ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Ανάπτυξη Υπηρεσίας Διαχείρισης Ανθρωπίνου Δυναμικού Μικρομεσαίων Επιχειρήσεων Πληροφορικής.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

---

του

**ΑΝΑΣΤΑΣΟΠΟΥΛΟΥ ΓΕΩΡΓΙΟΥ**

**Επιβλέπων:** Σαρηγιαννίδης Παναγιώτης,

Αναπληρωτής Καθηγητής

ΚΟΖΑΝΗ/ΜΑΡΤΙΟΣ/2024

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA

FUCULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING

# Development Of a Human Resource Management System for Small and Medium Software Enterprises

THESIS

**GEORGIOS ANASTASOPOULOS**

**SUPERVISOR:** Sarigiannidis Panagiotis

Associate Professor

KOZANI/MARCH/2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

## ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο “Ανάπτυξη Υπηρεσίας Διαχείρισης Ανθρωπίνου Δυναμικού Μικρομεσαίων Επιχειρήσεων Πληροφορικής.” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Παναγιώτη Σαρηγιαννίδη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.



Υπογραφή Φοιτητή: _____

# *Περίληψη*

Η ανάπτυξη λογισμικού είναι μια πολύπλοκη διαδικασία που απαιτεί τη συνεργασία πολλών διαφορετικών ειδικοτήτων. Προκειμένου οι εταιρείες να παράγουν λογισμικό υψηλής ποιότητας θα πρέπει να οργανώσουν τα στάδια της αναπτυξης με τέτοιο τρόπο, έτσι ώστε το έργο να είναι διαχειρίσιμο, βιώσιμο και με γνώμονα να εξυπηρετεί τελικά τις ανάγκες του πελάτη. Τα στάδια από τα οποία περνάει ένα λογισμικό, από τη σύλληψη της ιδέας μέχρι και τη διάθεση του τελικού προϊόντος, είναι πάρα πολλά, γεγονός που καθιστά, πολύ συχνά τα λάθη κατά τη διάρκεια της ανάπτυξης από τις εμπλεκόμενες ομάδες. Ο σκοπός της παρούσας διπλωματικής είναι να παρουσιάσει το SDLC και τις πτυχές του και να αναδείξει τα οφέλη που έχει ιδίως στις μικρομεσαίες επιχειρήσεις μέσω των μοντέλων που περιλαμβάνει. Συνοπτικά, ο κύκλος ζωής ανάπτυξης λογισμικού (SDLC) είναι μια μεθοδολογία που χρησιμοποιείται για να οργανώσει και να διαχειριστεί την ανάπτυξη λογισμικού καθώς την διαιρεί σε διάφορες φάσεις, καθεμία από τις οποίες έχει τους δικούς της στόχους και εργασίες. Προκειμένου να ολοκληρωθεί η ανάλυση του SDLC, τη διπλωματική αυτή συνοδεύει και μια εφαρμογή όπου αναπτύχθηκε με δημοφιλείς στο χώρο τεχνολογίες. Συγκεκριμένα επιλέχθηκε η χρήση της React.js για το front end της εφαρμογής και του Spring Boot με χρήση Restful Apis για το back end, λόγω της ευελιξίας που παρέχουν και της ταχύτητας αναπτυξης ποιοτικού κώδικα. Η εφαρμογή έχει σκοπό να προσφέρει ένα απλό και κατανοητό περιβάλλον μιας πλατφόρμας διαχείρισης έργων και προσωπικού που να ικανοποιεί τις ανάγκες μιας μικρής επιχείρησης, αναδεικνύοντας πως ακόμα και σε πρώιμο στάδιο, μια τέτοια εφαρμογή μόνο θετικά αποτελέσματα μπορεί να επιφέρει.

## Λέξεις Κλειδιά:

SDLC

Waterfall Model

Agile Model

Spiral Model

V-Shaped Model

React.js

Spring Boot

# *Abstract*

*Software development is a complex process that requires the collaboration of many different disciplines. In order for companies to produce high-quality software, they must organize the stages of development very well, so that the work is manageable, sustainable and ultimately serves the needs of the customer. The stages through which a software goes, from the conception of the idea to the release of the final product, are so many, a fact that makes the existence of mistakes more frequent during the development period by the project teams. The purpose of this thesis is to present the Software Development Life Cycle (SDLC) and its aspects and to highlight the benefits it has especially for small and medium enterprises through the models it includes. Briefly, the software development life cycle (SDLC) is a methodology used to organize and manage software development as it divides it into various phases, each with its own goals and tasks. In order to complete the analysis of the SDLC, this thesis is accompanied by an application where it was developed with popular technologies in the field. In particular, the use of React.js was chosen for the front end of the application and Spring Boot with the use of Rest Apis for the back end, due to the flexibility they provide and the speed of developing quality code. The application aims to offer a simple and understandable environment of a project management platform that satisfies the needs of a small business, showing that even at its early stage, such an application can only bring positive results.*

## Keywords:

SDLC

Waterfall Model

Agile Model

Spiral Model

V-Shaped Model

React.js

Spring Boot

# *Ευχαριστίες*

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Παναγιώτη Σαρηγιαννίδη που μου πρόσφερε την ευκαιρία να ασχοληθώ με ένα αντικείμενο που έχει έντονο αντίκτυπο στη σημερινές απαιτήσεις της αγοράς, τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο. Σημαντικές ήταν επίσης και οι συμβουλές του Πάρη Καρυπίδη και η καθοδήγηση που είχα καθόλη την διάρκεια της συγγραφής, όπως επίσης και των κοντινών μου συμφοιτητών και φίλων, για τις συμβουλές τους στο πρακτικό μέρος της διπλωματικής. Τέλος, θα ήθελα να ευχαριστήσω κάθε έναν ξεχωριστά από το συγγενικό μου περιβάλλον και ιδίως τους γονείς μου για κάθε οικονομική και ψυχολογική υποστήριξη τόσο στη διάρκεια εκπόνησης της διπλωματικής μου εργασίας όσο και κατά τη διάρκεια όλων των ετών φοίτησης μου.

# *Περιεχόμενα*

# *Κατάλογος Σχημάτων*

# *Κατάλογος Εικόνων*

# *Κατάλογος Πινάκων*

# *Πρόλογος*

Η συγκεκριμένη διπλωματική εργασία επικεντρώνεται στην ανάλυση και κατανόηση όλων αυτών των φάσεων από τις οποίες περνάει ένα λογισμικό για να αναπτυχθεί και πως το SDLC μπορεί να οργανώσει αυτές τις φάσεις μέσω μεθοδολογιων από τις πολλές που περιλαμβάνει. Καθώς οι απαιτήσεις και οι ανάγκες στο χώρο του software συνεχώς αλλάζουν, το SDLC περιέχει τόσες λύσεις και μεθοδολογίες, ώστε να μπορεί να εφαρμοστεί σε κάθε project, ασχέτως απαιτήσεων. Από έργα με υψηλούς κινδύνους και με έμφαση στη δοκιμή, μέχρι έργα με χαμηλό προϋπολογισμό ή μεταβαλλόμενες απαιτήσεις, το SDLC μπορεί να εφαρμόσει ένα πλάνο αναπτυξης προσαρμοσμένο στις ανάγκες του εκάστοτε project. Για να μελετηθεί και στην πράξη το SDLC, κατασκευάστηκε και μια εφαρμογή διαχείρισης projects, με σκοπό να εξυπηρετεί τις βασικές ανάγκες μιας μικρομεσαίας επιχείρησης και που διευκολύνει την διαδικασία ανάληψης και αναπτυξης project. Η πλατφόρμα θα αναπτυχθεί χρησιμοποιώντας το Spring Boot και την React. Το Spring Boot είναι ένα Java framework που διευκολύνει την ανάπτυξη οποιασδήποτε εφαρμογής, λόγω του τρόπου που απλοποιεί την διαδικασία συγγραφής κώδικα. Η React είναι μια βιβλιοθήκη της JavaScript που χρησιμοποιείται για την ανάπτυξη δυναμικών εφαρμογών ιστού και προσφέρει οργάνωση και επαναχρησιμοποίηση στον κώδικα.

# *Κεφάλαιο 1: Εισαγωγή*

Σε αυτό το κεφάλαιο γίνεται μία σύντομη εισαγωγή του θέματος της διπλωματικής εργασίας που εκπονήθηκε, καθώς επίσης και μια αναφορά στο σκοπό της.

## 1.1 Αντικείμενο και Σκοπός της Διπλωματικής

Μέσω της παρούσας εργασίας θα παρουσιαστεί λεπτομερώς το SDLC και πως διαχειρίζεται όλα τα στάδια της αναπτυξης λογισμικού, με τις διάφορες μεθοδολογίες του. Θα αναλυθούν οι γνωστότερες μεθοδολογίες, τα πλεονέκτημα και τα μειονεκτήματα τους, προκειμένου να γίνει κατανοητό ποτέ ενδείκνυται η χρήση της κάθε μεθοδολογίας. Θα γίνει αναφορά στις φάσεις του SDLC και τι εκτελείται σε κάθε μια από αυτές, ούτως ώστε να κατανοηθεί η σημασία τήρησης των διαδικασιών και των χρονοδιαγραμμάτων.

Με βάση τα όσα θα αναλυθούν θα καταστεί σαφές πλέον το θετικό αντίκτυπο που έχει για τα έργα αλλά και για τις επιχειρήσεις, η τήρηση της διαδικασίας αναπτυξης με βάση το SDLC. Με την ανάλυση των μεθόδων, θα αποδειχτεί η προσαρμοστικότητα που προσφέρει, έτσι ώστε να μπορεί να δομήσει οποιοδήποτε project και οποιαδήποτε απαίτηση του πελάτη.

Η κατασκευή της εφαρμογής επιδιώκει την πραγματική δοκιμή και υιοθέτηση του SDLC στην πράξη, προκειμένου να διαπιστωθούν τα οφέλη από πρώτο χέρι. Επίσης η επιλογή δημοφιλών στον χώρο τεχνολογιών, δίνει μεγαλύτερη βαρύτητα στην εφαρμογή, καθώς μέσω της έρευνας και της εμβάθυνσης που έγινε σε αυτές τις τεχνολογίες, η εφαρμογή πληροί τα σημερινά δεδομένα της αγοράς στον τομέα της ανάπτυξης λογισμικού. Με λίγα λόγια, η εφαρμογή επιδιώκει να παρουσιάσει σε ένα χρήστη η σε μια επιχείρηση, πως μέσω ενός απλουστευμένου περιβάλλοντος,

μπορεί να απλοποιηθεί η διαδικασία ανάπτυξης και να οργανώσουν οι ίδιοι τις φάσεις και τους υπάλληλους τους, με βάση πάντα το SDLC.

