ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΑΠΟΘΕΜΑΤΩΝ ΑΞΙΟΠΟΙΩΝΤΑΣ ΜΕΘΟΔΟΥΣ ΚΑΙ ΕΡΓΑΛΕΙΑ DEVOPS

Πτυχιακή εργασία

ΓΚΙΚΑΣ ΑΝΑΣΤΑΣΙΟΣ

Αθήνα, 2022

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATICS AND TELEMATICS

DEVELOPMENT OF INVENTORY MANAGEMENT APPLICATION UTILIZING DEVOPS METHODS AND TOOLS

Bachelor thesis

GKIKAS ANASTASIOS

Athens, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Επιβλέπων: Τσαδήμας Ανάργυρος
Ε.ΔΙ.Π., Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Μιχαήλ Δημήτριος
Δ.Ε.Π, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Τσερπές Κωνσταντίνος
Δ.Ε.Π, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Ο Γκίκας Αναστάσιος

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

‘Ο μόνος τρόπος για να κάνεις εξαιρετική δουλειά,
είναι να αγαπάς αυτό που κάνεις’

Steve Jobs

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα Πτυχιακή Εργασία εκπονήθηκε στα πλαίσια του προπτυχιακού προγράμματος του Τμήματος Πληροφορικής και Τηλεματικής, του Χαροκοπείου Πανεπιστημίου, υπό την επίβλεψη του κ. Τσαδήμα Ανάργυρου.

Αρχικά, θα ήθελα να ευχαριστήσω τον κύριο Τσαδήμα Ανάργυρο, μέλος Ε.ΔΙ.Π. του Τμήματος, ο οποίος υπήρξε επιβλέπων καθηγητής μου. Καθ' όλη τη διάρκεια της εκπόνησης της Πτυχιακής Εργασίας μου, με βοήθησε να εμπλουτίσω το γνωστικό πεδίο στο αντικείμενο της Πληροφορικής και κυρίως να με εξοικειώσει με νέες τεχνολογίες και μεθόδους, αναπτύσσοντας μεταξύ μας άριστη συνεργασία. Επίσης, θα ήθελα να ευχαριστήσω τα μέλη της εξεταστικής επιτροπής, κ. Μιχαήλ Δημήτριο, Επίκουρο Καθηγητή στο γνωστικό αντικείμενο Αλγόριθμοι και Πληροφορική, μέλος ΔΕΠ και τον κ. Τσερπέ Κωνσταντίνο, Αναπληρωτή Καθηγητή στο γνωστικό αντικείμενο Τεχνικές Προγραμματισμού Συστημάτων στον Παγκόσμιο Ιστό, μέλος ΔΕΠ, για τη πολύτιμη βοήθεια τους στην περάτωση των σπουδών μου. Πληροφορίες για τους κ. Μιχαήλ και κ. Τσερπέ στη σελίδα [professors-info].

Την ειλικρινή μου ευγνωμοσύνη θα ήθελα να δείξω και στους γονείς μου για τη συνεχή υποστήριξή τους καθ' όλη τη διάρκεια της φοίτησης προκειμένου να πετύχω τον απώτερο στόχο, αυτόν της αποφοίτησης και της απόκτησης των κατάλληλων γνώσεων. Η συμπαράσταση και η εμπιστοσύνη που έδειξαν στο πρόσωπό μου αποτέλεσε καθοριστικό παράγοντα αυτής της επιτυχίας.

Τέλος, οι φίλοι μου αποτέλεσαν και εκείνοι ένα επιπρόσθετο κίνητρο με τις παροτρύνσεις και συμβουλές τους σε όλη τη διάρκεια αυτών των χρόνων γεγονός το οποίο εκτιμώ ιδιαιτέρως.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη, υλοποίηση και παραμετροποίηση μιας διαδικτυακής εφαρμογής διαχείρισης αποθεμάτων που θα διευκολύνει ανθρώπους και επιχειρήσεις που δραστηριοποιούνται στον εμπορικό κλάδο να αυτοματοποιήσουν διάφορες διαδικασίες. Σε μία εποχή που αναδύονται συνεχώς νέες τεχνολογίες είναι σημαντικό οι άνθρωποι και οι επιχειρήσεις να προσαρμόζονται σε αυτά τα δεδομένα ώστε να βελτιστοποιούν την παραγωγικότητα τους.

Με βάση την αρχή αυτή, δημιουργήθηκε η εφαρμογή η οποία αφενός θα προσφέρει λειτουργίες σε εσωτερικούς χρήστες όπως διευθυντές και εργαζόμενους σχετικά με τη διαχείριση των αποθεμάτων, αφετέρου θα δίνει τη δυνατότητα και σε εξωτερικούς χρήστες να αλληλεπιδρούν και να πραγματοποιούν παραγγελίες. Το πιο ενδιαφέρον, όμως σημείο της παρούσας πτυχιακής εργασίας είναι η ενσωμάτωση τεχνολογιών στην εφαρμογή οι οποίες και αυτοματοποιούν όσο το δυνατόν περισσότερο τον κύκλο ανάπτυξης της και βοηθούν στην εκτέλεση της σε διαφορετικά περιβάλλοντα λογισμικού με τρόπο εύκολο, ασφαλή και γρήγορο.

Σημαντικό ρόλο, λοιπόν, στην υλοποίηση της εργασίας πέρα από την ανάπτυξη του βασικού συστήματος διαχείρισης αποθεμάτων, κατέχουν τα εργαλεία αυτοματοποίησης και κυρίως η παραμετροποίηση τους στα πλαίσια της ομαλής λειτουργικότητας με αυτό.

Το τεχνολογικό υπόβαθρο, στο οποίο αναπτύχθηκε η εφαρμογή είναι γλώσσες προγραμματισμού και λογισμικά όπως το Django framework το οποίο βασίζεται στη γλώσσα Python, η γλώσσα JavaScript, η HTML, CSS, Bootstrap, η χρήση βάσεων δεδομένων όπως η SQLite και η PostgreSQL, η χρήση wsgi και web servers όπως Gunicorn και Nginx, η χρήση εργαλείων αυτοματοποίησης, ενσωμάτωσης και παραμετροποίησης βασισμένα στη λογική του DevOps (Development-Operations) όπως Ansible, Jenkins, Docker, Kubernetes και η τέλος η χρήση εικονικών μηχανών (virtual machines) σε cloud πάροχο (βλ. Microsoft Azure).

Λέξεις κλειδιά: [διαχείριση αποθεμάτων, λογισμικό, αυτοματοποίηση, DevOps, Django]

Abstract

The main purpose of this thesis is the development, implementation and configuration of a web inventory management application that will facilitate people and industries operating in the commercial sector to automate different processes. In this age, where new technologies are emerging, it is highly important that people and industries can adapt so that they can maximize their production capabilities.

According to this purpose, a new application was created that on the one hand could offer new capabilities to internal users such as managers or employees regarding inventory management and on the other hand it could be possible for external users such as customers to interact with the application and place orders. However, the most interesting part of this thesis would be the integration of various technologies in the application that automate the software development lifecycle as much as possible and lead to the deployment in different environments in a safe, easy, and fast way.

Significant role to this implementation, except for the development of the main inventory management system, are playing the automation tools and mainly their configuration process in terms of normal intercommunication with the system.

The technological background, in which the application was developed, includes programming languages and software such as the Django framework based on Python language, the Javascript programming language, HTML, CSS, Bootstrap, the utilization of databases such as SQLite and PostgreSQL, the utilization of application and web servers such as Gunicorn and Nginx, the utilization of automation, integration and configuration management tools based on the DevOps concept such as Ansible, Jenkins, Docker, Kubernetes and finally the utilization of virtual machines on a cloud provider (e.g Microsoft Azure).

Keywords: [inventory management, software, automation, DevOps, Django]

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SQLite	Structured Query Lite
PostgreSQL	Postgre Structured Query Language
DevOps	**Dev**elopment-**Op**eration**s**
Vm	Virtual machine
CI/CD	Continuous Integration/Continuous Delivery
deploy	ανάπτυξη/εκτέλεση

Κεφάλαιο 1: Εισαγωγή

1.1 Εισαγωγή

Η σημασία του διαδικτύου εδώ και αρκετές δεκαετίες έχει γνωρίσει τεράστια αναγνώριση ανά τον κόσμο γεγονός που οφείλεται στη ραγδαία εξέλιξη της τεχνολογίας. Διανύουμε εξάλλου, την εποχή της πληροφορίας, γνωστή και ως ψηφιακή επανάσταση. Οι τεχνολογίες αναβαθμίζονται και εξελίσσονται, νέα εργαλεία ανάπτυξης λογισμικού αναδύονται όλο και πιο συχνά με αποτέλεσμα οι παραγωγικές δυνατότητες των κοινωνιών να αυξάνονται σημαντικά. Η εισχώρηση της τεχνολογίας και της πληροφορικής στον εμπορικό και παραγωγικό τομέα είναι αξιοσημείωτη και εκατομμύρια άνθρωποι και επιχειρήσεις ανά τον κόσμο κάνουν χρήση αυτών των δυνατοτήτων προκειμένου αναδειχθούν στον ολοένα και μεγαλύτερο ανταγωνισμό.

Στα πλαίσια αυτά, έχουν αναπτυχθεί εφαρμογές που βοηθούν επιχειρήσεις και ιδιώτες στην καθημερινότητα. Ένα σημαντικό ζήτημα, το οποίο λύθηκε με τη βοήθεια της τεχνολογίας ήταν ο τρόπος που θα μπορέσει να γίνει η διαχείριση των αποθεμάτων, με την αξιοποίηση κατάλληλων εφαρμογών προς αυτόν το σκοπό. Ωστόσο, το κυριότερο πρόβλημα δεν αποτέλεσε αυτό εδώ και χρόνια. Μπορεί η χρήση διαφόρων τέτοιων εφαρμογών να έλυνε πολλά προβλήματα αλλά όχι όλα τελικά.

Αναζητήθηκαν ποικίλοι τρόποι επίλυσης των προβλημάτων που προκύπτουν από τις παραδοσιακές μεθόδους προγραμματισμού (μεγάλες και ασύγχρονες εκδόσεις του λογισμικού, δύσκολος χρονοπρογραμματισμός, έλλειψη επικοινωνίας μεταξύ operators και developers κ.α). Την τελευταία δεκαετία η λογική του DevOps έχει έρθει να αντικαταστήσει την κλασική μέθοδο προγραμματισμού και να κάνει διαφορετικές ομάδες ενός project να συνεργάζονται προκειμένου να βελτιστοποιήσουν την παραγωγή και παράδοση υπηρεσιών και εφαρμογών σε μεγάλη ταχύτητα.

Στην παρούσα πτυχιακή εργασία, αναπτύχθηκε μία πλατφόρμα η οποία προσφέρει διάφορες λειτουργίες σχετικά με την διαχείριση των αποθεμάτων (όπως προσθήκη/διαγραφή/επεξεργασία προϊόντων, καταχώρηση/διαχείριση παραγγελιών, διαχείριση εταιρειών/εργαζομένων) και θα αξιοποιεί την λογική του DevOps προκειμένου να παράγει το λογισμικό αυτό με μεγαλύτερη, ταχύτητα, ασφάλεια, αποτελεσματικότητα και επεκτασιμότητα.

1.2 Στόχος

Η ηλεκτρονική υπηρεσία διαχείρισης αποθεμάτων αξιοποιώντας τη μέθοδο του DevOps έχει ως στόχο να διευκολύνει την επαγγελματική ζωή των ανθρώπων ώστε με τη βοήθεια των προγραμματιστών να αυτοματοποιούν διαδικασίες, να πετυχαίνουν τη συνεχή ενσωμάτωση (CI) και προώθηση σε περιβάλλον παραγωγής (CD) της εφαρμογής μετά από κάθε αλλαγή που θα είναι προσαρμοσμένη στις ανάγκες τους, να έχουν τη δυνατότητα παραμετροποίησης, επεκτασιμότητας της εφαρμογής σε διαφορετικά περιβάλλοντα εκτέλεσης αποφεύγοντας τα προβλήματα που προκύπτουν από την αναχρονιστική μέθοδο προγραμματισμού. Περισσότερες λεπτομέρειες σχετικά με αυτή την ενδιαφέρουσα τεχνολογία στα Κεφάλαια 3 και 5.

1.3 Δομή

Στο Κεφάλαιο 1 γίνεται εισαγωγή, ανασκόπηση και παρουσίαση της εφαρμογής και του τρόπου που αυτή είναι αναγκαία και βοηθά ανθρώπους και επιχειρήσεις.

Στο Κεφάλαιο 2 γίνεται αναφορά στην αρχιτεκτονική στην οποία βασίστηκε η παρούσα πτυχιακή εργασία για να υλοποιηθεί, καταγράφονται οι αναλύσεις απαιτήσεων του συστήματος και απεικονίζονται τα σχεδιαγράμματα που δείχνουν οντότητες-μοντέλα-ροές και την αλληλεπίδραση τους.

Στο Κεφάλαιο 3 περιγράφονται αναλυτικά τα πρότυπα των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της συγκεκριμένης πλατφόρμας, τα εργαλεία και οι μέθοδοι αυτοματοποίησης βασισμένα στη λογική DevOps καθώς και τα περιβάλλοντα στα οποία κατασκευάστηκε και εκτελέστηκε η εφαρμογή.

Στο Κεφάλαιο 4 περιγράφεται η υλοποίηση, η παραμετροποίηση και παραδοχές που έγιναν για το βασικό σύστημα καθώς και οι λειτουργικότητες που υποστηρίζονται.

Στο Κεφάλαιο 5 γίνεται λεπτομερής αναφορά στα εργαλεία και τις μεθόδους DevOps που χρησιμοποιήθηκαν και τον τρόπο που αυτά ενσωματώθηκαν στην εφαρμογή.

Στο Κεφάλαιο 6 περιγράφονται τα σενάρια χρήσης για κάθε ρόλο που περιλαμβάνεται στην εφαρμογή.

Στο Κεφάλαιο 7 αναφέρονται τα συμπεράσματα, οι στόχοι που επιτεύχθηκαν από το σύνολο της πτυχιακής εργασίας και πιθανές βελτιώσεις που θα μπορούσαν να γίνουν.

Τέλος, γίνεται παράθεση της βιβλιογραφίας και των αποθετηρίων κώδικα που χρησιμοποιήθηκαν.

Κεφάλαιο 2: Ανάλυση και Σχεδίαση της Εφαρμογής

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστεί η σχεδίαση και η αρχιτεκτονική σύμφωνα με την οποία έγινε η υλοποίηση της εφαρμογής. Θα γίνει, αρχικά, αναφορά στους ρόλους της εφαρμογής, στις απαιτήσεις του κάθε ρόλου μέσα από διαγράμματα όπως επίσης και στο σχεδιάγραμμα της αρχιτεκτονικής όπου θα φανεί η σημασία των εργαλείων DevOps που χρησιμοποιήθηκαν. Στο κεφάλαιο 4 αναλύονται και οι παραδοχές που έχουν γίνει σχετικά με την εφαρμογή.

2.2 Ανάλυση Απαιτήσεων

Η παρούσα εφαρμογή υποστηρίζει τους εξής ρόλους: προϊστάμενος (manager), εργαζόμενος (employee), πελάτης (customer) και διαχειριστής συστήματος (admin).

<u>Για τον προϊστάμενο:</u>

- Να μπορεί επεξεργάζεται (update) κάποιο προϊόν, ανανεώνοντας τα στοιχεία του (π.χ ανανέωση διαθέσιμου αποθέματος).
- Να μπορεί να διαγράψει κάποιο προϊόν.
- Να αναζητεί προϊόντα με βάση κάποιο κριτήριο (π.χ τιμή, απόθεμα, κατηγορία) και να ταξινομεί προϊόντα με βάση την τιμή.
- Να μπορεί να αναζητά εργαζόμενους και να ανενεώνει κάποια πληροφορία τους.
- Να μπορεί να βλέπει τις διαθέσιμες εταιρείες, να ανανεώνει τα στοιχεία τους και να έχει τη δυνατότητα να κάνει deactivate (όχι delete).

<u>Για τον εργαζόμενο:</u>

- Να μπορεί να βλέπει τις παραγγελίες που έχουν καταχωρηθεί από τους πελάτες.
- Να μπορεί να ανανεώνει την κατάσταση μιας παραγγελίας.

- Να μπορεί να αναζητά παραγγελίες, να βλέπει τα προϊόντα που περιέχονται στην παραγγελία και τις πληροφορίες παράδοσης του πελάτη.
- Να μπορεί να αναζητά πελάτες με βάση κάποια πληροφορία τους (π.χ username).

Για τον πελάτη:

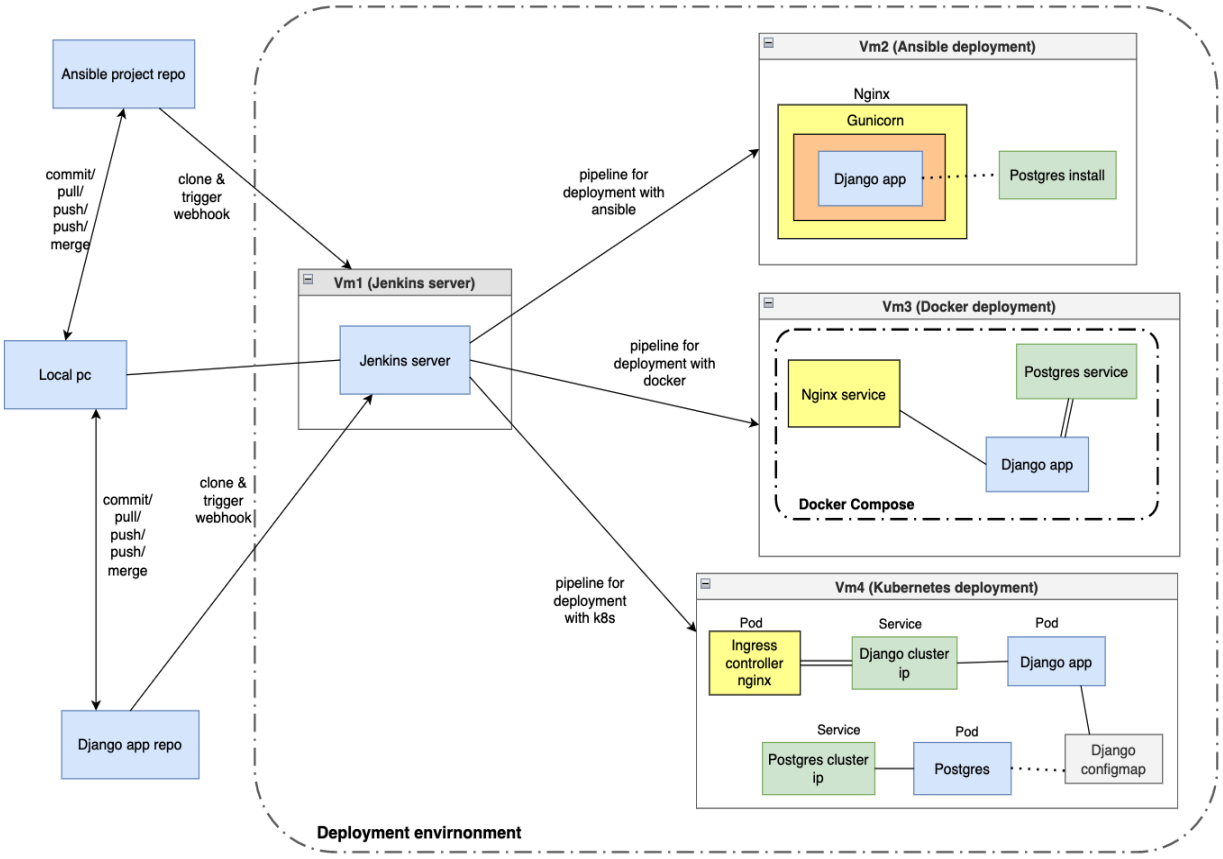
- Να μπορεί να επιλέξει κατηγορία προιόντος, να δει και να αναζητήσει προϊόντα με βάση κάποιο κριτήριο.
- Να προσθέσει προϊόντα στο καλάθι.
- Να μπορεί να δει το καλάθι του.
- Να μπορεί να δει τις παραγγελίες που έχει καταχωρήσει, τα προϊόντα μιας παραγγελίας του και τις πληροφορίες παράδοσης που έχει προσθέσει.
- Να πληρώσει μια παραγγελία.
- Να λάβει email επιβεβαίωσης σε περίπτωση καταχώρησης νέας παραγγελίας ή σε αλλαγής status παραγγελίας.

Για τον διαχειριστή συστήματος:

- Όλα τα παραπάνω
- Να μπορεί να βλέπει τους χρήστες που έχουν κάνει αίτημα για εγγραφή στο σύστημα (ως employees/managers ή ως customers)
- Να αποδέχεται ή να απορρίπτει τα αιτήματα εγγραφής στο σύστημα.

2.3 Σχεδιαγράμματα

Στις παρακάτω εικόνες διακρίνεται η αρχιτεκτονική του συστήματος καθώς επίσης τα διαγράμματα use case, class και activity.

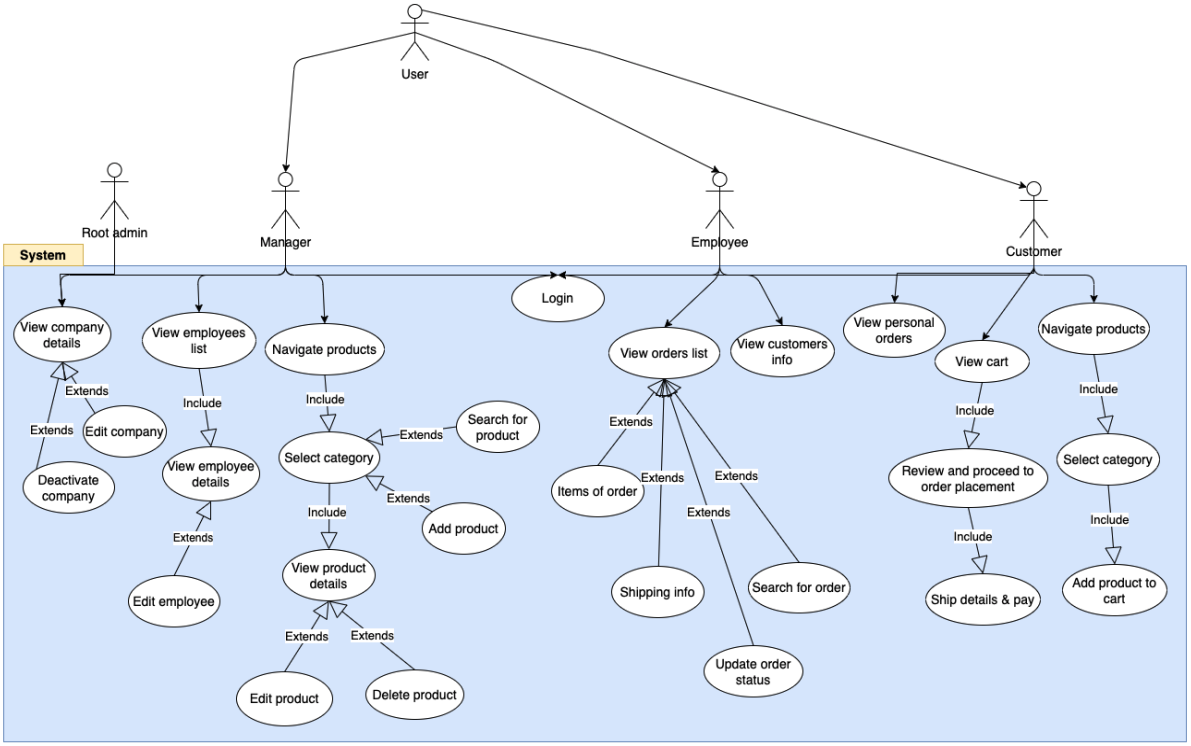


Σχήμα 1 Αρχιτεκτονική συστήματος

Στο παραπάνω διάγραμμα αρχιτεκτονικής παρατηρούμε τους τρόπους που έχει γίνει deploy η εφαρμογή μας δηλαδή τα devops εργαλεία που χρησιμοποιήθηκαν, τις τεχνολογίες για κάθε deployment, τον Jenkins server που βοηθά στην αυτοματοποίηση αυτής της διαδικασίας, τις εικονικές μηχανές (virtual machines) στις οποίες έγιναν deploy η εφαρμογή μας [12] και τέλος δύο αποθετήρια κώδικα (ansible project repo, django app repo [11]) στα οποία περιέχεται ο κώδικας της εφαρμογής.

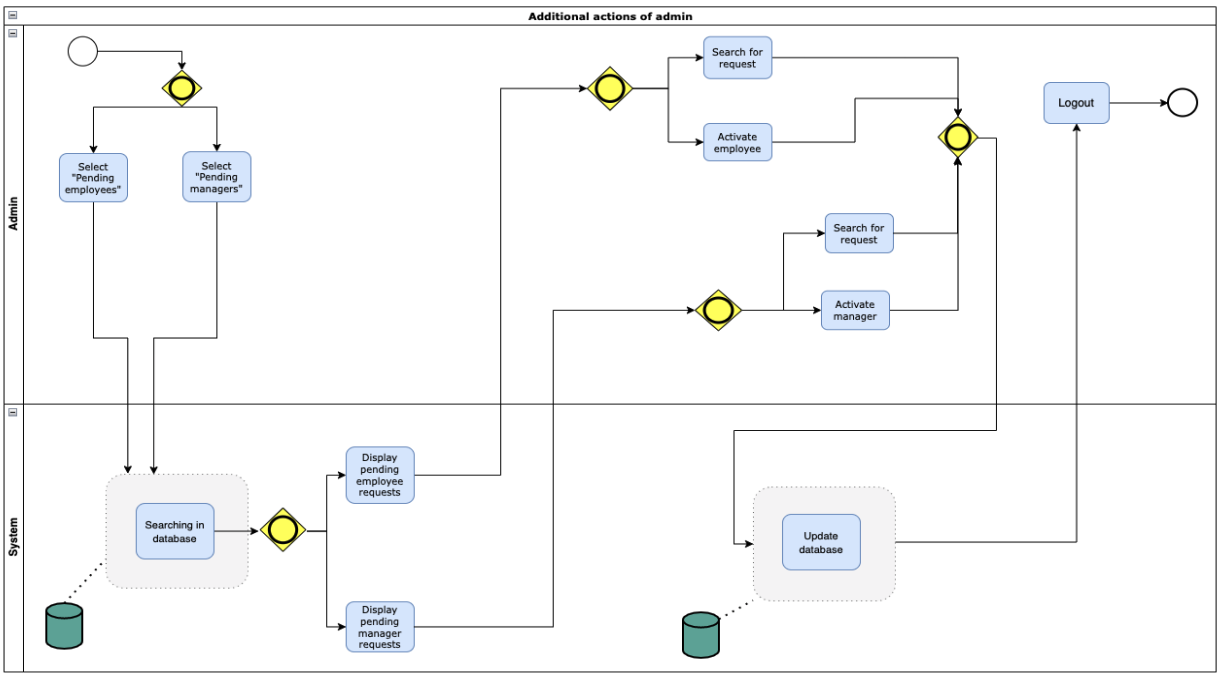
Τόσο τοπικά ο υπολογιστής μας όσο και οι εικονικές μηχανές (vm2, vm3, vm4) όπου γίνεται deploy η εφαρμογή, η σύνδεση και η επικοινωνία γίνεται με κλειδιά ssh.

Προκειμένου να γίνει το deployment από το vm1 στα υπόλοιπα 3 vms και να μπορούμε να δούμε την εφαρμογή από τον υπολογιστής μας τοπικά, χρησιμοποιήθηκαν τα Jenkins pipelines (αναλυτικά στα Κεφάλαια 3 και 5).



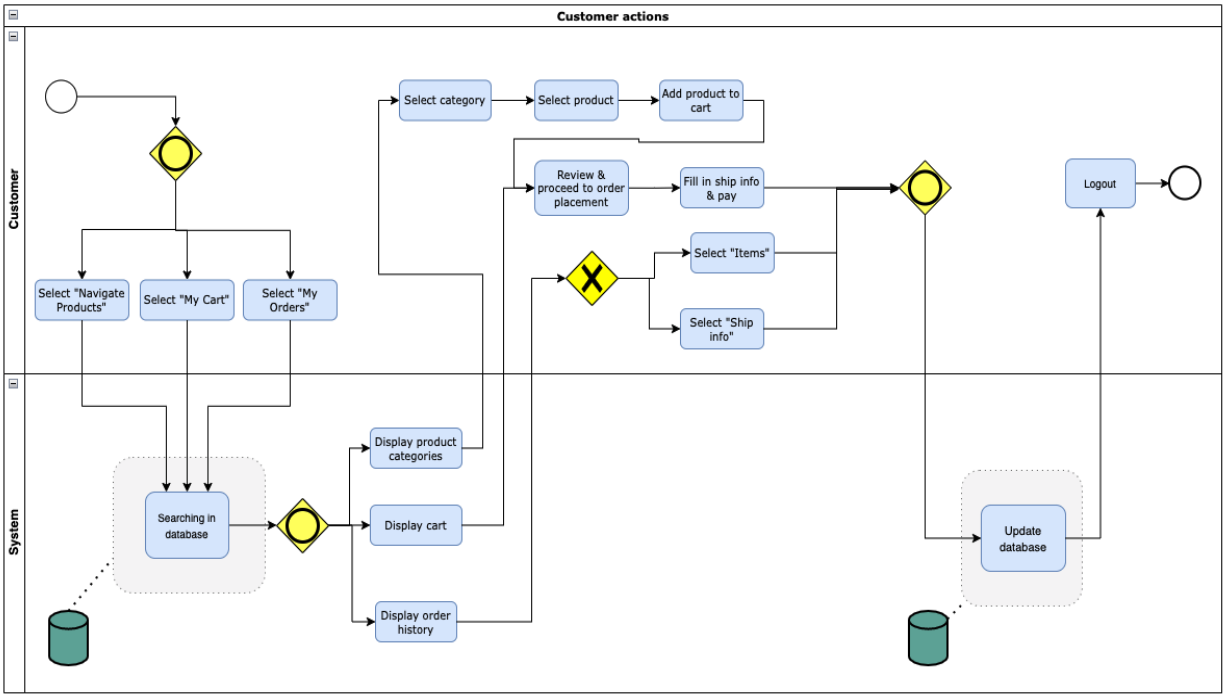
Σχήμα 2 Use Case Diagram

Στο παραπάνω διάγραμμα αποτυπώνονται όλες οι απαιτήσεις των χρηστών που προαναφέρθηκαν στο κεφ. 2.2. Εξάλλου, ο σκοπός του use case διαγράμματος είναι να απεικονιστούν οι διαφορετικοί ρόλοι σε ένα σύστημα και η αλληλεπίδρασή τους με αυτό, σύμφωνα με μια υψηλού επιπέδου προβολή της λειτουργικότητας του.



