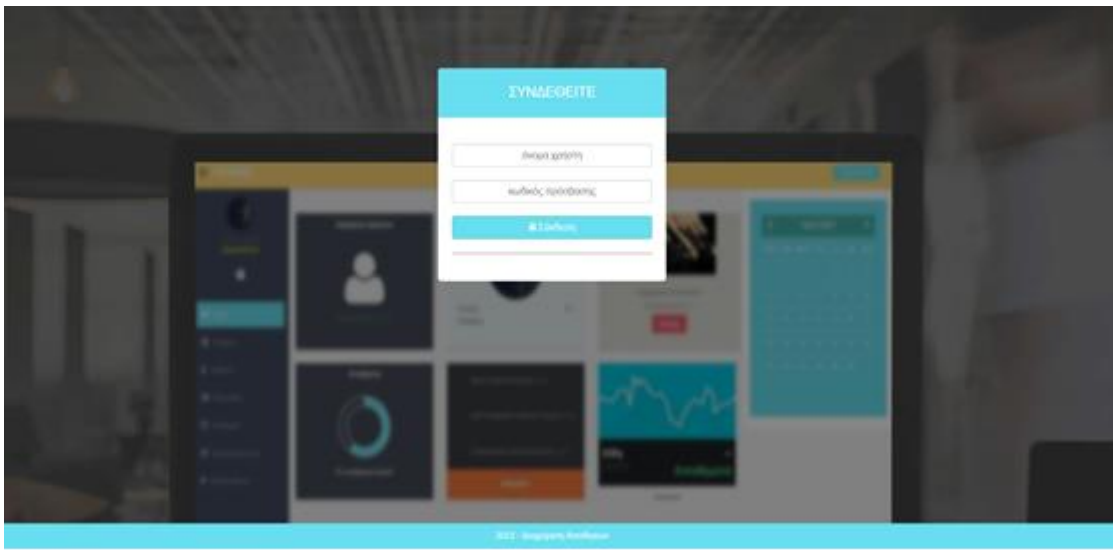
Τα διαπιστευτήριά του διαχειριστή είναι : admin / Admin!1234

Τα διαπιστευτήριά του τοπικού διαχειριστή είναι : user1 / User1!

Ο σύνδεσμος πλοήγησης είναι: http://localhost/storage/index

Κεφάλαιο 7ο: Παρουσίαση εφαρμογής

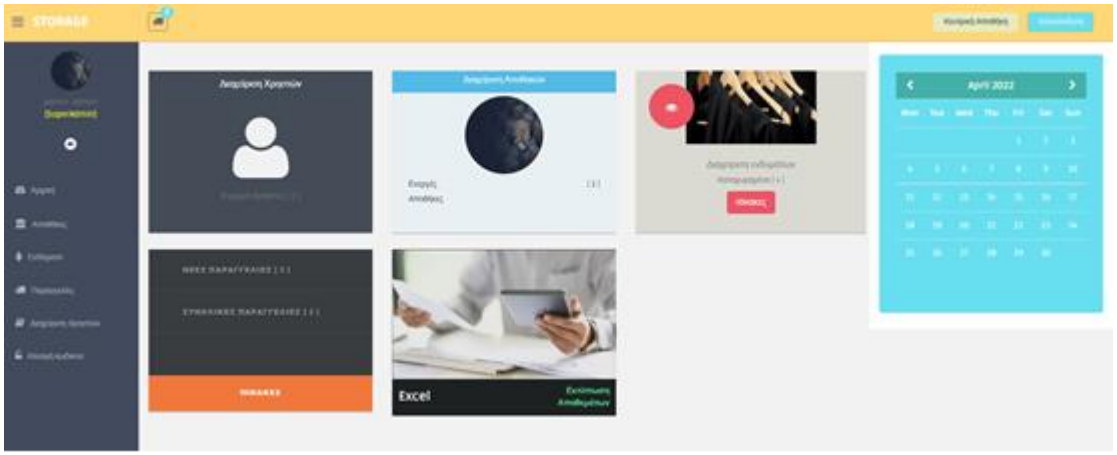
Είσοδος στο πρόγραμμα και καταχώρηση των διαπιστευτηρίων.



Εικόνα 8 Είσοδος χρήστη

Ο κωδικός χρήστη πρέπει να εγγραφεί στην λατινική, από τουλάχιστον 5 χαρακτήρες, με τουλάχιστον 1 κεφαλαίο γράμμα, 1 μικρό γράμμα, 1 αριθμό και 1 ειδικό χαρακτήρα {@ $! % * ? &}