## 1.2 Οργάνωση του τόμου

Το παρόν κεφάλαιο αποτελεί την εισαγωγή της διπλωματικής εργασίας, στο οποίο παρουσιάστηκε το θέμα που θα ερευνηθεί και οι στόχοι που έχουν τεθεί.

Στη συνέχεια στο κεφάλαιο 2, θα αναλυθεί το θεωρητικό υπόβαθρο του έργου, παρουσιάζοντας το SDLC και εξετάζοντας διάφορες μεθοδολογίες που περιλαμβάνει.

Συνεχίζοντας, το κεφάλαιο 3 θα εστιάσει στον σχεδιασμό και την ανάπτυξη του συστήματος, αναλύοντας τις απαιτήσεις και τις λειτουργίες του συστήματος, τα σενάρια χρήσης, το μέσο αποθήκευσης και γενικότερα τις τεχνολογίες που χρησιμοποιήθηκαν.

Στο κεφάλαιο 4, θα γίνει περιγραφή του σχεδιασμού της γραφικής διεπαφής του χρήστη (UI), θα αναλυθεί η εμπειρία χρήσης (UX) και θα παρουσιαστούν στιγμιότυπα της εφαρμογής και του τρόπου λειτουργίας της.

Στο κεφάλαιο 5, θα παρουσιαστούν τα συμπεράσματα και οι μελλοντικές επεκτάσεις της εργασίας. Θα γίνει ανασκόπηση των στόχων που έχουν επιτευχθεί και θα αξιολογηθούν τα αποτελέσματα. Επίσης, θα γίνει αναφορά στις περαιτέρω βελτιώσεις και δυνατότητες ανάπτυξης για το μέλλον.

Τέλος, στο τελευταίο κεφάλαιο, θα γίνει η αναφορά στην βιβλιογραφία που χρησιμοποιήθηκε για την έρευνα και τον σχεδιασμό της εφαρμογής.

# Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

Το κεφάλαιο 2 παρουσιάζει το θεωρητικό υπόβαθρο της ανάπτυξης λογισμικού και εφαρμογών. Ξεκινάει εξηγώντας το Software Development Life Cycle (SDLC) και επισημαίνοντας τα οφέλη του, ιδίως από οικονομικής άποψης αλλά και από άποψη ποιότητας στους τομείς εφαρμογής του. Στη συνέχεια, εξετάζεται η λειτουργία του SDLC, οι φάσεις στις οποίες χωρίζεται η διαδικασία αναπτυξης, τους ρόλους που παίρνουν τα μέλη σε αυτές τις φάσεις, καθώς και τα διάφορα μοντέλα υλοποίησης του SDLC αναλόγως εφαρμογής. Οι πληροφορίες για το γενικό θεωρητικό υπόβαθρο του Software Development Life Cycle (SDLC) αντλούνται κατά βάση από τις αναφορές [1], [2].

Το κεφάλαιο 2 έχει σαν στόχο να παρέχει μια συνοπτική εξήγηση του όρου SDLC στον αναγνώστη και να κάνει ξεκάθαρες όλες τις απαιτήσεις που θα μπορούσε να έχει μια εφαρμογή, η οποία θα προσομοίωνε τη διαδικασία αυτή. Τα κύρια θέματα που θα εξεταστούν στο παρόν κεφάλαιο προσφέρουν την απαραίτητη βάση για την κατανόηση και ανάλυση της ανάπτυξης ενός λογισμικού, αλλά είναι και μια εισαγωγή για την ευκολότερη κατανόηση του κεφαλαίου 3, που αναφέρεται σε πιο τεχνικά θέματα, σχετικά με την υλοποίηση της εφαρμογής που συνοδεύει τη διπλωματική.

## 2.1 Software Development Life Cycle (SDLC)

Το "SDLC" ορίζεται ως ο κύκλος ζωής ανάπτυξης λογισμικού (Software Development Life Cycle) και αναφέρεται σε μια μεθοδολογία που περιγράφει τη διαδικασία αναπτυξης λογισμικού σε φάσεις. Το SDLC περιγράφει με βήματα, τις φάσεις και τις διαδικασίες, από την αρχική ιδέα μέχρι την παράδοση του προϊόντος στον τελικό χρήστη, η ακόμα και μετά την παράδοση, στη συντήρηση του.

Τα βήματα του SDLC μπορούν να ποικίλουν, αλλά συνήθως περιλαμβάνουν τα εξής στάδια [3][4] :

1. **Ανάλυση Απαιτήσεων και Σχεδιασμού (Requirements Analysis & Planning):** Καθορισμός στόχων, αναγνώριση απαιτήσεων σύμφωνα με τις ανάγκες του πελάτη και καθορισμός αρχιτεκτονικής και τεχνικών στοιχείων.

2. **Σχεδίαση (Design):** Καθορισμός πρότυπων σχεδίων και πλάνων για την σχεδίαση της γραφικής διεπαφής και των λειτουργιών του λογισμικού

3. **Ανάπτυξη (Development):** Κωδικοποίηση και υλοποίηση του λογισμικού σύμφωνα με τις προδιαγραφές.

4. **Δοκιμές (Testing):** Έλεγχος και επαλήθευση της λειτουργίας του λογισμικού, επιδιόρθωση σφαλμάτων.

5. **Εφαρμογή (Deployment):** Εγκατάσταση και εκτέλεση του λογισμικού σε παραγωγικό περιβάλλον.

6. **Συντήρηση (Maintenance):** Ενημέρωση και προσθήκη νέων χαρακτηριστικών, συμβατότητας.

## Σημασία και οφέλη του SDLC στην ανάπτυξη λογισμικού

Με την επιτυχημένη εκτέλεση της μεθοδολογίας του SDLC επιτυγχάνεται αποτελεσματικότερα η ανάπτυξη του λογισμικού συνολικά καθώς πέραν της υψηλότερης ποιότητας του τελικού προϊόντος, διασφαλίζεται μειωμένο ρίσκο [3] και πλήρης ικανοποίηση των πελατών. Χρησιμοποιώντας το SDLC, οι ομάδες ανάπτυξης μπορούν να διαχειριστούν αποτελεσματικά το έργο τους, να εντοπίζουν προβλήματα εγκαίρως και να λαμβάνουν τα απαραίτητα μέτρα για την αντιμετώπισή τους.

Πιο συγκεκριμένα όμως, τα σημαντικότερα οφέλη είναι τα εξής :

- **Μείωση δαπανών ανάπτυξης:** Ο σχεδιασμός και ο καλός προγραμματισμός που επιτυγχάνεται με το SDLC μειώνουν τις πιθανότητες εμφάνισης δαπανηρών και χρονοβόρων σφαλμάτων και βελτιστοποιούν το συνολικό κόστος.

- **Αύξηση παραγωγικότητας:** Η μεθοδολογία SDLC ενισχύει τη συνεργασία και την επικοινωνία μεταξύ των μελών της ομάδας, αλλά και όλων των ομάδων μεταξύ τους καθόλη την διάρκεια της έργου, βοηθώντας στην αποτελεσματική διαχείριση του χρόνου, των πόρων και των σφαλμάτων.

- **Μείωση χρόνου ανάπτυξης:** Το SDLC βοηθά στην οργάνωση της διαδικασίας ανάπτυξης, μειώνοντας τον χρόνο που απαιτείται για την παραγωγή και την παράδοση του λογισμικού. Η μείωση του χρόνου αναπτυξης δίνει περισσότερο χώρο για δοκιμές και επαλήθευση πριν την διάθεση του.

- **Αύξηση ποιότητας του προϊόντος:** Η συστηματική και δομημένη προσέγγιση του SDLC βελτιώνει την ποιότητα του παραγόμενου λογισμικού, προσφέροντας ένα πιο αξιόπιστο και αποτελεσματικό προϊόν.

- **Αποτελεσματικότερη συντήρηση:** Οι καλά καθορισμένες διαδικασίες και η τεκμηρίωση που προκύπτει από το SDLC διευκολύνουν την αντιμετώπιση αλλαγών αλλά και την προσθήκη βελτιώσεων στο λογισμικό, κάνοντας τη συντήρηση πιο αποτελεσματική και αποδοτική.

## Οργανισμοί και επιχειρήσεις που χρησιμοποιούν το SDLC

Το SDLC χρησιμοποιείται ευρέως από διάφορες επιχειρήσεις και οργανισμούς που επιθυμούν να αναπτύξουν λογισμικό και εφαρμογές υψηλής ποιότητας. Είναι κρίσιμο ιδίως όμως για εταιρείες

που εξαρτώνται από λογισμικό για τις δραστηριότητές τους, ανεξαρτήτως μεγέθους και βιομηχανίας. Η υιοθέτηση της μεθοδολογίας αφορά πολλούς τομείς επιχειρήσεων και φορέων, όπως :

**Τεχνολογία Πληροφορικής:** Εταιρείες που αναπτύσσουν λογισμικό και παρέχουν υπηρεσίες πληροφορικής.

**Χρηματοοικονομικοί Φορείς:** Τράπεζες, χρηματοπιστωτικές εταιρείες και ασφαλιστικές εταιρείες που χρησιμοποιούν λογισμικό για τις χρηματοοικονομικές τους λειτουργίες και για τη διαχείριση κρίσιμων οικονομικών δεδομένων. Διασφαλίζει την ασφάλεια, την ακεραιότητα και την αξιοπιστία των συστημάτων [5].

**Λιανικό Εμπόριο:** Η οργανωμένη ανάπτυξη λογισμικού συμβάλλει στην απρόσκοπτη λειτουργία των e-Shops και στην αποτελεσματική διαχείριση των παραγγελιών και των αποθεμάτων, επιχειρήσεων που δραστηριοποιούνται στο ηλεκτρονικό εμπόριο.

**Υγειονομική Περίθαλψη:** Νοσοκομεία, ιατρικά κέντρα και φαρμακευτικές εταιρείες που χρησιμοποιούν λογισμικό για τη διαχείριση ασθενών, ιατρικών εγγραφών και άλλων λειτουργιών, διασφαλίζοντας την ασφάλεια και ακεραιότητα των ευαίσθητων ιατρικών δεδομένων, ενώ παράλληλα βελτιώνεται η ποιότητα της παρεχόμενης φροντίδας [6][7].