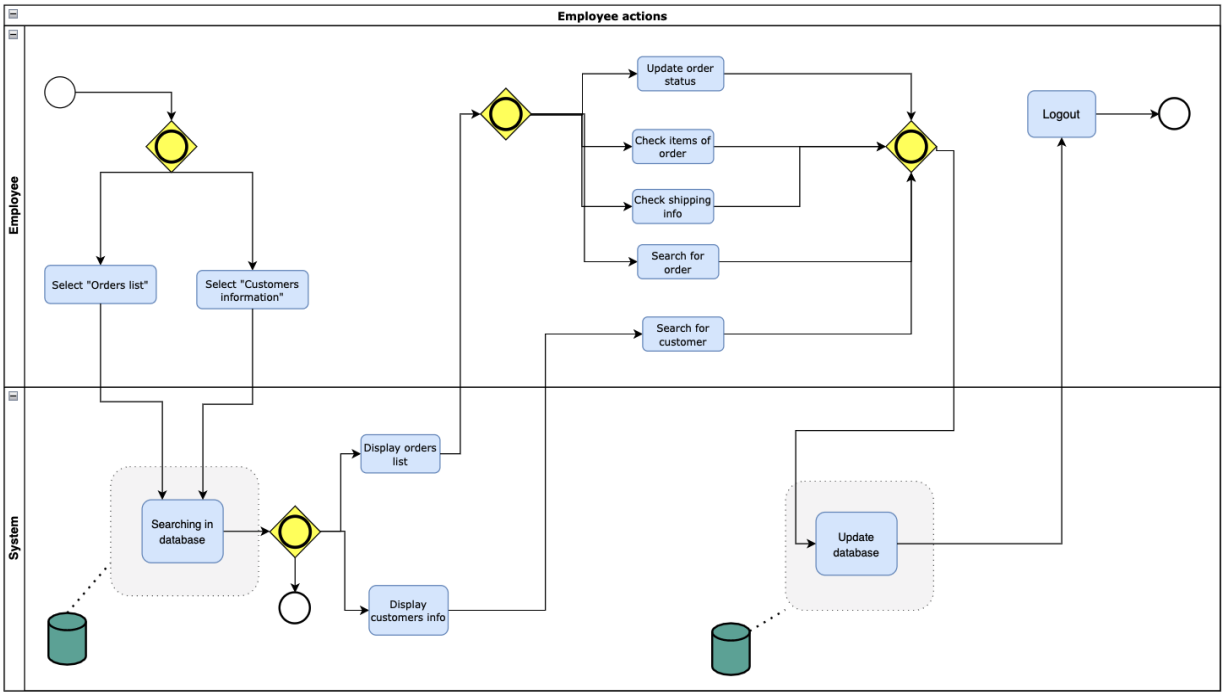
Σχήμα 3 Activity Diagram (admin)

Στο παραπάνω activity διάγραμμα φαίνονται οι επιπλέον ενέργειες που κάνει ένας διαχειριστής όπως αναφέρθηκαν και στην ανάλυση απαιτήσεων. Τα activity διαγράμματα περιγράφουν τη συμπεριφορά ενός ρόλου στο σύστημα και μία ροή από το αρχικό σημείο προς το τελικό έχοντας ενδιάμεσα ορισμένα σημεία απόφασης καθώς και την αλληλεπίδραση με components όπως μια βάση δεδομένων.



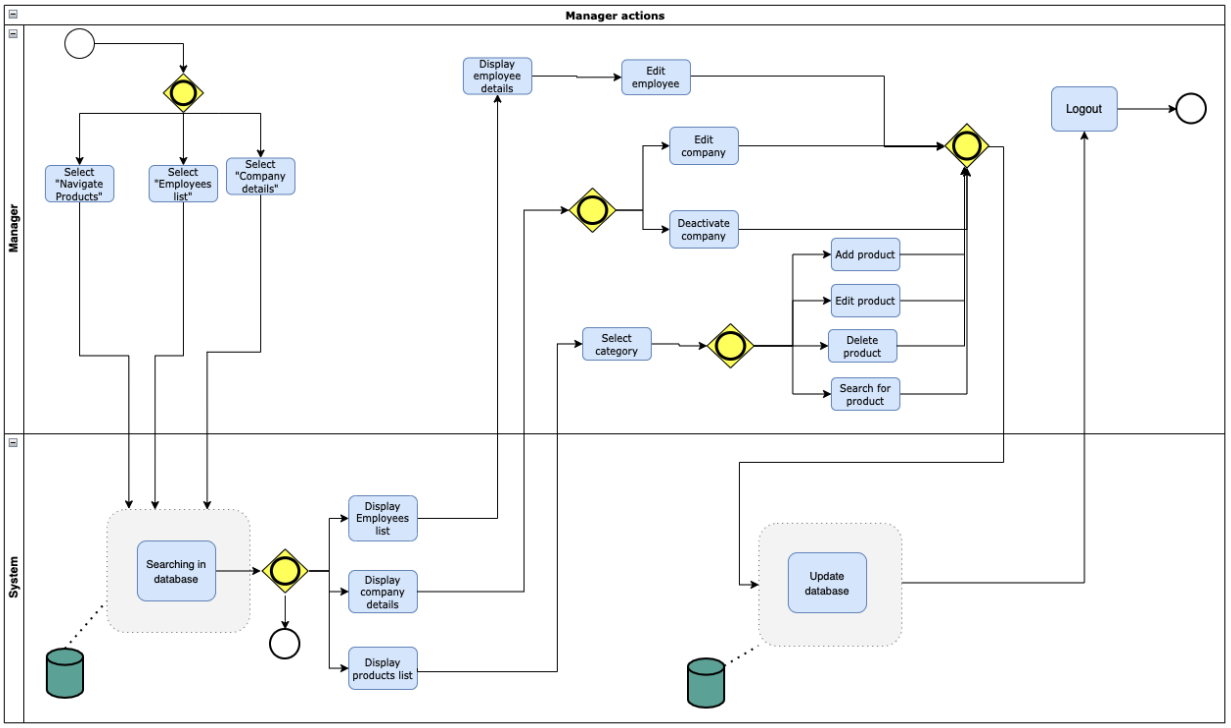
Σχήμα 4 Activity Diagram (customer)

Όπως και στο παραπάνω activity διάγραμμα, έτσι και σε αυτό βλέπουμε τις ενέργειες που πραγματοποιεί ένας customer όπως περιγράφηκαν, μαζί με κάποια σημεία απόφασης.



Σχήμα 5 Activity Diagram (employee)

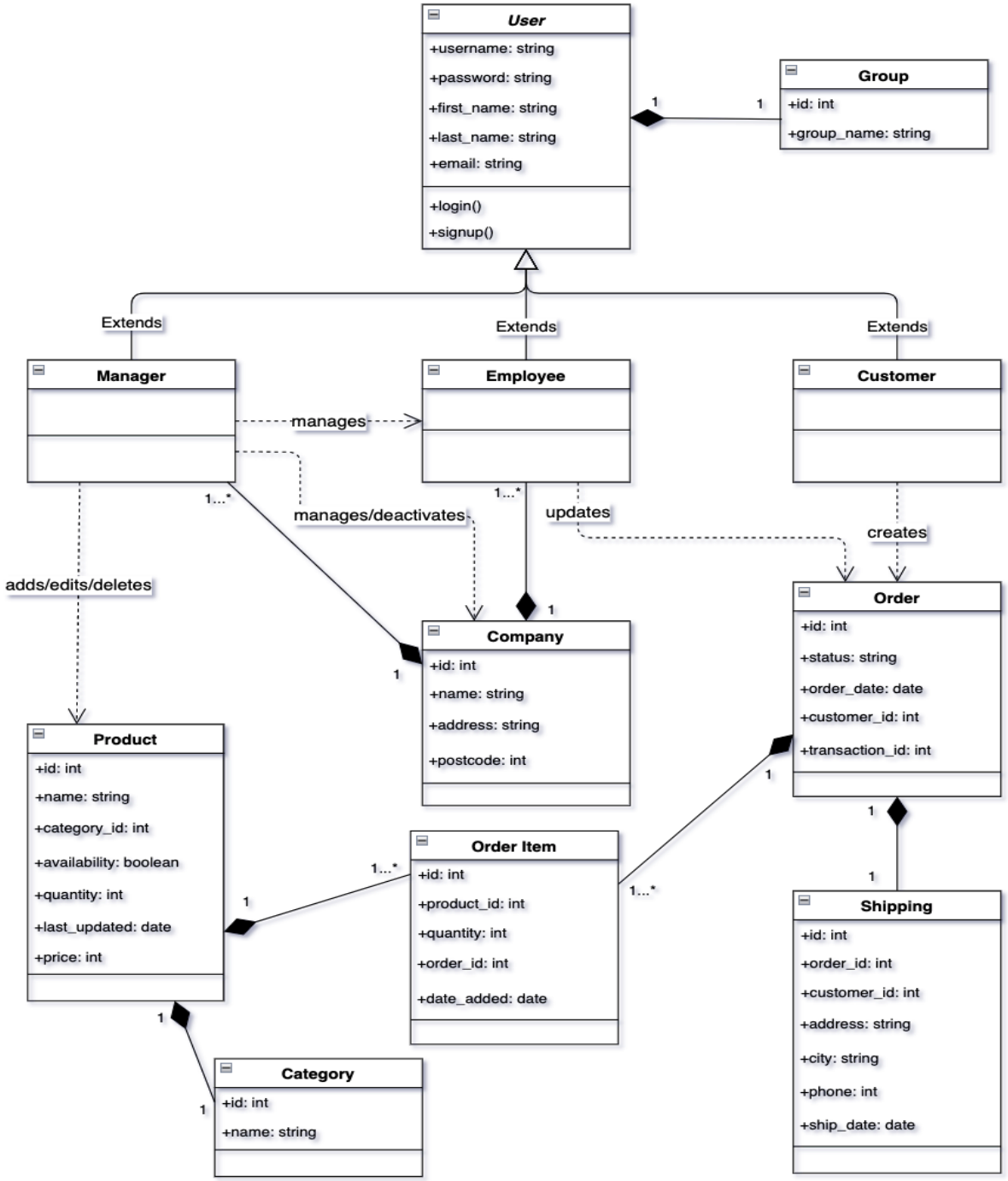
Όπως και στα παραπάνω activity διαγράμματα, έτσι και σε αυτό βλέπουμε τις ενέργειες που πραγματοποιεί ένας employee όπως περιγράφηκαν, μαζί με κάποια σημεία απόφασης.



Σχήμα 6 Activity Diagram (manager)

Όπως και στα παραπάνω activity διαγράμματα, έτσι και σε αυτό βλέπουμε τις ενέργειες που πραγματοποιεί ένας manager όπως περιγράφηκαν, μαζί με κάποια σημεία απόφασης.

Προκειμένου να γίνει πιο ξεκάθαρη η αλληλεπίδραση μεταξύ των αντικειμένων, των μοντέλων και των ρόλων της εφαρμογής, δημιουργήθηκε το class διάγραμμα το οποίο και δείχνει αυτή τη σύνδεση μεταξύ των οντοτήτων. Θα μπορούσε να θεωρηθεί ως κάποιο βαθμό ότι τα μοντέλα αυτά απεικονίζουν την μορφή που έχει η βάση δεδομένων μας.



Σχήμα 7 Class Diagram

Κεφάλαιο 3: Περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν

3.1 Python - Django Framework

Η Python είναι μία διερμηνευόμενη δυναμική γλώσσα προγραμματισμού υψηλού επιπέδου και υποστηρίζει το διαδραστικό και το αντικειμενοστραφές μοντέλο. Δημιουργήθηκε από τον Ολλανδό Γκίντο βαν Ρόσσουμ στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε για πρώτη φορά το 1991.

Το Django είναι ένα από τα δημοφιλέστερα web frameworks και έχει γραφτεί σε γλώσσα Python. Ένα web framework είναι ένα σύνολο από στοιχεία που μας βοηθά να αναπτύξουμε ιστοσελίδες πιο γρήγορα και εύκολα. Το κυριότερο είναι ότι το Django προσφέρει μία σειρά από χαρακτηριστικά όπως είναι οι web servers, διακομιστές δηλαδή οι οποίοι διαβάζουν μηνύματα και αποκρίνονται με μία ιστοσελίδα. Γι' αυτό το Django βοηθάει ώστε να δημιουργηθεί το περιεχόμενο που θα σταλεί ώστε να εμφανιστεί η ιστοσελίδα. Βασίζεται στο μοτίβο MTV (Model-Template-View) το οποίο περιλαμβάνει μοντέλα για τη δημιουργία των οντοτήτων της εφαρμογής, views όπου εκεί διαμορφώνεται η λειτουργικότητα της εφαρμογής και templates που αφορούν οτιδήποτε σχετικά με την εμφάνιση στην ιστοσελίδα. Μέσω των μοντέλων admin γίνεται η διαχείριση της εφαρμογής. Λόγω της αντικειμενοστρεφούς συμπεριφοράς του αλληλεπιδρά με ευκολία με βάσεις δεδομένων και τα δεδομένα που χρησιμοποιούνται στον κώδικα ανακτώνται και αποθηκεύονται με μεγάλη ταχύτητα [1] [2].

Για να είναι εφικτή η δημιουργία μιας εφαρμογής που θα βασίζεται στο Django είναι απαραίτητη προϋπόθεση να έχει εγκατασταθεί προηγουμένως κάποια έκδοση της Python.

Στη συνέχεια προκειμένου να γίνει η εγκατάσταση απαιτείται η εκτέλεση ορισμένων εντολών:

```
python3.9 -m venv myvenv
```

Εικόνα 1 Δημιουργία εικονικού περιβάλλοντος

Το εικονικό περιβάλλον (virtual environment) είναι απαραίτητο επειδή δημιουργεί ένα απομονωμένο περιβάλλον, ένα φάκελο δηλαδή που περιέχει όλα τα απαραίτητα εκτελέσιμα που χρησιμοποιούν τα πακέτα της εφαρμογής.

```
source myvenv/bin/activate
```

Εικόνα 2 Ενεργοποίηση εικονικού περιβάλλοντος

Έπειτα θα πρέπει να εγκαταστήσουμε το Django με την παρακάτω εντολή:

```
pip install -U django
```

Εικόνα 3 Εγκατάσταση Django

Για την αρχικοποίηση νέου project:

```
django-admin.py startproject myproject
```

Εικόνα 4 Δημιουργία νέου project

Η δομή του Django project θα πρέπει να είναι κάπως έτσι:

```
myproject/                  <-- higher level folder
 |-- myproject/             <-- django project folder
 |    |-- myproject/
 |    |    |-- __init__.py
 |    |    |-- settings.py
 |    |    |-- urls.py
 |    |    |-- wsgi.py
 |    +-- manage.py
 +-- venv/                  <-- virtual environment folder
```

Εικόνα 5 Δομή Django project

Για να δημιουργηθούν οι πίνακες στη βάση:

```
python manage.py migrate
```

Εικόνα 6 Δημιουργία πινάκων στη βάση

Για να μπορούμε να έχουμε πρόσβαση στο django-admin panel αρχικά θα πρέπει:

```
python manage.py createsuperuser
```

Εικόνα 7 Δημιουργία superuser

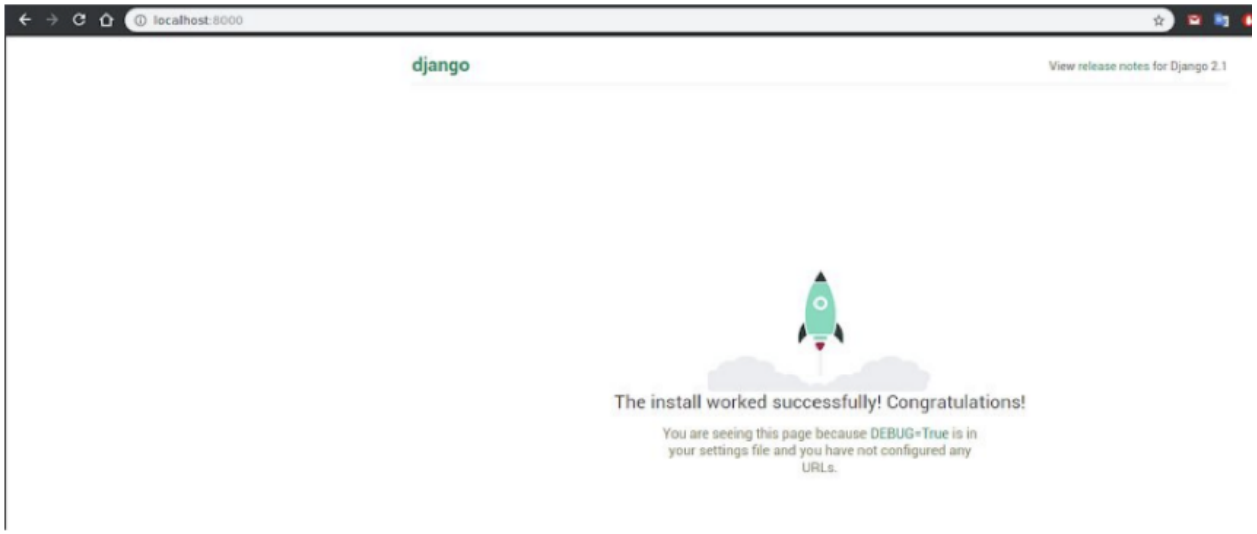
Για να τρέξουμε τοπικά το project μέσα από τον development server:

```
python myproject/manage.py runserver 0.0.0.0:8000
```

Εικόνα 8 Εκτέλεση εφαρμογής τοπικά

Τέλος για να εμφανιστεί η εφαρμογή πληκτρολογούμε στη γραμμή αναζήτησης: http://127.0.0.1:8000/

Εάν η εφαρμογή μας έχει εκτελεστεί σωστά τότε θα πρέπει να εμφανίζεται στην οθόνη μας:



Εικόνα 9 Επιτυχής εκτέλεση της εφαρμογής

3.2 HTML, CSS, JavaScript, Bootstrap

Η HTML (**H**yper**T**ext **M**arkup **L**anguage) είναι η κύρια γλώσσα για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων τα οποία αποτελούνται από *ετικέτες* (tags), περικλείονται μέσα στα σύμβολα < και > (για παράδειγμα `<html>`), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα `<h1>` και `</h1>`), με την πρώτη να ονομάζεται *ετικέτα έναρξης* και τη δεύτερη *ετικέτα λήξης* (ή σε άλλες περιπτώσεις *ετικέτα ανοίγματος* και *ετικέτα κλεισίματος* αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες που μπορεί κανείς να διαβάσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές.

Η CSS (***C**ascading **S**tyle **S**heets*) είναι μια γλώσσα που ανήκει στην κατηγορία των γλωσσών φύλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστότοπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα αφού διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript.

Η JavaScript (**JS**) είναι διερμηνευμένη γλώσσα προγραμματισμού. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του

πελάτη (client-side scripts) να μπορούν να επικοινωνούν, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξη της είναι επηρεασμένη από τη γλώσσα προγραμματισμού C. Η JavaScript αντιγράφει πολλές ονομασίες από τη Java, μία ευρέως διαδεδομένη γλώσσα προγραμματισμού , αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστραφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές (virtual machines) και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Ο κώδικας Javascript μιας σελίδας περικλείεται από τις ετικέτες της HTML <script type="text/javascript"> και </script> [1].

3.3 PyCharm IDE

Το PyCharm είναι ένα δωρεάν περιβάλλον ανάπτυξης το οποίο χρησιμοποιείται στον προγραμματισμό εφαρμογών. Δημιουργήθηκε από την εταιρία Jetbrains, αφορά τη γλώσσα προγραμματισμού Python και είναι κατάλληλο ειδικά όταν θέλουμε να χρησιμοποιήσουμε κάποιο web framework όπως το Django.

Η επιλογή του συγκεκριμένου περιβάλλοντος έγινε για καθαρά πρακτικούς λόγους αφού εκτός του ότι πρόκειται για ένα εξαιρετικά δημοφιλές περιβάλλον ανάπτυξης, προσφέρει μία πληθώρα επιλογών στον προγραμματιστή όσον αφορά βιβλιοθήκες, πακέτα, διαχείριση σφαλμάτων, ομαδοποίηση, αλληλεπίδραση με συστήματα ελέγχου ανοικτού κώδικα (π.χ git) και είναι εξαιρετικά κατανοητό στη χρήση ακόμα τόσο από αρχάριους όσο και από επαγγελματίες χρήστες.

3.4 SQLite - DB Browser for SQLite, PostgreSQL - PgAdmin

Το SQLite είναι μια σχεσιακή βάση δεδομένων συμβατή με SQL. Σε αντίθεση με άλλα συστήματα που βασίζονται σε SQL όπως MySQL και PostgreSQL, το SQLite δεν χρησιμοποιεί αρχιτεκτονική πελάτη-διακομιστή (client-server). Ολόκληρο το λογισμικό περιέχεται σε μια βιβλιοθήκη C, η οποία είναι ενσωματωμένη σε εφαρμογές.

Το SQLite επικεντρώνεται στην παροχή μιας ισχυρής βάσης δεδομένων συμβατής με SQL χωρίς εξαρτήσεις. Όπως υποδηλώνει το όνομα, είναι ένα ελαφρύ λογισμικό που μπορεί να τρέξει σχεδόν σε οτιδήποτε υποστηρίζει C και μόνιμο χώρο αποθήκευσης αρχείων. Οι δεσμεύσεις είναι διαθέσιμες για τις πιο δημοφιλείς γλώσσες προγραμματισμού υψηλού επιπέδου.

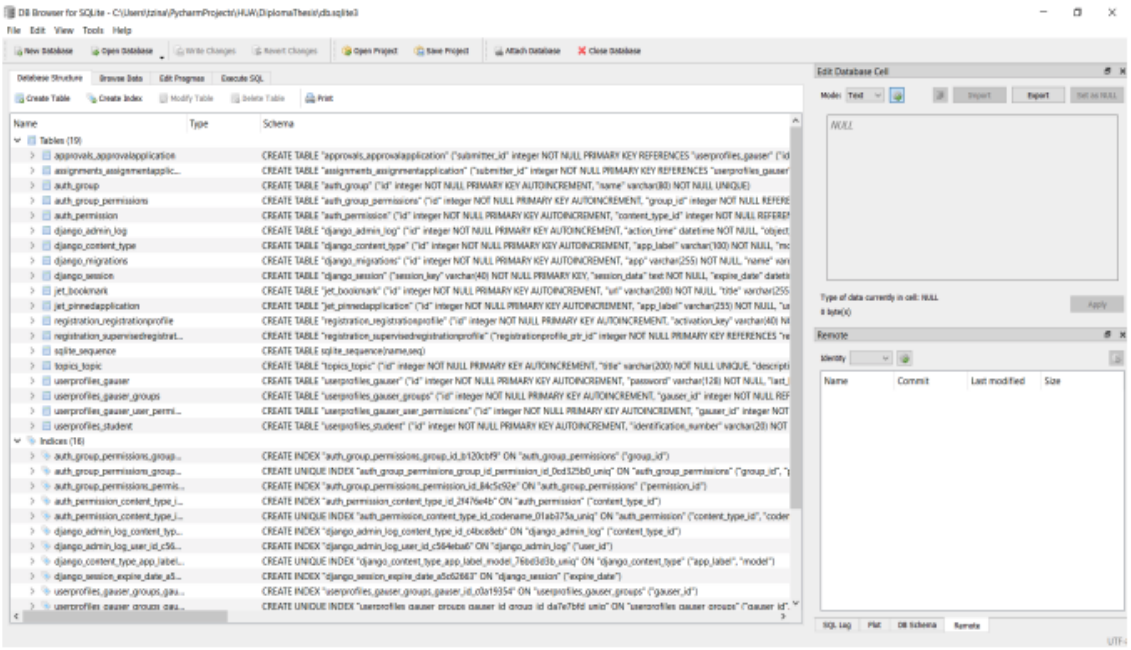
Δεδομένου ότι οι βάσεις δεδομένων SQLite είναι κανονικά αρχεία, είναι εξαιρετικά φορητές και δημιουργούνται εύκολα αντίγραφα ασφαλείας. Επίσης, η έλλειψη ενός στοιχείου διακομιστή καθιστά το SQLite πολύ πιο εύκολο να ρυθμιστεί, ακόμα κι αν δεν βρισκόμαστε σε ένα πλήρες περιβάλλον ανάπτυξης (production).

Το SQLite λειτουργεί καλύτερα όταν θέλουμε να συνδυάσουμε τις ισχυρές δυνατότητες αναζήτησης και αποθήκευσης του SQL με την ευκολία χρήσης της συμβατικής πρόσβασης στο σύστημα αρχείων. Σε αυτά τα σενάρια, προσφέρει καλύτερη απόδοση και ανθεκτικότητα σε σύγκριση με τις κανονικές λειτουργίες ανάγνωσης και εγγραφής.

Το σύστημα λειτουργεί καλά σε περιβάλλοντα όπου οι τελικοί χρήστες δεν πρέπει ποτέ να γνωρίζουν την ύπαρξη της βάσης δεδομένων και δεν απαιτεί συντήρηση ή διαχείριση, καθιστώντας το ιδανικό για εφαρμογές για κινητά και συσκευές IoT.

Συνοπτικά, η απλότητα, η φορητότητα και η αξιοπιστία τον καθιστούν το προτιμώμενο σύστημα αποθήκευσης για σύγχρονα λειτουργικά συστήματα και ενσωματωμένες πλατφόρμες. Δεν καταναλώνει σχεδόν καθόλου πόρους, είναι παραμετροποιήσιμο και εύκολο στη χρήση για προγραμματιστές λειτουργώντας ‘αόρατα’ για τους τελικούς χρήστες.

Το DB Browser for SQLite είναι μία διαδραστική, ανοικτού κώδικα τεχνολογία που χρησιμοποιείται για να σχεδιάσει, να δημιουργήσει και να επεξεργαστεί αρχεία βάσεων δεδομένων συμβατά με SQLite. Αυτή η τεχνολογία επιτρέπει στους χρήστες να αλληλεπιδρούν με δεδομένα σε ένα περιβάλλον βασισμένο στα φύλλα.



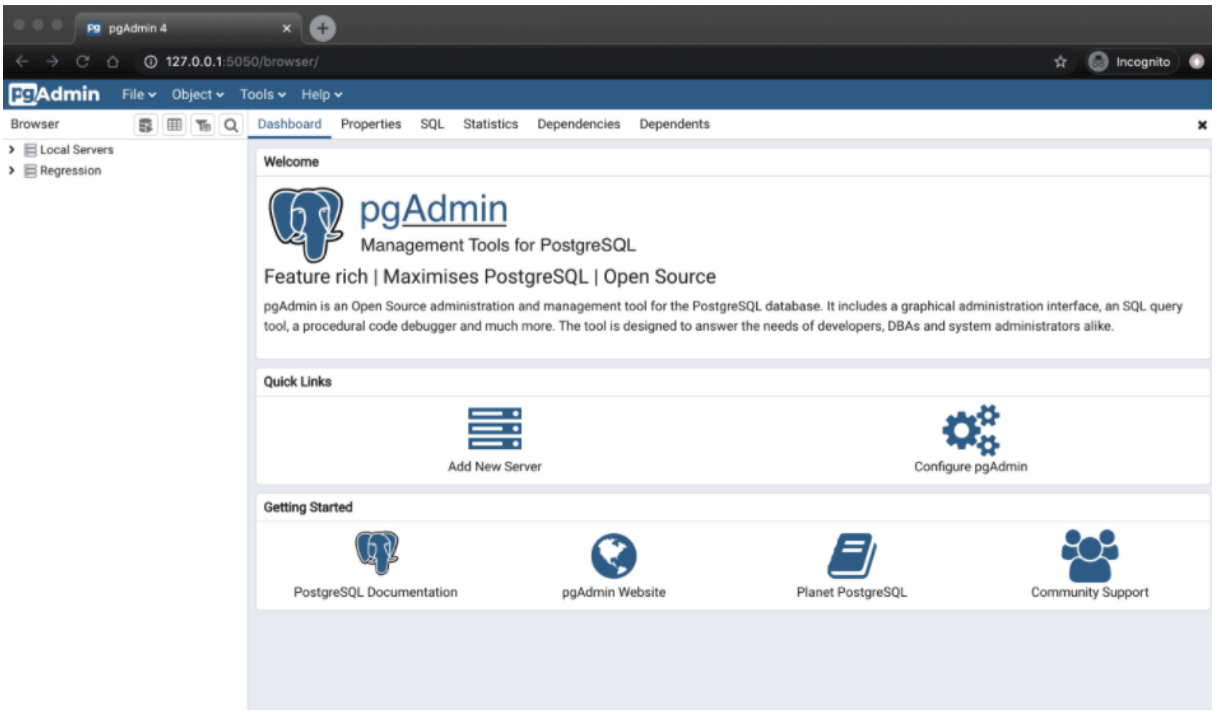
Εικόνα 10 DB Browser for SQLite

Όσον αφορά την PostgreSQL, είναι μια σχεσιακή βάση δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Η ανάπτυξη της διαρκεί ήδη πάνω από δύο δεκαετίες και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία.

Η PostgreSQL δεν ακολουθεί την αρχιτεκτονική client – server. Αντιθέτως, η βάση δεδομένων PostgreSQL είναι ενσωματωμένη στην εφαρμογή που έχει πρόσβαση στη βάση δεδομένων. Η εφαρμογή αλληλεπιδρά με τη βάση δεδομένων που διαβάζει και γράφει απευθείας από τα αρχεία βάσης δεδομένων που είναι αποθηκευμένα στο δίσκο.

Η PostGreSQL δεν έχει η ίδια κάποιο γραφικό περιβάλλον διεπαφής με τον χρήστη. Για την αλληλεπίδραση με την βάση δεδομένων χρησιμοποιείται ένα πρόγραμμα χρήστη (client) το PgAdmin που παρέχεται και εγκαθίσταται μαζί με την PostGreSQL. Το PgAdmin και έχει τα παρακάτω χαρακτηριστικά :

1) Είναι ένα ολοκληρωμένο σύστημα σχεδιασμού και διαχείρισης βάσεων δεδομένων και αποτελεί το μέσο αλληλεπίδρασης της βάσης δεδομένων με το χρήστη.
2) Είναι γραμμένο σε C++ και είναι δυνατό να χρησιμοποιηθεί σε περιβάλλοντα Linux, FreeBSD, Solaris, Mac OS X και Windows για τη διαχείριση της PostgreSQL.
3) Προσφέρει ένα απλό γραφικό περιβάλλον για την ανάπτυξη πολύπλοκων βάσεων δεδομένων μέσω της διατύπωσης ερωτημάτων σε SQL, με στόχο την απλούστευση των διαδικασιών για το χρήστη
4) Είναι ελεύθερο λογισμικό και δεν απαιτεί επιπλέον προγράμματα για την επικοινωνία με τον διακομιστή της βάσης δεδομένων.



Εικόνα 11 PgAdmin

3.5 Gunicorn, Nginx

Όταν μία Python web εφαρμογή τρέχει στην παραγωγή (production) συνήθως υπάρχουν τρία βασικά σημεία τα οποία χρειάζονται για να εμφανιστεί: ένας web server (π.χ nginx), ένας wsgi application application server (π.χ gunicorn) και βασικά η εφαρμογή μας.

Ο web server αποδέχεται αιτήματα, φροντίζει για μία γενική λογική στο domain και χειρίζεται συνδέσεις βασισμένες στα http/https.

Ο Gunicorn έχει φτιαχτεί έτσι ώστε πολλοί διαφορετικοί web servers να μπορούν να αλληλεπιδράσουν. Δεν ‘νοιάζεται΄ για το τι χρησιμοποιούμε για να χτίσουμε την εφαρμογή αφού μπορεί να αλληλεπιδράσει με αυτή χρησιμοποιώντας τη διεπαφή wsgi. Η διεπαφή wsgi (web server gateway interface) είναι ένας τρόπος να σιγουρευτούμε ότι οι web server και οι web εφαρμογές μας επικοινωνούν μεταξύ τους. Έτσι κάπου στην εφαρμογή

(συνήθως στο αρχείο wsgi.py) ορίζεται ένα αντικείμενο το οποίο καλείται από τον Gunicorn. Το αντικείμενο αυτό χρησιμοποιείται για να περάσει τα αιτήματα δεδομένων στην εφαρμογή και για να λάβει απάντηση.