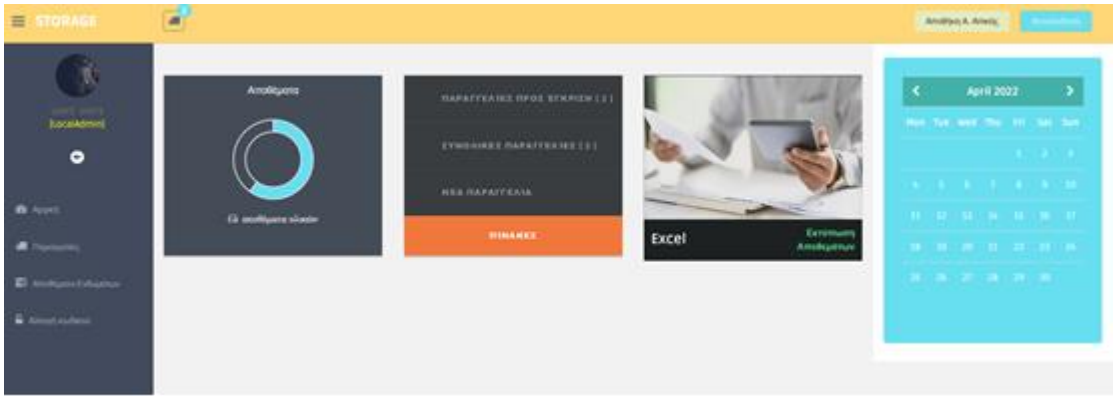
Αρχικό μενού



Εικόνα 9 Αρχικό Μενού

Αυτό το παράθυρο εμφανίζει τις επιλογές που έχει στη διάθεσή του ο διαχειριστής της εφαρμογής. (admin).

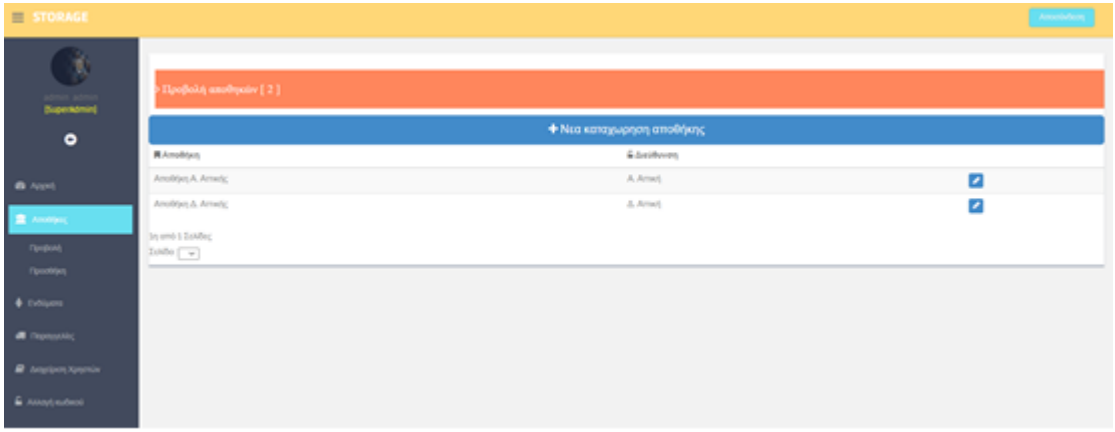
Από την άλλη, σε έναν τοπικό διαχειριστή αποθήκης, (user1) της εφαρμογής, οι επιλογές διαφοροποιούνται.