**Κυβέρνηση:** Κρατικοί φορείς και δημόσιες υπηρεσίες που χρησιμοποιούν λογισμικό για την υλοποίηση διαδικασιών και την παροχή δημόσιων ποιοτικών υπηρεσιών προς τους πολίτες. Όπως αποδεικνύεται [8], οι ηλεκτρονικές δημόσιες υπηρεσίες κάνουν την εμφάνιση τους όλο και περισσότερο ανά τις χώρες, με καλύτερο παράδειγμα αυτό της Μεγάλης Βρετάνιας, που έχει χρησιμοποιήσει ήδη αρκετά μοντέλα για τις κρατικές της εφαρμογές και συνεχίζει ακόμα να εξελίσσεται σε αυτό το τομέα [9].

Υπάρχουν πολλά παραδείγματα γνωστών εταιρειών που υιοθετούν το SDLC ήδη εδώ και χρόνια και αναφέρουν ότι τα οφέλη είναι αισθητά ιδίως ως προς την διαχείριση του χρόνου και τη συνεχή βελτίωση των υπηρεσιών τους. Πιο συγκεκριμένα, αναφέρεται ότι η Microsoft το 2001, προκειμένου να διασφαλίσει την σωστή διαχείριση των κινδύνων ασφαλείας της, δημιούργησε, στα πλαίσια των μοντέλων SDLC που χρησιμοποιούσε ήδη, το μοντέλο SDL-IT, το οποίο έδινε περισσότερη έμφαση στην ασφάλεια και την ιδιωτικότητα [10]. Η IBM επίσης, υιοθέτησε το SDLC για την ανάπτυξη της cloud πλατφόρμας της, διασφαλίζοντας ότι η πλατφόρμα θα ήταν ευέλικτη και προσαρμοσμένη στις ανάγκες των πελατών.

## Ρόλος των μελών της ομάδας ανάπτυξης στο SDLC

Ο ρόλος των μελών της ομάδας ανάπτυξης είναι κρίσιμος για την επιτυχία του Software Development Life Cycle (SDLC). Κάθε μέλος συμβάλλει με τις δεξιότητές του και την ειδίκευσή του σε κάθε φάση του SDLC [11], επιτρέποντας την ομαλή πρόοδο και την αποτελεσματική ολοκλήρωση του έργου. Κυριότεροι ρόλοι των μελών είναι οι εξής:

**Project Manager (Διευθυντής Έργου):** Ο ρόλος του Project Manager είναι να διαχειρίζεται το έργο συνολικά [12]. Αναλαμβάνει την προγραμματισμένη εκτέλεση των φάσεων του SDLC, τον προσδιορισμό των προθεσμιών και των στόχων, καθώς και την ανάθεση αρμοδιοτήτων στα μέλη της ομάδας.

**Business Analyst (Αναλυτής Επιχειρησιακών Απαιτήσεων):** Ο Business Analyst είναι υπεύθυνος για την κατανόηση των αναγκών και απαιτήσεων του πελάτη ή του τελικού χρήστη. Συνεργάζεται με τους ενδιαφερόμενους φορείς για τη συλλογή, την ανάλυση και την καταγραφή των λειτουργικών και μη λειτουργικών απαιτήσεων του συστήματος.

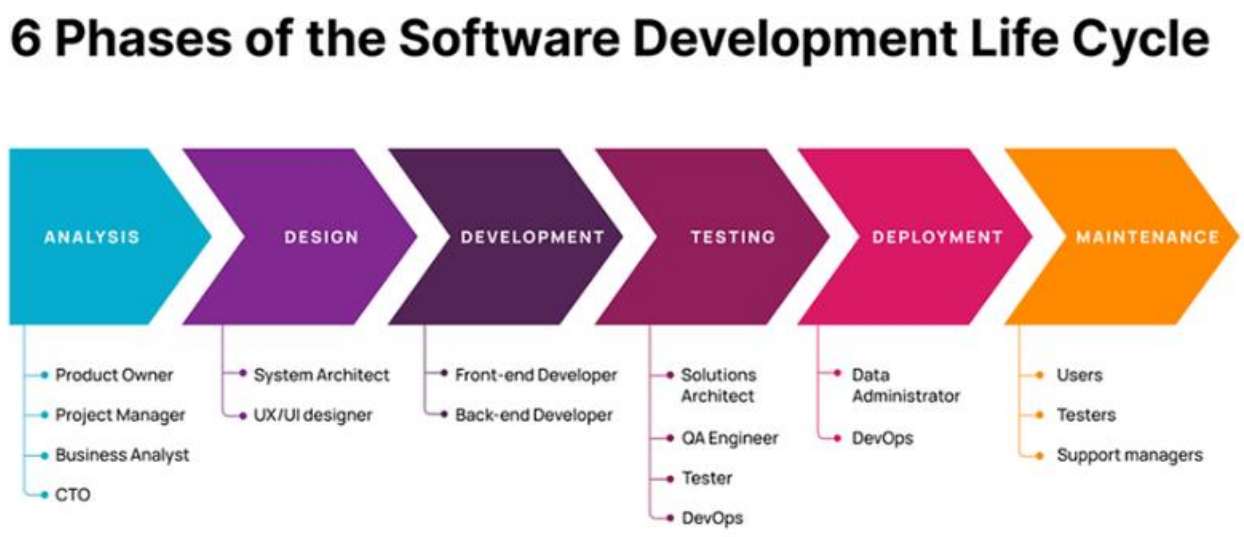
**UI/UX Designers (Σχεδιαστές Χρήσης και Εμπειρίας Χρήστη):** Ο σχεδιαστής UI/UX δημιουργεί τον σχεδιασμό της διεπαφής χρήστη (UI) και βελτιστοποιεί την εμπειρία του χρήστη (UX). Στόχος του είναι να δημιουργήσει μια ευχάριστη και λειτουργική εμπειρία για τους χρήστες.

**Software Developers (Προγραμματιστής Λογισμικού):** Οι προγραμματιστές είναι υπεύθυνοι για την υλοποίηση του λογισμικού σύμφωνα με τις προδιαγραφές και απαιτήσεις του έργου. Αναπτύσσουν τον κώδικα της εφαρμογής, επιλέγοντας τις κατάλληλες προγραμματιστικές γλώσσες, εργαλεία και τεχνολογίες.

**Quality Assurance Analysts ( Αναλυτές Αξιολόγησης Ποιότητας):** Η ομάδα Quality Assurance είναι υπεύθυνη για την εκτέλεση δοκιμών και ελέγχων, για τον έλεγχο της ποιότητας του λογισμικού. Βεβαιώνονται ότι το λογισμικό λειτουργεί σωστά, ανιχνεύοντας και διορθώνοντας τυχόν σφάλματα ή προβλήματα.

Συνοψίζοντας, το SDLC αποτελεί μια κρίσιμη μεθοδολογία για την ανάπτυξη λογισμικού, προσφέροντας οφέλη σε οργανισμούς και επιχειρήσεις που επιδιώκουν την παραγωγή υψηλής ποιότητας προϊόντων. Με το SDLC, οι επιχειρήσεις μπορούν να επιτύχουν αποτελεσματική διαχείριση έργου, μείωση του ρίσκου και βελτίωση της ποιότητας του παραγόμενου λογισμικού. Στο επόμενο κεφάλαιο, θα επικεντρωθούμε στις διάφορες φάσεις του SDLC και τις λεπτομερείς διαδικασίες που ακολουθούνται σε κάθε μία από αυτές.

## *2.2 SDLC Phases*




*Σχήμα 1. Οι ρόλοι των μελών σε κάθε φαση του SDLC (https://productcoalition.com/a-comprehensive-guide-to-the-software-development-life-cycle-sdlc-15b7892e1d44)*

Όπως ήδη αναφέρθηκε, το SDLC χωρίζεται σε 6 ή 7 στάδια (ανάλογα με τις ανάγκες του εκάστοτε έργου) [13][2]. Παρακάτω θα αναλυθούν καθένα από αυτά και οι λειτουργίες που εκτελούνται.

## Στάδιο Ανάλυσης Απαιτήσεων (Requirements Analysis)

Στο στάδιο του σχεδιασμού, ξεκινά η διαδικασία προετοιμασίας για την επιτυχή ανάπτυξη του λογισμικού. Η ομάδα που εμπλέκεται στο έργο έχει ως βασικό στόχο να καθορίσει το πεδίο εφαρμογής του έργου, τους στόχους που πρέπει να επιτευχθούν και τις λύσεις που αναμένονται . Πραγματοποιούνται συνεντεύξεις με τους ενδιαφερόμενους φορείς και συλλέγονται πληροφορίες για τις λειτουργίες που πρέπει να υποστηριχθούν, τα δεδομένα που θα αναλυθούν, και τους περιορισμούς που πρέπει να ληφθούν υπόψη. Έτσι εξασφαλίζεται η άμεση και σωστή αντιμετώπιση των απαιτήσεων και η ακριβής ανάλυση των επιχειρηματικών διαδικασιών που πρέπει να καλυφθούν και να επιλυθούν από το μελλοντικό σύστημα, με γνώμονα σαφώς τις απαιτήσεις του πελάτη.

Οι στόχοι αυτοί θα πρέπει να είναι σαφείς, ρεαλιστικοί με το πεδίο εφαρμογής και με την πραγματοποίηση τους καθώς και να συμφωνούν με τα χρονικά περιθώρια.

Σε αυτό το στάδιο, τεκμηριώνονται όλες οι απαιτήσεις και οι προδιάγραφες του έργου, προσδιορίζονται απαιτήσεις υλικού και απαιτήσεις δικτύου, προτείνονται πρωτότυπες ιδέες και αξιολογούνται εναλλακτικές λύσεις και περιορισμοί που πρέπει να ληφθούν υπόψη. Οι προγραμματιστές δημιουργούν ένα Έγγραφο Προδιαγραφής Απαιτήσεων (Software Requirements Specification – SRS), στο οποίο περιγράφεται τι θα κάνει η εφαρμογή και πως αναμένεται να λειτουργήσει. Δημιουργούνται μοντέλα όπως οι λειτουργικές απαιτήσεις (functional requirements) και οι μη λειτουργικές απαιτήσεις (non-functional requirements), προκειμένου να περιγράψουν με σαφήνεια τι πρέπει να πετύχει το λογισμικό και ποια περιβάλλοντα πρέπει να εξυπηρετούν.