Ο Gunicorn αναλαμβάνει ό,τι συμβαίνει μεταξυ του web server και της εφαρμογής και τρέχει πολλα στιγμιότυπα της εφαρμογής καθιστώντας σίγουρο ότι συμπεριφέρονται σωστά, διανέμοντας εισερχόμενα αιτήματα μεταξύ αυτών των στιγμιοτύπων και επικοινωνώντας με τον web server.

Όσον αφορά τους web servers, συναντάμε συχνά τον όρο web hosting (ή διακομιστή/web server). Σε αυτήν την εποχή της πληροφορίας βρίσκουμε πολλές παραλλαγές υπηρεσιών web hosting ειδικά από τις Ηνωμένες Πολιτείες (ΗΠΑ), όπου η τεχνολογία του Διαδικτύου συνεχίζει να αναπτύσσεται.

Για να εκτελέσουμε ένα σύστημα web hosting, χρειαζόμαστε μια συσκευή web server. Μεταξύ των πολλών λογισμικών που χρησιμοποιούν οι προγραμματιστές, υπάρχει και το διάσημο Nginx. Είναι λογισμικό web server που κυκλοφόρησε ως ανοιχτή πηγή. Εκτός από γνωστό ως web server, το Nginx είναι επίσης γνωστό ως αντίστροφος διακομιστής μεσολάβησης, HTTP cache, και εξισορρόπησης φορτίου. Πολλές εταιρείες βασισμένες σε τεχνολογία υπολογιστών μεγάλης κλίμακας σε όλο τον κόσμο επιλέγουν να χρησιμοποιήσουν web server. Μεταξύ των ονομάτων αυτών είναι οι Google, Twitter, Facebook.

Το Nginx λειτουργεί ως web server, δηλαδή προσομοιώνει μια συσκευή υπολογιστή ως μηχανή παροχής υπηρεσιών σελίδας στο διαδίκτυο.

Τα χαρακτηριστικά του Nginx τα οποία μπορούν να χρησιμοποιηθούν για την υποστήριξη της απόδοσης του διαδικτύου είναι: [1]

1. Υποστήριξη IPv6 (δικτυακό πρότυπο)
   Το IPv6 έχει διαφορετική δομή διευθύνσεων από το IPv4, το οποίο έχει μήκος 128-bit και είναι γραμμένο με 8 δεκαεξαδικων ομάδες.
2. Εξισορρόπηση φορτίου
   Η εξισορρόπηση φορτίου είναι μια τεχνική για την κατανομή φορτίων κίνησης σε δύο ή περισσότερες γραμμές σύνδεσης με ισορροπημένο τρόπο.

3. Υποστήριξη FastCGI με διαδικασίες cache

   Το FastCGI (Fast Common Gateway Interface) είναι ένα πρωτόκολλο δυαδικό για σύνδεση προγραμμάτων (για παράδειγμα web browser) με web server διαδραστικά.

4. Websockets

   Το WebSocket είναι ένα πρωτόκολλο επικοινωνίας υπολογιστή, η λειτουργία του είναι να παρέχει σύνδεση μέσων επικοινωνίας πλήρως αμφίδρομη μέσω μιας σύνδεσης TCP (Transmission Control Protocol).

5. Διαχειρίζεται στατικά αρχεία, αρχεία ευρετηρίου και αυτόματη ευρετηρίαση

3.6 Ansible, Jenkins, Docker, Kubernetes

Η Ansible είναι ένα απλό λογισμικό αυτοματοποίησης πληροφορικής που αυτοματοποιεί την παροχή cloud (cloud provisioning) , τη διαχείριση παραμετροποιήσεων (configuration management), την ανάπτυξη εφαρμογών, και την ενορχήστρωση (orchestration) εντός των υπηρεσιών. Σχεδιασμένη για πολυεπίπεδες αναπτύξεις, η Ansible μοντελοποιεί την υποδομή πληροφορικής (infrastructure, περισσότερα εδώ [IaS]) περιγράφοντας πώς όλα τα συστήματά σχετίζονται μεταξύ τους, αντί να γίνεται διαχείριση ενός συστήματος κάθε φορά.

Η ansible τρέχει σε πολλά unix-based συστήματα (βλ. Linux, Mac OS X) ωστόσο μπορεί να παραμετροποιήσει και συστήματα Windows. Περιλαμβάνει το δικό της περιγραφικό λεξιλόγιο ώστε να περιγράψει την παραμετροποίηση ενός συστήματος. Γράφτηκε από τον Michael DeHaan και αποκτήθηκε από την εταιρεία Red Hat το 2015.

Δεν χρησιμοποιεί πράκτορες (agents, επιπλέον λογισμικά δήλαδή) και καμία πρόσθετη προσαρμοσμένη υποδομή ασφαλείας, οπότε είναι εύκολο στην ανάπτυξη - και το πιο σημαντικό, χρησιμοποιεί μια πολύ απλή γλώσσα (YAML, με τη μορφή των Ansible Playbooks) που επιτρέπει να περιγράφουμε τις εργασίες αυτοματοποίησης. Η Ansible λειτουργεί με τη σύνδεση στους κόμβους και την προώθηση μικρών προγραμμάτων, που ονομάζονται "Ansible modules", σε αυτούς. Αυτά τα προγράμματα γράφονται έτσι ώστε να είναι μοντέλα πόρων της επιθυμητής κατάστασης του συστήματος. Στη συνέχεια, η Ansible

εκτελεί αυτά τα modules (μέσω SSH πρωτοκόλλου σύνδεσης, περισσότερα εδώ [SSH]) και τα αφαιρεί όταν τελειώσουν.

Το module "authorized_key" του Ansible είναι ένας πολύ καλός τρόπος για να χρησιμοποιήσουμε την ansible για να ελέγξουμε ποιες μηχανές μπορούν να έχουν πρόσβαση σε ποιους hosts. Μπορούν επίσης να χρησιμοποιηθούν και άλλες επιλογές, όπως το kerberos ή τα συστήματα διαχείρισης ταυτότητας.

```
ssh-agent bash
ssh-add ~/.ssh/id_rsa
```

Εικόνα 12 Σύνδεση ssh [Πηγή: https://www.ansible.com/overview/how-ansible-works]

Επιπλέον, η Ansible αναπαριστά τα μηχανήματα που διαχειρίζεται χρησιμοποιώντας ένα πολύ απλό αρχείο INI που τοποθετεί όλα τα διαχειριζόμενα μηχανήματα σε ομάδες της επιλογής μας.

Για να προσθέσουμε νέα μηχανήματα, δεν εμπλέκεται κανένας πρόσθετος διακομιστής SSL, οπότε δεν υπάρχει πρόβλημα να αποφασίσουμε γιατί ένα συγκεκριμένο μηχάνημα δεν συνδέθηκε λόγω προβλημάτων NTP ή DNS.

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

Εικόνα 13 Δομή του inventory file [Πηγή: https://www.ansible.com/overview/how-ansible-works]

Τα playbooks μπορούν να ενορχηστρώσουν με λεπτομέρεια πολλαπλά σημεία της τοπολογίας της υποδομής, με πολύ λεπτομερή έλεγχο σχετικά με το πόσες μηχανές θα υπάρχουν κάθε φορά. Αυτό είναι το σημείο όπου η Ansible αρχίζει να γίνεται πιο ενδιαφέρουσα.

Η προσέγγιση της Ansible για την ενορχήστρωση (orchestration) είναι μια προσέγγιση απλότητας, καθώς ο κώδικας αυτοματοποίησης που χρησιμοποιούμε έχει νόημα ακόμα και μετά από καιρό αφού δεν πρέπει να θυμόμαστε πράγματα σχετικά με την ειδική σύνταξη ή τα χαρακτηριστικά [3] [10].

```
---
- hosts: webservers
serial: 5 # update 5 machines at a time
roles:
- common
- webapp
```

Εικόνα 14 Δομή playbook [Πηγή: https://www.ansible.com/overview/how-ansible-works]

Ο Jenkins είναι δρομολογητής αυτοματοποίησης ανοιχτού κώδικα. Βοηθάει στην αυτοματοποίηση κομματιών της ανάπτυξης λογισμικού σχετικά με το χτίσιμο, τη δοκιμή, και την ανάπτυξη (build-test-deploy).

Ο Jenkins διαχειρίζεται και ελέγχει τις διαδικασίες παράδοσης λογισμικού σε ολόκληρο τον κύκλο ζωής, συμπεριλαμβανομένης της κατασκευής, της τεκμηρίωσης, της δοκιμής, του σταδίου, της ανάπτυξης, της στατικής ανάλυσης κώδικα. Μπορούμε να ρυθμίσουμε το Jenkins ώστε να παρακολουθεί τυχόν αλλαγές κώδικα σε μέρη όπως το GitHub, το Bitbucket ή το GitLab και να κάνει αυτόματα μια κατασκευή (build) με εργαλεία όπως το Maven και το Gradle.

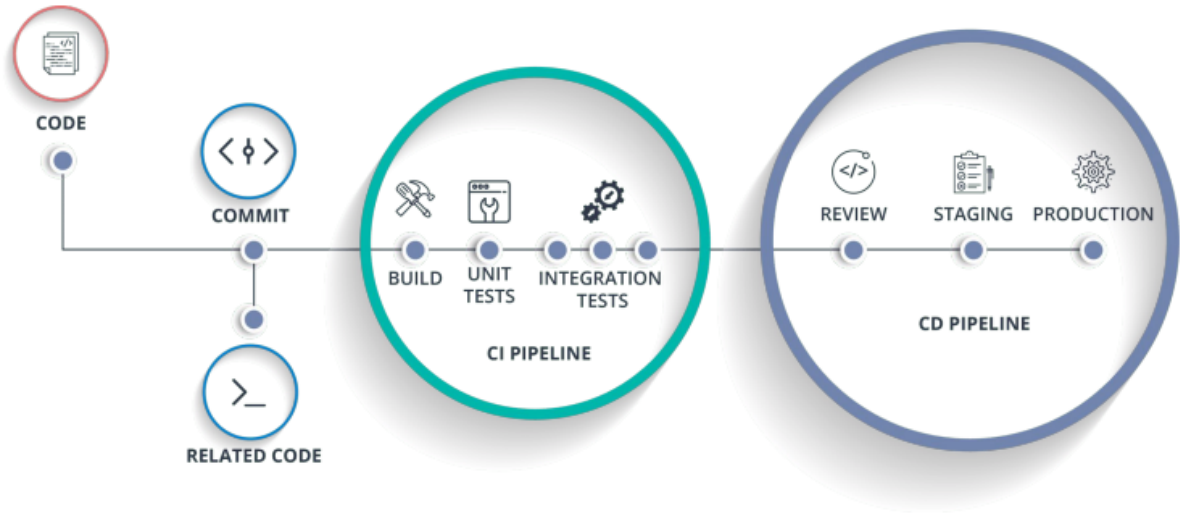
Η συνεχής ενσωμάτωση (Continuous integration/CI) είναι μια φιλοσοφία κωδικοποίησης και ένα σύνολο πρακτικών που οδηγούν τις ομάδες ανάπτυξης να εφαρμόζουν μικρές αλλαγές και να ελέγχουν συχνά τα αποθετήρια ελέγχου μετατροπής κώδικα. Επειδή οι περισσότερες σύγχρονες εφαρμογές απαιτούν την ανάπτυξη κώδικα σε διαφορετικές πλατφόρμες και εργαλεία, χρειάζεται ένας μηχανισμός για την ενσωμάτωση και την επικύρωση των αλλαγών της.

Η συνεχής παράδοση (Continuous Delivery/CD) συνεχίζει από εκεί που τελειώνει η συνεχής ενσωμάτωση. Το CD αυτοματοποιεί την παράδοση των εφαρμογών σε επιλεγμένα περιβάλλοντα υποδομής. Οι περισσότερες ομάδες εργάζονται σε πολλαπλά περιβάλλοντα, όπως περιβάλλοντα ανάπτυξης και δοκιμών, και το CD εξασφαλίζει ότι

υπάρχει ένας αυτοματοποιημένος τρόπος για την προώθηση των αλλαγών κώδικα σε αυτά.
Η συνεχής ενσωμάτωση και η συνεχής παράδοση απαιτούν διαρκείς δοκιμές, επειδή ο στόχος είναι να παραδίδονται ποιοτικές εφαρμογές και κώδικας στους χρήστες. Οι συνεχείς δοκιμές συχνά υλοποιούνται ως ένα σύνολο αυτοματοποιημένων δοκιμών παλινδρόμησης, επιδόσεων και άλλων δοκιμών που εκτελούνται στο ευρέως γνωστό CI/CD pipeline (αγωγός).

Το pipeline είναι ένα σύνολο αυτοματοποιημένων διαδικασιών που επιτρέπουν στους προγραμματιστές να μεταγλωττίζουν, να χτίζουν και να αναπτύσσουν αξιόπιστα και αποτελεσματικά τον κώδικά τους σε πλατφόρμες παραγωγής. Δεν υπάρχει κανένας αυστηρός κανόνας που να ορίζει πώς πρέπει να μοιάζει ένα pipeline και τα εργαλεία που πρέπει να χρησιμοποιεί, ωστόσο τα πιο συνηθισμένα στοιχεία ενός pipeline είναι τα εξής: χτίσιμο αυτοματοποίησης/συνεχούς ενσωμάτωσης, αυτοματοποίηση δοκιμών και αυτοματοποίηση ανάπτυξης.

Ένα pipeline αποτελείται γενικά από ένα σύνολο εργαλείων που συνήθως αναλύονται στις εξής κατηγορίες:
- Έλεγχος πηγής (Source Control)
- Εργαλεία κατασκευής (Build tools)
- Containerisation (πληροφορίες παρακάτω)
- Διαχείριση παραμετροποίησης (Configuration Management)
- Παρακολούθηση (Monitoring)



Εικόνα 15 CI/CD pipeline [Πηγή: https://medium.com/@singlanitish29/automated-ci-cd-pipeline-8124eb8dc136]

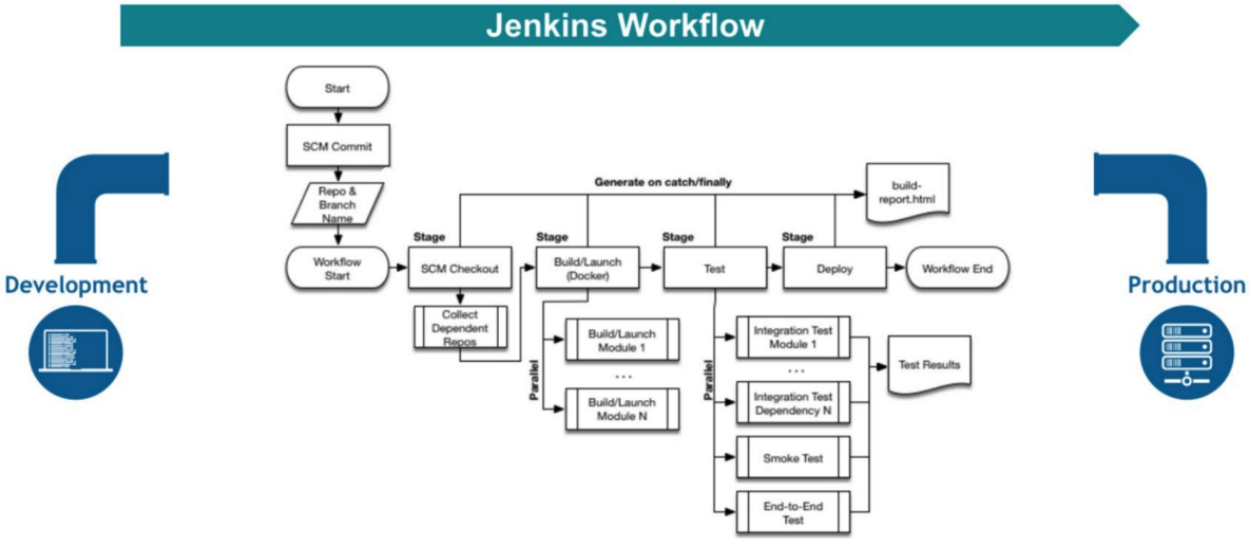
Τα συστατικά (components) ενός CI/CD pipeline είναι:

- Αποθετήριο κώδικα (π.χ. github)
- Αποθετήριο εικόνων (π.χ. Docker Hub)
- Περιβάλλον CI/CD (π.χ. Jenkins)
- Περιβάλλον(τα) ανάπτυξης (π.χ. VMs, Kubernetes Cluster)

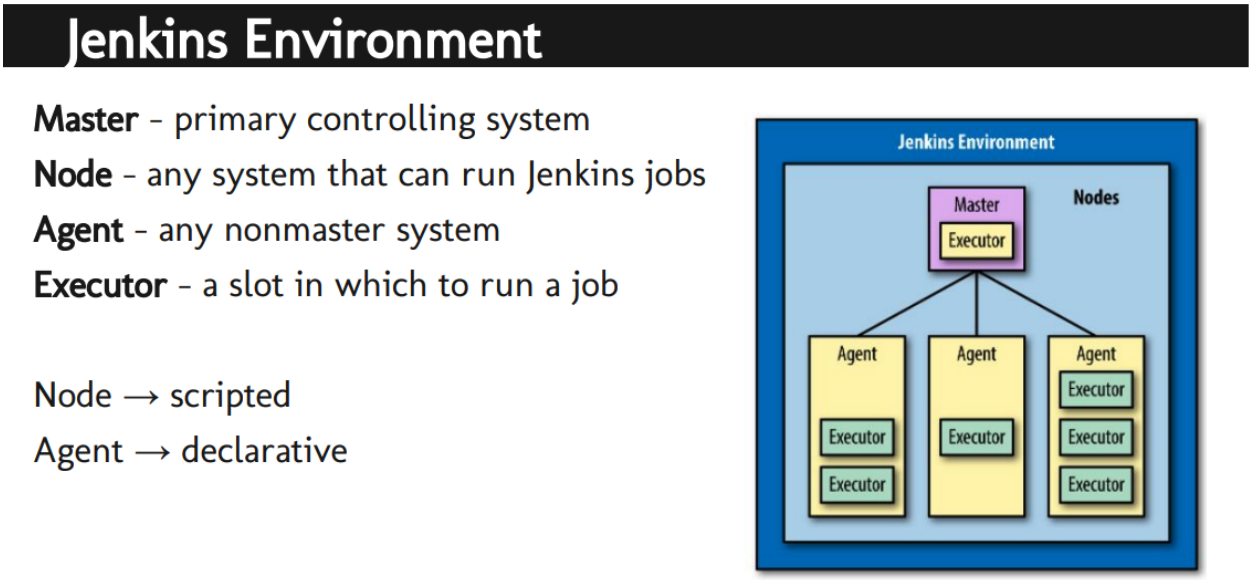
Παρακάτω απεικονίζεται η σύνταξη και η δομή ενός pipeline. Ένα stage στον Jenkins περιγράφει τη δουλειά που γίνεται κάθε φορά σε ένα συγκεκριμένο κομμάτι του κώδικα. Τα steps δείχνουν κάθε βήμα που ακολουθείται μέσα σε ένα stage [4].

```
pipeline {
    agent any
    stages {
        stage('Example Build') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Example Deploy') {
            when {
                branch 'production'
                environment name: 'DEPLOY_TO', value: 'production'
            }
            steps {
                echo 'Deploying'
            }
        }
    }
}
```

Εικόνα 16 Σύνταξη pipeline



Εικόνα 17 Ροή Jenkins

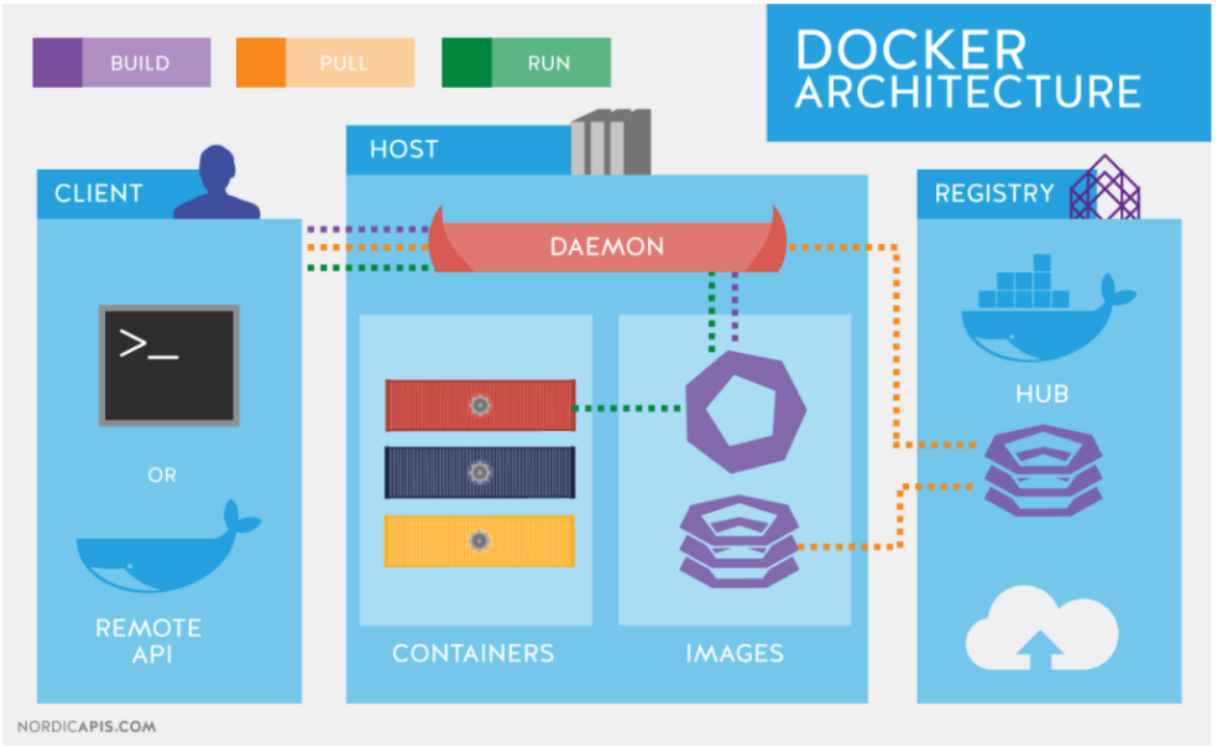


Εικόνα 18 Περιβάλλον Jenkins

Η ανάπτυξη εφαρμογών, σήμερα, απαιτεί πολλά περισσότερα από κώδικα. Οι πολλαπλές γλώσσες, τα πλαίσια, οι αρχιτεκτονικές και οι διεπαφές μεταξύ εργαλείων για κάθε στάδιο του κύκλου ζωής δημιουργούν τεράστια πολυπλοκότητα. Το Docker απλοποιεί και επιταχύνει τη ροή αυτή, ενώ δίνει στους προγραμματιστές την ελευθερία να καινοτομούν με την επιλογή εργαλείων, στοίβας εφαρμογών και περιβαλλόντων ανάπτυξης για κάθε έργο.

Το Docker, λοιπόν, είναι ένα λογισμικό ανοικτού κώδικα που αυτοματοποιεί την ανάπτυξη εφαρμογών λογισμικού μέσα σε containers (βλ. παρακάτω) παρέχοντας ένα πρόσθετο επίπεδο αφαίρεσης και αυτοματοποίησης σε επίπεδο λειτουργικού συστήματος στο Linux.

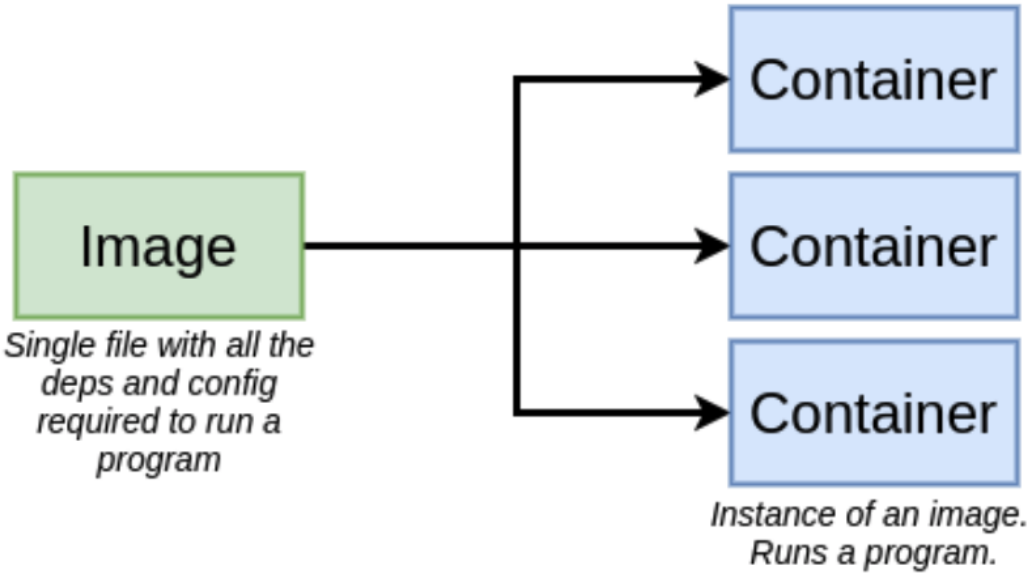
Για εκατομμύρια προγραμματιστές σήμερα, το Docker είναι το βασικό πρότυπο για τη δημιουργία και την κοινή χρήση εφαρμογών με containers - από την επιφάνεια εργασίας, μέχρι το cloud.



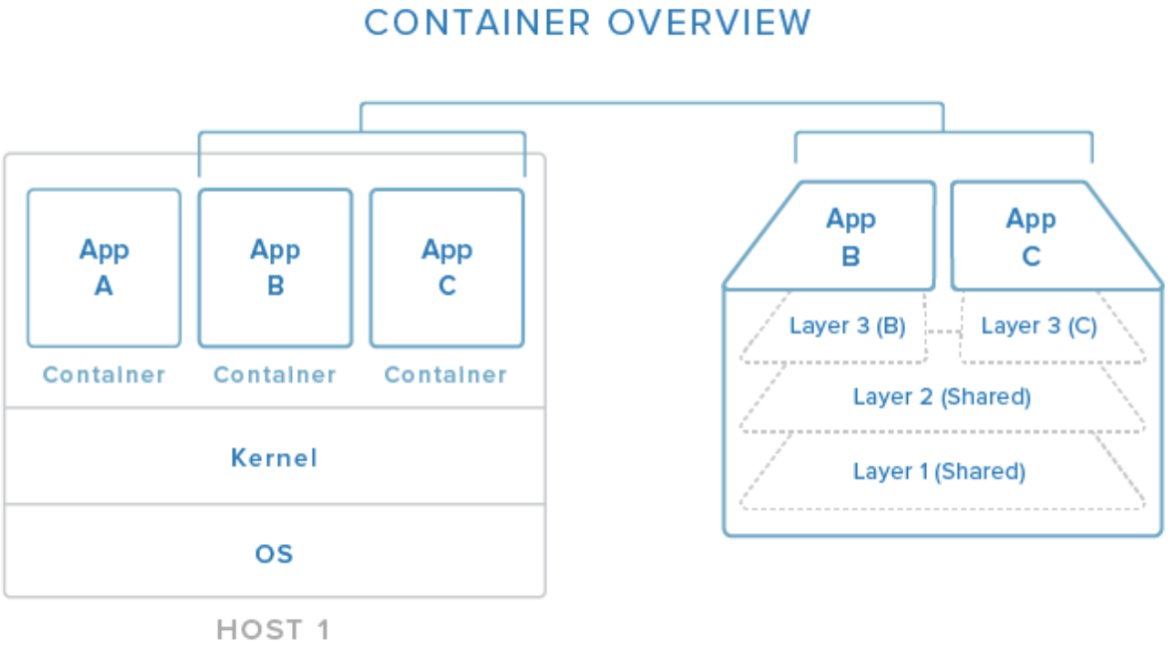
Εικόνα 19 Αρχιτεκτονική Docker [Πηγή: https://nordicapis.com/api-driven-devops-spotlight-on-docker/]

Ένα container είναι μια μονάδα λογισμικού που πακετάρει τον κώδικα και όλα τα αρχεία του, ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα από το ένα υπολογιστικό περιβάλλον στο άλλο. Ένα container image είναι ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο λογισμικού που περιλαμβάνει όλα όσα απαιτούνται για την εκτέλεση μιας εφαρμογής: κώδικα, χρόνο εκτέλεσης, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις.

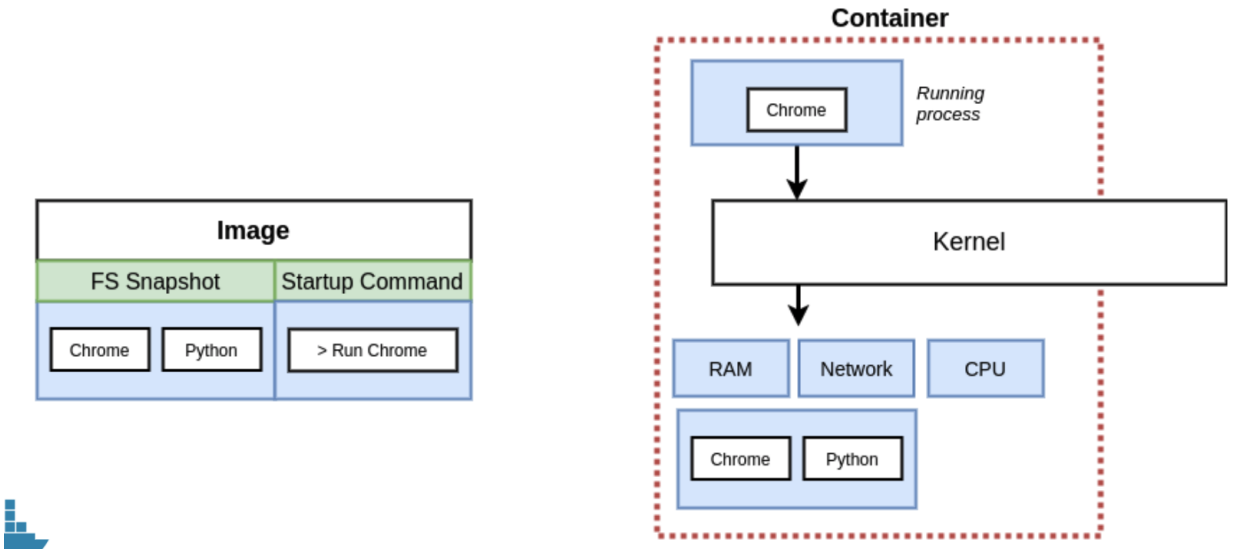
Τα images γίνονται containers κατά την εκτέλεση και στην περίπτωση των containers - τα images γίνονται containers όταν εκτελούνται στην Docker Engine. Είναι διαθέσιμο τόσο για εφαρμογές που βασίζονται σε Linux όσο και για εφαρμογές που βασίζονται σε Windows, το λογισμικό που περιέχει containers θα εκτελείται πάντα το ίδιο, ανεξάρτητα από την υποδομή. Τα containers απομονώνουν το λογισμικό από το περιβάλλον του και διασφαλίζουν ότι λειτουργεί ομοιόμορφα παρά τις διαφορές, για παράδειγμα, μεταξύ ανάπτυξης και παραγωγής.