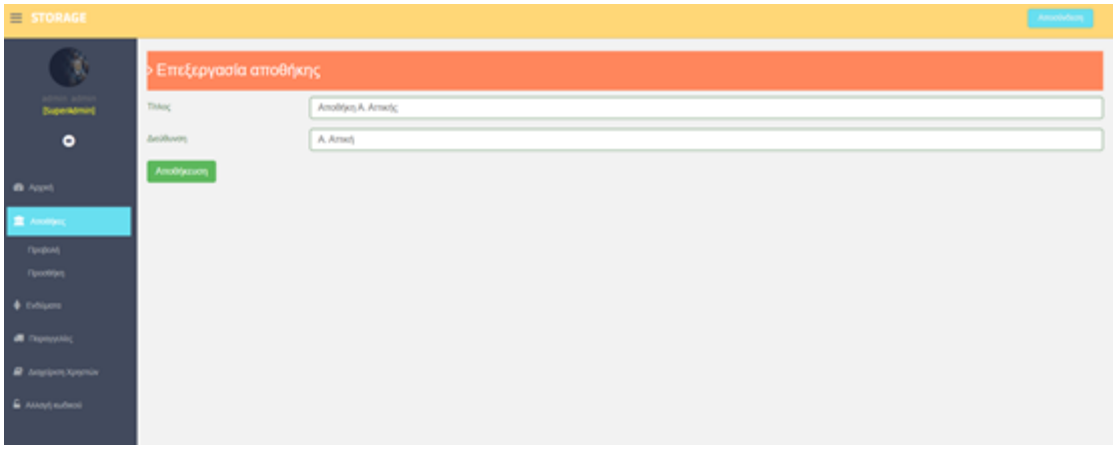
Εικόνα 10 Παράθυρο Διαχειριστή

Αποθήκες

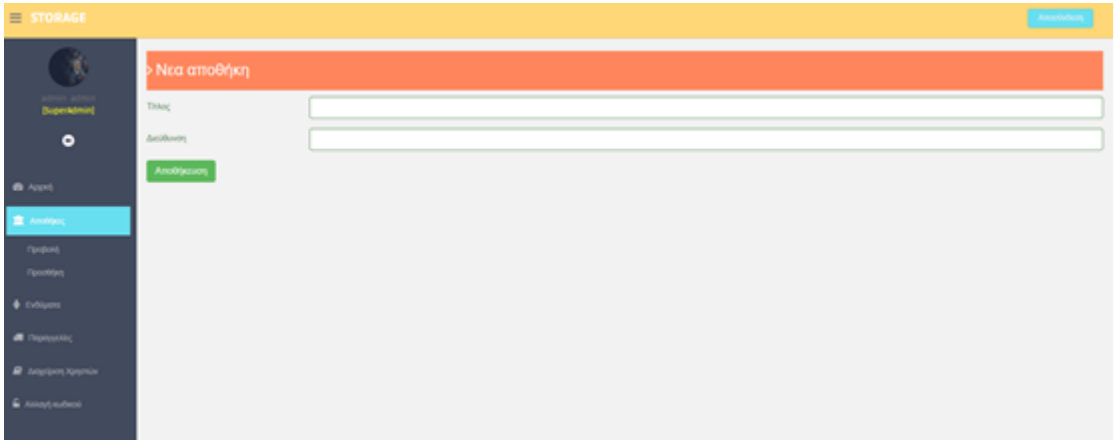
Εδώ ο κεντρικός διαχειριστής διαχειρίζεται τις αποθήκες, προσθέτει νέα. Επίσης βλέπει τις πληροφορίες τους.