Μια σωστά προετοιμασμένη φάση σχεδιασμού εξασφαλίζει την σωστή χρηματοδότηση και τους πόρους που χρειάζονται για την ανάληψη του εκάστοτε συστήματος. Εξασφαλίζει επίσης την μεγαλύτερη αποδοτικότητα κατά την ανάπτυξη και μειώνει τον κίνδυνο εμφάνισης σφαλμάτων.

Θεωρείται από τα πιο σημαντικά στάδια, καθώς επίσης καθορίζει το χρονοδιάγραμμα ολοκλήρου του έργου. Ιδίως σε εμπορικές συμφωνίες και με την συνεχή αύξηση των απαιτήσεων των εταιρειών, ένα σωστό χρονοδιάγραμμα θα επιφέρει μικρότερο κόστος και διαμοιρασμό του χρόνου της κάθε φάσης καλυτέρα. Οπότε η ομάδα συνήθως χρειάζεται, αλλά και αφιερώνει αρκετό χρόνο, για να αναλύσει σωστά κάθε λεπτομέρεια του έργου, προκειμένου να συντάξει ένα σωστό χρονοδιάγραμμα [14].

## Στάδιο Σχεδίασης (Design)

Επόμενο στάδιο και από τα πιο απαραίτητα στάδια του SDLC, είναι το στάδιο της σχεδίασης. Τα μέλη του έργου, αφού έχουν καθορίσει σωστά τις απαιτήσεις και την τεκμηρίωση που υλοποίησαν στη φάση της ανάλυσης απαιτήσεων, καλούνται όλα αυτά τα στοιχεία, να τα χρησιμοποιήσουν για να αναπτύξουν την τεχνική, πλέον, τεκμηρίωση του έργου.

Στοιχεία όπως η αρχιτεκτονική του λογισμικού, η διεπαφή του χρήστη και του συστήματος, το δίκτυο, οι απαιτήσεις του, η διαχείριση των δεδομένων, θα θέσουν τις βάσεις για το σχέδιο του έργου. Η διεπαφή θα πρέπει να είναι φιλική και εύχρηστη, ώστε οι χρήστες να μπορούν να αλληλοεπιδρούν με το λογισμικό χωρίς δυσκολία και προβλήματα, αλλά ταυτόχρονα να είναι σύμφωνη και με τις απαιτήσεις των πελατών.

Συγκεκριμένα για τη σχεδίαση της διεπαφής δημιουργούνται πρωτότυπα σχέδια του λογισμικού και των σελίδων με βάση την χρηστικότητα και την οπτική αισθητική. Έτσι ο πελάτης μπορεί να γνωρίζει σε αρχικό στάδιο τη μορφή που θα έχει το γραφικό περιβάλλον του λογισμικού και να ανατροφοδοτεί την ομάδα σχεδίασης σε περαιτέρω αλλαγές, προτού αυτό αναπτυχθεί.

Όλα αυτά τα στοιχεία αναλύονται και παρουσιάζονται στο Έγγραφο Προδιαγραφής Απαιτήσεων Επιχειρησιακού Επιπέδου (Business Requirements Specification – BRS), το οποίο είναι ένα επιχειρηματικό έγγραφο, που περιγράφει τις υψηλού επιπέδου απαιτήσεις από την οπτική γωνία της επιχείρησης για το σύστημα που θα αναπτυχθεί. Περιλαμβάνει πληροφορίες σχετικά με τους σκοπούς, τους στόχους και τα οφέλη του συστήματος.

Με την ολοκλήρωση της καταγραφής, τα μέλη της ομάδας θα είναι σε θέση να γνωρίζουν τι πρέπει να κάνουν σε κάθε στάδιο του SDLC, αλλά δίνει τη δυνατότητα και στην άλλη πλευρά, αυτήν του πελάτη, να κατανοήσει τα σχέδια λειτουργίας. Το έγγραφο SRS που δημιουργήθηκε στην προηγουμένη φαση, θα μετατραπεί σε μια πιο απλή γλώσσα, ούτως ώστε να μπορεί να μετατραπεί σε μια γλώσσα προγραμματισμού. Επίσης, στο στάδιο του σχεδιασμού, οι ομάδες συνηθίζουν να κατασκευάζουν διαγράμματα UML, με πιο σημαντικά να είναι το Class Diagram, UML Sequence Diagram και το Use Case Diagram, για να αποδώσουν πιο κατανοητά την αρχιτεκτονική του συστήματος.

## Στάδιο Ανάπτυξης (Development)

Μια από τις πιο χρονοβόρες, αλλά ενδιαφέρουσες, ταυτόχρονα. φάσεις ενός έργου, είναι αυτή της κατασκευής. Είναι το στάδιο όπου οι προγραμματιστές γράφουν πραγματικό κώδικα και κατασκευάζουν την εφαρμογή σύμφωνα με τα έγγραφα σχεδιασμού και τις περιγραφόμενες προδιαγραφές, που έχουν αναλυθεί στα προηγούμενα στάδια. Διαμορφώνονται τα cloud συστήματα, τα πρωτόκολλα και το υλικοτεχνικό κομμάτι, εκτός της σύνταξης του κώδικα, για αυτό και αυτό το στάδιο είναι τόσο σημαντικό.

Τα διαγράμματα UML, που κατασκευάστηκαν νωρίτερα, κατέχουν σημαντικό ρόλο στην έγκαιρη και ορθή εκπλήρωση αυτού του σταδίου. Λανθασμένα διαγράμματα θα καθυστερούσαν συνολικά το έργο, δεδομένο που δείχνει ακόμα πιο έντονα την αναγκαιότητα της όσο το δυνατόν σωστότερης ολοκλήρωσης της κάθε φάσης.

Παράλληλα με τον κώδικα, οι προγραμματιστές, ιδανικά σε αυτή την φάση, δημιουργούν ένα λεπτομερές έγγραφο για τον κώδικα τους, γνωστό και ως Έγγραφο Σχέδιου Αποδοχής (Acceptance Test Plan – ATP), που περιγράφει πώς θα δοκιμαστεί το σύστημα λογισμικού για να διασφαλιστεί ότι πληροί τις απαιτήσεις που καθορίζονται στα προηγούμενα δυο έγγραφα, το SRS και το BRS. Αυτή η τεκμηρίωση βοηθά άλλα μέλη της ομάδας να κατανοήσουν τον κώδικα, τον σκοπό του, τη χρήση του και τυχόν εξαρτήσεις. Η κατάλληλη τεκμηρίωση είναι ζωτικής σημασίας για τη μελλοντική συντήρηση και ενημέρωση, αλλά και για τη φάση της δοκιμής στο επόμενο στάδιο.

Οι προγραμματιστές ακολουθούν συγκεκριμένες οδηγίες και χρησιμοποιούν συγκεκριμένα εργαλεία και compilers, debuggers όσον αφορά τον κώδικα, με βάση τις ανάγκες που έχουν συμφωνηθεί σε προηγούμενο στάδιο με τον πελάτη.

## Στάδιο Δοκιμής (Testing)

Σε αυτή τη φάση, το λογισμικό που έχει αναπτυχθεί ελέγχεται διεξοδικά με βάση το Έγγραφο Σχέδιου Αποδοχής (ATP) που συντάχθηκε, για τυχόν ελαττώματα και σφάλματα, ξανά και ξανά, έως ότου τα αποτελέσματα ταιριάζουν με το αναμενόμενο αποτέλεσμα. Αν υπάρχουν και διαπιστωθούν ελαττώματα και σφάλματα, ανατίθενται στους προγραμματιστές, για να κάνουν εκ νέου διορθώσεις και στην συνέχεια επανεξετάζονται, μέχρις ότου το λογισμικό να είναι σύμφωνο με τις προσδοκίες του πελάτη και η εμπειρία του τελικού χρήστη να μην επηρεαστεί αρνητικά σε κανένα σημείο.

Η δοκιμή πρέπει να διασφαλίζει ότι κάθε λειτουργία και διαφορετικά μέρη της εφαρμογής λειτουργούν ταυτόχρονα σωστά, με στόχο να μειωθούν τυχόν καθυστερήσεις και δυσλειτουργίες αλλά και να βελτιστοποιηθεί το ποσοστό χρήσης.

Υπάρχουν διάφοροι τύποι δοκιμών κατά τη φάση της δοκιμής, πιο γνωστοί από τους οποίους είναι των δοκιμών διασφάλισης ποιότητας (QA) [15], δοκιμών ολοκλήρωσης συστήματος (SIT) [16] και δοκιμών αποδοχής χρήστη (UAT) [17].

### STLC (Software Testing Life Cycle)

Το STLC αναφέρεται στη διαδικασία που χρησιμοποιείται για τις δοκιμές λογισμικών και περιγράφει τη σειρά των βημάτων που πρέπει να ακολουθήσει η ομάδα δοκίμων για να διασφαλιστεί η ποιότητα του λογισμικού.

Κάθε επιτυχημένο SDLC περιλαμβάνει τη φάση των δοκιμών, και για τον λόγο αυτό, το STLC θεωρείται ως ένα υποσύνολο του SDLC και συνδέεται άρρηκτα με τις άλλες φάσεις της αναπτυξης λογισμικού [18]. Καθορίζει τις διαδικασίες, τις τεχνικές και τα εργαλεία που χρησιμοποιούνται για την εκτέλεση των δοκιμών και τον έλεγχο της ποιότητας του λογισμικού πριν από την παράδοση του στον πελάτη. Οι βασικές φάσεις του STLC περιλαμβάνουν τα εξής:

- **Σχεδιασμός των δοκιμών:** Σε αυτό το στάδιο ορίζονται οι στόχοι και το σχέδιο των δοκιμών και δημιουργούνται τα σενάρια δοκιμών. Καθορίζεται η στρατηγική δοκιμών, ο σχεδιασμός των σεναρίων και ορίζονται οι πόροι που θα χρησιμοποιηθούν.
- **Ανάπτυξη των δοκιμών και των σεναρίων:** Σε αυτή την φαση των δοκίμων, δημιουργούνται οι δοκιμαστικοί κώδικες και τα σενάρια δοκίμων και ετοιμάζονται τα δεδομένα που θα χρησιμοποιηθούν κατά τις δοκιμές.
- **Εκτέλεση των δοκιμών:** Οι δοκιμαστικοί κώδικες και τα προγραμματισμένα σενάρια εκτελούνται και καταγράφονται τα αποτελέσματα. Εάν στα αποτελέσματα ανιχνευθούν σφάλματα, τότε αναφέρονται και παρακολουθείται η διαδικασία επιδιόρθωσής τους.