Εικόνα 20 Images-Containers



Εικόνα 21 Επίπεδο των containers



Εικόνα 22 Από image σε container

```
$ docker run hello-world

Hello from Docker.
This message shows that your installation appears to be working correctly.
...
```

Εικόνα 23 Εκτέλεση container με βάση ένα image

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

Εικόνα 24 Εμφάνιση των containers

Ένα Dockerfile είναι ένα απλό αρχείο κειμένου που περιέχει μια λίστα εντολών που καλείται κατά τη δημιουργία μιας εικόνας. Είναι δηλαδή ένας απλός τρόπος αυτοματοποίησης της διαδικασίας δημιουργίας εικόνας. Οι εντολές που γράφουμε σε ένα Dockerfile είναι παρόμοιες με εντολές που χρησιμοποιούμε στο Linux. Αυτό σημαίνει ότι

δεν χρειάζεται να γνωρίζει κάποιος ειδική σύνταξη προκειμένου να δημιουργήσει ένα Dockerfile.

```
FROM python:3

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .

# install dependencies
RUN pip install --no-cache-dir -r requirements.txt
```

Εικόνα 25 Σύνταξη Dockerfile

Η εντολή docker build αναλαμβάνει τη δημιουργία ενός Docker image από ένα Dockerfile.

Παρακάτω μπορούμε να δούμε τη δημιουργία ενός image. Πριν εκτελέσουμε την εντολή θα πρέπει να αντικαταστήσουμε το όνομα χρήστη με το δικό μας, το οποίο θα πρέπει να είναι το ίδιο με αυτό που φτιάχνουμε κατά την εγγραφή μας στο Docker hub. Το Docker hub είναι ένα registry στο οποίο υπάρχουν images. Η εντολή docker build είναι αρκετά απλή καθώς παίρνει ένα προαιρετικό όνομα ετικέτας με -t και μια τοποθεσία του καταλόγου που περιέχει το αρχείο Docker.

```
$ docker build -t yourusername/catnip .
Sending build context to Docker daemon 8.704 kB
Step 1 : FROM python:3
# Executing 3 build triggers...
Step 1 : COPY requirements.txt /usr/src/app/
 ---> Using cache
Step 1 : RUN pip install --no-cache-dir -r requirements.txt
```

Εικόνα 26 Δημιουργία image

Αφού δημιουργηθεί το image αξιοποιούμε το docker compose, ένα εργαλείο το οποίο εγκαθίσταται μαζί με το docker και μας βοηθά ώστε να εκτελέσουμε πολλαπλά containers την ίδια στιγμή, αυτοματοποιώντας τη διαδικασία. Γι' αυτό φτιάχνουμε ένα αρχείο docker-compose.yml. Αυτό περιέχει μία λίστα από containers, αντιστοιχίσεις πορτών (port mapping) και το πώς θα γίνει το χτίσιμο.
Προκειμένου να γίνει εκτέλεση γράφουμε την εντολή docker-compose up.

Αναλυτικότερες λεπτομέρειες σχετικά με το docker, το docker compose και τα γνωρίσματα τους, μπορούμε να βρουμε στις σελίδες [5] και [6]

Στο επόμενο στάδιο έχουμε το Kubernetes. Το Kubernetes, γνωστό και ως K8s, είναι ένα σύστημα ανοικτού κώδικα για την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης και της διαχείρισης εφαρμογών με container.
Ομαδοποιεί τα containers που αποτελούν μια εφαρμογή σε λογικές μονάδες για εύκολη διαχείριση.
Το Kubernetes επιτρέπει να ενορχηστρώσουμε ένα σύνολο/συστάδα/cluster εικονικών μηχανών και να προγραμματίσουμε την εκτέλεση containers σε αυτές τις εικονικές μηχανές με βάση τους διαθέσιμους υπολογιστικούς πόρους και τις απαιτήσεις πόρων κάθε container. Τα containers ομαδοποιούνται σε pods, τη βασική λειτουργική μονάδα του Kubernetes. Τα containers και τα pods μπορούν να κλιμακωθούν στην επιθυμητή κατάσταση και είμαστε σε θέση να διαχειριστούμε τον κύκλο ζωής τους για να διατηρούμε τις εφαρμογές σας σε λειτουργία.
Το Kubernetes προσφέρει ορισμένα πολύ σημαντικά πλεονεκτήματα τα οποία αφορούν την αρχιτεκτονική του. Υποστηρίζει την επεκτασιμότητα δηλαδή μπορεί να κλιμακώσει τη δημιουργία pods ανάλογα με τη χρήση της CPU, την υψηλή διαθεσιμότητα και σε επίπεδο εφαρμογής αλλά και σε επίπεδο υποδομής, την ασφάλεια πολλαπλών επιπέδων (εφαρμογής, δικτύου κτλ) και τέλος τη φορητότητα.

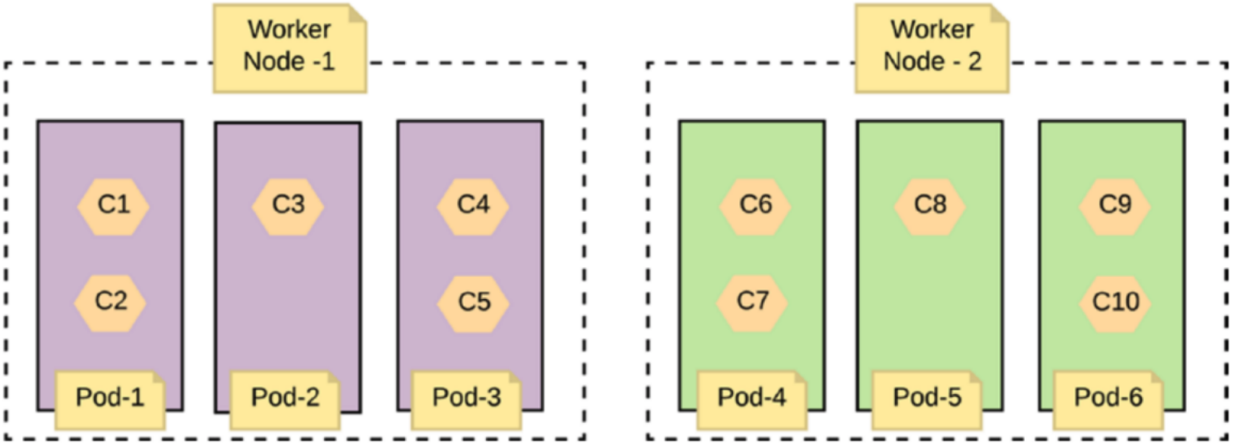
Κάποιος, ωστόσο μπορεί να αναρωτηθεί πώς διαφοροποιείται σε σχέση με το Docker. Εδώ αξίζει να αναφέρουμε ότι μια θεμελιώδης διαφορά μεταξύ του Kubernetes και του Docker είναι ότι το Kubernetes προορίζεται να εκτελείται σε ένα cluster, ενώ το Docker

εκτελείται σε έναν μόνο κόμβο. Το Kubernetes είναι πιο εκτεταμένο από το Docker και προορίζεται να συντονίζει συστάδες κόμβων σε κλίμακα παραγωγής με αποτελεσματικό τρόπο. Τα pods του Kubernetes (μονάδες προγραμματισμού που μπορούν να περιέχουν ένα ή περισσότερα containers στο οικοσύστημα Kubernetes) κατανέμονται μεταξύ των κόμβων για να παρέχουν υψηλή διαθεσιμότητα.

Μπορούμε επομένως να αντιληφθούμε τις τεράστιες δυνατότητες που παρουσιάζονται από τον συνδυασμό του Docker και του Kubernetes σε μια εφαρμογή.

Η από κοινού χρήση τους μας βοηθά:

1) Να κάνουμε την υποδομή πιο ισχυρή και την εφαρμογή πιο υψηλής διαθεσιμότητας. Η εφαρμογή θα παραμείνει σε λειτουργία, ακόμη και αν κάποιοι από τους κόμβους τεθούν εκτός λειτουργίας.
2) Να κάνουμε την εφαρμογή πιο επεκτάσιμη. Αν η εφαρμογή αρχίσει να δέχεται πολύ μεγαλύτερο φορτίο και πρέπει να επεκταθεί για να μπορέσουμε να παρέχουμε καλύτερη εμπειρία στους χρήστες, δημιουργούμε περισσότερα containers ή προσθέτουμε περισσότερους κόμβους στο cluster.



Εικόνα 27 Pods και Containers

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: croc-hunter
  labels:
    app: croc-hunter
spec:
  containers:
    - name: croc-hunter
      image: quay.io/lachie83/croc-hunter
      ports:
        - containerPort: 8080
```

Εικόνα 28 Σύνταξη Pod

Εκτέλεση/deployment ενός pod:

```
kubectl apply -f <pod-file>.yaml
```

Εικόνα 29 Δημιουργία deployment

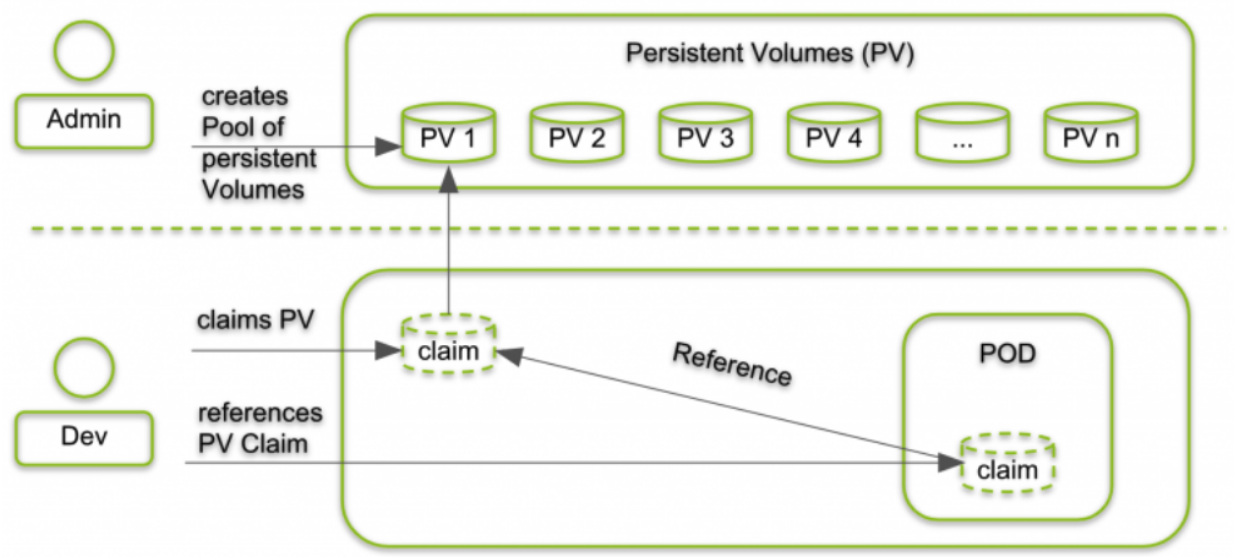
Αφού είδαμε τα pods, αξίζει να σημειώσουμε το τρόπο που γίνεται η προσπέλαση τους. Αυτό γίνεται με ένα Service, μια αφαιρετικότητα δηλαδή που ορίζει ένα σύνολο από pods και κανόνες με τους οποίους μπορούμε να τα προσπελάσουμε. Το σύνολο των pods που ‘στοχεύει’ ένα service καθορίζεται από ένα selector.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: db-service
  kind: ClusterIP
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

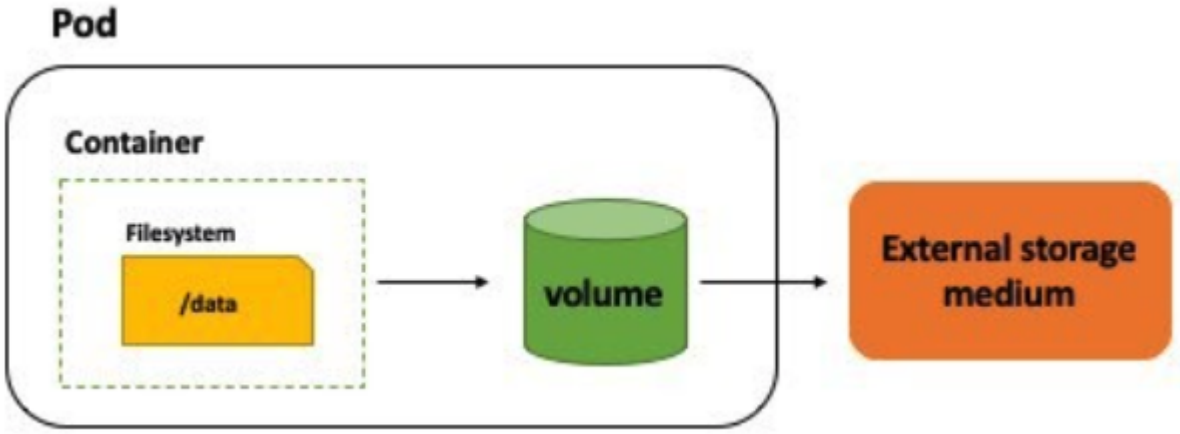
Εικόνα 30 Σύνταξη Service

Περισσότερες πληροφορίες σχετικά με τα services και για τους τύπους τους, βρίσκονται εδώ: [SVC]

Στο περιβάλλον του kubernetes θα συναντήσουμε αρκετές ακόμα ορολογίες που αφορούν την εκτέλεση των εφαρμογών.
Στα πλαίσια της παρούσας πτυχιακής εργασίας, μερικές από τις ορολογίες που θα ήταν χρήσιμο να αναφερθούν είναι ο Ingress controller, ένα pod που ουσιαστικά επιτρέπει την επικοινωνία της εφαρμογής με τον 'έξω κόσμο' λειτουργώντας δηλαδή σαν proxy server για να δούμε την εφαρμογή, τα volumes τα οποία είναι χώροι όπου αποθηκεύονται δεδομένα σχετικά με τα deployments, τα secrets στα οποία αποθηκεύονται key-value ζεύγη π.χ για τη σύνδεση με τη βάση μας, και τα configmaps τα οποία περιέχουν key-value ζεύγη όσον αφορά την παραμετροποίηση της εφαρμογής μας (π.χ .env αρχεία). [7]



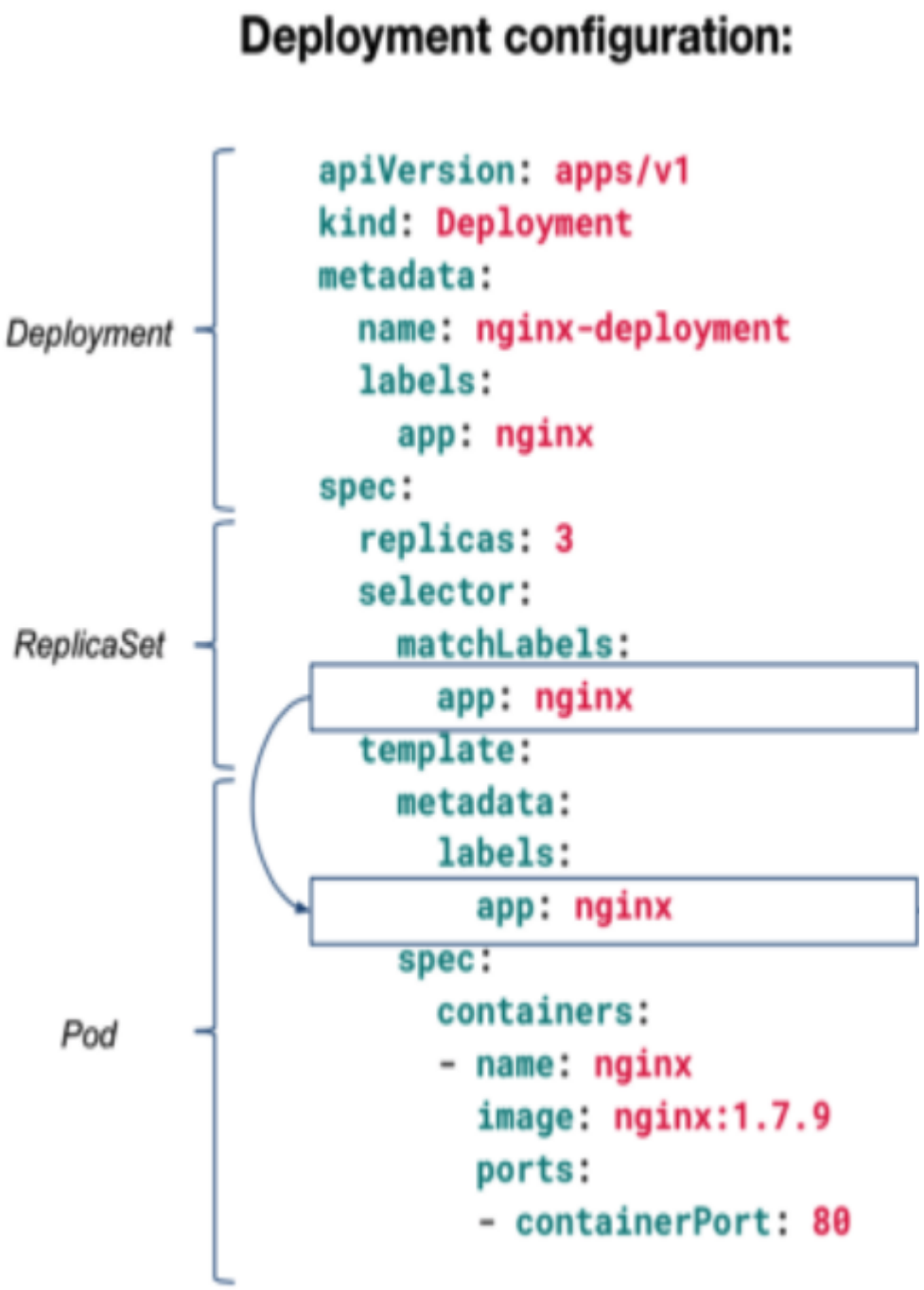
Εικόνα 31 Volumes 1



Εικόνα 32 Volumes 2

```
kubectl   get        po
          describe   deployment
          delete     svc          -n NAMESPACE   -o wide
                     cm                          -o yaml
                     secrets
                     nodes
```

Εικόνα 33 Βασικές εντολές kubernetes



Εικόνα 34 Σύνταξη ενός Deployment

Κεφάλαιο 4: Υλοποίηση και Παραμετροποίηση του Βασικού Συστήματος

4.1 Βασική Δομή Project - Παραδοχές

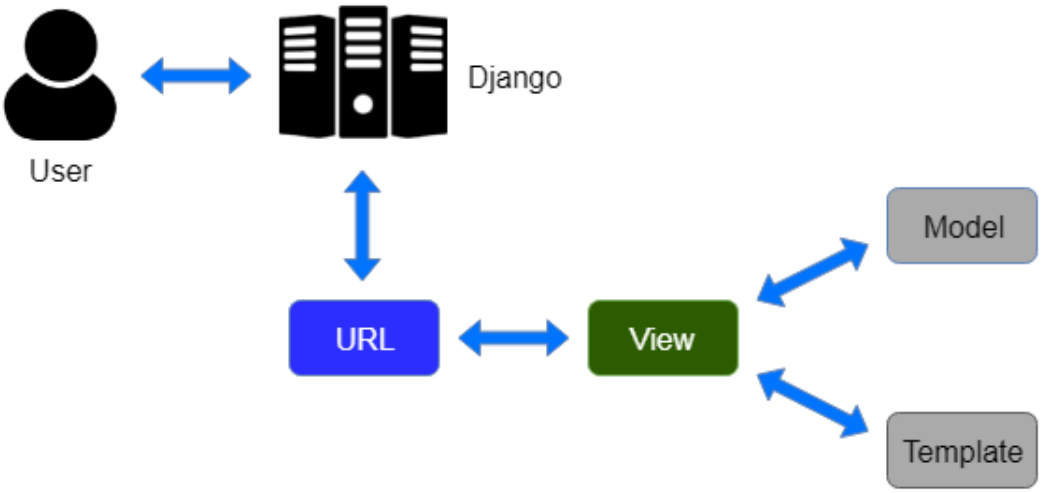
Το Django project βασίζεται στο μοτίβο MVT (Model-View-Template). Το MVT είναι ένα σχεδιαστικό πρότυπο λογισμικού, μία συλλογή που αποτελείται από τρία βασικά μέρη, το Model, το View και το Template.

Το Model αφορά τα μοντέλα (επίπεδο δεδομένων) δηλαδή την αλληλεπίδραση των οντοτήτων με τη βάση δεδομένων και τη διαχείριση των δεδομένων.

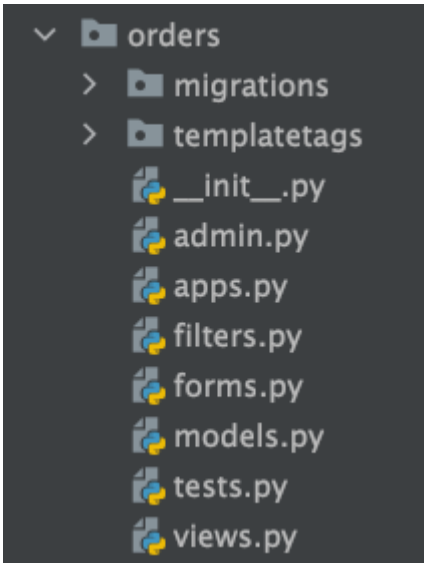
Το Template ανήκει στο επίπεδο εμφάνισης και χειρίζεται ολοκληρωτικά το κομμάτι του user interface.

Το View αναλαμβάνει το ρόλο του διαμεσολαβητή και του εκτελεστή των λειτουργιών, αλληλεπιδρώντας με τα μοντέλα ώστε να μεταφέρει τα δεδομένα και να δημιουργήσει templates.

Το σύστημα διαχείρισης αποθεμάτων, στην παρούσα πτυχιακή εργασία, βασίζεται στο παραπάνω μοτίβο αφού πρόκειται για ένα Django project. Χωρίζεται σε 4 διαφορετικά apps κάθε ένα από τα οποία υποδηλώνει τις οντότητες που υπάρχουν στο project.
Σε ένα ξεχωριστό κατάλογο, παρατηρείται και το αρχείο των ρυθμίσεων, μαζί με το αρχείο των urls (μονοπατιών της εφαρμογής) και το αρχείο παραμέτρων σύνδεσης παραμέτρων με τη βάση.
Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, το project περιλαμβάνει τους ρόλους manager, employee, customer και admin. Σε αυτούς περιέχονται οι οντότητες προιόν (product), παραγγελία (order), αντικείμενο παραγγελίας (orderitem), εταιρεία (company), κατηγορία (category) και παράδοση (shipping).
Στις παρακάτω εικόνες μπορούμε να παρατηρήσουμε μία γραφική αναπαράσταση ενός MVT μοτίβου αλλά και τη δομή ενός Django application. [8] [9]



Εικόνα 35 MVT pattern [Πηγή: https://www.javatpoint.com/django-mvt]



Εικόνα 36 Δομή ενός app

Στα πλάσια του βασικού συστήματος θα ήταν χρήσιμο να καταγραφούν επιπλέον ορισμένες παραδοχές που έχουν γίνει, οι οποίες αφορούν την υλοποίηση:

1) Υπάρχουν 2 φόρμες register. Μία για τους customers και μία για τους employees/managers.
2) Οι managers και οι employees δεν προυπάρχουν στο σύστημα και θα πρέπει αρχικά να κάνουν register request και στη συνέχεια ο admin να τους κάνει activate προκειμένου να μπορούν να κάνουν login.
3) Κατά το register request ο υποψήφιος manager/employee μπορεί να επιλέξει τη θέση την οποία επιθυμεί (manager/employee) προκειμένου στη συνέχεια να γίνει activate.

4) Αν κάποιος επιθυμεί να κάνει register ως customer, καταχωρείται αυτόματα στο σύστημα ως customer αφου κάνει register (σε διαφορετική register form).
5) Ο manager μπορεί να δει τη λίστα με τα products, να τα επεξεργαστεί ή να τα διαγράψει.
6) Ο manager μπορεί να επεξεργαστεί τα στοιχεία των employees και της company.
7) Ο manager μπορει δεν μπορεί να διαγράψει κάποιον employee αλλά αν κάνει deactivate την company τότε γίνονται αυτόματα deactivate όλοι οι employees που ανήκουν σε αυτή.
8) Ο employee διαχειρίζεται μόνο orders, βλέπει τους καταχωρημένους customers ενώ ο manager διαχειρίζεται products, employees, company (όχι orders).
9) Οι customers μπορούν να επιλέξουν κατηγορία προϊόντων, να δουν τη λίστα με τα products, το ιστορικό παραγγελιών τους (myorders), να προσθέσουν προϊόντα στο καλάθι (add to cart) και να δημιουργήσουν νέα παραγγελία προϊόντος.
10) Οι employees επεξεργάζονται orders (αλλαγή του status π.χ pending/approved κτλ).
11) Στο σύστημα προυπάρχουν καταχωρημένες ήδη 3 companies.
12) Έχει γίνει οι παραδοχή ότι το σύστημα δεν περιλαμβάνει τον ρόλο προμηθευτής.
13) Μόνο οι εγγεγραμμένοι χρήστες ως customers μπορούν να καταχωρήσουν μία παραγγελία (όχι guests). Μέχρι να αλλάξει το order status μιας παραγγελίας (από κάποιον employee) από την κατάσταση Pending που βρίσκεται μόλις καταχωρείται αρχικά σε κάποια άλλη κατάσταση (π.χ Processing), ο customer διατηρεί τη δυνατότητα επεξεργασίας της παραγγελίας (προσθήκη ή αφαίρεση προϊόντων). Μόλις το status αλλάξει (π.χ Approved) τότε αδειάζει το καλάθι και η επόμενη παραγγελία θα λογίζεται ως νέα.
14) Υποστηρίζεται λειτουργία αποστολής email τόσο σε περίπτωση μεταβολής του order status (π.χ Pending->Processing) όσο και σε περίπτωση καταχώρησης new order (αποστολή confirmation).
15) Υποστηρίζεται λειτουργία πληρωμής παραγγελίας (paypal ή credit card).

4.2 URLs

Τα urls χρησιμοποιούνται για να ανακατευθύνουν http requests στο σωστό view ανάλογα με αυτό που καλείται κάθε φορά. Επιστρέφουν επίσης διαφορετικό view για κάθε αντικείμενο αφού τα urls μπορούν να παίρνουν σαν όρισμα αριθμούς ή χαρακτήρες.
Η παρούσα εφαρμογή αποτελείται από τα εξής urls:

- register/ (για εγγραφή χρηστών employees/managers)
- register-customer/ (για εγγραφή χρηστών customers)
- login/ (σύνδεση χρηστών)
- logout/ (αποσύνδεση χρηστών)
- admin/ (διαχειριστικό πάνελ)
- inactive_employees/ (register requests των employees)
- inactive_managers/ (register requests των managers)
- activate_user/<id>/ (για αποδοχή εγγραφής χρήστη)
- orders_list/ (λίστα παραγγελιών)
- orders_list/<id>/orderitems (αντικείμενα της παραγγελίας)
- orders_list/<id>/shipping (πληροφορίες παράδοσης παραγγελίας)
- orders_list/update_order/<id> (ανανέωση κατάστασης παραγγελίας)
- employee/customers (καταχωρημένοι customers)
- products/categories/ (κατηγορίες προϊόντων)
- products/categories/<id> (επιλογή κατηγορίας προϊόντος)
- create_product/ (προσθήκη νέου προϊόντος)
- update_product/ (ανανέωση στοιχείων προϊόντος)
- delete_product/ (διαγραφή προϊόντος)
- employees/ (λίστα προϊόντων)
- update_employee/ (ανανέωση στοιχείων employee)
- companies/ (λίστα καταχωρημένων εταιρειών)
- update_company/ (ανανέωση στοιχείων εταιρείας)
- deactivate/<id>/ (απενεργοποίηση εταιρείας->απενεργ. εργαζομένων κτλ.)
- customer/products/choose_category/ (κατηγορίες προϊόντων-> customer)
- customer/products/choose_category/<id> (επιλογή κατηγορίας ->customer)
- customer/mycart/ (καλάθι customer)
- customer/checkout/ (σύνοψη παραγγελίας)
- customer/update_item/ (ανανέωση ποσότητας προϊόντος στο καλάθι)
- customer/process_order/ (στοιχεία παράδοσης-πληρωμή-καταχώρηση παραγγ.)

- customer/myorders/ (ιστορικό παραγγελιών)

4.3 Models

Τα models αποτελούν αντικείμενα τα οποία σχηματικά αναπαριστούν τη μορφή και τα δεδομένα της βάσης. Περιλαμβάνουν πεδία για τα δεδομένα που υπάρχουν και προσφέρουν διάφορες λειτουργίες προσθήκης,διαγραφής, επεξεργασίας των δεδομένων.

Στις παρακάτω φωτογραφίες απεικονίζονται τα models της εφαρμογής:

```
class Company(models.Model):
    name = models.CharField(max_length=50, blank=False, null=True)
    date_added = models.DateTimeField(auto_now_add=True, blank=True, null=True)
    address = models.CharField(max_length=100, null=True)
    postcode = models.IntegerField(default='0', blank=True, null=True)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return self.name
```

Εικόνα 37 Company model

```
class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True, blank=True)
    name = models.CharField(max_length=200, null=True)
    phone = models.IntegerField(null=True)
    email = models.EmailField(null=True)

    def __str__(self):
        return self.name

    def get_customer_by_email(email):
        try:
            return Customer.objects.get(email=email)
        except:
            return False

    def does_exits(self):
        return Customer.objects.filter(email=self.email)
```

Εικόνα 38 Customer model

```python
class Employee(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    def __str__(self):
        return self.user
```

Εικόνα 39 Employee model

```python
STATUS = (
        ('Pending', 'Pending'),
        ('Approved', 'Approved'),
        ('Declined', 'Declined'),
        ('Processing', 'Processing'),
        ('Delivered', 'Delivered'),
    )
```

Εικόνα 40 Order status

```python
class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
    date_ordered = models.DateTimeField(auto_now_add=True)
    status = models.CharField(max_length=50, null=True, blank=True, choices=STATUS)
    transaction_id = models.CharField(max_length=100, null=True)

    def __str__(self):
        return str(self.id)

    @property
    def shipping(self):
        shipping = False
        orderitems = self.orderitem_set.all()
        for i in orderitems:
            shipping = True
        return shipping

    @property
    def get_cart_total(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.get_total for item in orderitems])
        return total

    @property
    def get_cart_items(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.quantity for item in orderitems])
        return total
```

Εικόνα 41 Order model

```python
class OrderItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.SET_NULL, blank=True, null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, blank=True, null=True)
    quantity = models.IntegerField(default=0, blank=False, null=True)
    date_added = models.DateTimeField(auto_now_add=True)

    @property
    def get_total(self):
        total = self.product.price * self.quantity
        return total
```

Εικόνα 42 OrderItem model

```python
class Shipping(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, blank=True, null=True)
    address = models.CharField(max_length=200, null=True)
    city = models.CharField(max_length=100, null=True)
    phone = models.CharField(max_length=50, null=True)
    date_added = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.address
```

Εικόνα 43 Shipping model

```python
class Category(models.Model):
    name = models.CharField(max_length=50, blank=True, null=True)
    def __str__(self):
        return self.name


class Product(models.Model):
    category = models.ForeignKey(Category, on_delete=models.SET_NULL, blank=False, null=True)
    prod_name = models.CharField(max_length=50, blank=False, null=True)
    quantity = models.IntegerField(default='0', blank=False, null=True)
    availability = models.BooleanField(default=False)
    last_updated = models.DateTimeField(auto_now_add=False, auto_now=True)
    price = models.IntegerField(default=0, blank=False)
    image = models.ImageField(null=True, blank=True)

    def __str__(self):
        return self.prod_name

    def get_products_by_id(cart_product_id):
        return Product.objects.filter(id__in=cart_product_id)