Εικόνα 11 Διαχείριση Αποθήκης



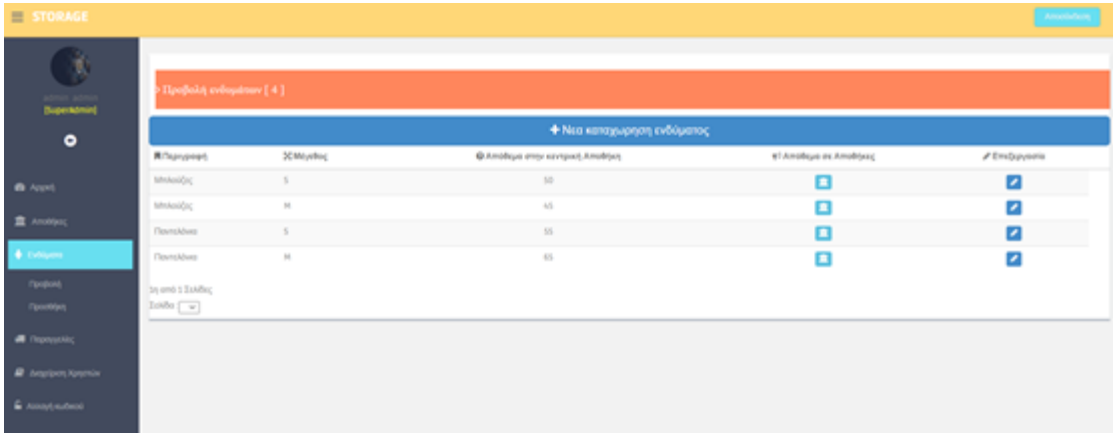
Εικόνα 12 Επεξεργασία Αποθήκης



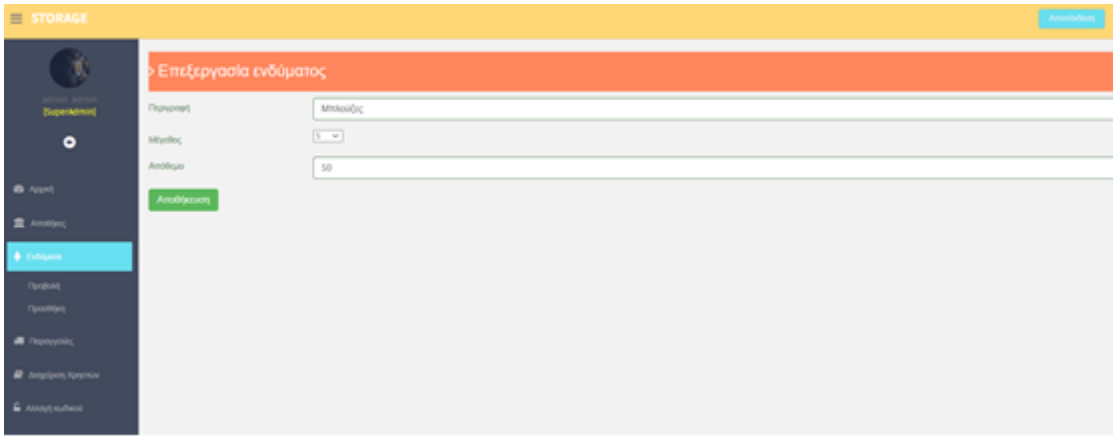
Εικόνα 13 Νέα καταχώρηση

Ενδύματα

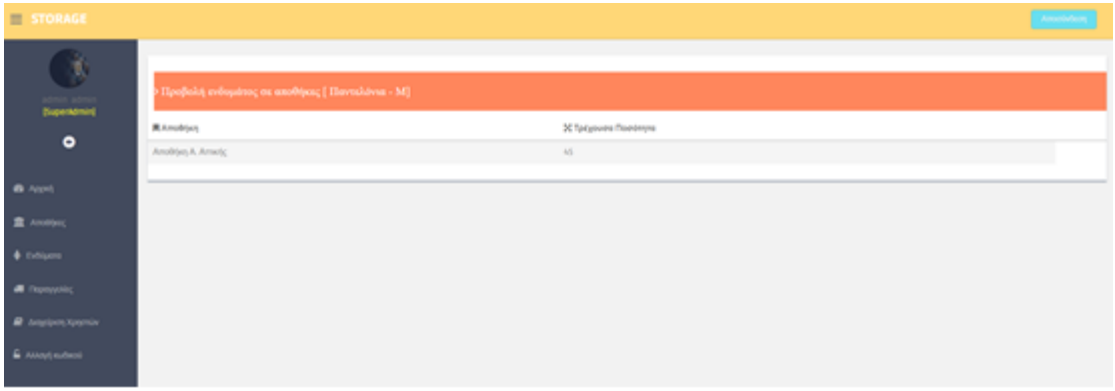
Εδώ ο κεντρικός διαχειριστής διαχειρίζεται τα ενδύματα, προσθέτει νέο. Επίσης διαχειρίζεται τα αποθέματα κάθε είδους που βρίσκονται στην κεντρική αποθήκη. Τέλος εποπτεύει τα αποθέματα κάθε είδους ανά αποθήκη.