- **Αναφορά Σφαλμάτων:** Οι προγραμματιστές αναλύουν τα σφάλματα της προηγούμενης φάσης και προχωρούν στην επιδιόρθωση τους. Οι επιδιορθωμένοι κώδικες επαναδοκιμάζονται με σκοπό την επαλήθευση της επιτυχούς επίλυσης των σφαλμάτων.

- **Αναθεώρηση δοκιμών:** Σε αυτό το στάδιο περιλαμβάνεται η αναθεώρηση των αποτελεσμάτων των δοκιμών και δημιουργείται μια συνοπτική αναφορά για αυτά. Ελέγχεται η ποιότητα του λογισμικού και τυχόν προβλήματα σημειώνονται για επίλυση σε επόμενη φάση. Ο πελάτης αξιολογεί με την σειρά του το λογισμικό και αποφασίζει εάν είναι έτοιμο για παράδοση ή όχι.

Με την εφαρμογή του STLC στο στάδιο της δοκιμής, επιτυγχάνεται η ασφάλεια και η αξιοπιστία, μειώνοντας την εμφάνιση κρίσιμων σφαλμάτων και βελτιώνοντας συνολικά την ποιότητα του λογισμικού και την εμπειρία των χρηστών. Επιτυγχάνεται επίσης εξοικονόμηση χρόνου και πόρων από την έγκαιρη ανίχνευση και διόρθωση των σφαλμάτων, διασφαλίζοντας την ευελιξία και τη λειτουργικότητα, αφού είναι προσαρμοσμένο στις ανάγκες των πελατών. Μέσω της συμμόρφωσης και της πλήρης κατανόησης των προδιαγραφών, των αναγκών και των κανονισμών που έχουν τεθεί, επιτυγχάνεται η αξιοπιστία του φορέα και η εμπιστοσύνη των πελατών και θέτονται θεμέλια μακροχρόνιας συνεργασίας. Συνεπώς, η διαδικασία του STLC είναι ένα απαραίτητο εργαλείο για την εξασφάλιση υψηλής ποιότητας λογισμικού και απαραίτητο κομμάτι συνολικά του SDLC.

Η φάση της δοκιμής είναι κρίσιμη, καθώς αποτελεί την τελευταία φάση πριν διατεθεί το λογισμικό στον τελικό χρήστη, διασφαλίζοντας ταυτόχρονα ότι θα πληροί τα κριτήρια ποιότητας που είχαν οριστεί και συμφωνηθεί εγγράφως στα προηγούμενα στάδια.

Αναλόγως λοιπόν, με την πολυπλοκότητα του λογισμικού, τις απαιτήσεις του τελικού χρήστη, τον αριθμό των σφαλμάτων αλλά και τις ικανότητες των ομάδων ανάπτυξης και δοκιμών, το στάδιο της δοκιμής μπορεί είτε να είναι μια σύντομη διαδικασία , είτε να αποδειχθεί πολύ χρονοβόρα τελικά.

## Στάδιο Εφαρμογής/Υλοποίησης (Deployment)

Αφού τελειώσει η φαση της δοκιμής, σειρά έχει αυτή της εφαρμογής, της διάθεσης δηλαδή του λογισμικού στο χρήστη [19]. Το λογισμικό, αφού εγκατασταθεί και ενσωματωθεί στο περιβάλλον λειτουργίας του από τους προγραμματιστές και δοθούν οι οδηγίες χρήσης στους χρήστες, θα είναι έτοιμο για χρήση.

Για αυτό το λόγο, αυτό το στάδιο περιλαμβάνει την εκπαίδευση και την υποστήριξη των χρηστών πριν ή και κατά τη διάρκεια λειτουργίας του λογισμικού.

Ωστόσο, ακόμα και μετά την κυκλοφορία της εφαρμογής, πραγματοποιείται ακόμα ένας μικρός κύκλος δοκίμων, αν αυτό κρίνεται απαραίτητο, κυρίως για επίλυση θεμάτων συμβατότητας ή παρατηρήσεων από τους χρήστες, για τη βελτίωση του λογισμικού αλλά και για την προσθήκη τεκμηρίωσης για τον κώδικα.

### Στάδιο Συντήρησης (Maintenance)

Η φάση της συντήρησης είναι πολύ σημαντική για την εξέλιξη και την αδιάλειπτη λειτουργία της νέας εφαρμογής, καθώς κύριος στόχος αυτής της φάσης είναι να διασφαλίσει ότι οι ανάγκες συνεχίζουν να ικανοποιούνται και ότι το σύστημα συνεχίζει να λειτουργεί σύμφωνα με τις προδιαγραφές που αναφέρονται στην πρώτη φάση.

Ακόμα όμως και να ικανοποιούνται πλήρως οι ανάγκες, πέραν της διαχείρισης και της διόρθωσης σφαλμάτων, οι προγραμματιστές είναι υπεύθυνοι για την τροποποίηση , το σχεδιασμό πρόσθετων δυνατοτήτων αλλά και των αναβαθμίσεων σε νεότερες βελτιστοποιημένες εκδόσεις.

Κατά τη φάση συντήρησης, παρέχονται υπηρεσίες υποστήριξης και προστασίας του λογισμικού από πιθανά σφάλματα ή προβλήματα. Οι πελάτες μπορούν να ζητήσουν και να αναπτύξουν νέες λειτουργίες ή να προσαρμόσουν το λογισμικό με βάση τις εξελίξεις της αγοράς και τις ανάγκες τους. Η συντήρηση και ανάπτυξη του λογισμικού εξασφαλίζει ότι η εφαρμογή παραμένει χρήσιμη και λειτουργική για τους τελικούς χρήστες.

Όπως είναι λογικό, όλες αυτές οι εργασίες και ιδιαίτερα αυτή της συνεχούς αναβάθμισης μπορεί να δημιουργούν νέους κύκλους ανάπτυξης, οπότε ουσιαστικά αυτό το στάδιο, θα συνεχίζει να είναι σε λειτουργία και να απασχολεί τους προγραμματιστές όσο η εφαρμογή είναι και θα συνεχίσει να είναι διαθέσιμη προς πώληση ή προς όποια γενικότερη εκμετάλλευση [20].

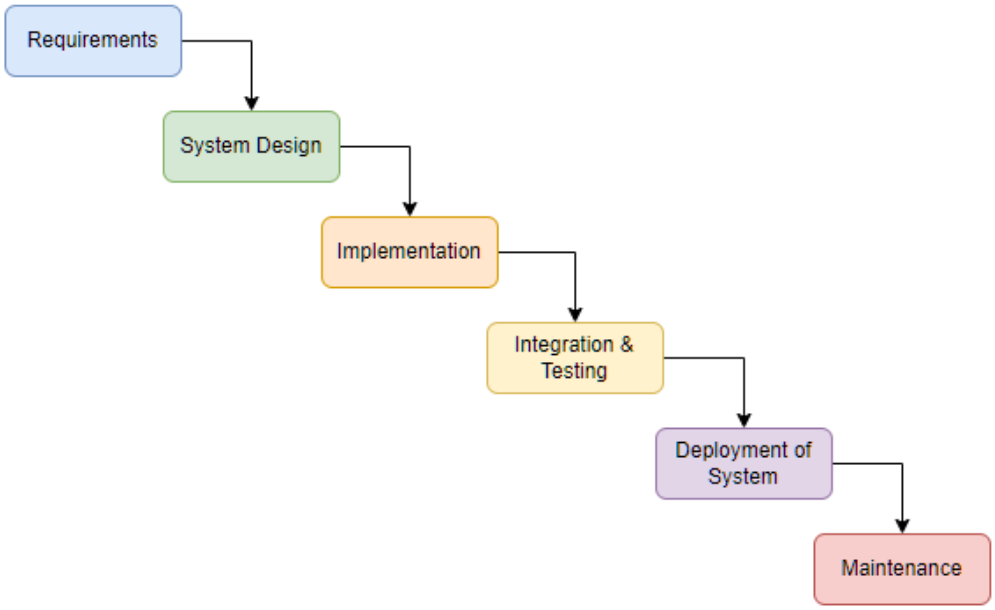
## *2.3 SDLC Models & Methodologies*

Η μέθοδος SDLC χρησιμοποιεί πολλά διαφορετικά μοντέλα που καθορίζουν και σχεδιάζουν την διαδικασία της ανάπτυξης, τα οποία ακολουθούνται σε όλη την φάση της. Αυτά τα μοντέλα ονομάζονται επίσης "Μοντέλα Διαδικασίας Ανάπτυξης Λογισμικού". Κάθε μοντέλο διαδικασίας ακολουθεί μια σειρά συγκεκριμένων φάσεων ανάλογα τον τύπο του, με σκοπό να εξασφαλίσει την επιτυχία του εκάστοτε βήματος της ανάπτυξης λογισμικού [21] [22]. Τα πιο σημαντικά και γνωστά από τα μοντέλα είναι το Waterfall Model, V-Shaped Model, Iterative Model, Spiral Model, Agile Model, RAD Model, Big Bang Model και αρκετά άλλα και συνεχώς όλο και εμφανίζονται νέα ή παραλλαγές των ήδη υπαρχόντων. Παρακάτω όμως θα αναλυθούν τα σημαντικότερα από αυτά.

### 2.3.1 Waterfall Model

Το μοντέλο του “καταρράκτη” είναι το πιο κλασσικό και διαδεδομένο μοντέλο διαδικασίας ανάπτυξης λογισμικού καθώς και το πιο ευθύ από όλα τα υπόλοιπα μοντέλα. Πρωτοεμφανίστηκε το 1970 από τον Winston W. Royce και η μέθοδος του βασίζεται στην απλή υποδιαίρεση του έργου σε διάφορες φάσεις. Κάθε φάση έχει το δικό της έργο και κάθε φαση, αφού τελειώσει, διαδέχεται την επόμενη [23].

Σαν τον καταρράκτη, έτσι και αυτή η μέθοδος φαίνεται να ρέει σταθερά προς τα κάτω, μέσω των φάσεων και προτιμάται σε έργα όπου το κόστος και ο χρόνος δεν είναι τόσο σημαντικά όσο η ποιότητα. Με αυτό το μοντέλο διασφαλίζεται η συνέπεια του κάθε σταδίου και κατ' επέκταση η συνέπεια ολόκληρου του συστήματος, αφού κάθε στάδιο επαληθεύεται πριν γίνει η μετάβαση στο επόμενο.