    @property
    def imageURL(self):
        try:
            url = self.image.url
        except:
            url = ''
        return url
```

Εικόνα 44 Category-Product model

4.4 Views

Τα views είναι συναρτήσεις που διαχειρίζονται αιτήματα του χρήστη. Προκειμένου να υλοποιήσουν τα αιτήματα αυτά έχουν πρόσβαση στα μοντέλα δεδομένων τα οποία και μπορούν να επεξεργαστούν προκειμένου να στείλουν ένα response στο χρήστη μέσω των templates.

4.4.1 Application Companies

- **company(request):** view το οποίο χρησιμοποιείται για να εμφανίσει τη λίστα των εταιρειών ταξινομημένες κατά id.
- **update_company(request, pk):** view το οποίο χρησιμοποιείται για να γίνει η επεξεργασία των δεδομένων μιας εταιρείας.
- **deactivate(request, pk):** view το οποίο χρησιμοποιείται για να γίνει deactivate μια εταιρεία από τον admin ή τον manager.

4.4.2 Application Invmanagement

- **registerPage(request):** view το οποίο χρησιμοποιείται για να καταχωρήσει ένας manager/employee register request ώστε μετά να έχει πρόσβαση στη εφαρμογή.
- **registerCustomerPage(request):** view το οποίο χρησιμοποιείται για να κάνει register ένας customer ώστε μετά να έχει πρόσβαση στη εφαρμογή.
- **loginPage(request):** view το οποίο χρησιμοποιείται για να συνδεθεί κάποιος χρήστης στην εφαρμογή.
- **logoutUser(request):** view το οποίο χρησιμοποιείται για να αποσυνδεθεί κάποιος χρήστης από την εφαρμογή.
- **inactive_employees(request):** view το οποίο χρησιμοποιείται ώστε ο admin να βλέπει τα register request των employees και να τα αποδέχεται ή να τα απορρίπτει.

- **inactive_managers(request):** view το οποίο χρησιμοποιείται ώστε ο admin να βλέπει τα register request των managers και να τα αποδέχεται ή να τα απορρίπτει.
- **activate_user(request, pk):** view το οποίο χρησιμοποιείται για να γίνει αποδεκτό το αίτημα εγγραφής του employee/manager και να μπορεί στη συνέχεια να έχει πρόσβαση την εφαρμογή.
- **home(request):** view το οποίο χρησιμοποιείται για να εμφανιστεί η αρχική σελίδα της εφαρμογής.
- **employees(request):** view το οποίο χρησιμοποιείται για να εμφανιστούν οι employees.
- **create_employee(request):** view το οποίο χρησιμοποιείται για να καταχωρηθεί νέος employee.
- **update_employee(request, pk):** view το οποίο χρησιμοποιείται για να γίνει επεξεργασία των στοιχείων ενός employee.
- **customers(request):** view το οποίο χρησιμοποιείται για να εμφανιστεί η λίστα των καταχωρημένων customers.
- **choose_categories(request):** view το οποίο χρησιμοποιείται για να εμφανιστούν οι διαθέσιμες κατηγορίες στον customer.
- **product_list_customer(request, pk):** view το οποίο χρησιμοποιείται για να εμφανιστεί η λίστα προϊόντων στον customer ανάλογα με την κατηγορία που επέλεξε.

4.4.3 Application Orders

- **orders(request):** view το οποίο χρησιμοποιείται για να εμφανιστούν οι καταχωρημένες παραγγελίες.
- **order_items(request, pk):** view το οποίο χρησιμοποιείται για να εμφανιστούν τα αντικείμενα μιας συγκεκριμένης παραγγελίας.
- **shipping_info(request, pk):** view το οποίο χρησιμοποιείται για να εμφανιστούν οι πληροφορίες παράδοσης μιας συγκεκριμένης παραγγελίας.
- **update_order(request, pk):** view το οποίο χρησιμοποιείται για να γίνει αλλαγή της κατάστασης μιας παραγγελίας.

- **send_mail(status, order_id, mail, customer_name):** view το οποίο χρησιμοποιείται για να σταλεί αυτόματα email στο χρήστη όταν αλλάξει η κατάσταση της παραγγελίας του.
- **mycart(request):** view το οποίο χρησιμοποιείται για να εμφανιστεί το καλάθι παραγγελιών του customer.
- **checkout(request):** view το οποίο χρησιμοποιείται για να εμφανιστεί η σύνοψη παραγγελίας του customer.
- **update_item(request):** view το οποίο χρησιμοποιείται για να ενημερωθει το καλάθι παραγγελιών του customer σε περίπτωση αυξομείωσης της ποσότητας ενός προϊόντος.
- **processOrder(request):** view το οποίο χρησιμοποιείται για να εμφανιστεί η σελίδα συμπλήρωσης των στοιχείων παράδοσης και πληρωμής της παραγγελίας.
- **send_order_confirmation(status, order_id, mail, customer_name):** view το οποίο χρησιμοποιείται για να σταλεί αυτόματα email επιβεβαίωσης στον customer όταν καταχωρήσει νέα παραγγελία.
- **order_page(request)**: view το οποίο χρησιμοποιείται για να εμφανιστεί το ιστορικό παραγγελιών ενός customer.

4.4.4 Application Products

- **categories(request):** view το οποίο χρησιμοποιείται για να εμφανιστούν οι κατηγορίες προϊόντων σε manager.
- **product_list(request, pk):** view το οποίο χρησιμοποιείται για να εμφανιστεί η λίστα προϊόντων σε manager.
- **create_product(request):** view το οποίο χρησιμοποιείται για να καταχωρηθεί νέο προιόν.
- **update_product(request, pk):** view το οποίο χρησιμοποιείται για να γίνει επεξεργασία των στοιχείων ενός προϊόντος.
- **delete_product(request, pk):** view το οποίο χρησιμοποιείται για να γίνει διαγραφή ενός προϊόντος.

4.5 Templates

Τα templates είναι αρχεία τα οποία αφορούν αποκλειστικά την εμφάνιση, τη δομή και το σχεδιασμό της εφαρμογής και γράφονται συνήθως στη γλώσσα HTML. Τα views δημιουργούν δυναμικά μία σελίδα HTML χρησιμοποιώντας κάποιο αντίστοιχο template. Στην εφαρμογή μας υπάρχει ένας φάκελος template μέσα στον οποίο υπάρχουν κατάλογοι για κάθε εφαρμογή με τα αντίστοιχα templates της. Αναλυτικότερα:

4.5.1 Auth

- **login.html:** template που αφορά την εμφάνιση της σελίδας σύνδεσης.
- **register_html:** template που αφορά την εμφάνιση της σελίδας εγγραφής manager/employee.
- **register-customer.html:** template που αφορά την εμφάνιση της σελίδας εγγραφής customer.

4.5.2 Companies

- **list_companies.html:** template που αφορά την εμφάνιση σελίδας διαθέσιμων εταιρειών.
- **update_company.html:** template που αφορά την εμφάνιση σελίδας ανανέωσης στοιχείων εταιρείας.

4.5.3 Users

- **activate_user.html:** template που αφορά την εμφάνιση σελίδας για την αποδοχή της εγγραφής χρήστη και την ενεργοποίηση του προκειμένου να συνδέεται στην εφαρμογή.
- **create_employee.html:** template που αφορά την εμφάνιση σελίδας για τη καταχώρηση νέου employee.
- **inactive_employees.html:** template που αφορά την εμφάνιση σελίδας για τα αιτήματα εγγραφών των employees.
- **inactive_managers.html:** template που αφορά την εμφάνιση σελίδας για τα αιτήματα εγγραφών των managers.

- **list_customers.html:** template που αφορά την εμφάνιση σελίδας των καταχωρημένων customers.
- **list_employees.html:** template που αφορά την εμφάνιση σελίδας των καταχωρημένων employees.
- **product_list_customer.html:** template που αφορά την εμφάνιση σελίδας προϊόντων με βάση την κατηγορία που επέλεξε ο customer.
- **update_employee.html:** template που αφορά την εμφάνιση σελίδας επεξεργασίας των στοιχείων ενός employee.

4.5.4 Orders

- **checkout.html:** template που αφορά την εμφάνιση σελίδας για τη σύνοψη νέας παραγγελίας.
- **myorders.html:** template που αφορά την εμφάνιση σελίδας ιστορικού παραγγελιών ενός customer.
- **list_orders.html:** template που αφορά την εμφάνιση σελίδας των καταχωρημένων παραγγελιών.
- **my_cart.html:** template που αφορά την εμφάνιση σελίδας του καλαθιού του customer.
- **order_items.html:** template που αφορά την εμφάνιση σελίδας των αντικειμένων μιας παραγγελίας.
- **shipping_info.html:** template που αφορά την εμφάνιση σελίδας πληροφοριών παράδοσης μιας παραγγελίας.

4.5.5 Products

- **categories.html:** template που αφορά την εμφάνιση σελίδας κατηγοριών σε manager.
- **create_product.html:** template που αφορά την εμφάνιση σελίδας καταχώρησης νέου προϊόντος.
- **delete_product.html:** template που αφορά την εμφάνιση σελίδας διαγραφής προϊόντος.
- **product_list.html:** template που αφορά την εμφάνιση σελίδας προιοντων σε manager.

- **choose_categories.html:** template που αφορά την εμφάνιση σελίδας για την επιλογή κατηγορίας προϊόντος.
- **footer.html:** template που αφορά την εμφάνιση του κάτω μέρους των σελίδων.
- **home.html:** template που αφορά την εμφάνιση της αρχικής σελίδας.
- **navbar.html:** template που αφορά την εμφάνιση μπάρας οδηγιών στις σελίδες.

4.6 DevOps αρχεία

Στην εφαρμογή μας συμπεριλαμβάνονται και κάποια αρχεία τα οποία χρησιμεύουν στην παραμετροποίηση με τα εργαλεία DevOps (που θα αναλυθούν στο επόμενο κεφάλαιο).

Πρόκειται είτε για αρχεία Jenkinsfile τα οποία αφορούν την ενσωμάτωση του κώδικα στον Jenkins server και τον τρόπο που θα γίνει το deployment της εφαρμογής μέσω αυτού σε κάποια εικονική μηχανή, είτε για τα αρχεία Dockerfile και docker-compose.yml που περιγράφουν τον τρόπο και τις προϋποθέσεις ώστε να γίνει deploy η εφαρμογή σε εικονική μηχανή με τη χρήση του Docker, είτε για το φάκελο k8s ο οποίος περιέχει αρχεία σχετικά με τη διαδικασία του deployment σε εικονική μηχανή με τη χρήση του Kubernetes.

Το deployment με τη χρήση της Ansible περιγράφεται στο επόμενο κεφάλαιο καθώς στα πλαίσια της παρούσας πτυχιακής εργασίας δημιουργήθηκε ένα ακόμα αποθετήριο κώδικά εκτός από αυτό του βασικού συστήματος ώστε να εξυπηρετηθεί ευκολότερα ο σκοπός αυτός.

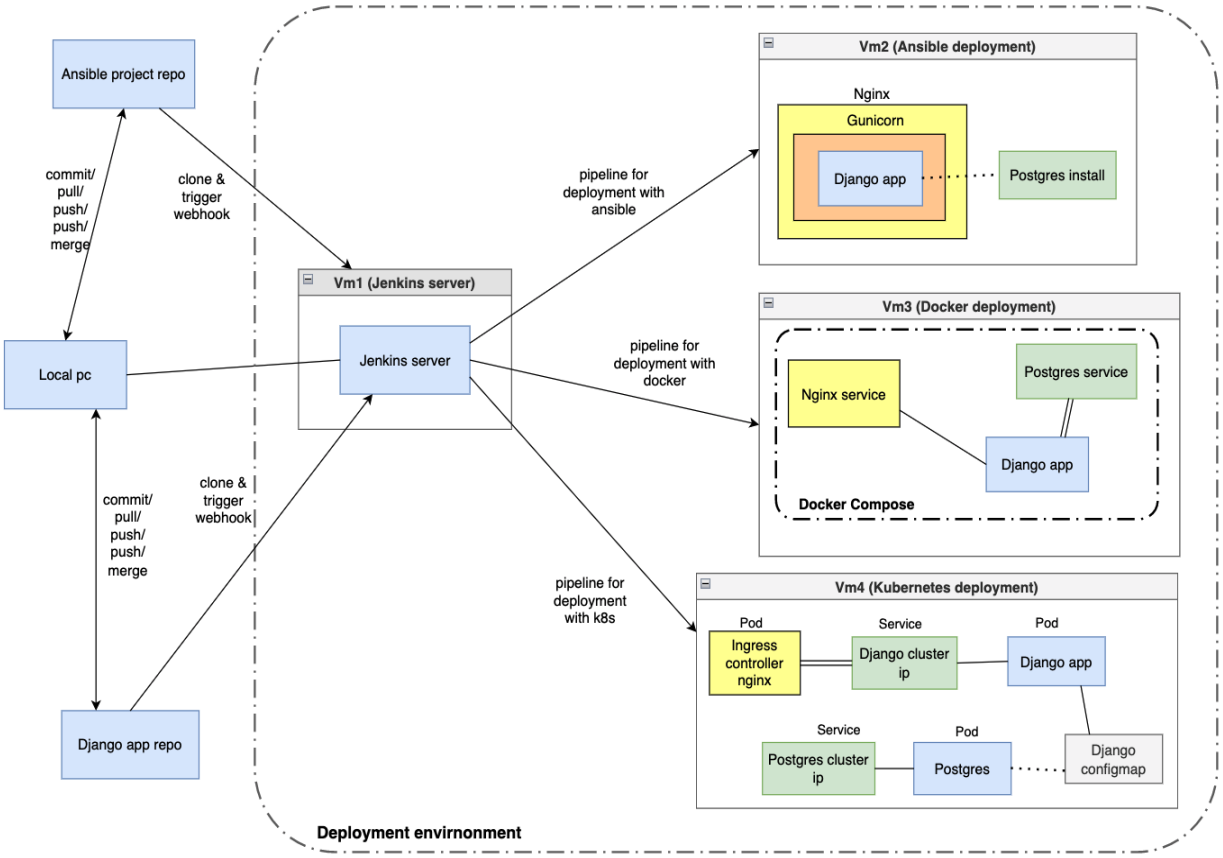
4.7 Υποστηριζόμενες λειτουργίες

Η βασική εφαρμογή αξιοποιεί ορισμένες βιβλιοθήκες προκειμένου να αναπτύξει λειτουργίες που θα τη καταστήσουν περισσότερο ρεαλιστική και εφαρμόσιμη. Συγκεκριμένα οι λειτουργίες αυτές περιλαμβάνουν:

- σελιδοποίηση (pagination) των προιόντων προκειμένου η εφαρμογή να είναι εμφανίσιμη και λειτουργική.
- φίλτρα (filtering) αναζήτησης ώστε ο χρήστης να έχει τη δυνατότητα να βρει εύκολα και γρήγορα αυτό που επιθυμεί.

- ταξινόμηση (sorting) προϊόντων με βάση την τιμή είτε σε αύξουσα είτε σε φθίνουσα σειρά.
- αποστολή email ενημέρωσης στον customer σε περίπτωση που μεταβληθεί η κατάσταση της παραγγελίας του ή σε περίπτωση που καταχωρήσει μία νέα παραγγελία.
- δυνατότητα πληρωμής μιας παραγγελίας αξιοποιώντας περιβάλλον sandbox.

Κεφάλαιο 5: Παραμετροποίηση και Ενσωμάτωση των Εργαλείων DevOps



Σχήμα 1 Αρχιτεκτονική Συστήματος

5.1 Ansible & Project Repository

Η παρούσα πτυχιακή εργασία δίνει τη δυνατότητα αυτοματοποίησης, ανάπτυξης και εκτέλεσης της εφαρμογής διαχείρισης αποθεμάτων σε διαφορετικά περιβάλλοντα (βλ. εικονικές μηχανές, οι εικονικές μηχανές που χρησιμοποιήθηκαν είναι από τον πάροχο Microsoft). Η πρώτη περίπτωση με την οποία καθίσταται αυτό εφικτό είναι αξιοποιώντας την Ansible (βλ. Κεφάλαιο 3, 3.6).

Για την επίτευξη του σκοπού αυτού απαιτήθηκαν ορισμένες ενέργειες:

α) Δημιουργία του αρχείου Jenkinsfile στο project του βασικού συστήματος το οποίο περιέχει κάποια stages και μέσα σε αυτά κάποια βήματα (steps) που θα ακολουθήσει ο jenkins για να γίνει deploy η εφαρμογή (π.χ πώς να εκτελέσει το ansible playbook που περιέχει οδηγίες εγκαταστασης).

β) Δημιουργία και παραμετροποίηση ενός νέου project (ansible-inventory-management-system). Στο project αυτό έχουν οριστεί μεταβλητές περιβάλλοντος (π.χ host, στοιχεία σύνδεσης με βάση δεδομένων κτλ.), και κυρίως τα playbooks τα οποία περιέχουν τα βήματα (tasks) που θα πρέπει να γίνουν ώστε να εγκατασταθεί και να εκτελεστεί η εφαρμογή στην εικονική μηχανή. Εκτός του playbook με τις οδηγίες εγκατάστασης της εφαρμογής, έχει δημιουργηθεί και ένα playbook που περιγράφει τα βήματα εγκατάστασης και της δημιουργίας ενός χρήστη στην βάση δεδομένων ώστε να αλληλεπιδρούν με τη βασική εφαρμογή. Μέσα στα playbooks ορίζονται μεταξύ άλλων και δύο πολύ σημαντικά εργαλεία για την λειτουργία και την πρόσβαση στην εφαρμογή απομακρυσμένα, ενός WSGI server (Gunicorn) και ενός Web Server (Nginx), όπως και το αρχείο hosts που αναφέρει πληροφορίες (π.χ port, ip κτλ.) για τον απομακρυσμένο server που θα εκτελεστεί η εφαρμογή.

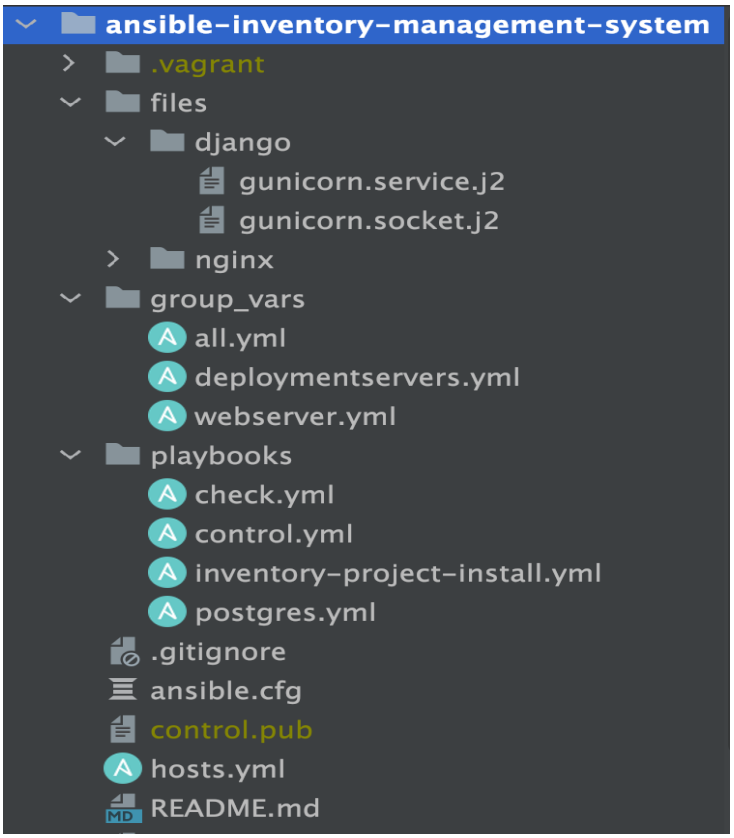
γ) Η παραμετροποίηση του Jenkins server, δημιουργώντας credentials σύνδεσης με το απομακρυσμένο μηχάνημα εκτέλεσης και η δημιουργία ενός pipeline που θα αναφέρεται στο παραπάνω Jenkinsfile του project μας ώστε να καταλάβει ο Jenkins σε ποιο αρχείο είναι οι οδηγίες που πρέπει να εκτελέσει.

δ) Η δημιουργία webhooks στα repositories του βασικού project και του ansible project προκειμένου οι αλλαγές που γίνονται στον κώδικα να ενσωματώνονται αυτόματα στον Jenkins server (π.χ κάνοντας commit ή push).

Εφόσον τα παραπάνω βήματα έχουν υλοποιηθεί, η εφαρμογή μας είναι έτοιμη να γίνει deploy είτε κάνοντας απλά build στο job του Jenkins Server είτε κάνοντας κάποια αλλαγή στον κώδικα μας και προωθώντας τη στο αποθετήριο (github).

Με τον τρόπο αυτό, θα έχουμε τη δυνατότητα να δούμε την εφαρμογή, που εκτελείται απομακρυσμένα, στο μηχάνημα μας έχοντας κάνει χρήση της τεχνολογίας Ansible.

Εικόνα 45 Deployment stage (Ansible)



Εικόνα 46 Δομή Ansible project

5.2 Jenkins Server

Σε ότι αφορά τον Jenkins server, όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, πρόκειται για ένα εργαλείο αυτοματοποίησης το οποίο υποστηρίζει την τεχνική του CI/CD, δηλαδή τη συνεχή ενσωμάτωση των αλλαγών που γίνονται στον κώδικα πραγματοποιώντας build κι tests και την προώθηση τους σε περιβάλλον production.

Ο Jenkins server, λοιπόν λειτουργεί σαν συνδετικός κρίκος ανάμεσα στο αποθετήριο του κώδικα μας και στα περιβάλλοντα που θα γίνει deploy η εφαρμογή. Εγκαθίσταται σε μία εικονική μηχανή και εφόσον έχουν παραμετροποιηθεί σωστά τα credentials σύνδεσης τόσο με τα αποθετήρια κώδικα όσο και με τις εικονικές μηχανές τότε μπορεί να υπηρετήσει αποτελεσματικά το ρόλο για τον οποίο χρησιμοποιείται.

Απαραίτητη προϋπόθεση είναι για κάθε περιβάλλον που θέλουμε να κάνουμε την εφαρμογή μας deploy, να έχουμε το αντιστοιχο Jenkinsfile. Έτσι θα μπορεί ο Jenkins (δηλώνοντας το όνομα του Jenkinsfile σε ένα pipeline) να βρίσκει το αρχείο με τις οδηγίες εγκατάστασης.

Εικόνα 47 Script Path

5.3 Docker - Docker Compose

Προκειμένου να γίνει το deployment στην εικονική μηχανή αξιοποιώντας το docker χρειάζεται να δημιουργηθούν δύο αρχείο στο project του βασικού συστήματος, το docker-compose.yml και το Dockerfile. Το Dockerfile μέσα από βήματα που εκτελούνται σειριακά ορίζει τι χρειάζεται η εφαρμογή για να τρέξει σε περιβάλλον docker. Συγκεκριμένα, στο Dockerfile της εφαρμογής μας ορίζουμε:

α) μία έκδοση της python που χρειάζεται να γίνει install

β) ένα working directory

γ) τον κατάλογο που πρέπει να γίνει copy και περιέχει τα αρχεία που θέλουμε

δ) τα requirements (βιβλιοθήκες) που πρέπει να γίνουν install

ε) την port που θα γίνει expose η εφαρμογή

στ) την εντολή όταν ξεκινήσει το container για να γίνει expose αξιοποιώντας τον gunicorn

docker-compose.yml | Dockerfile

```
# Use an existing docker image as a base
FROM python:3.9-buster

#Change working directory
WORKDIR /app

COPY . ./

RUN pip install -r requirements.txt

RUN python manage.py collectstatic --noinput

EXPOSE 8000/tcp

# Tell what to do when it starts as a container
CMD ["gunicorn", "inventoryproject.wsgi:application", "--bind", "0.0.0.0:8000"]
```

Εικόνα 48 Dockerfile

Στο αρχείο docker-compose.yml ορίζουμε κάποια services τα οποία χρειάζεται η εφαρμογή για να εκτελεστεί και συγκεκριμένα τη βάση δεδομένων με τις παραμέτρους σύνδεσης, τη βασική εφαρμογή διαχείρισης αποθεμάτων έχοντας ορίσει το dockerfile που πρέπει να διαβαστεί, το αρχείο με τις μεταβλητές περιβάλλοντος, volumes τα οποία αποθηκεύουν

διάφορα δεδομένα, την port που θα τρέχει η εφαρμογή στον container, και ένα nginx image ώστε να γίνει expose η εφαρμογή εξωτερικά.

docker-compose.yml | Dockerfile

```yaml
services:
  db:
    image: postgres:13
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_USER=dbuser
      - POSTGRES_PASSWORD=pass123
      - POSTGRES_DB=test_db
    volumes:
      - dj_postgres_data:/var/lib/postgresql/data/
  django:
    build:
      context: .
      dockerfile: Dockerfile
    env_file:
     - ./inventoryproject/.env
    ports:
      - "8000:8000"
    volumes:
      - static_volume:/app/static
    depends_on:
      - db
  nginx:
    image: nginx:latest
    volumes:
      - ./assets/nginx/nginx.http.config:/etc/nginx/nginx.conf
      #- ./assets/nginx/certs:/etc/nginx/certs
      - static_volume:/app/static
    ports:
      # - "443:443"
      - "80:80"
```

Εικόνα 49 docker-compose

Τέλος, έχει δημιουργηθεί ένα δεύτερο Jenkinsfile (Jenkinsfile2) το οποίο αντίστοιχα περιγράφει τα βήματα του deployment με docker. Αυτό επιτυγχάνεται αν στο deployment

stage ορίσουμε την εντολή docker-compose up, έχοντας παράλληλα δημιουργήσει απομακρυσμένη σύνδεση με την εικονική μηχανή που θέλουμε να εκτελεστεί η εφαρμογή.

5.4 Kubernetes

Η τελευταία μέθοδος που αξιοποιήθηκε στα πλαίσια της πτυχιακής εργασίας για να γίνει deploy η εφαρμογή μας είναι το Kubernetes. Για την επίτευξη αυτού του σκοπού χρησιμοποιήθηκαν:

α) ένας φάκελος k8s εντός του project της βασικής εφαρμογής, μέσα στον οποίο δημιουργήθηκαν δύο ακόμα φάκελοι, ο db και ο django. Από τη μία ο db περιέχει ό,τι χρειάζεται σχετικά με τη βάση δεδομένων, δηλαδή έχει οριστεί ένα service τύπου cluster ip το οποίο είναι χρήσιμο ώστε τα υπόλοιπα components της εφαρμογής να βρίσκουν τη βάση δεδομένων, ένα persistent volume claim το οποίο αφορά τα volumes ορίζοντας το χώρο, τα modes κτλ. που απαιτούνται για την αποθήκευση των δεδομένων και τέλος η δημιουργία ενός deployment της βάσης για την εκτέλεση της containerized εφαρμογής μας. Το deployment καθοδηγεί το Kubernetes πώς να φτιάξει και να ανανεώσει τα instances της εφαρμογής ώστε να τρέξουν σε συγκεκριμένα nodes. Αν ένα node πέσει τότε ο deployment controller αντικαθιστά το instance με ένα άλλο σε κάποιο άλλο node. Στο δικό μας deployment της βάσης έχουν οριστεί replicas, τα volumes (βλ. persistent volume claim) και ένα container αξιοποιώντας ένα postgres image. Από την άλλη στο φάκελο django ακολουθείται παρόμοια διαδικασία δηλαδή δημιουργία ενός ακόμα service τύπου cluster ip ωστε τα υπόλοιπα components να βρίσκουν τη βασική εφαρμογή, έναν ingress controller δηλαδή ένα pod που τρέχει στο cluster μας και θα λειτουργεί σαν proxy server για να γίνεται expose η εφαρμογή (δηλώνοντας παράλληλα ένα hostname το οποίο αντιστοιχεί σε μία dns υπηρεσία με την public ip που έχει το cluster), το deployment με το αντίστοιχο container που περιέχει το image της εφαρμογής από το docker hub και το configmap που ορίζει τα key-value pairs του .env αρχείου (δηλ. παραμετροποίηση).

β) ένα αρχείο Jenkinsfile (Jenkinsfile3) με τα βήματα ώστε να γίνει deploy (βλ. δημιουργία του image και προώθησης του στο registry, χρήση της εντολής kubectl apply κτλ.).

Δεν θα πρέπει και σε αυτήν τη περίπτωση να παραληφθεί η παραμετροποίηση του Jenkins server, όπως και παραπάνω (βλ. 5.3), με τη δημιουργία ενός pipeline που θα αναφέρεται στο Jenkinsfile3. Παρακάτω απεικονίζονται ενδεικτικά οι περιπτώσεις που περιγράφονται στο κεφάλαιο αυτό.

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pg-pvc-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Εικόνα 50 Persistent Volume Claim (postgres)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: pg-cluster-ip
spec:
  type: ClusterIP
  selector:
    component: postgres
  ports:
    - port: 5432
      targetPort: 5432
```

Εικόνα 51 Cluster-ip (postgres)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      component: postgres
  template:
    metadata:
      labels:
        component: postgres
    spec:
      volumes:
        - name:  pg-database-persistent-volume-storage
          persistentVolumeClaim:
            claimName: pg-pvc-claim
      containers:
        - name: postgres
          image: "postgres:11"
          ports:
            - containerPort: 5432
          env:
            - name: POSTGRES_USER
              valueFrom:
                secretKeyRef:
                  name: pg-user
                  key: PGUSER
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: pg-user
                  key: PGPASSWORD
```

Εικόνα 52 Deployment (postgres)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: django-clip
spec:
  type: ClusterIP
  selector:
    component: django
  ports:
    - port: 8000
      targetPort: 8000

```

Εικόνα 53 Cluster ip (inventory app)

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: django-ingress
spec:
  rules:
  - host: "django.westeurope.cloudapp.azure.com"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: django-clip
            port:
              number: 8000
```

Εικόνα 54 Ingress Controller (inventory app)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: django
spec:
  replicas: 1
  selector:
    matchLabels:
      component: django
  template:
    metadata:
      labels:
        component: django
    spec:
      initContainers:
        - name: django-init
          image: stas33/django
          command: ["/bin/bash"]
          args: ["-c", "python manage.py makemigrations && python manage.py migrate"]
      containers:
        - name: django
          image: stas33/django
          ports:
            - containerPort: 8000
          envFrom:
            - configMapRef:
                name: django-config
          imagePullPolicy: Always
```

Εικόνα 55 Deployment (inventory app)

```
23
24          stage('Docker build & push image') {
25              environment {
26                  IMAGE='stas33/django'
27                  DOCKER_USERNAME='stas33'
28                  DOCKER_PASSWORD=credentials('docker-passwd')
29              }
30              steps {
31                  sh '''
32                  echo $BUILD_ID
33                  COMMIT_ID=$(git rev-parse --short HEAD)
34                  echo $COMMIT_ID
35                  TAG=$COMMIT_ID-$BUILD_ID
36                  docker build -t $IMAGE -t $IMAGE:$TAG .
37                  docker login -u="$DOCKER_USERNAME" -p="$DOCKER_PASSWORD"
38                  docker push $IMAGE --all-tags
39                  '''
40              }
41          }
42
43          stage('Deploy to K8s') {
44              steps {
45                  sh '''
46                  kubectl config use-context microk8s
47                  cd k8s/db
48                  ls *.yaml | while read x; do kubectl apply -f $x; done
49                  cd ../django
50                  ls *.yaml | while read x; do kubectl apply -f $x; done
51                  '''
52              }
53          }
```

Εικόνα 56 Build & Deployment stages (K8s)

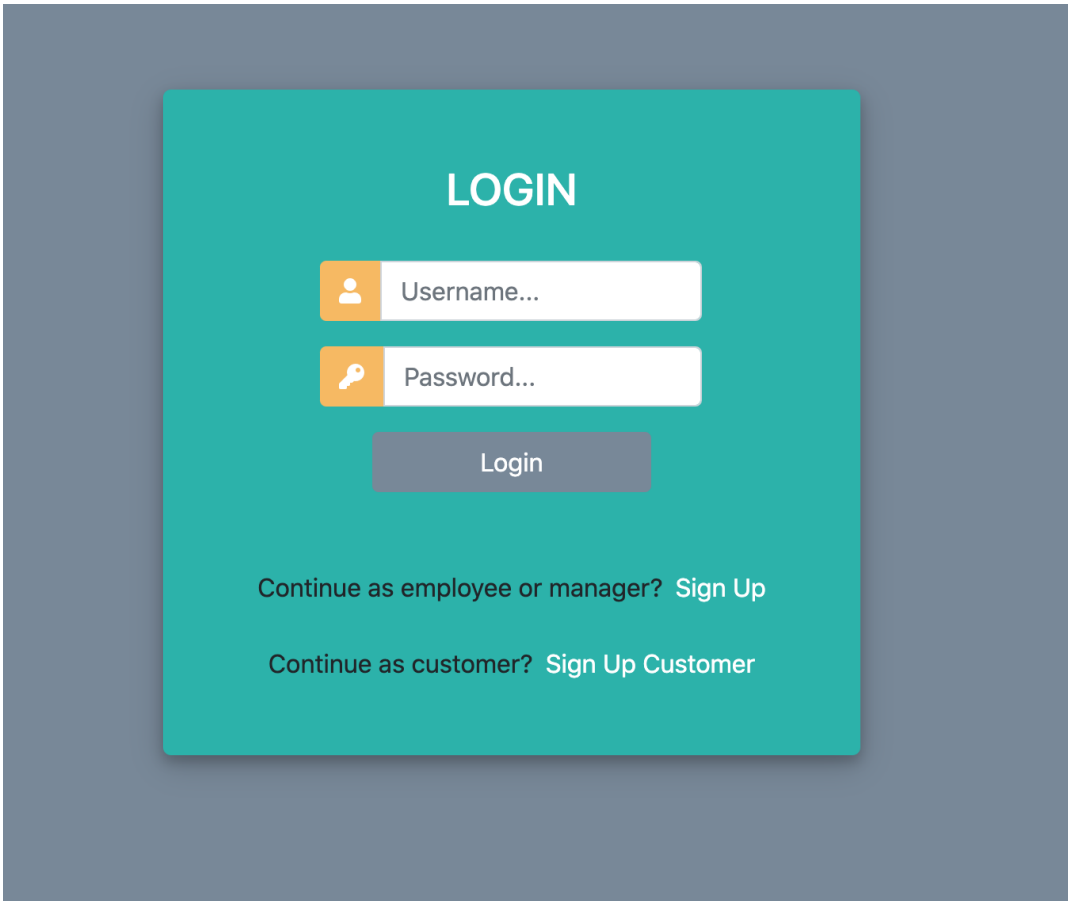
Κεφάλαιο 6: Σενάρια Χρήσης

Στο Κεφάλαιο αυτό απεικονίζονται και αναλύονται τα σενάρια χρήσης και η λειτουργικότητα της βασικής εφαρμογής για τον κάθε ρόλο. Έχουν ληφθεί υπόψη οι παραδοχές που έχουν γίνει στο κεφάλαιο 4.1 .