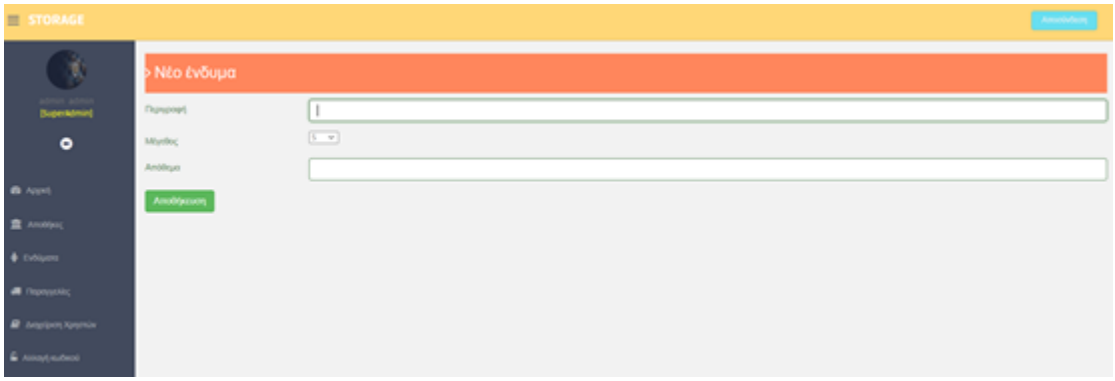
Εικόνα 14 Διαχείριση ενδυμάτων



Εικόνα 15 Επεξεργασία ενδυμάτων



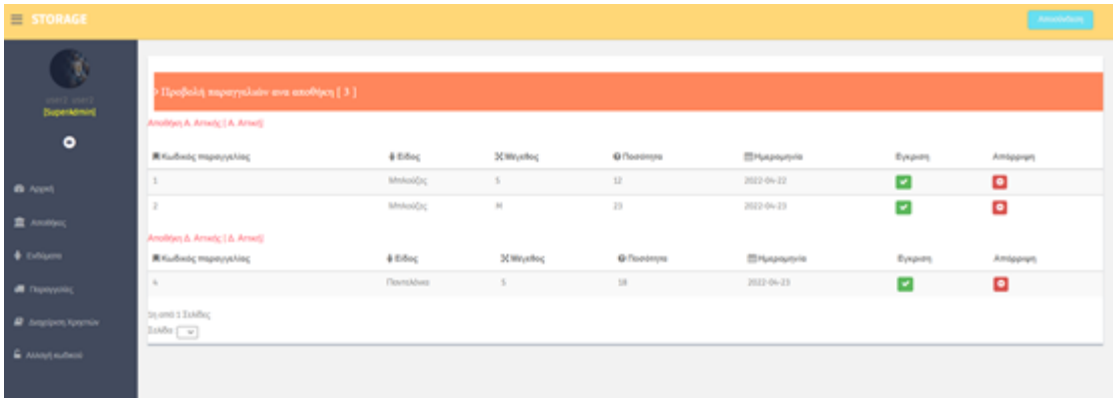
Εικόνα 16 Προβολή ενδυμάτων



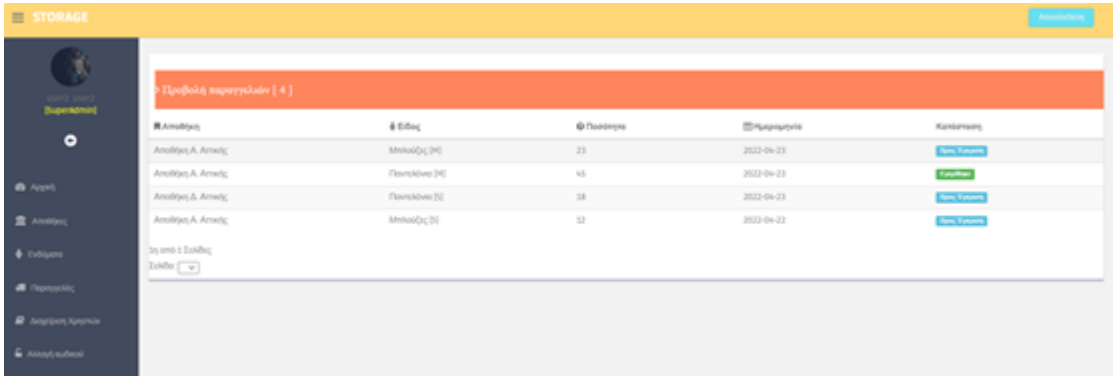
Εικόνα 17 Δημιουργία νέου ενδύματος

Παραγγελίες

Εδώ ο κεντρικός διαχειριστής βλέπει τις νέες παραγγελίες που έχουν υποβληθεί ανά αποθήκη. Για κάθε παραγγελία μπορεί να την εγκρίνει ή και να την απορρίψει. Όταν την εγκρίνει, η ζητούμενη ποσότητα αφαιρείται από το αρχικό απόθεμα της κεντρικής αποθήκης.

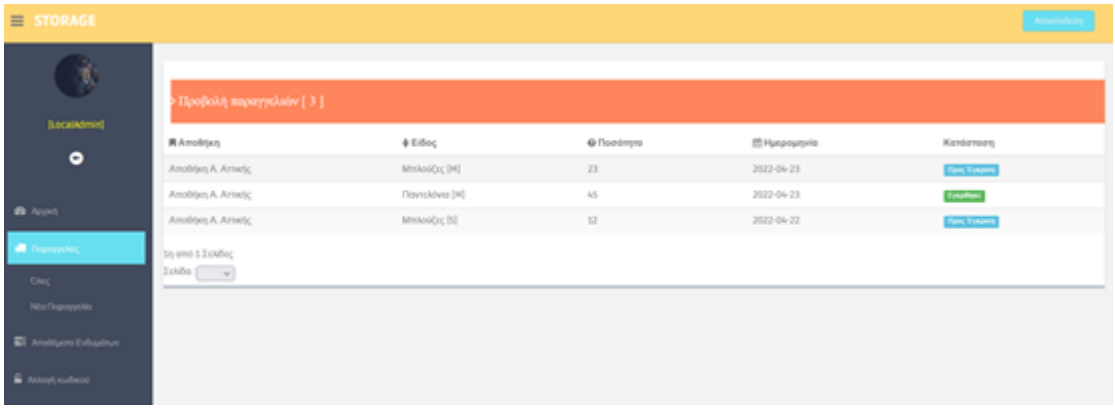


Εικόνα 18 Προβολή παραγγελιών

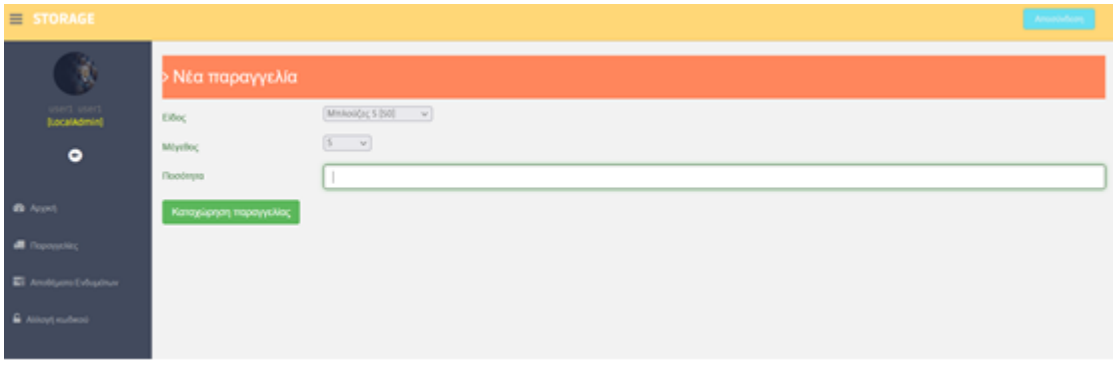


Εικόνα 19 Προβολή παραγγελιών 2

Ο τοπικός διαχειριστής βλέπει τις δικές του μόνο υποβαλλόμενες παραγγελίες.