*Σχήμα 2. Waterfall Model (created with draw.io)*

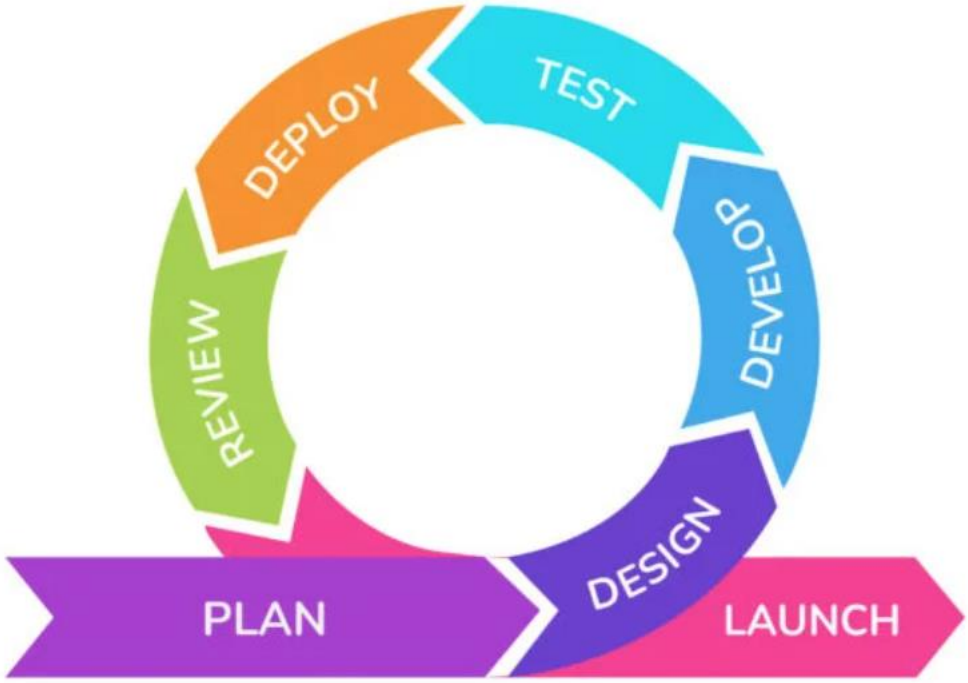
Πλεονεκτήματα αυτού του μοντέλου είναι η απλότητα που προσφέρει, οπότε επιτυγχάνεται η σταθερότητα ως προς το σχεδιασμό, τον κώδικα, και τα παραδοτέα σε κάθε φάση. Επιτυγχάνεται ακόμα η ελαχιστοποίηση των απαιτούμενων πόρων σε σύγκριση με άλλες μεθόδους και η σαφήνεια του έργου ως προς τα μέλη της ομάδας αλλά και τους πελάτες.

Το Waterfall είναι ένα γραμμικό μοντέλο ανάπτυξης λογισμικού, που σημαίνει ότι κάθε στάδιο πρέπει να ολοκληρωθεί πριν ξεκινήσει το επόμενο. Λόγω της σταθερότητας όμως που προσφέρει το μοντέλο του "καταρράκτη", αποτυγχάνεται η ευελιξία. Χάνεται η δυνατότητα επιστροφής και αλλαγής απαιτήσεων (από το στάδιο του σχεδιασμού και έπειτα), αφού μια τέτοια αλλαγή θα οδηγούσε σε αναπήδηση προηγουμένων σταδίων, οπότε και σε ελαττώματα στο σχεδιασμό και στον κώδικα . Σφάλματα, που οι προγραμματιστές αναγνωρίζουν δηλαδή, συνεχίζουν να υπάρχουν μέχρι το τέλος του έργου, οδηγώντας σε νέες εργασίες επεκτείνοντας και άλλο το χρόνο αναμονής για τον πελάτη. Επίσης η ομάδα δοκίμων εμπλέκεται μόνο στη φάση των δοκιμών και όχι σε όλο το έργο συνολικά. Ο πελάτης, τελικά, δεν θα μείνει ικανοποιημένος, είτε λόγω των χρονικών περιθωρίων, είτε από το τελικό αποτέλεσμα, αφού αν θέλει να είναι εντός χρονικών περιθωρίων, θα πρέπει να συμβιβαστεί με την μη προσθήκη των αλλαγών στο τελικό προϊόν.

Εν κατακλείδι, το μοντέλο του καταρράκτη, προτείνεται για έργα με σταθερές και γνωστές απαιτήσεις και για βραχυπρόθεσμα έργα με πιο άμεση παράδοση [24].

### 2.3.2 Agile Model

Η ταχεία εξέλιξη των τεχνολογιών και η πολυπλοκότητα των προς επίλυση ζητημάτων καθιστούν δύσκολο τον ακριβή προσδιορισμό των απαιτήσεων για την υλοποίηση προγραμματιστικών έργων, οπότε η υιοθέτηση μοντέλων όπως το Waterfall δεν ενθαρρύνουν την επικοινωνία και την συνεργασία των εμπλεκόμενων μερών. Σε τέτοιες περιπτώσεις το μοντέλο Agile προσαρμόζεται καλυτέρα στις ανάγκες του project, γιατί επικεντρώνεται στην ικανοποίηση του πελάτη και στη γρήγορη παράδοση του συστήματος [25][26]. Η συνεργασία όλων των εμπλεκόμενων μερών είναι απαραίτητη για την κατανόηση των πραγματικών αναγκών των πελατών και των επιχειρήσεων [27].




*Σχήμα 3. Agile Model (https://www.krasamo.com/agile-development-process/)*

Η ιδέα πίσω από τη μέθοδο Agile, είναι στην υλοποίηση της ανάπτυξης σε επαναληπτικούς κύκλους χωρίζοντας την σε μικρά, συνεχή επαναλαμβανόμενα βήματα, γνωστά ως "επαναλήψεις" ή "σπριντ", τα οποία συνήθως διαρκούν από μια έως τρεις εβδομάδες. Σε καθεμιά από αυτές τις επαναλήψεις, εργάζονται ταυτόχρονα αρκετές ομάδες μεταξύ τους σε τομείς όπως την ανάλυση απαιτήσεων, την σχεδίαση, τον κώδικα κλπ., και τους αντιμετωπίζουν διαφορετικά από έργο σε έργο. Οι απαιτήσεις κάθε έργου είναι διαφορετικές και το μοντέλο Agile προσαρμόζει τη μέθοδο ανάπτυξης στις εκάστοτε ανάγκες. Το μοντέλο αυτό εφαρμόζει, επίσης, επαναληπτική προσέγγιση, έως ότου, μέσα από όλες τις σταδιακές επαναλήψεις, το έργο να καταλήξει στην τελική μορφή με τα όλα τα χαρακτηριστικά που εξυπηρετούν τον πελάτη. Η συμφωνία των εμπλεκόμενων μερών για το αντικείμενο που θα υλοποιηθεί στον επόμενο κύκλο είναι απαραίτητη για την εξασφάλιση της συνοχής του έργου. Η συμφωνία αυτή πρέπει να βασίζεται στα αποτελέσματα των ενδιάμεσων ελέγχων και στις ανάγκες των τελικών χρηστών. Οι εργασίες χωρίζονται σε μικρά χρονικά πλαίσια,

έτσι ώστε σε κάθε πλαίσιο να παραδίδεται συγκεκριμένο χαρακτηριστικό, παρέχοντας απλότητα και την ύπαρξη λιγοστών κανόνων στην ανάπτυξη.

Οι κύριες αρχές του Agile είναι η επικοινωνία και η συνεργασία μεταξύ των μελών της ομάδας, η ανταπόκριση σε αλλαγές και η υποβολή συχνών, λειτουργικών παραδόσεων. Οι βασικές φάσεις του μοντέλου Agile περιλαμβάνουν:

1. Σχεδιασμός και καθορισμός απαιτήσεων: Καθορίζονται οι αρχικές απαιτήσεις και δημιουργείται ένα αρχικό σχέδιο για την ανάπτυξη.
2. Επαναλήψεις (Sprints): Η ανάπτυξη γίνεται σε μικρά, προκαθορισμένα χρονικά διαστήματα, με την ομάδα να παραδίδει λειτουργικό λογισμικό σε κάθε επανάληψη.
3. Δοκιμές και αξιολόγηση: Κάθε παράδοση δοκιμάζεται και αξιολογείται από την ομάδα και τους πελάτες.
4. Επανεκτίμηση και προσαρμογή: Μετά από κάθε επανάληψη, η ομάδα επανεκτιμά τις απαιτήσεις και προσαρμόζει το σχεδιασμό ανάλογα.

Γενικότερα, η μέθοδος Agile γίνεται όλο και πιο διαδεδομένη στο χώρο της ανάπτυξης λογισμικού, αφού σαν μοντέλο δεν απαιτεί πολλούς πόρους και είναι εύκολο στην διαχείριση. Οι προγραμματιστές του έργου είναι πιο ευέλικτοι, καθώς το μοντέλο τους επιτρέπει να εργάζονται σε έργα που οι απαιτήσεις μεταβάλλονται, ελαχιστοποιώντας τον γενικό σχεδιασμό και προσφέροντας μια γρήγορη και ρεαλιστική εικόνα του λογισμικού εκ των προτέρων. Επίσης, λόγω της ταυτόχρονης αναπτυξης και υλοποίησης των σταδίων, προωθείται η ομαδικότητα μεταξύ των μελών και των ομάδων.