6.1 Σελίδες Εγγραφής & Σύνδεσης

Προκειμένου να εμφανιστεί η σελίδα σύνδεσης θα πρέπει να εισάγουμε το σωστό URL path. Δεδομένου ότι η εφαρμογή μπορεί να γίνει deploy σε 3 διαφορετικά περιβάλλοντα, το URL θα είναι είτε η ip διεύθυνση της εικονικής μηχανής (περιπτώσεις Ansible και Docker deployment) είτε ένα όνομα domain name που έχουμε δώσει (περίπτωση Kubernetes deployment) τα οποία περιγράφονται και στο αρχείο README.md.

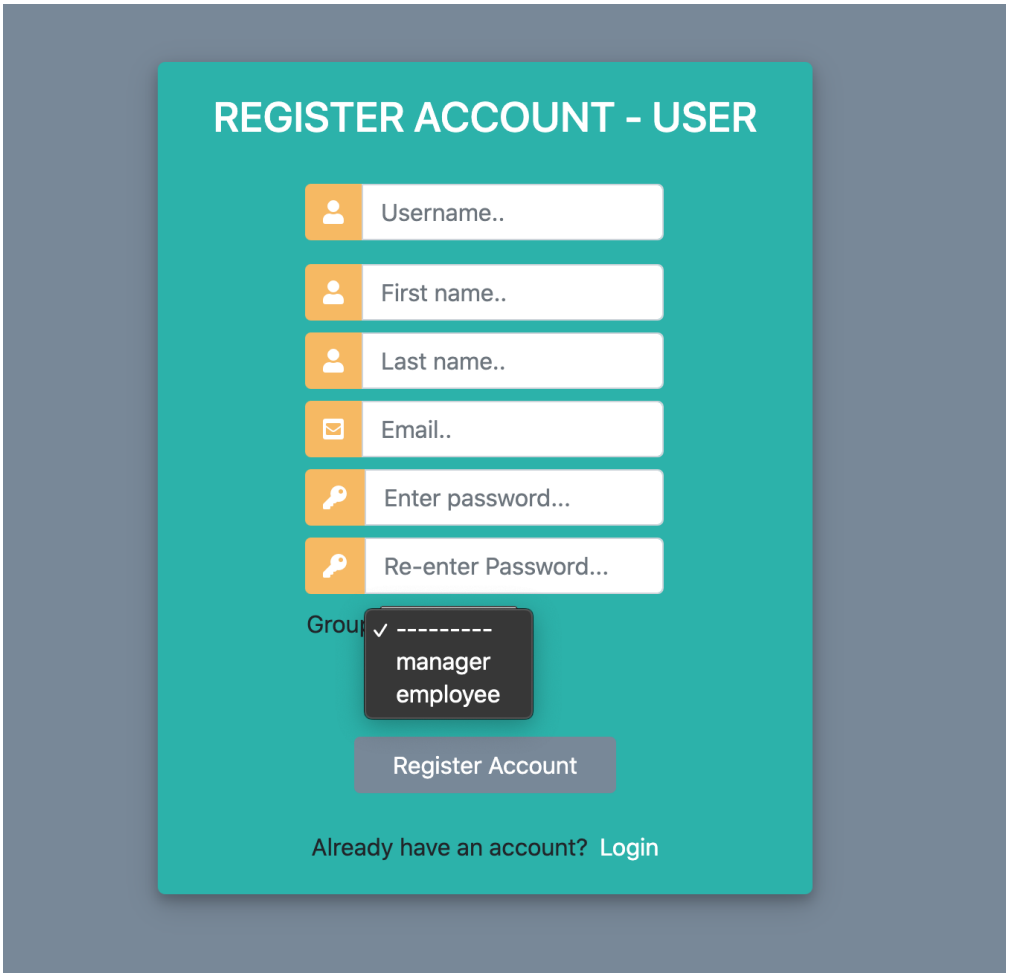
Η σελίδα σύνδεσης περιέχει πεδία για username και password και links εγγραφής ανάλογα με το ρόλο του χρήστη.



Εικόνα 57 Σελίδα Σύνδεσης

6.1.1 Manager/Employee Registration

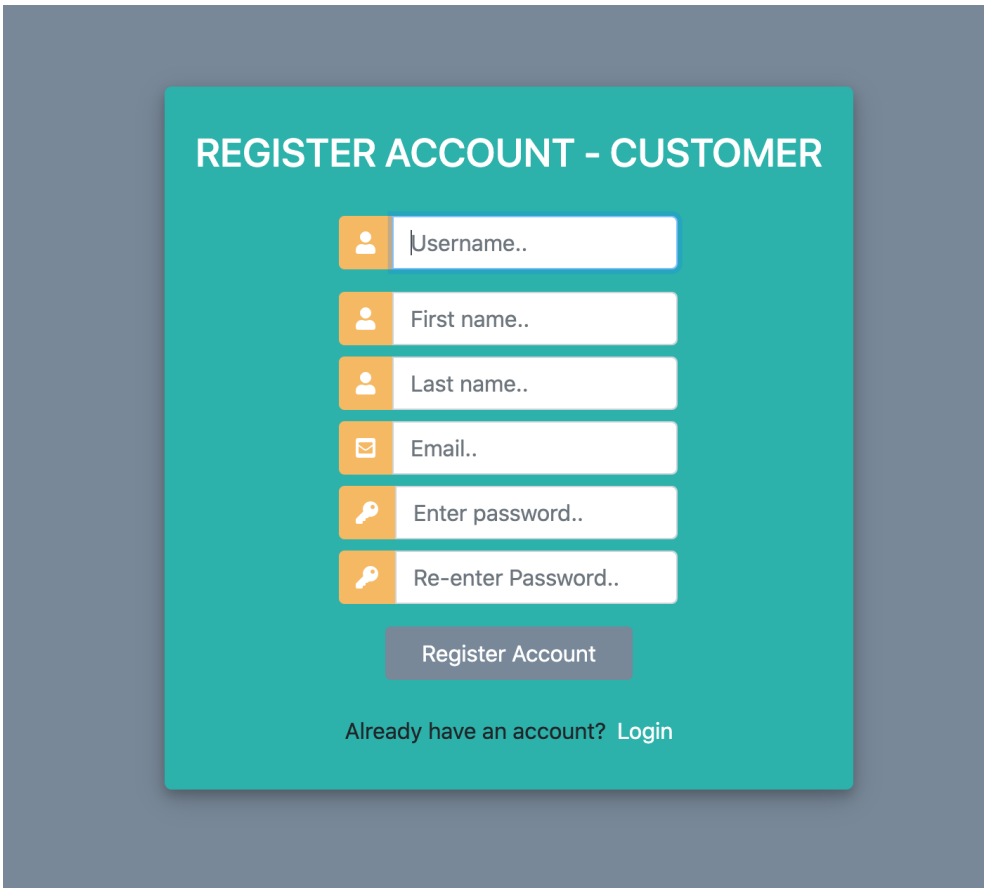
Σε περίπτωση που κάποιος χρήστης επιθυμεί να εγγραφεί ως employee ή manager θα πρέπει να συμπληρώσει την αντίστοιχη φόρμα εγγραφής και στη συνέχεια να εγκριθεί από τον admin. Η συγκεκριμένη φόρμα περιλαμβάνει πεδία όπως username, first name, last name, email, password και group ώστε να επιλέξει (manager/employee).



Εικόνα 58 Σελίδα Εγγραφής (employee/manager)

6.1.2 Customer Registration

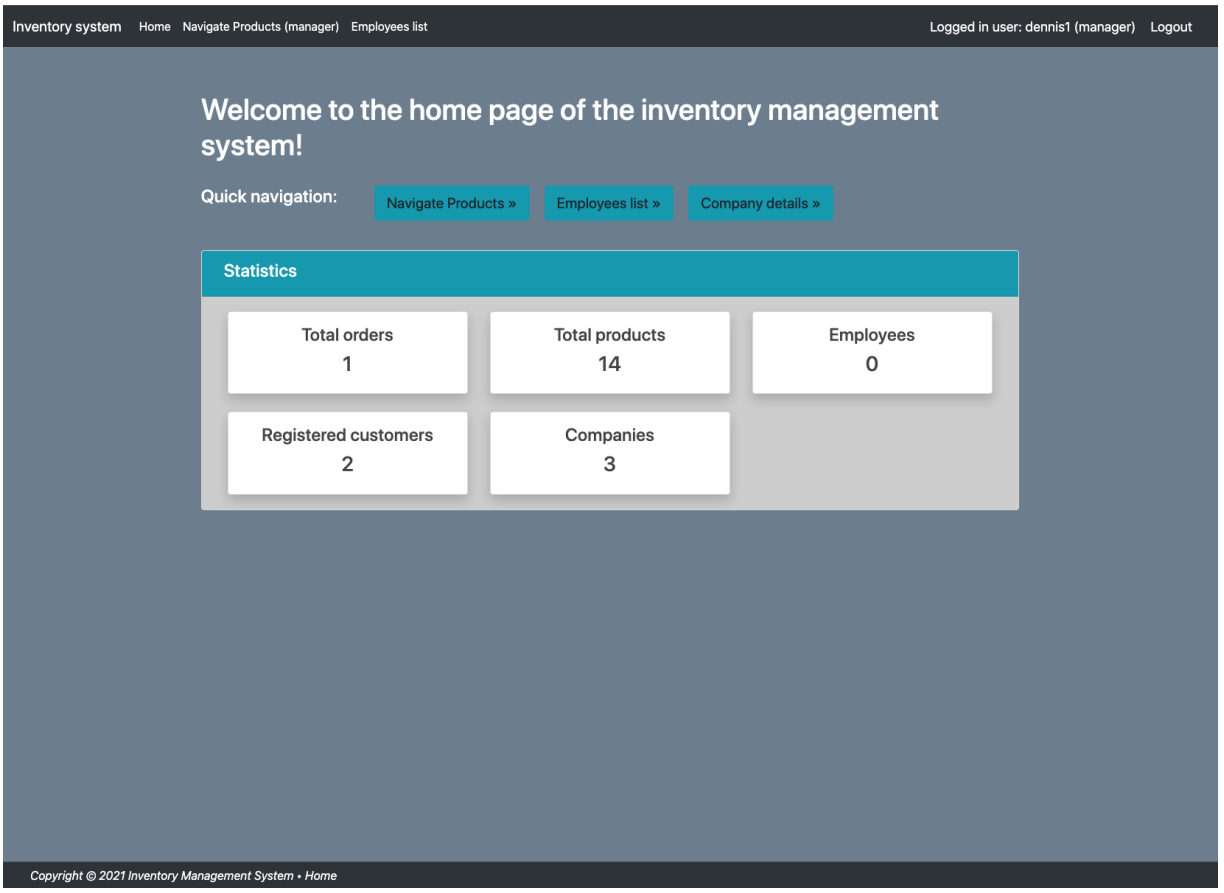
Σε περίπτωση που κάποιος χρήστης επιθυμεί να εγγραφεί ως customer θα πρέπει να συμπληρώσει την αντίστοιχη φόρμα εγγραφής χωρίς να χρειάζεται κάποια έγκριση (καταχωρείται αυτόματα σαν νέος χρήστης). Η συγκεκριμένη φόρμα περιλαμβάνει τα ίδια πεδία με τη διαφορά ότι απουσιάζει εκείνο της επιλογής group.



Εικόνα 59 Σελίδα εγγραφής (customer)

6.2 Λειτουργίες Manager

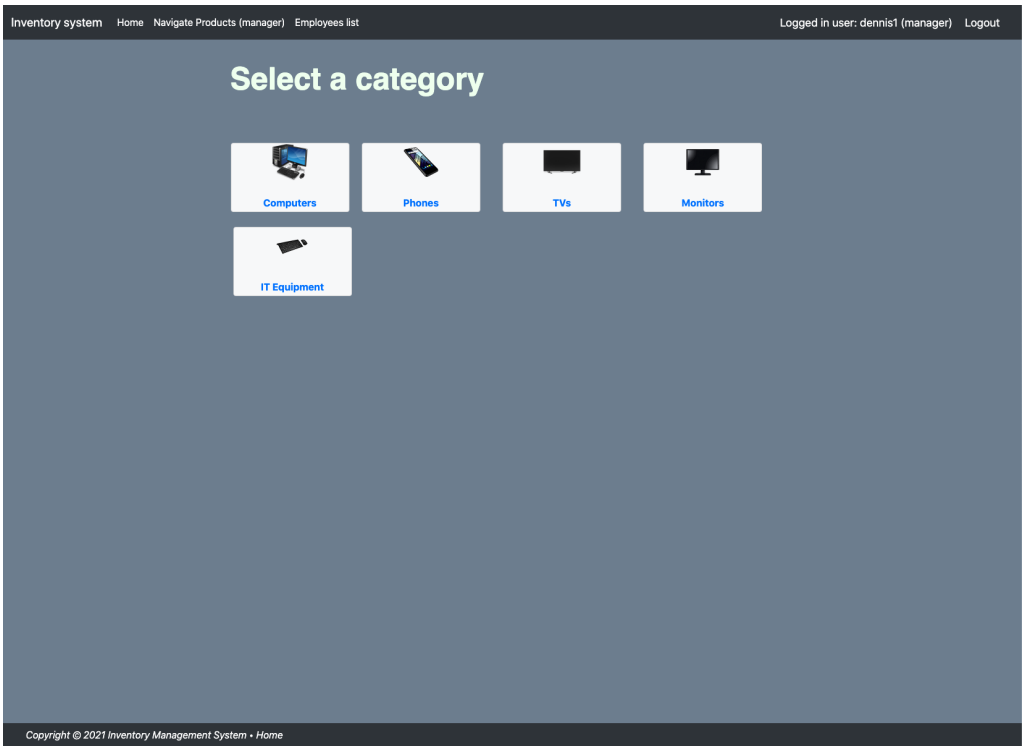
Αφότου κάποιος χρήστης γίνει αποδεκτός από τον admin ως manager τότε έχει τη δυνατότητα να κάνει login, ώστε να οδηγηθεί στην αρχική σελίδα. Στην αρχική σελίδα περιλαμβάνονται οι τρεις λειτουργίες του ρόλου manager και από κάτω ένας πίνακας στατιστικών.



Εικόνα 60 Αρχική σελίδα manager

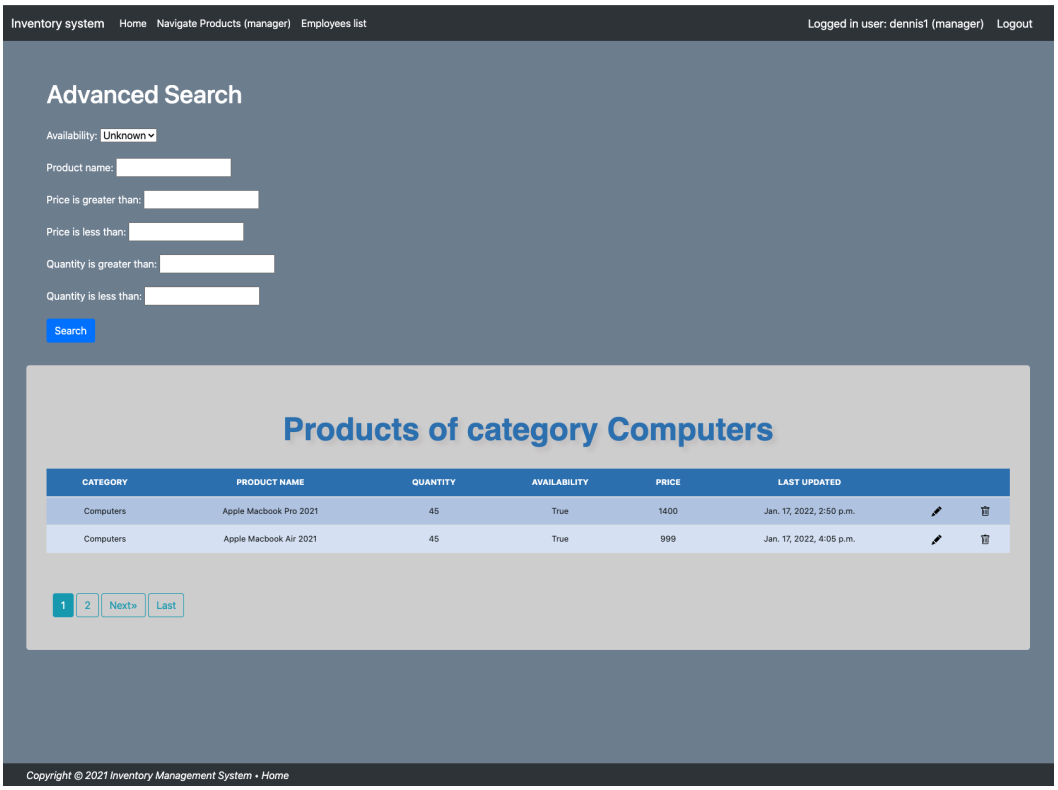
6.2.1 Επιλογή Navigate Products

Σε περίπτωση που ο χρήστης επιλέξει Navigate Products, εμφανίζεται σελίδα ώστε να επιλέξει κατηγορία προιόντος.



Εικόνα 61 Επιλογή κατηγορίας

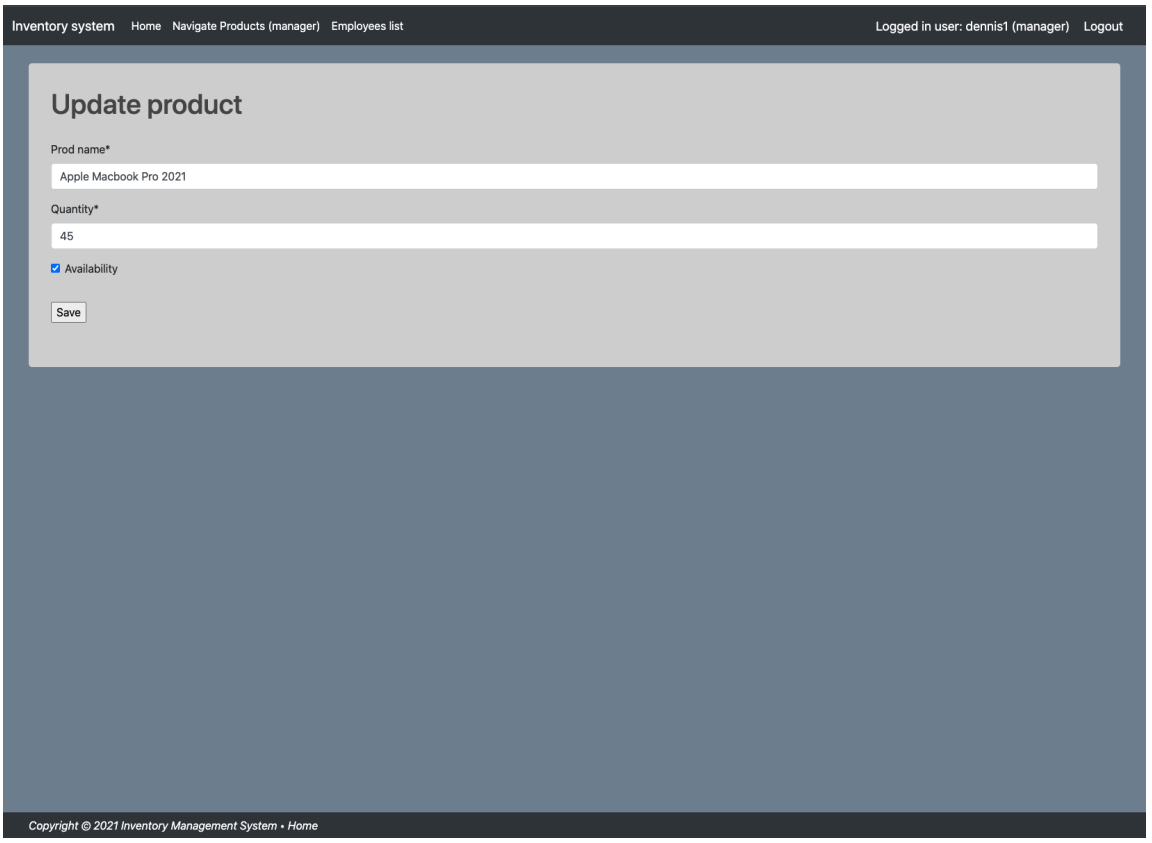
Στη συνέχεια αφού επιλέξει κάποια κατηγορία, εμφανίζεται ο πίνακας με τα προϊόντα της κατηγορίας, στον οποίο μπορεί να κάνει πλήθος ενεργειών όπως αναζήτηση με κάποιο κριτήριο, επεξεργασία στοιχείων προϊόντος, διαγραφή προϊόντος.



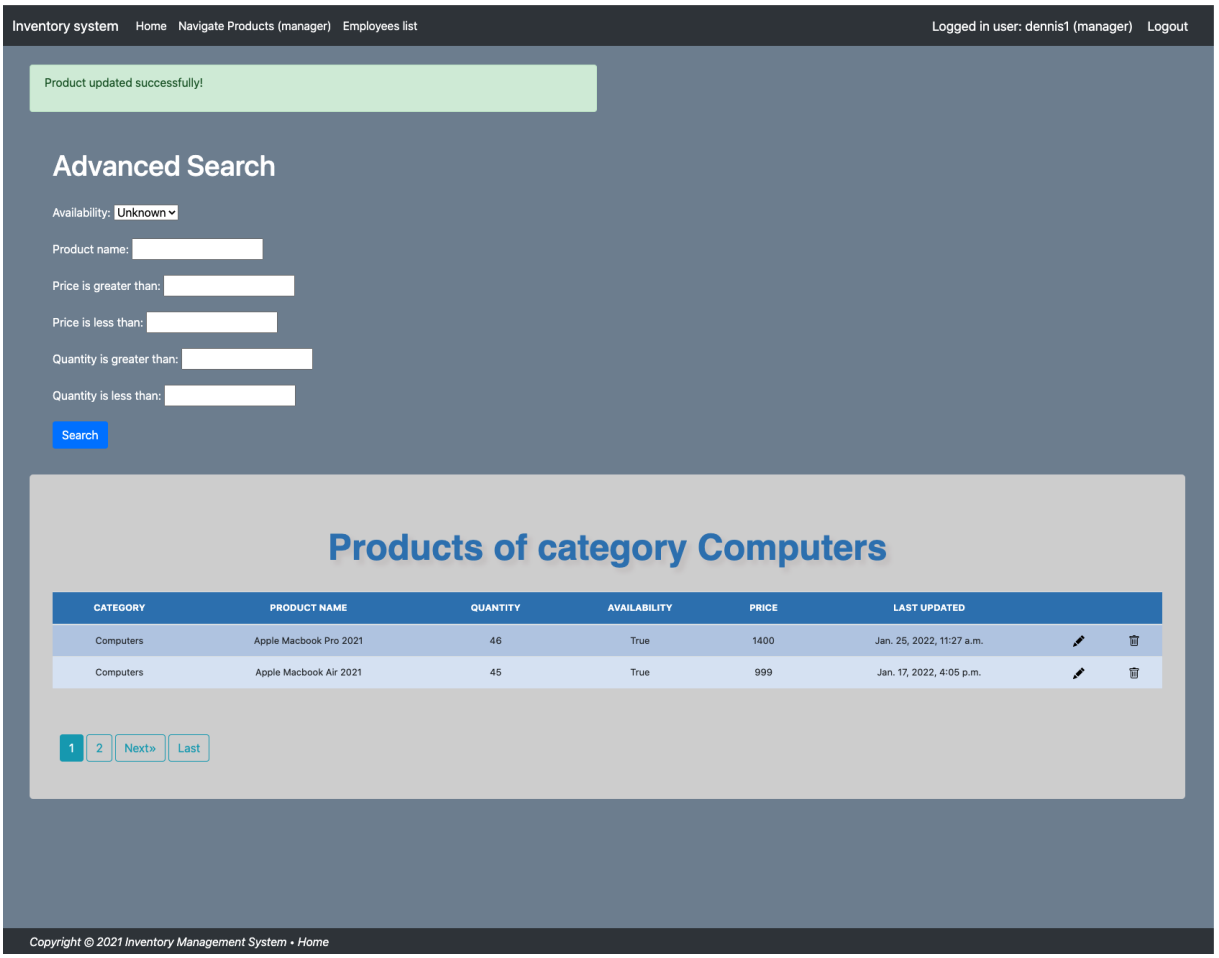
Εικόνα 62 Εμφάνιση προϊόντων (manager)

6.2.1.1 Επεξεργασία προιόντος

Εφόσον επιλέξει επεξεργασία προϊόντος, εμφανίζεται σελίδα όπου μπορεί να επεξεργαστεί το όνομα, την ποσότητα και τη διαθεσιμότητα του προϊόντος . Μόλις ολοκληρώσει την ενέργεια τότε πατάει το κουμπί save και οδηγείται στην αρχική σελίδα όπου και εμφανίζεται μήνυμα επιτυχίας.