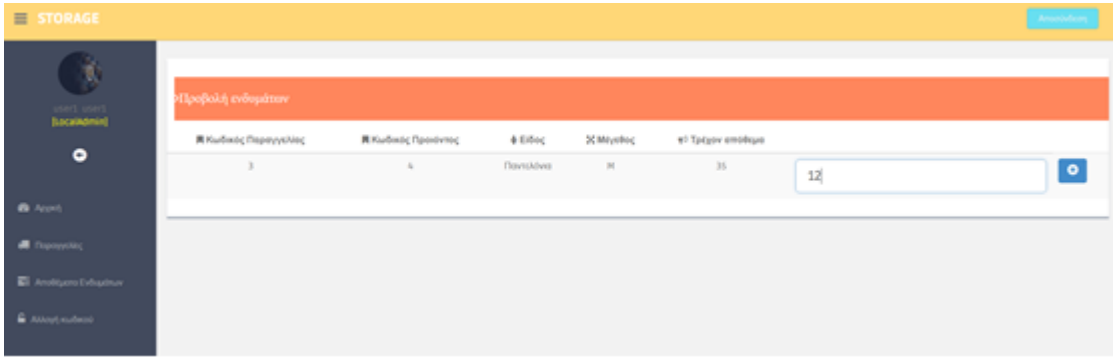
Εικόνα 20 Προβολή παραγγελιών 3



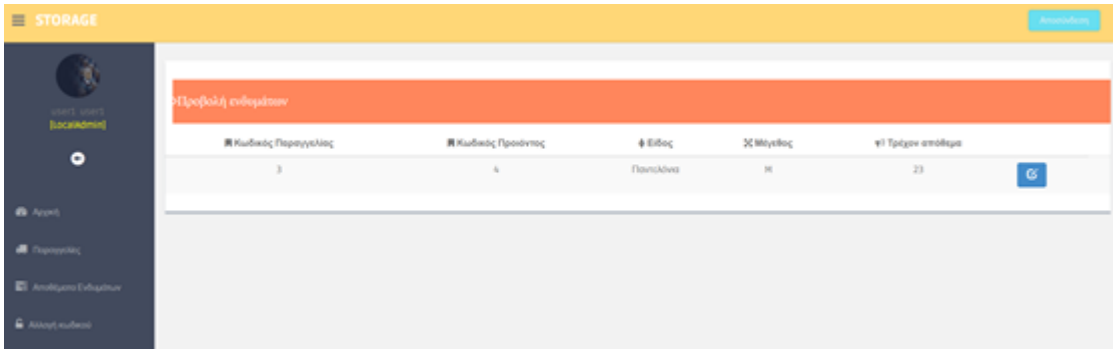
Εικόνα 21 Δημιουργία νέας παραγγελίας

Αποθέματα ενδυμάτων

Εδώ μόνο οι τοπικοί διαχειριστές έχουν πρόσβαση και μπορούν να τροποποιούν τα αποθέματα τους εφόσον καταχωρήσουν τις ποσότητες ενδυμάτων που πουλήθηκαν από την αποθήκη τους. Έτσι το τρέχον απόθεμα ενημερώνεται αυτόματα.