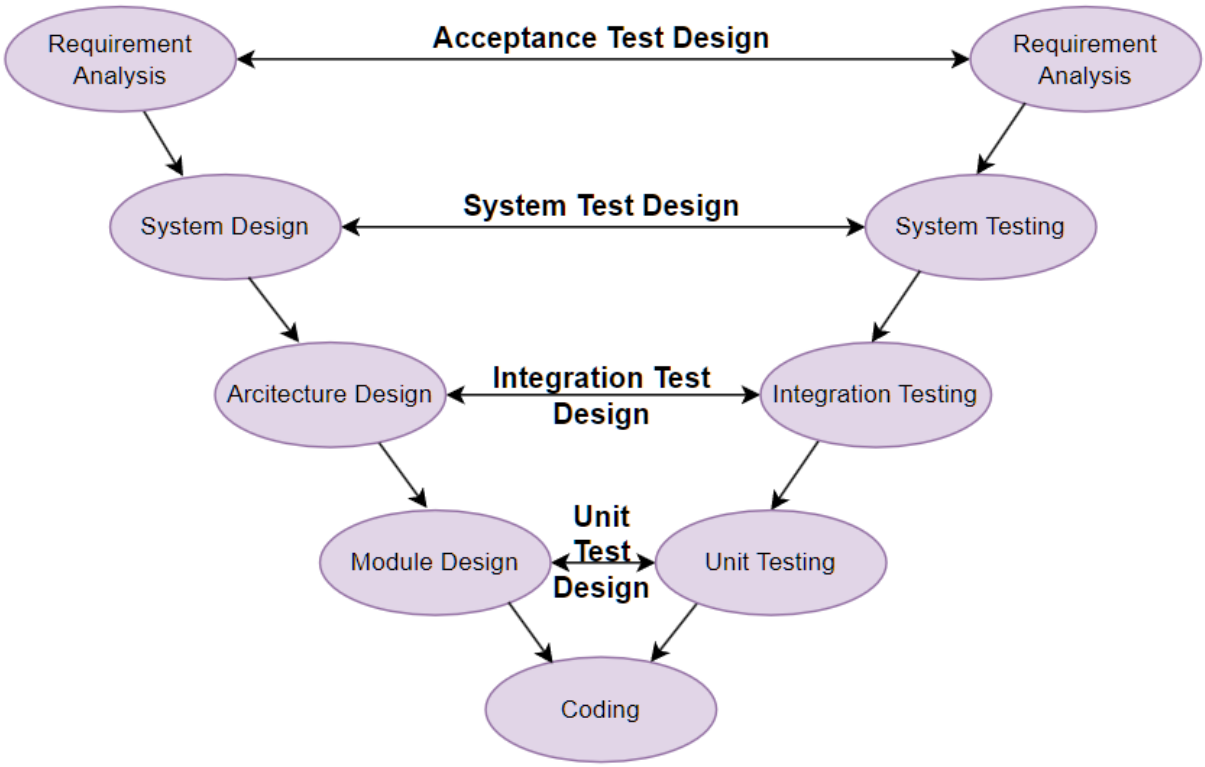
Βέβαια όπως και σε κάθε άλλο μοντέλο, το μοντέλο Agile έχει και αυτό κάποια μειονεκτήματα. Η υλοποίηση του έργου εξαρτάται σε μεγάλο βαθμό από τον πελάτη, ο οποίος πρέπει να είναι συγκεκριμένος ως προς τις απαιτήσεις, για να μην οδηγηθεί η ομάδα σε λάθος συμπεράσματα. Λόγω όμως τις απλότητας και των λιγοστών κανόνων του μοντέλου, το έργο εξαρτάται αρκετά και από τους εργαζομένους σε ατομικό επίπεδο, επειδή μπορεί να είναι δύσκολο να διαχειριστεί κανείς τις συχνές αλλαγές, πράγμα που μπορεί να δημιουργήσει αστοχίες στο τελικό προϊόν. Το κόστος καθώς και το χρονοδιάγραμμα του έργου είναι πιο δύσκολο να εκτιμηθούν αφού μπορεί οι επαναλήψεις που θα γίνουν και η εργασία από την ομάδα να αυξηθούν, ειδικά αν το έργο είναι μεγάλο ή περίπλοκο.

Συνοψίζοντας, η μέθοδος Agile δεν βασίζεται στην γραμμική προσέγγιση όπως οι πιο παραδοσιακές μέθοδοι (Waterfall, V-Shaped, κλπ.), αλλά χρησιμοποιεί μια πιο προσαρμοστική προσέγγιση, όπου δεν υπάρχει λεπτομερής σχεδιασμός και υπάρχει σαφήνεια σχετικά με τις μελλοντικές εργασίες μόνο σε σχέση με τα χαρακτηριστικά που πρέπει να αναπτυχθούν. Υπάρχει ανάπτυξη με γνώμονα τα χαρακτηριστικά και η ομάδα προσαρμόζεται δυναμικά στις μεταβαλλόμενες απαιτήσεις του προϊόντος. Το προϊόν δοκιμάζεται πολύ συχνά, μέσω των επαναλήψεων, ελαχιστοποιώντας τα μετέπειτα προβλήματα και τον χρόνο των δοκιμών.

### 2.3.3 V-Shaped Model

Το μοντέλο V, γνωστό και ως μοντέλο επαλήθευσης και επικύρωσης (V&V) ακολουθεί μια διαδικασία σχεδιασμού όπως το μοντέλο καταρράκτη, καθώς κάθε φάση του μοντέλου πρέπει να ολοκληρωθεί πριν ξεκινήσει η επόμενη φάση, με τη διαφορά ότι απαλείφει τα μειονεκτήματα του δευτέρου. Σε αυτό το μοντέλο η ομάδα δοκιμής συμμετέχει από τις πρώτες φάσεις του έργου,

διορθώνοντας και αποφεύγοντας σφάλματα που ενδεχομένως θα προκύπταν, μειώνοντας έτσι τον χρόνο εργασίας στο στάδιο δοκιμών. Η ιδέα είναι ότι από την μια πλευρά του “V” βρίσκονται οι φάσεις επαλήθευσης και από την άλλη πλευρά οι φάσεις επικύρωσης, και μια προς μια συνδέονται παράλληλα μέσω των δοκιμών, ώστε να διασφαλίζεται ότι κάθε απαίτηση του αναπτυσσόμενου συστήματος ελέγχεται σε κάθε στάδιο των δοκιμών και κάθε στάδιο ανάπτυξης (Development) και συσχετίζεται με το αντίστοιχο στάδιο δοκιμών (Testing), σχηματίζοντας "την βάση του V" [28][21].




*Σχήμα 4. V-Shaped model (created with draw.io)*

Τα παραδοτέα εκτελούνται παράλληλα καθώς ενώ, η ομάδα προγραμματισμού εργάζεται με τα έγγραφα απαιτήσεων συστήματος SRS (System Requirements Specification), η ομάδα δοκιμών εργάζεται με τα επιχειρησιακά έγγραφα απαιτήσεων BRS (Business Requirements Specification) και προετοιμάζουν το σχέδιο δοκιμής αποδοχής ATP (Acceptance Test Plan) και τα σενάρια δοκιμής αποδοχής ATC (Acceptance Test Cases) και ούτω καθεξής. Από την στιγμή που οι developers παραδώσουν το τελικό προϊόν, οι testers καθίστανται άμεσα ενημερωμένοι και έτοιμοι να αναλάβουν τις δοκιμές με όλα τα απαιτούμενα πλάνα (όπως το σχέδιο δοκιμής, τα test cases).

Επαλήθευση (Validation) : Η φάση επαλήθευσης αναφέρεται στην πρακτική αξιολόγησης της διαδικασίας ανάπτυξης προϊόντος για να διασφαλιστεί ότι η ομάδα πληροί τις καθορισμένες απαιτήσεις.

Επικύρωση (Validation): Η επικύρωση είναι η διαδικασία αξιολόγησης του λογισμικού μετά την ολοκλήρωση της διαδικασίας ανάπτυξης, για να προσδιοριστεί εάν το λογισμικό ανταποκρίνεται στις προσδοκίες και τις απαιτήσεις των πελατών.

Τα στάδια της επαλήθευσης είναι τα εξής :

**Ανάλυση απαιτήσεων πελάτη (Business requirements analysis):** Στο πρώτο βήμα γίνεται η πρώτη επαφή με τον πελάτη και γίνεται λεπτομερής συζήτηση σχετικά με τις απαιτήσεις του προϊόντος. Είναι μια πολύ σημαντική διαδικασία καθώς ο πελάτης μπορεί να μην έχει σαφείς απαιτήσεις από την αρχή.

**Σχεδίαση συστήματος (System Design) :** Σε επόμενο στάδιο οι developers αναλύουν τις απαιτήσεις του εγγράφου απαιτήσεων και σχεδιάζουν την αρχιτεκτονική του συστήματος. Επίσης το σχέδιο δοκιμής συστήματος (ATP) αναπτύσσεται με βάση τη σχεδίαση του συστήματος, αφήνοντας περισσότερο χρόνο για την πραγματική εκτέλεση της δοκιμής αργότερα

**Σχεδίαση αρχιτεκτονικής (Architecture Design):** Σε αυτό το στάδιο γίνονται κατανοητές οι αρχιτεκτονικές προδιαγραφές. Προσδιορίζονται οι τεχνικές προσεγγίσεις ενώ ο σχεδιασμός (αναφέρεται και σαν High Level Design, HLD) του συστήματος διαιρείται σε μικρότερα διαφορετικά και πιο τεχνικά κομμάτια, καθώς εδώ καθορίζονται τα μέσα μεταφοράς, οι βάσεις δεδομένων, η αλληλεπίδραση, οι σχέσεις και οι εξαρτήσεις των modules και τεχνικές λεπτομέρειες.

**Σχεδίαση module (Module Design):** Σε αυτή τη φάση, καθορίζεται η λεπτομερής εσωτερική σχεδίαση και η συμβατότητα όλων των μονάδων του συστήματος. Βάσει αυτής της εσωτερικής σχεδίασης αναπτύσσονται το σχέδιο δοκιμής αποδοχής ATP (Acceptance Test Plan) και τα σενάρια δοκιμής αποδοχής ATC (Acceptance Test Cases) που είναι πολύ σημαντικά για την εξάλειψη σφαλμάτων και ασυμβατοτήτων σε πρώιμο στάδιο.

Τα στάδια της επικύρωσης είναι τα εξής :

**Unit Testing :** Τα τεστ μονάδων εκτελούνται για να ελέγξουν τον κώδικα για ατομικά modules. Τα τεστ μονάδων βοηθούν στην εξάλειψη των σφαλμάτων σε πρώιμο στάδιο, αν και δεν μπορούν να εντοπίσουν όλα τα σφάλματα.

**Integration Testing:** Τα τεστ ολοκλήρωσης εκτελούνται για να ελέγξουν τη συνύπαρξη και την επικοινωνία μεταξύ των modules εντός του συστήματος. Τα τεστ ολοκλήρωσης βοηθούν να διασφαλιστεί ότι τα modules μπορούν να συνεργαστούν σωστά μεταξύ τους.