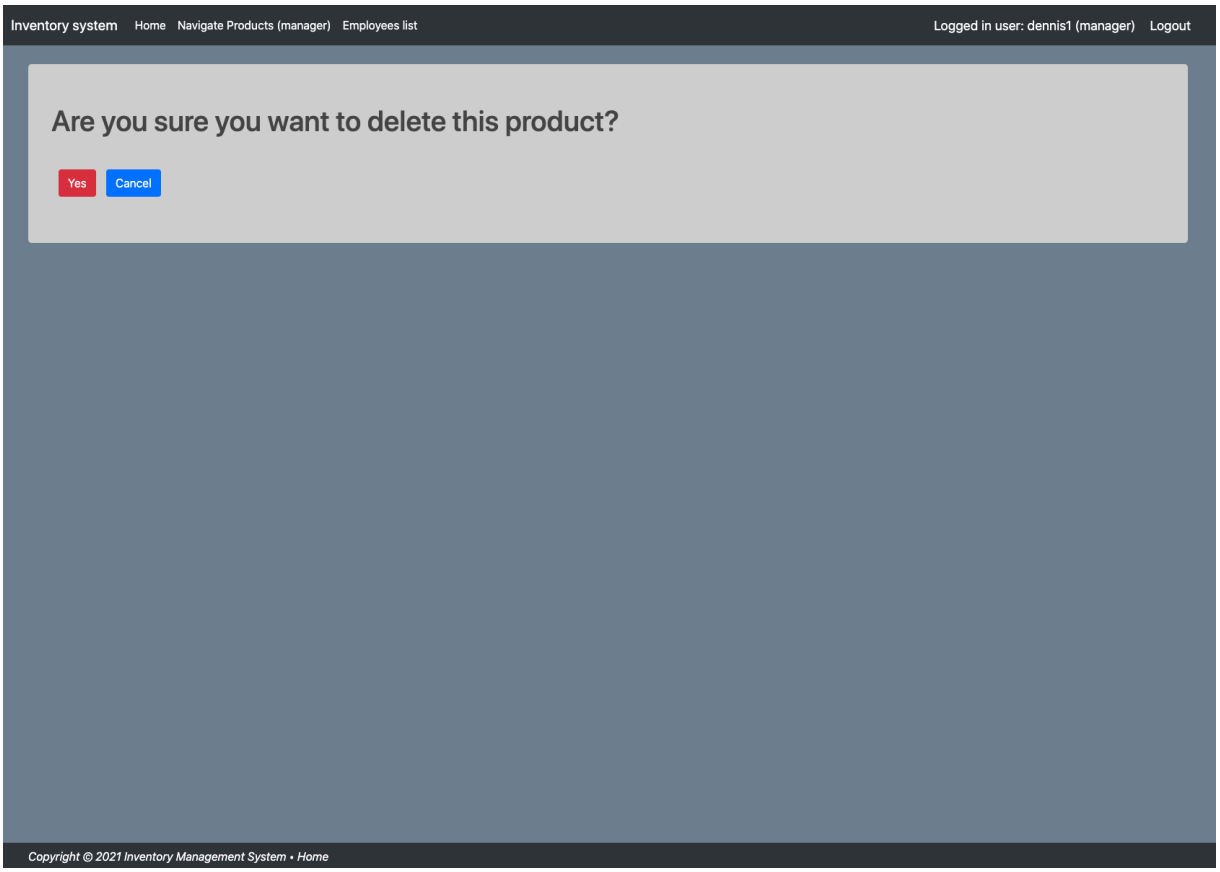
Εικόνα 63 Επεξεργασία προϊόντος (manager)



Εικόνα 64 Επιτυχής επεξεργασία

6.2.1.2 Διαγραφή προιόντος

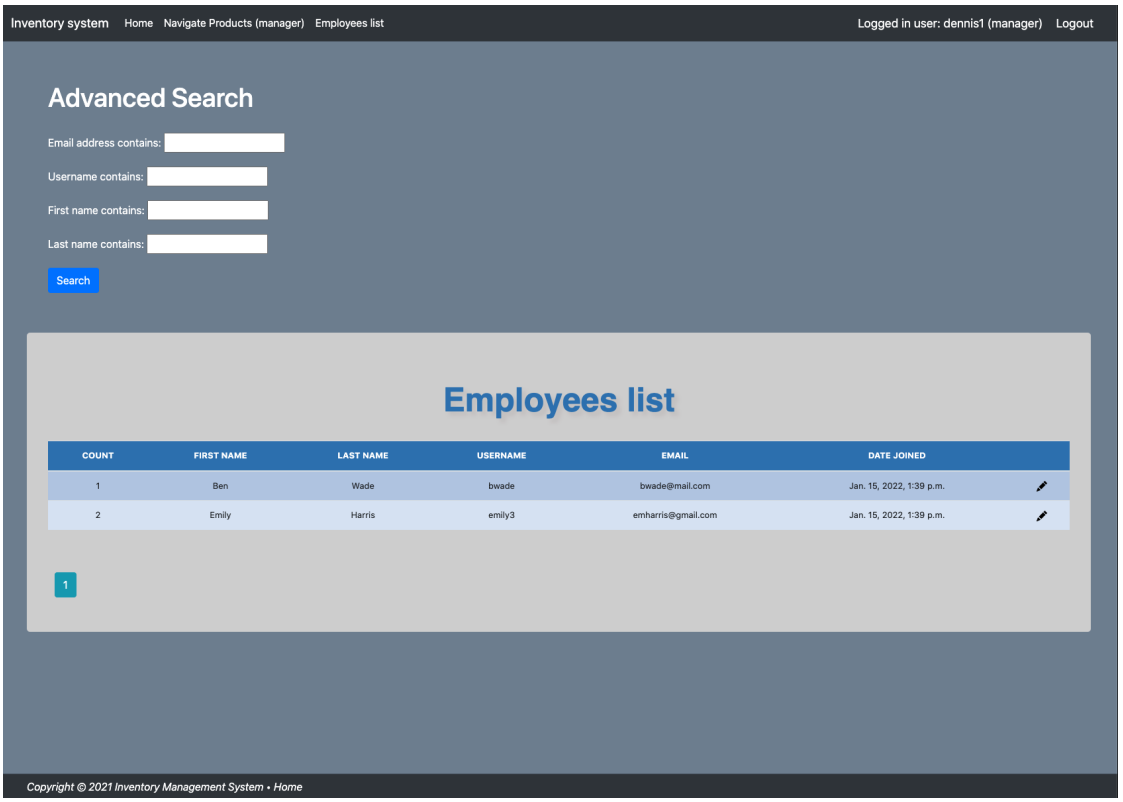
Σε περίπτωση που ο χρήστης επιλέξει διαγραφή προϊόντος εμφανίζεται σελίδα επιβεβαίωσης διαγραφής όπου καλείται να επιβεβαιώσει τη διαγραφή ή όχι. Στη συνέχεια και εφόσον επιλέξει ναι, εμφανίζεται το αντίστοιχο μήνυμα επιτυχής διαγραφής και οδηγείται στην αρχική σελίδα.



Εικόνα 65 Επιβεβαίωση διαγραφής

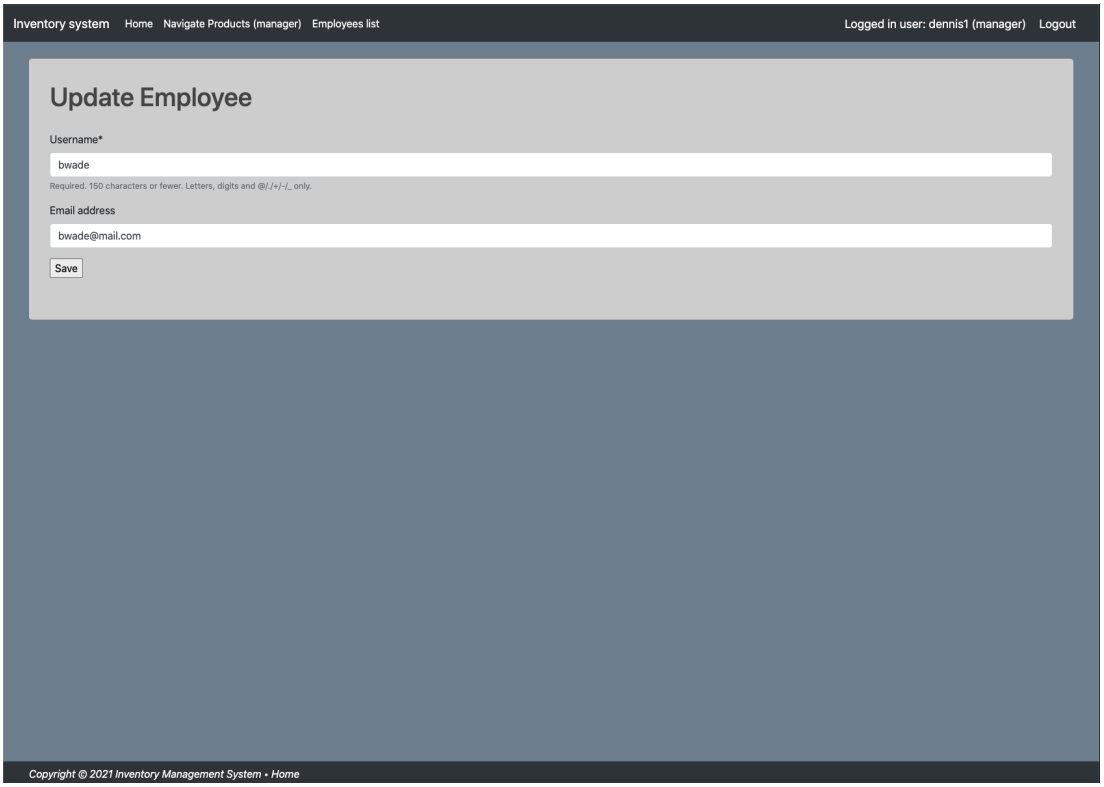
6.2.2 Επιλογή Employees list

Σε περίπτωση που ο manager επιλέξει Employees list, τότε εμφανίζεται σελίδα με τους διαθέσιμους εργαζόμενους στην οποία υπάρχει η δυνατότητα αναζήτησης βάσει κριτηρίων και επεξεργασίας των στοιχείων των εργαζομένων.



Εικόνα 66 Λίστα εργαζομένων

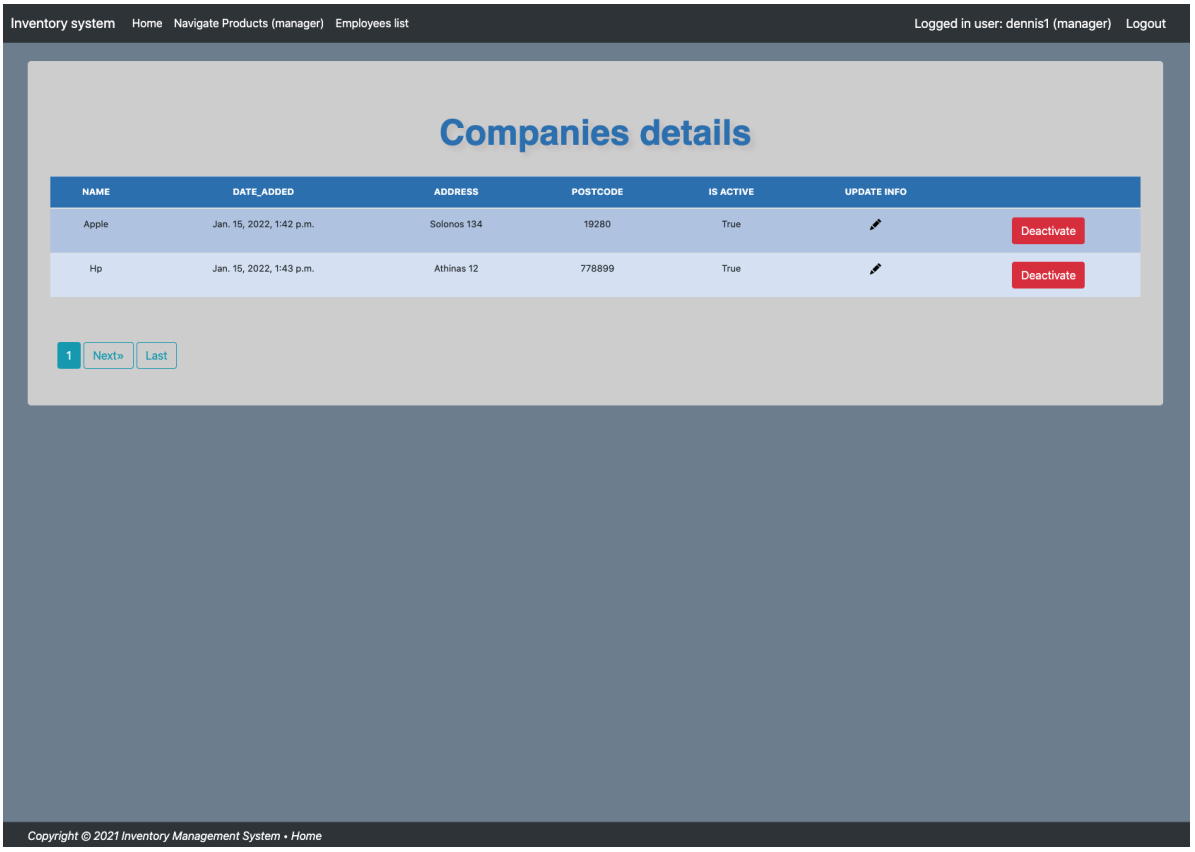
Στη σελίδα που θα εμφανιστεί, ο manager μπορεί να επεξεργαστεί το username και το email του εργαζόμενου και να αποθηκεύσει τις αλλαγές πατώντας save.



Εικόνα 67 Επεξεργασία εργαζομένου

6.2.3 Επιλογή Companies details

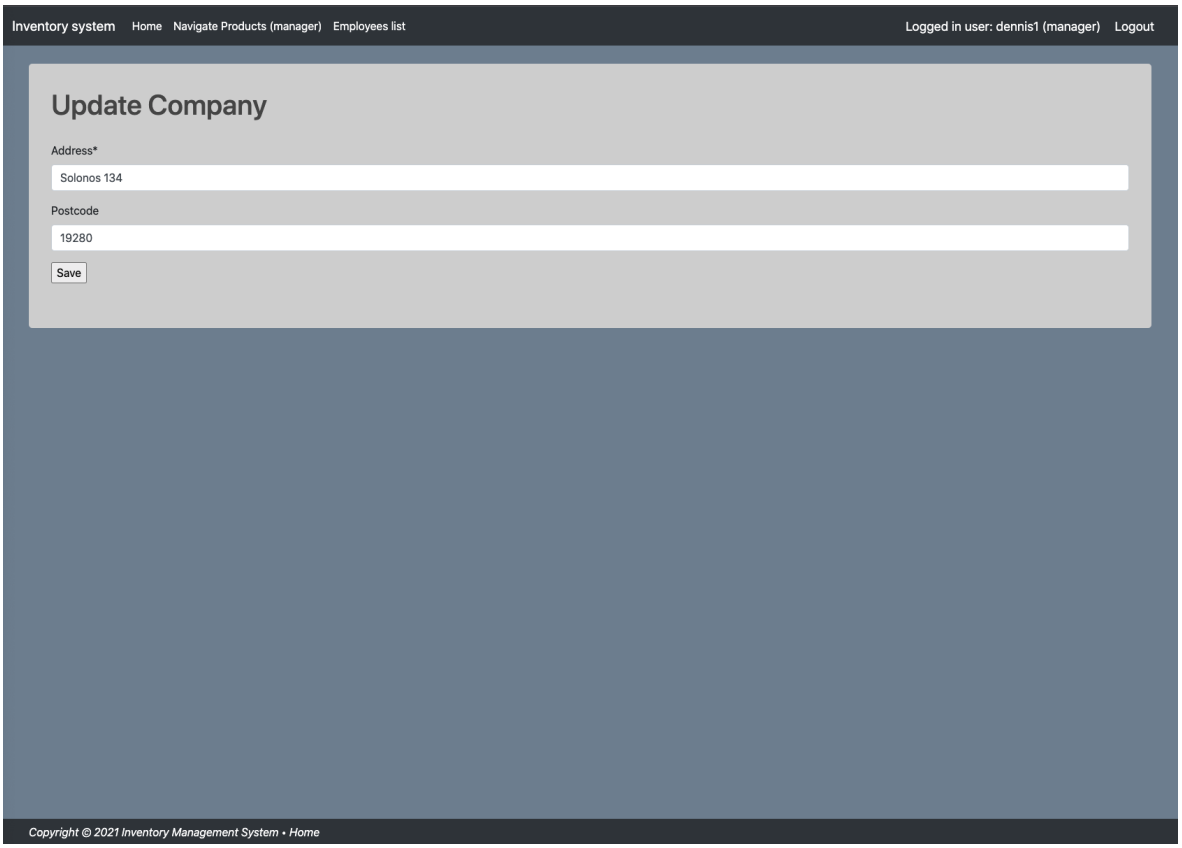
Εφόσον ο manager επιλέξει Companies details τότε εμφανίζεται σελίδα με τις διαθέσιμες εταιρείες και εχει την επιλογή επεξεργασίας των στοιχείων της εταιρείας ή την απενεργοποίηση (deactivate) της εταιρείας.



Εικόνα 68 Λίστα εταιρειών

6.2.3.1 Επεξεργασία στοιχείων εταιρείας

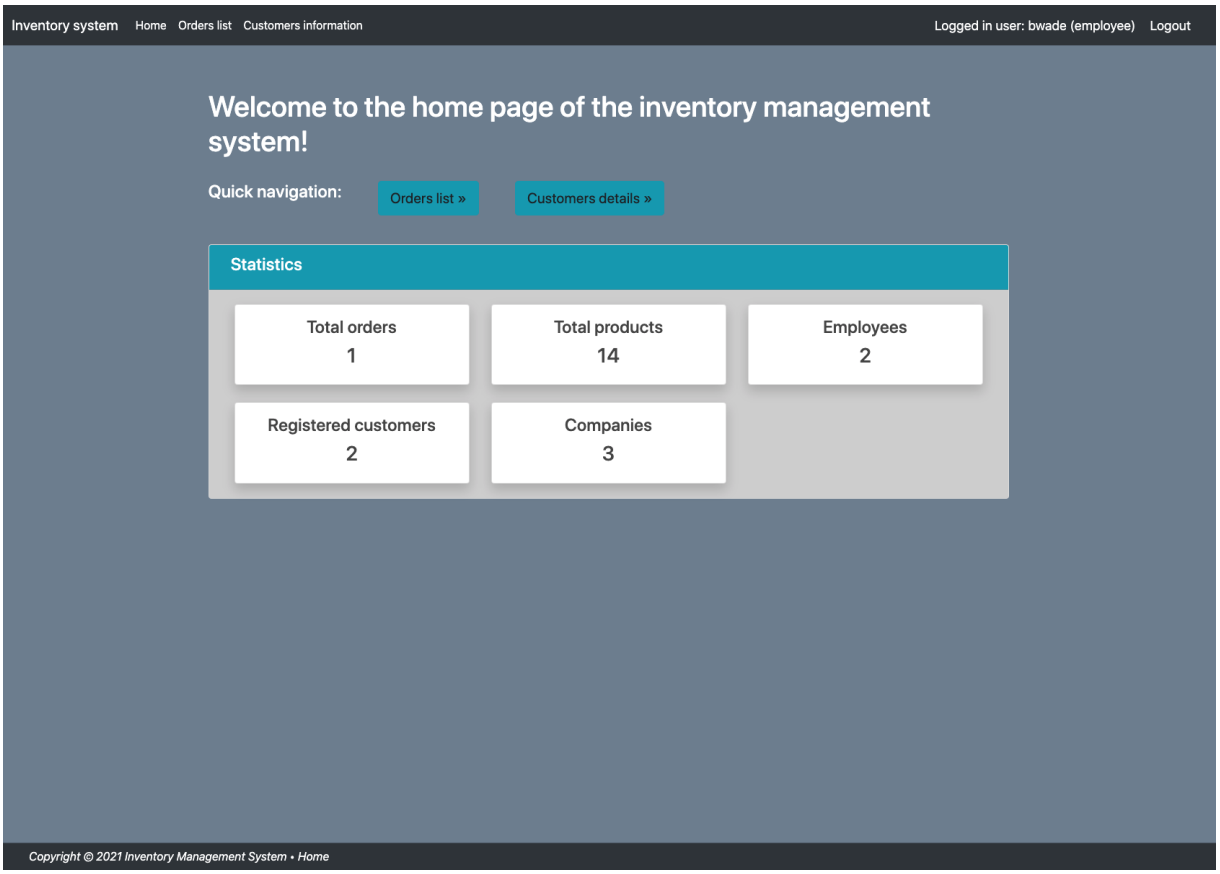
Στην επεξεργασία των στοιχείων της εταιρείας ο manager μπορεί να αλλάξει τη διεύθυνση και τον τ.κ της και να αποθηκεύσει τις αλλαγές.



Εικόνα 69 Επεξεργασία εταιρείας

6.3 Λειτουργίες Employee

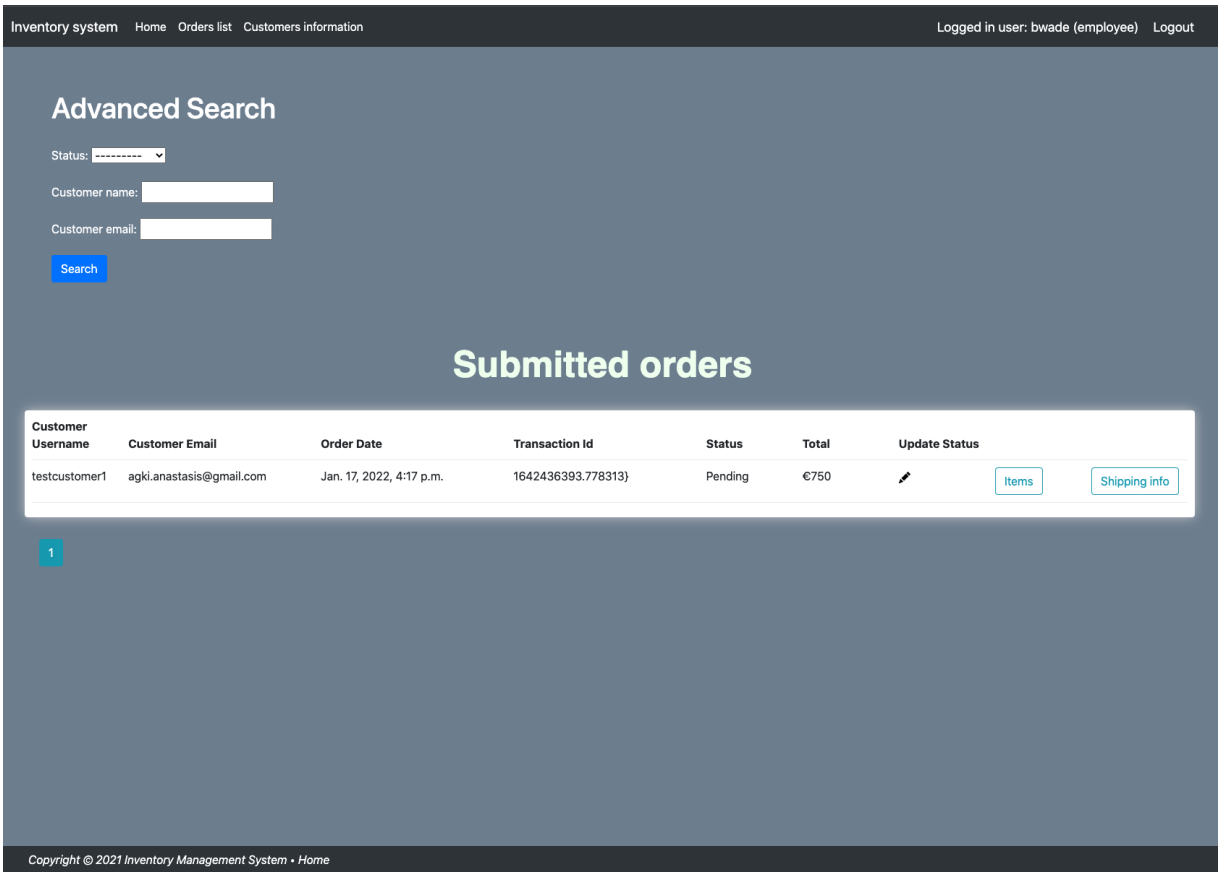
Σε περίπτωση που ο χρήστης συνδεθεί ως employee τότε ανακατευθύνεται στην αρχική σελίδα η οποία περιέχει δύο επιλογές (orders list, customers details) και ένα πίνακα στατιστικών.



Εικόνα 70 Αρχική σελίδα employee

Σε περίπτωση που ο employee επιλέξει Orders list τότε εμφανίζεται σελίδα με τις καταχωρημένες παραγγελίες. Στη σελίδα αυτή υπάρχει η δυνατότητα αναζήτησης παραγγελίας βάσει κριτηρίων, η αλλαγή της κατάστασης της παραγγελίας, η δυνατότητα εμφάνισης των αντικειμένων της παραγγελίας και η δυνατότητα εμφάνισης των πληροφοριών παράδοσης.

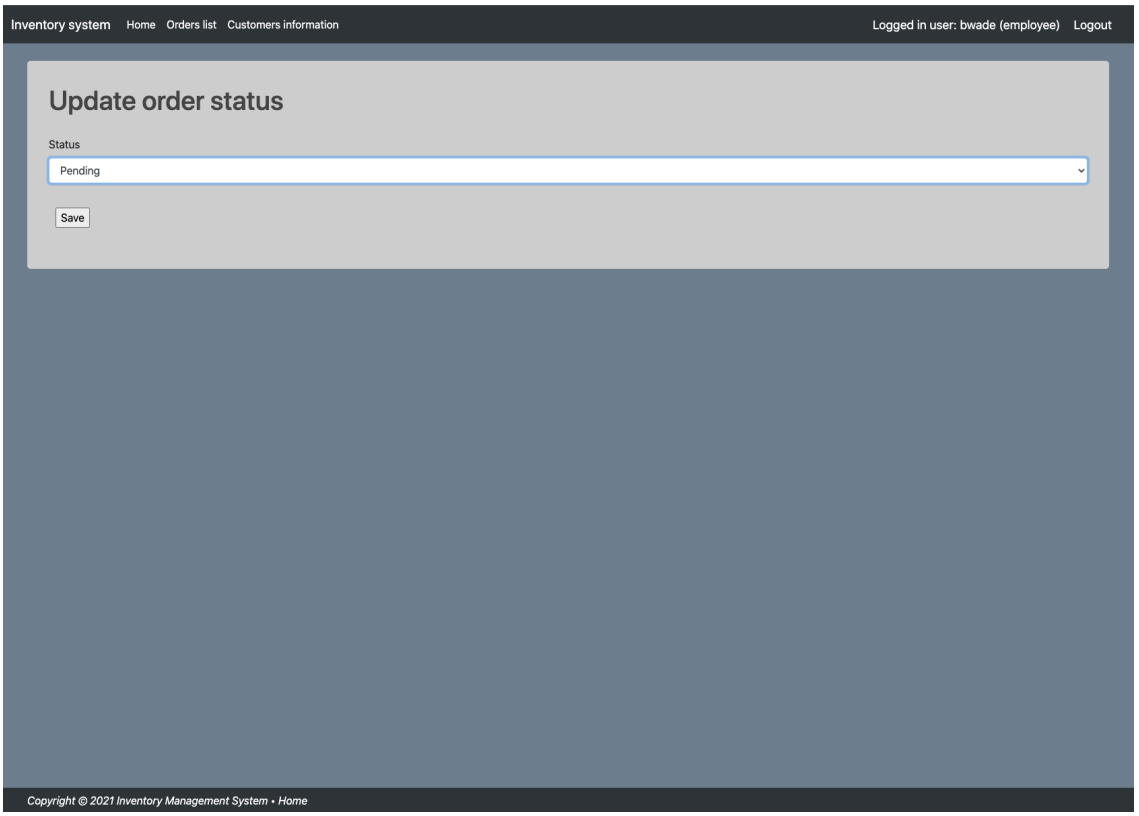
6.3.1 Επιλογή Orders list



Εικόνα 71 Λίστα παραγγελιών

6.3.1.1 Επιλογή Update status

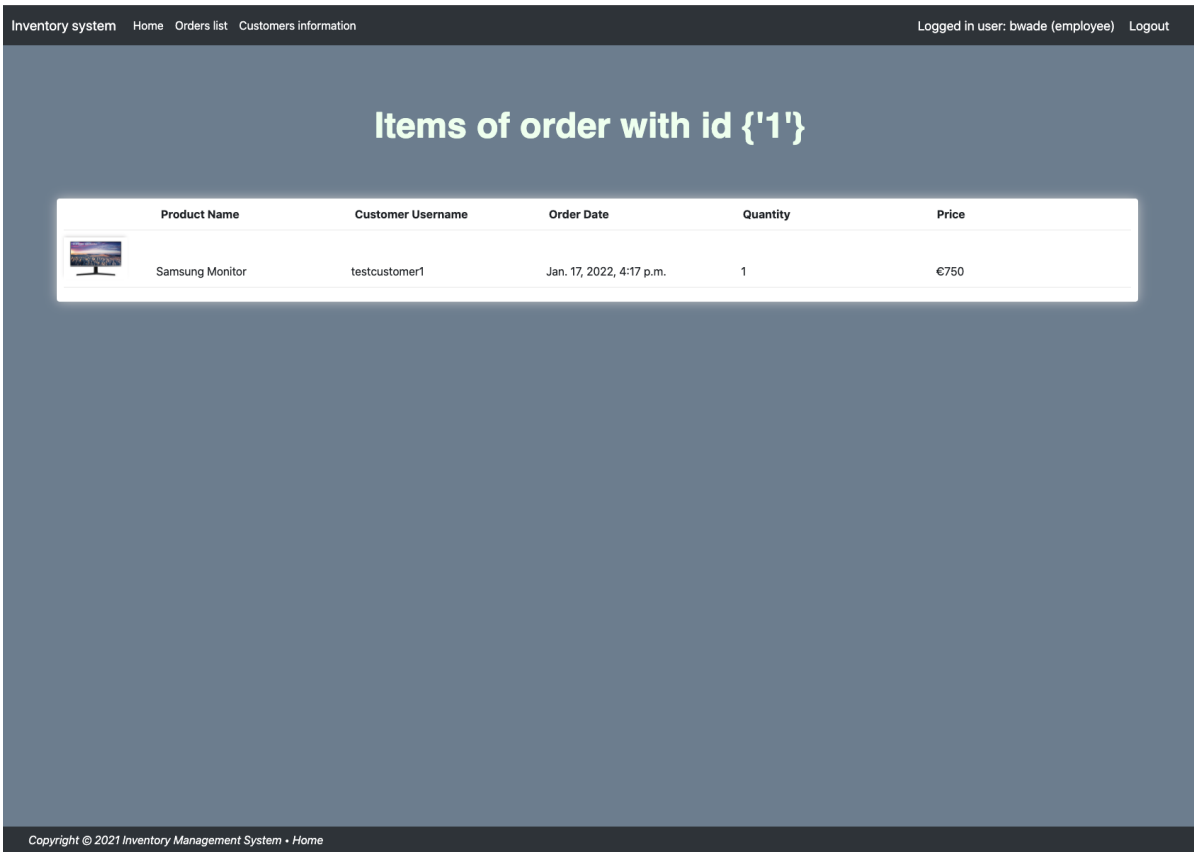
Εφόσον ο employee επιλέξει update status τότε ανακατευθύνεται σε σελίδα όπου μπορεί να αλλάξει την κατάσταση της παραγγελίας. Υπάρχουν πέντε διαθέσιμες επιλογές (Pending, Processing, Approved, Declined, Delivered). Μόλις μεταβληθεί το status τότε στέλνεται αυτόματα email ενημέρωσης στον συγκεκριμένο πελάτη.



Εικόνα 72 Αλλαγή κατάστασης παραγγελίας

6.3.1.2 Επιλογή Items

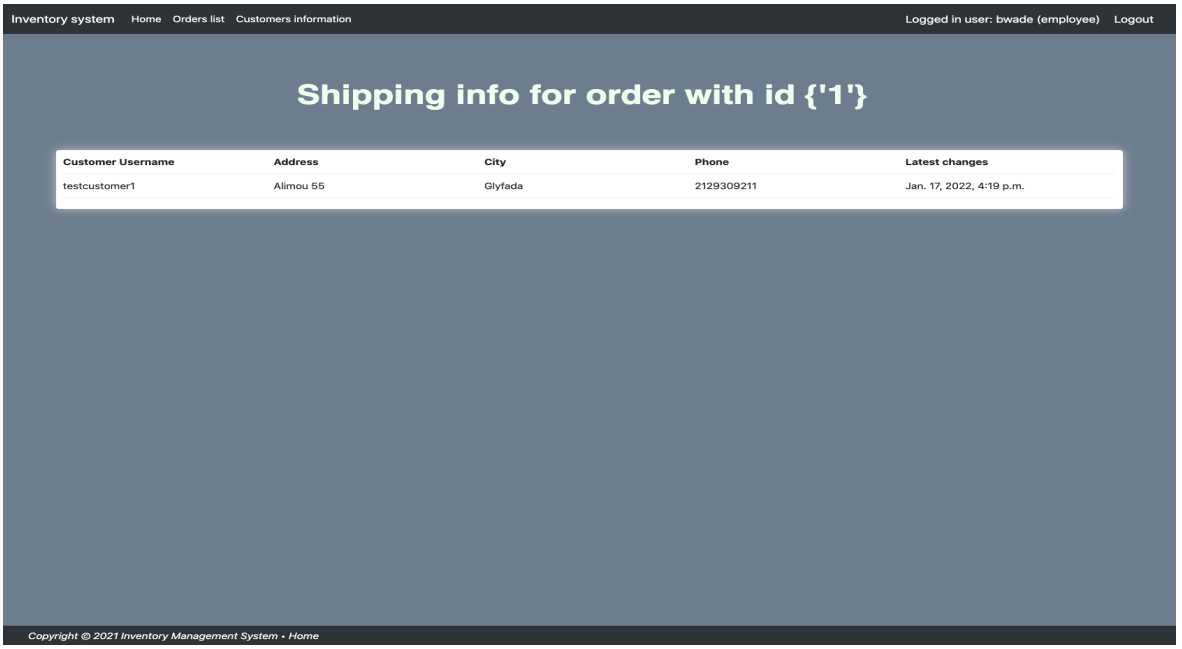
Αν ο συνδεδεμένος employee επιλέξει Items τότε εμφανίζεται η λίστα με τα προϊόντα που περιέχονται στη συγκεκριμένη παραγγελία.