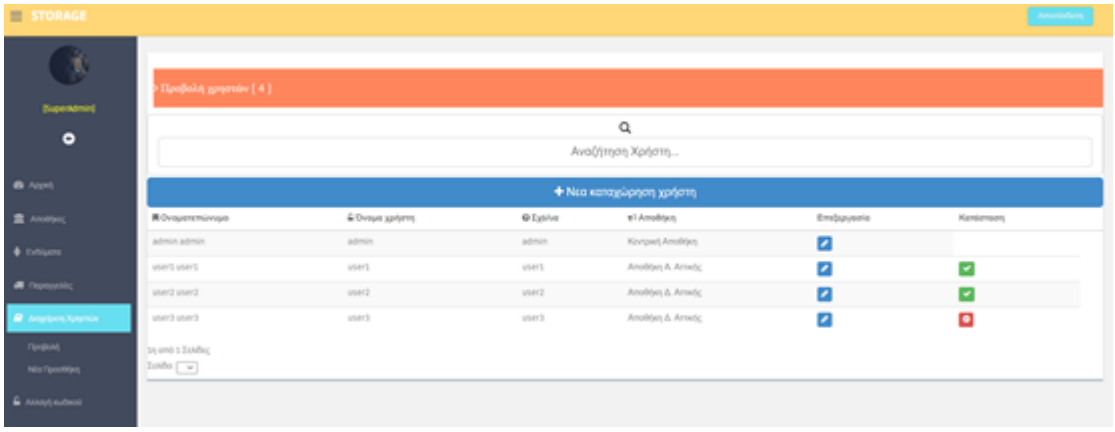
Εικόνα 22 Προβολή αποθεμάτων



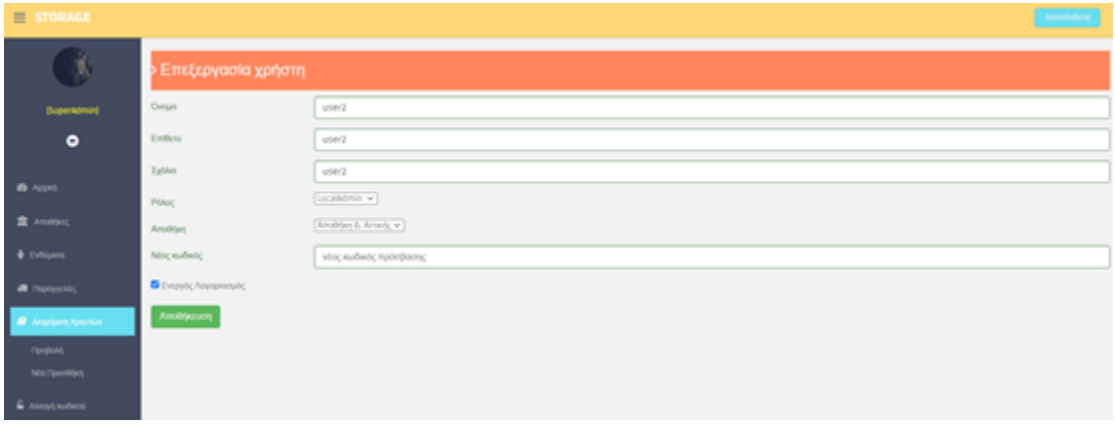
Εικόνα 23 Προβολή αποθεμάτων 2

Διαχείριση χρηστών

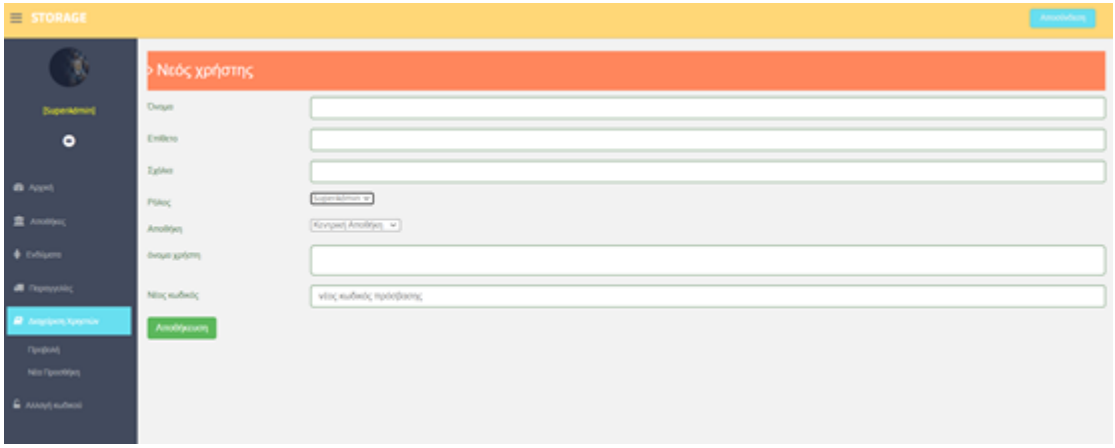
Εδώ ο κεντρικός διαχειριστής διαχειρίζεται τους χρήστες, προσθέτει νέο, αναζητά με βάση το ονοματεπώνυμο η username. Επίσης βλέπει ποιος χρήστης είναι ενεργοποιημένος και ποιος απενεργοποιημένος.



Εικόνα 24 Διαχείριση χρηστών

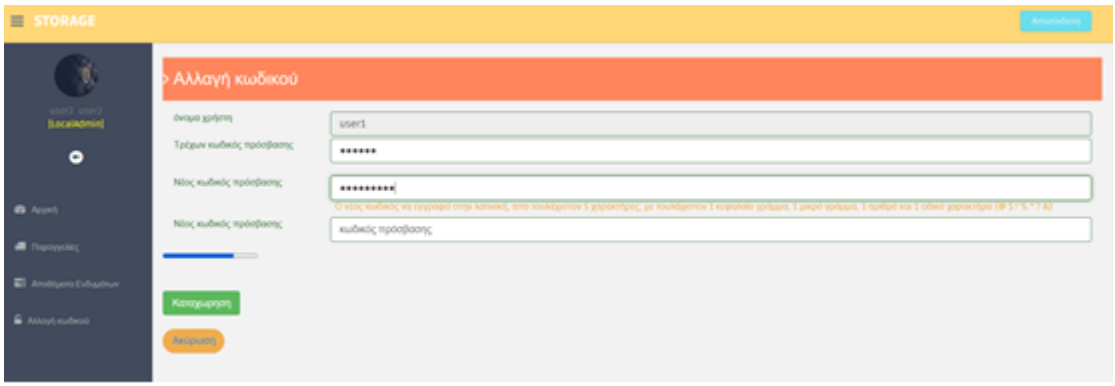


Εικόνα 25 Επεξεργασία χρηστών



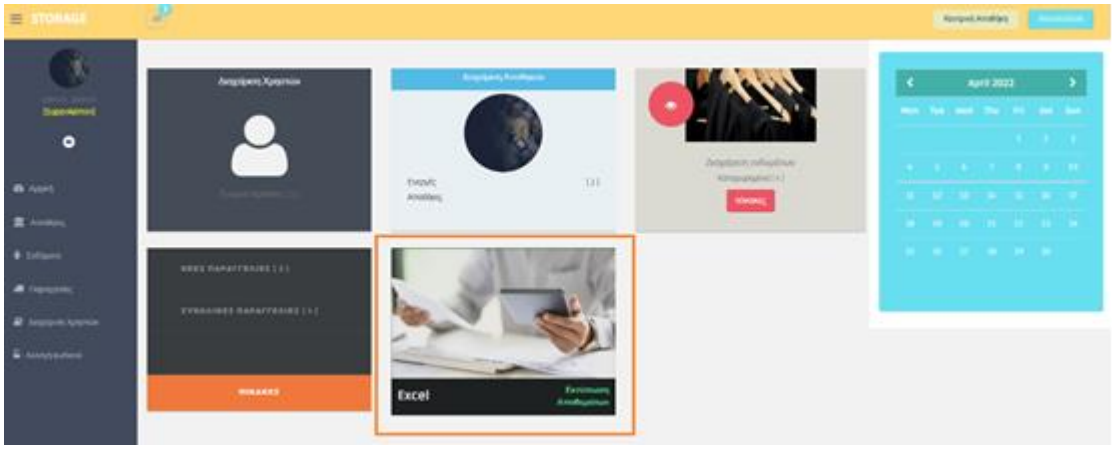
Εικόνα 26 Δημιουργία χρηστών

Αλλαγή Κωδικού



Εικόνα 27 Αλλαγή κωδικού

Εκτύπωση αποθεμάτων (με την επιλογή του σημειωμένου εικονιδίου)