Εικόνα 73 Προϊόντα παραγγελίας

6.3.1.3 Επιλογή Shipping info

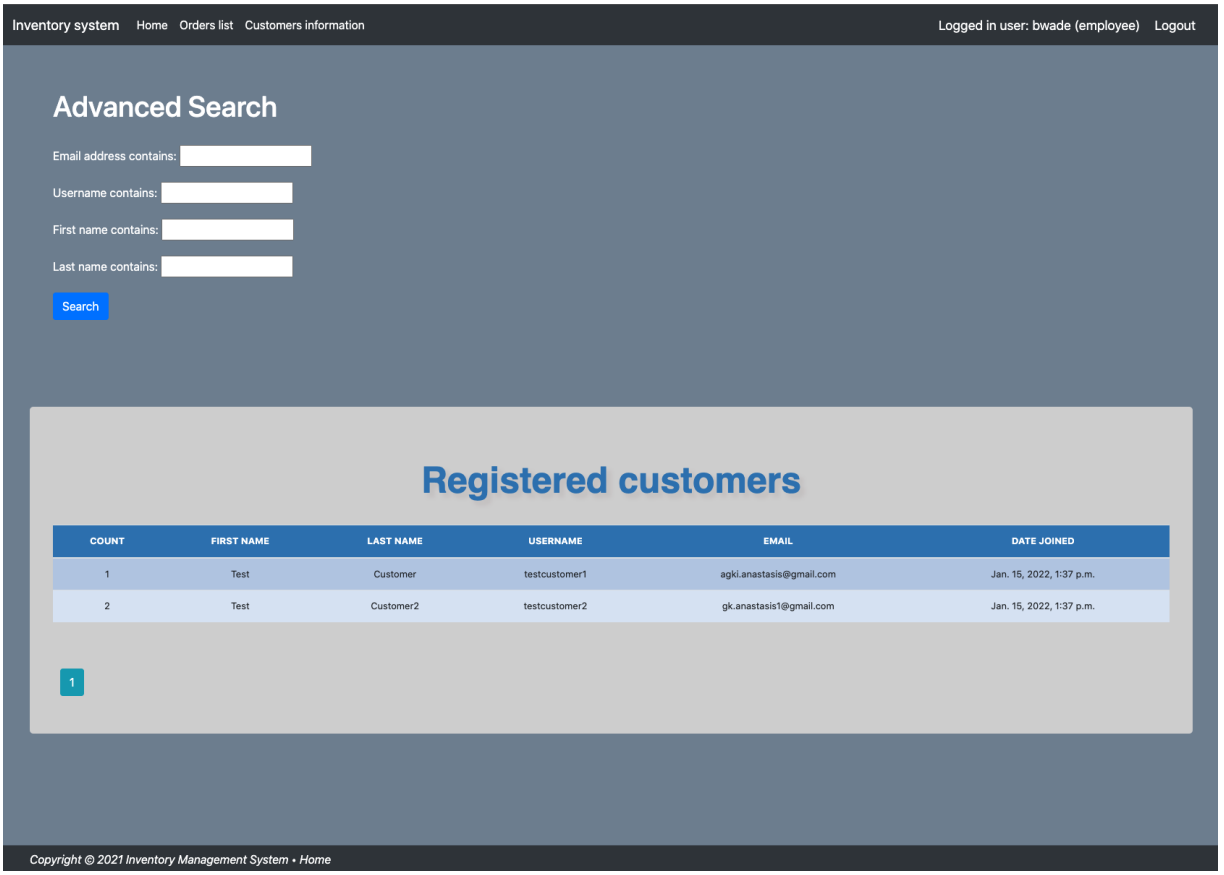
Εφόσον ο συνδεδεμένος employee επιλέξει shipping info σε μια παραγγελία τότε ανακατευθύνεται σε σελίδα με τις πληροφορίες παράδοσης.



Εικόνα 74 Πληροφορίες παράδοσης

6.3.2 Επιλογή Customers information

Τέλος, ο employee έχει την επιλογή να δει τη λίστα με τους καταχωρημένους πελάτες έχοντας τη δυνατότητα αναζήτησης με κάποιο κριτήριο.

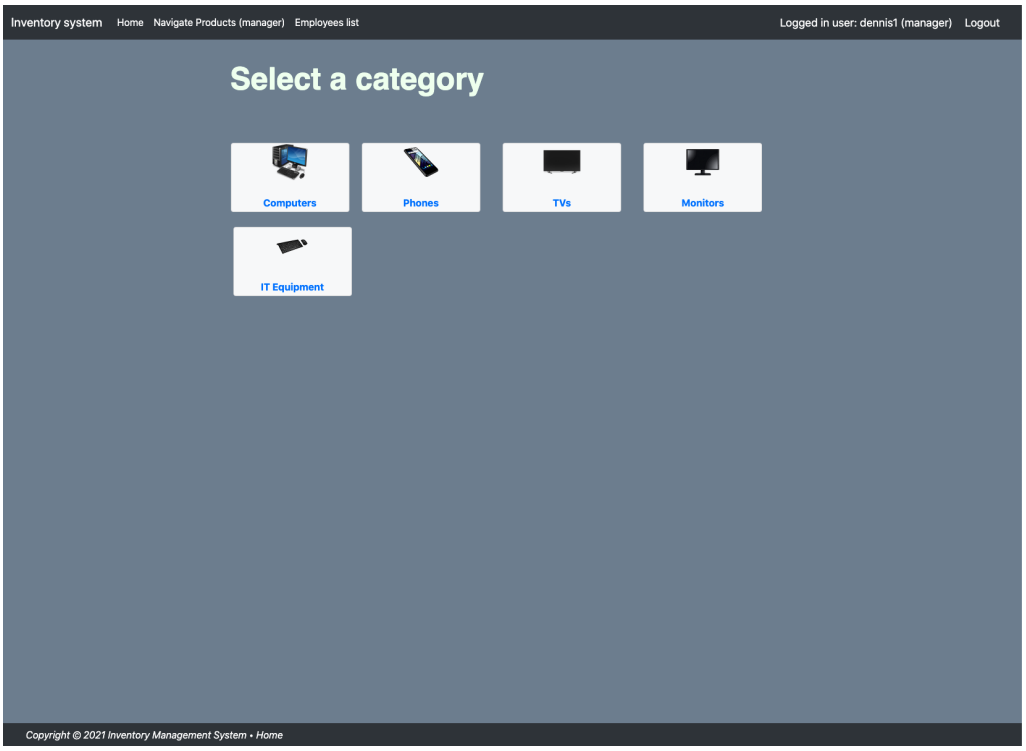


Εικόνα 75 Καταχωρημένοι πελάτες

6.4 Λειτουργίες Customer

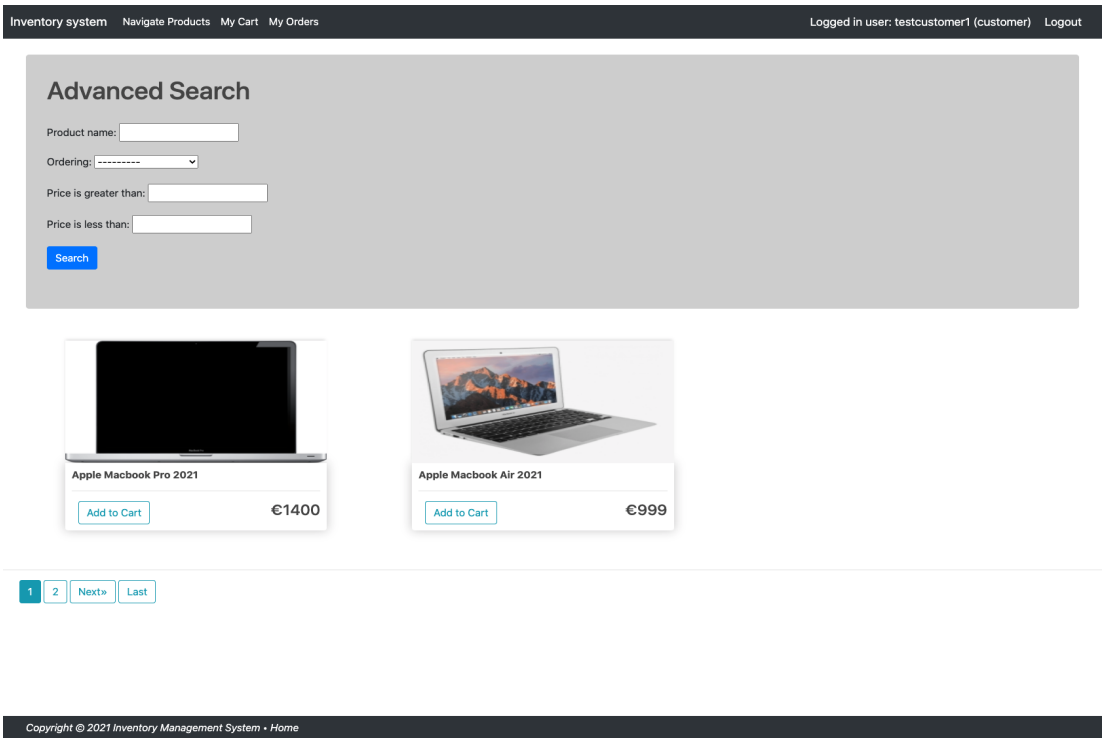
Μόλις κάποιος χρήστης συνδεθεί στην εφαρμογή ως customer τότε ανακατευθύνεται στην αρχική οθόνη έχοντας τρεις επιλογές (Navigate Products, My Cart, My Orders).

6.4.1 Επιλογή Navigate Products



Εικόνα 61 Επιλογή κατηγορίας

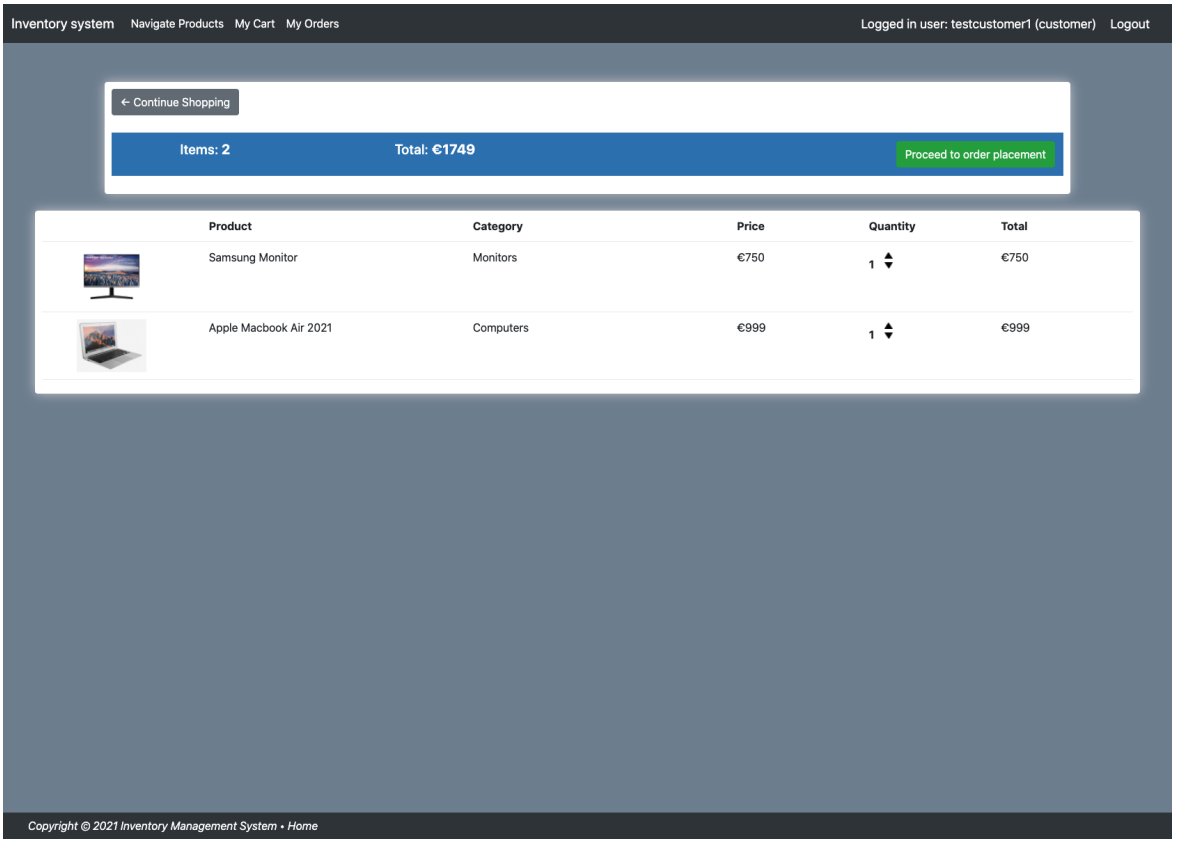
Μόλις γίνει η επιλογή της κατηγορίας εμφανίζεται η λίστα με τα διαθέσιμα προϊόντα. Ο συνδεδεμένος customer μπορεί να προσθέσει κάποιο/α προϊόν στο καλάθι πατώντας Add to cart ώστε αυτό να προστεθεί. Δίνεται η δυνατότητα επίσης να γίνει αναζήτηση κάποιου προϊόντος και ταξινόμηση με βάση την τιμή.



Εικόνα 76 Λίστα προϊόντων customer

6.4.2 Επιλογή My Cart

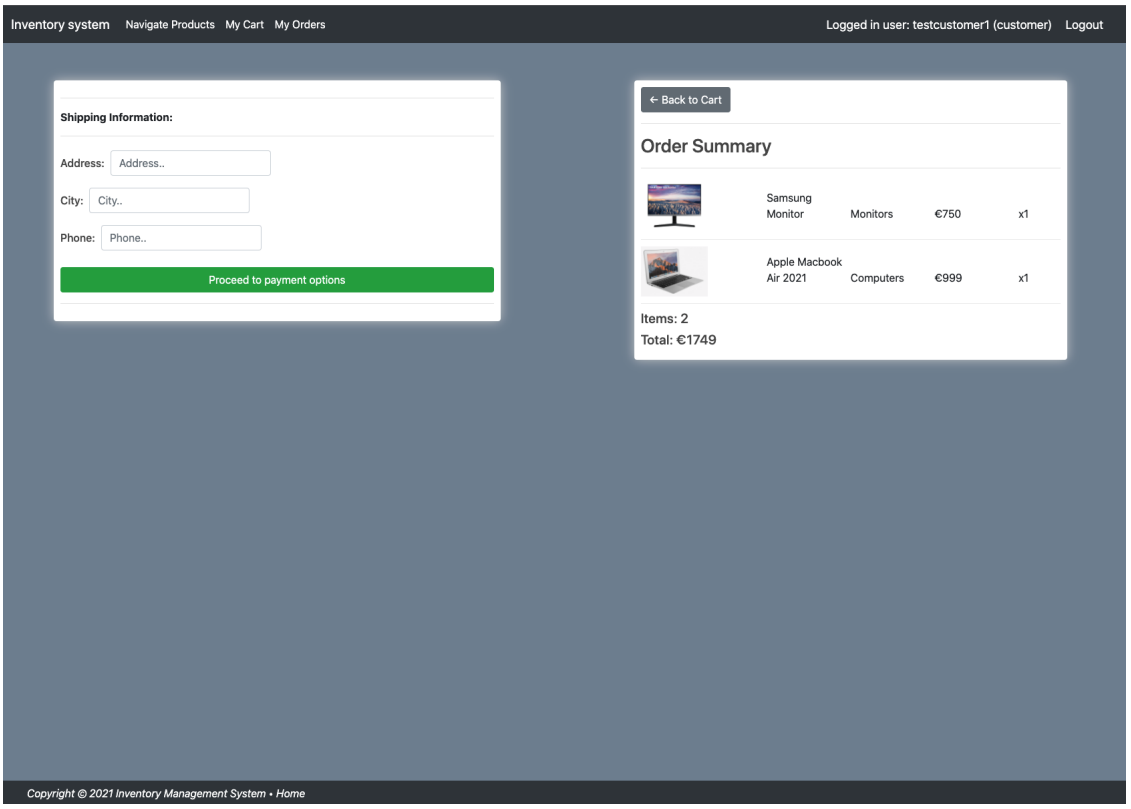
Αφού προστεθεί το προϊόν στο καλάθι πατώντας Add to cart, ο customer μπορεί να επιλέξει My Cart ώστε να δει τα προιοντα που βρίσκονται στο καλάθι. Μέσα από τα βέλη στη στήλη quantity μπορεί να αυξομειώσει τη ποσότητα του προϊόντος.



Εικόνα 77 Περιεχόμενα καλαθιού

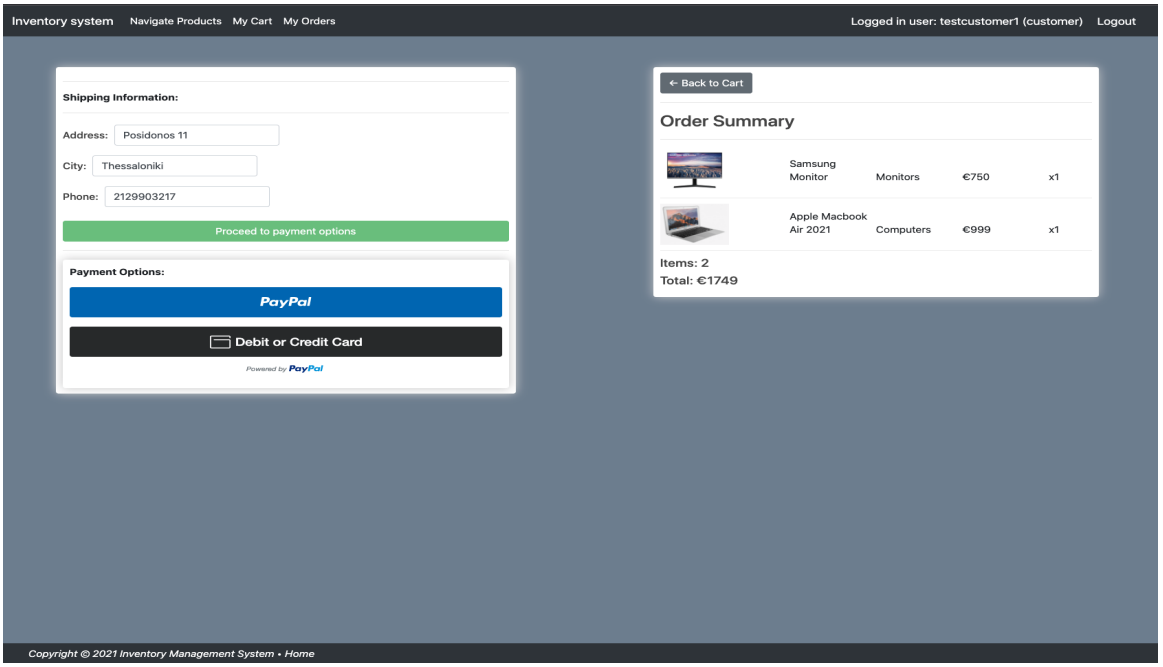
6.4.2.1 Επιλογή Proceed to order placement

Μόλις ο customer επιλέξει proceed to order placement, ανακατευθύνεται σε σελίδα που συμπληρώνονται τα στοιχεία παράδοσης και στα δεξια φαίνεται το σύνολο του καλαθιού.

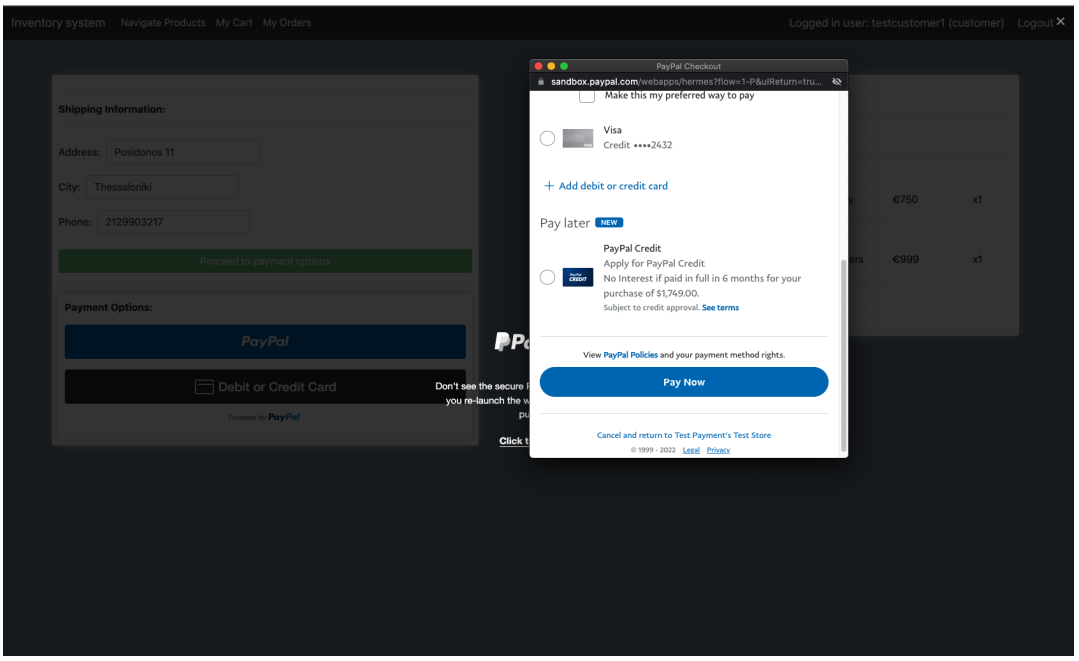


Εικόνα 78 Σύνολο και πληροφορίες παράδοσης

Μόλις συμπληρώσει τα στοιχεία και επιλέξει Proceed to payment options, εμφανίζονται οι επιλογές πληρωμής. Επιλέγοντας έναν από του δύο τρόπους εμφανίζεται ένα pop up παράθυρο (sandbox περιβάλλον) ώστε να γίνει η πληρωμή. Όταν ο customer επιλέξει πληρωμή ανακατευθύνεται αυτόματα στην αρχική σελίδα επιλογής κατηγορίας.



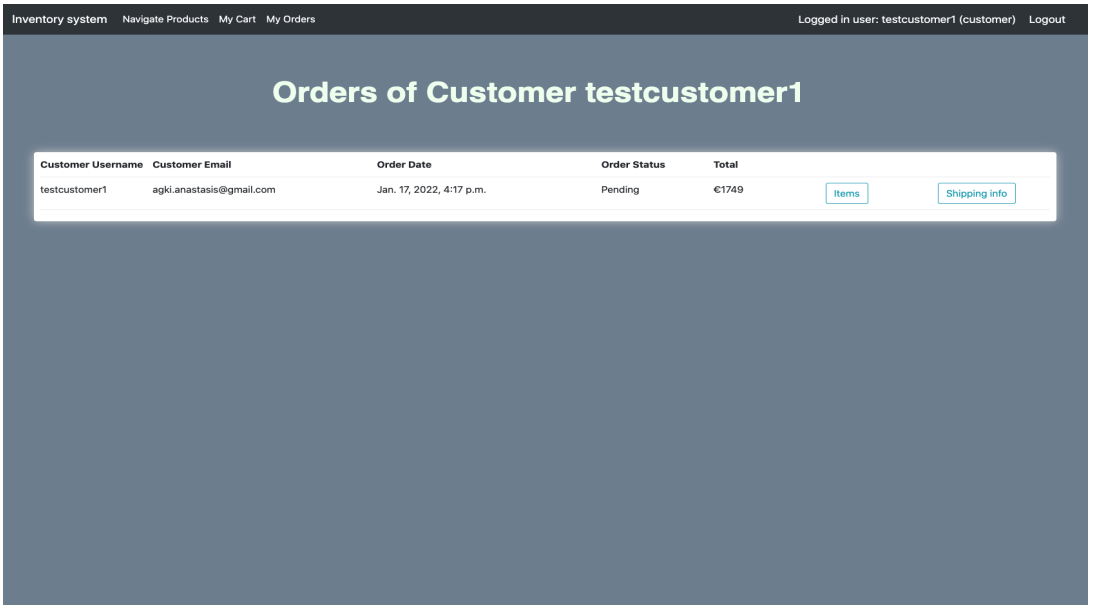
Εικόνα 79 Επιλογή τρόπου πληρωμής



Εικόνα 80 Πληρωμή

6.4.3 Επιλογή My Orders

Επιλέγοντας My Orders, ο customer έχει τη δυνατότητα να δει τις παραγγελίες που έχει καταχωρήσει. Επιπλέον, όπως και στο κεφάλαιο 6.3.1, μπορεί να επιλέξει είτε Items είτε Shipping info ώστε να δει τις αντίστοιχες πληροφορίες. Να σημειωθεί ότι ωσότου μία καταχωρημένη παραγγελία (κατάσταση Pending) αλλάξει status από κάποιον employee, ο customer μπορεί ακόμα να επεξεργάζεται τα περιεχόμενα της παραγγελίας (τροποποιόντας προϊόντα/διεύθυνση χωρίς να καταχωρείται σαν νέα παραγγελία).



Εικόνα 81 Παραγγελίες πελάτη

Κεφάλαιο 7: Συμπεράσματα

7.1 Συμπεράσματα

Μέσα από την εφαρμογή διαχείρισης αποθεμάτων κατέστη δυνατό να κατανοήσουμε ότι μια εφαρμογή δεν περιορίζεται μονάχα στα στενά πλαίσια της ανάπτυξη της.
Οι κύριοι στόχοι που επιτεύχθηκαν στα πλαίσια της παρούσας πτυχιακής εργασίας ήταν:

α) η παρουσίαση μιας σύγχρονης και ραγδαία αναπτυσσόμενης τεχνολογικής τάσης (DevOps) και η κατανόηση του κύκλου ζωής των εφαρμογών
β) η ανάπτυξη μιας πρακτικής εφαρμογής που χρησιμοποιείται στην επαγγελματική ζωή
γ) η εξοικείωση με τεχνολογίες αυτοματοποίησης, συνεχούς ενσωμάτωσης των αλλαγών του κώδικα και προώθησης των αλλαγών σε production περιβάλλον (βλ. Jenkins)
δ) η εξοικείωση με τεχνολογίες διαχείρισης παραμετροποιήσεων (configuration management) και ενορχήστρωσης (orchestration) και η αξιοποίηση τους για εγκατάσταση της εφαρμογής σε production περιβάλλον (βλ. Ansible)
ε) η εξοικείωση με τεχνολογιες βασισμένες στη λογική των containers και της απομόνωσης της εφαρμογής από την υπόλοιπη υποδομή (βλ. Docker)
στ) η εξοικείωση με τεχνολογιες βασισμένες στη λογική διαχείρισης των containers σε διαφορετικούς κόμβους διασφαλιζοντας την ομαλη λειτουργία της εφαρμογής (βλ. Kubernetes)
ζ) η διαδικασία επιτυχούς ενσωμάτωσης των παραπάνω τεχνολογιών στην εκδοχή της εφαρμογής μας
η) η κατανόηση του τρόπου εκτέλεσης της εφαρμογής σε εικονικές μηχανές αξιοποιώντας τις παραπάνω τεχνολογίες
θ) η διασύνδεση πολλαπλών κόμβων

7.2 Μελλοντικές κατευθύνσεις

Η παρούσα εφαρμογή καλύπτει σε αρκετά ικανοποιητικό βαθμό τις απαιτήσεις των χρηστών αλλά και την ενσωμάτωση των τεχνολογιών βασισμένες στο DevOps που αφορούν τον προγραμματιστή. Ωστόσο, όπως σε όλες τις εφαρμογές έτσι και στην παρούσα, υπάρχουν προοπτικές βελτίωσης και περαιτέρω ανάπτυξης της.

Συγκεκριμένα, μία πρόταση θα ήταν η βελτίωση του user interface με σκοπό ο χρήστης να έχει όσο το δυνατόν πιο ευχάριστη αλληλεπίδραση με το σύστημα. Είναι σημαντικό για κάθε εφαρμογή πέρα από την εμφάνιση, να προσφέρει στον χρήστη ευκολία στη χρήση και όχι μεγάλη πολυπλοκότητα ώστε να είναι κατανοητή και να προσφέρει μία ευχάριστη εμπειρία.

Επιπλέον, θα μπορούσαν να γίνουν βελτιώσεις όσον αφορά το security enhancement της εφαρμογής. Τα τελευταία χρόνια παρατηρούνται συνεχώς παραβιάσεις ασφαλείας με αποτέλεσμα την υποκλοπή σημαντικών δεδομένων, για αυτό η ασφάλεια αποτελεί πλέον ίσως το σημαντικότερο μέλημα για τις επιχειρήσεις και τους προγραμματιστές που σχεδιάζουν/υλοποιούν τις εφαρμογές. Το security enhancement θα μπορούσε να αφορά και την ενίσχυση της εφαρμογής αυτής καθαυτής αλλά και το κομμάτι της εκτέλεσης της σε διαφορετικά περιβάλλοντα, ενισχύοντας δηλαδή την ασφάλεια της σύνδεσης μεταξύ των εικονικών μηχανών και των κόμβων.

Τέλος, δεν θα μπορούσε να παραληφθεί ενδεχόμενη επέκταση που να αφορά το κομμάτι του (όσο το δυνατόν) καλύτερου testing της εφαρμογής. Προκειμένου μία εφαρμογή να προωθηθεί σε περιβάλλον production, είναι βαρύνουσας σημασίας να έχουν προβλεφθεί και υλοποιηθεί επαρκή σενάρια που θα αφορούν τη δοκιμή και τη συμπεριφορά της προτού παραδοθεί στον πελάτη για κανονική χρήση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] : https://en.wikipedia.org/wiki/Main_Page

[2] : https://tutorial.djangogirls.org/en/

[3] : https://docs.ansible.com/

[4] : https://www.jenkins.io/doc/

[5] : https://docs.docker.com/

[6] : https://docker-curriculum.com/

[7] : https://kubernetes.io/docs/home/

[8] : https://stackoverflow.com/

[9] : https://docs.djangoproject.com/en/4.0/

[10] : https://eclass.hua.gr/

[11] : https://github.com/

[12] : https://azure.microsoft.com/en-us/

[professors-info] : https://dit.hua.gr/index.php/el/

[IaS] : https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

[SSH] : https://en.wikipedia.org/wiki/Secure_Shell

[SVC] : https://kuberneters.io/docs/concepets/services-networking/service/

ΒΟΗΘΗΤΙΚΟΣ ΚΩΔΙΚΑΣ

- https://itsourcecode.com/free-projects/python-projects/inventory-management-system-project-using-django-with-source-code/
- https://simpleisbetterthancomplex.com/tutorial/2018/01/18/how-to-implement-multiple-user-types-with-django.html
- https://arbcoms.com/store-management-system/?_page=3&_ga=2.187384296.1921521588.1634563086-1673047771.1634301705

ΑΠΟΘΕΤΗΡΙΑ ΚΩΔΙΚΑ ΕΡΓΑΣΙΑΣ

Βασική εφαρμογή: https://github.com/stas33/inventory-management-project

Ansible project: https://github.com/stas33/ansible-inventory-management-system