Εικόνα 28 Εκτύπωση αποθεμάτων

Κεντρική αποθήκη :

	A	B	C	D	E	F
1	**Κωδικός προιόντος**		**Περιγραφή**	**Μέγεθος**	**Απόθεμα**	
2	1		Μπλούζες	S	50	
3	2		Μπλούζες	M	45	
4	3		Παντελόνια	S	55	
5	4		Παντελόνια	M	20	
6						

Δευτερεύουσα αποθήκη :

	A	B	C	D	E	F	G	H
1	Αποθήκη Α. Αττικής	**Κωδικός Παραγγελίας**	**Κωδικός προιόντος**		**Περιγραφή**	**Μέγεθος**	**Απόθεμα**	
2		3	4		Παντελόνια	M	23	
3								

Κεφάλαιο 8ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Η διαχείριση ενός δικτύου αποθηκών ενδυμάτων απαιτεί υπεύθυνη και καλά σχεδιασμένη διαχείριση. Αξιοποιώντας τη σύγχρονη τεχνολογία, η δουλειά του διαχειριστή μπορεί να απλοποιηθεί με πολλούς τρόπους. Η τρέχουσα εφαρμογή απευθύνεται στον τομέα της λογιστικής, καθώς έχει ως στόχο να παρέχει σε πραγματικό χρόνο εικόνα του τρέχοντος αποθέματος υλικών, το οποίο μεταβάλλεται συνεχώς καθώς αποστέλλονται σε άλλες αποθήκες. Η εφαρμογή έχει υλοποιηθεί σε ποικίλες γλώσσες προγραμματισμού στο παρελθόν, αλλά ήταν πρόκληση να υλοποιηθεί σε μια καινούρια, σύγχρονη και ευρέως χρησιμοποιούμενη τεχνολογία όπως η Spring Boot. Στο μέλλον, η εφαρμογή αυτή θα μπορούσε να χρησιμοποιηθεί στα χαμηλότερα επίπεδα της αποθήκης. Για παράδειγμα, θα μπορούσε να δημιουργηθεί μια διεπαφή με έναν απλό χρήστη που ζητάει ρούχα και παρακολουθεί τη ροή των παραγγελιών από τη δευτερεύουσα αποθήκη στην κύρια αποθήκη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Internet Site Spring Boot, Available:

https://www.baeldung.com/spring-boot.

[2] Spring Security, Available:

https://docs.spring.io/spring-security/ site/ docs/ current/ reference/ html5/

[3] Thymeleaf, Available:

https://www.baeldung.com/spring-boot-crud-thymeleaf,

https://www.thymeleaf.org/doc/articles/springsecurity.html

[4] Wikipedia:

https://el.m.wikipedia.org/wiki/Πληροφοριακά_συστήματα

[5] ΠΤΥΧΙΑΚΉ ΕΡΓΑΣΙΑ ΑΠΟ ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΚΗΣ:

http://repository.library.teiwest.gr/xmlui/bitstream/handle/123456789/6267/ΑΝΑΛΥΣΗ%20ΣΧΕΔΙΑΣΜΟΣ%20ΚΑΙ%20ΥΛΟΠΟΙΗΣΗ%20ΚΙΝΗΤΗΣ%20ΕΦΑΡΜΟΓΗΣ%20%28MOBILE%20APPLICATION%29%20ΚΑΙ%20ΔΙΑΣΥΝΔΕΣΗ%20ΤΗΣ%20ΜΕ%20ΤΑ%20ΠΛΗΡΟΦΟΡΙΑΚΑ%20ΣΥΣΤΗΜΑΤΑ%20ΜΙΑΣ%20ΕΠΙΧΕΙΡΗΣΗΣ..pdf?sequence=1&isAllowed=y

[6] ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΑΠΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ:

https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/4101/Filios.pdf?sequence=2&isAllowed=y

[7] ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕΙ ΚΡΗΤΗΣ:

https://www.teicrete.gr/ie/sites/teicrete.gr.ie/files/tmima_mihanikon_pliroforikis_tei_kritis_-_ptyhiakes_ergasies_tmimatos_mihanikon_pliroforikis_1.pdf

[8] ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ (ΙΕΠ):

Γ ΕΠΑΛ - ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΣΕ ΕΠΙΧΕΙΡΗΣΕΙΣ ΚΑΙ ΟΡΓΑΝΙΣΜΟΥΣ (iep.edu.gr)