**System Testing:** Τα τεστ συστήματος εκτελούνται για να ελέγξουν ολόκληρη τη λειτουργικότητα του συστήματος και τη επικοινωνία του υπό ανάπτυξη συστήματος με εξωτερικά συστήματα. Τα τεστ συστήματος βοηθούν να διασφαλιστεί ότι το σύστημα πληροί όλες τις απαιτήσεις και ότι είναι συμβατό με άλλα συστήματα.

**Acceptance Testing:** Τα τεστ αποδοχής εκτελούνται για να ελέγξουν το προϊόν σε περιβάλλον χρήστη. Τα τεστ αποδοχής βοηθούν να διασφαλιστεί ότι το προϊόν είναι αξιόπιστο και ικανό να ανταποκριθεί στις ανάγκες των χρηστών.

Η "βάση του V" (Coding Phase) : Αυτή είναι η τελική φαση όπου συνδέονται όλα και όπου γράφεται ο πραγματικός κώδικας για τις μονάδες συστήματος. Επιλέγεται η καλύτερη γλώσσα προγραμματισμού και ο κώδικας γράφεται σύμφωνα με τις οδηγίες και τα πρότυπα κωδικοποίησης του έργου. Στη συνέχεια ελέγχεται και βελτιστοποιείται πριν παραδοθεί στον πελάτη και διατεθεί στην αγορά.

Αυτό το μοντέλο έχει εφαρμογή και στοχεύει κυρίως σε περίπλοκα και υψηλής ποιότητας έργα, με μεγάλο συνήθως χρόνο ανάπτυξης και καλά ορισμένων απαιτήσεων αλλά και για έργα που απαιτούν ασφάλεια, καθώς δίνεται έμφαση στην ενδελεχή δοκιμή και στην εξασφάλιση της αξιοπιστίας και της ποιότητας του λογισμικού, όπως για παράδειγμα σε αεροδιαστημικά και αμυντικά συστήματα [29]. Είναι ένα γενικά αυστηρό μοντέλο, καθώς κάθε φαση έχει συγκεκριμένα παραδοτέα και ολοκληρώνονται η μια μετά την άλλη. Εξοικονομείται όμως πολύς χρόνος, άρα και πόροι αφού, αφού μέχρι η ομάδα προγραμματισμού να φτάσει στο στάδιο της έκδοσης, η ομάδα δοκίμων έχει ήδη ετοιμάσει το σχέδιο δοκιμών της. Έχει αρκετά ακόμα πλεονεκτήματα όπως την έναρξη των δοκιμών από τα αρχικά στάδια της αναπτυξης του προϊόντος έως και το τέλος της, δίνοντας μεγαλύτερο ποσοστό επιτυχίας και ένα πιο ολοκληρωμένο συνολικά προϊόν.

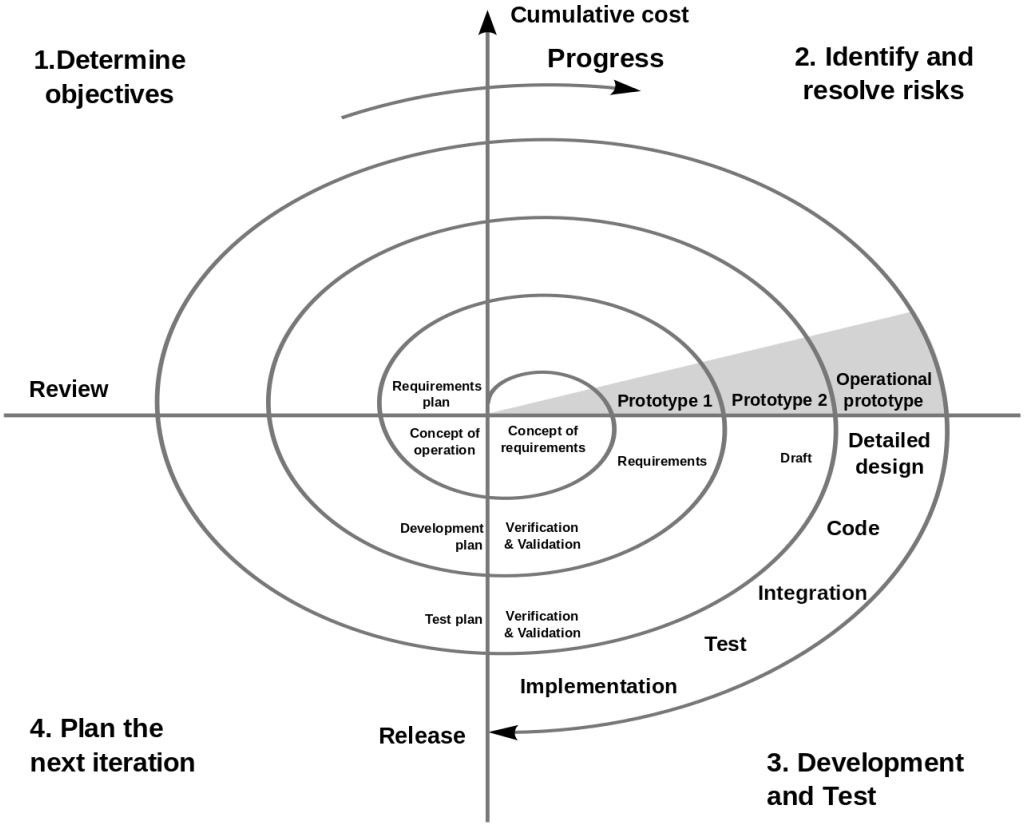
Στα μειονεκτήματα αυτού του μοντέλου είναι ο υψηλότερος κίνδυνος αλλά και ο υψηλότερος αρχικός προϋπολογισμός, αφού και η ομάδα προγραμματισμού και η ομάδα δοκιμών ξεκινάνε να εργάζονται από το πρώτο κιόλας στάδιο. Επίσης αν χρειαστεί κάποια αλλαγή στις απαιτήσεις, χρειάζεται πολύς χρόνος προσαρμογής καθώς απαιτείται εκ νέου πολύ τεκμηρίωση (documentation) και δοκιμή. Ακριβώς επειδή δίνει πολύ έμφαση στην τεκμηρίωση, το έργο μπορεί να εξαρτηθεί υπερβολικά από αυτή και να αποκλίνει τελικά της πραγματικής υλοποίησης.

### 2.3.4 Spiral Model

Το μοντέλο Spiral, που έκανε την εμφάνιση του το 1988 από τον Barry Boehm, αποτελεί μια προηγμένη μέθοδο ανάπτυξης που ενσωματώνει τα θετικά στοιχεία του μοντέλου Waterfall και του μοντέλου Iterative [21][22]. Το σπειροειδές μοντέλο συνδυάζει το γραμμικό μοντέλο, που χωρίζει τη διαδικασία ανάπτυξης σε μια σειρά από διαδοχικές φάσεις και κάθε φάση ολοκληρώνεται πριν ξεκινήσει η επόμενη και το επαναληπτικό μοντέλο, όπου χωρίζει τη διαδικασία ανάπτυξης σε μια σειρά από επαναλήψεις και κάθε επανάληψη περιλαμβάνει ένα μικρό σύνολο εργασιών που ολοκληρώνονται πριν ξεκινήσει η επόμενη επανάληψη. Αυτή η επαναλαμβανόμενη διαδικασία ανάπτυξης, επιτρέπει την προσαρμογή στις μεταβαλλόμενες απαιτήσεις, με κάθε επανάληψη να ονομάζεται "Σπείρα". Το όνομα προκύπτει από το γεγονός ότι η γραφική αναπαράσταση του μοντέλου, μοιάζει με σπείρα με πολλές στροφές, όπου ο ακριβής αριθμός των στροφών της σπείρας είναι άγνωστος και μπορεί να διαφέρει από έργο σε έργο. Κάθε στροφή της σπείρας ονομάζεται Φάση της διαδικασίας ανάπτυξης λογισμικού και αντιπροσωπεύει έναν πλήρη κύκλο ανάπτυξης λογισμικού, από την συλλογή και ανάλυση απαιτήσεων έως το σχεδιασμό, την υλοποίηση, τη δοκιμή και τη συντήρηση.

Ο ακριβής αριθμός των φάσεων που απαιτούνται για την ανάπτυξη του προϊόντος μπορεί να ποικίλλει κυρίως επειδή το μοντέλο αυτό δίνει έμφαση στον κίνδυνο, δεδομένου κιόλας ότι ο Project manager προσδιορίζει δυναμικά τον αριθμό των φάσεων, οπότε ο Project manager έχει σημαντικό ρόλο στην ανάπτυξη ενός προϊόντος στο μοντέλο Spiral. Κάθε σπείρα περιλαμβάνει τέσσερις κύριες φάσεις: Σχεδιασμό, Ανάλυση Κινδύνου, Υλοποίηση, Αξιολόγηση.

Στην αρχική φάση του Spiral, καθορίζονται οι αρχικές απαιτήσεις και πραγματοποιείται ένας αρχικός σχεδιασμός. Στη συνέχεια, γίνεται η ανάλυση κινδύνων, προσδιορίζονται οι κρίσιμες περιοχές του συστήματος και προτείνονται πιθανές λύσεις για την καλύτερη προσέγγιση. Στη φάση της υλοποίησης, ξεκινάει ο πραγματικός προγραμματισμός του συστήματος. Μετά τη φάση της υλοποίησης, η επόμενη στροφή ξεκινάει με μια νέα φάση σχεδιασμού, η μια νέα σπείρα, με βάση τα αποτελέσματα της αξιολόγησης.




*Σχήμα 5. Γραφική Αναπαράσταση Μοντέλου Spiral (https://en.wikipedia.org/wiki/Spiral_model)*

Στο μοντέλο Spiral σε μορφή διαγράμματος, το κόστος του project είναι ανάλογο της ακτίνας της σπείρας σε οποιοδήποτε σημείο και η γωνία που σχηματίζεται αντιπροσωπεύει την πρόοδο που έχει σημειωθεί μέχρι στιγμής στην τρέχουσα φάση.
Κάθε φάση του μοντέλου Spiral είναι χωρισμένη σε τέσσερις κατηγορίες, όπως φαίνεται στο παραπάνω διάγραμμα. Οι λειτουργίες αυτών των τεσσάρων κατηγοριών είναι οι εξής:

1. Καθορισμός στόχων και εναλλακτικών λύσεων : Καθορίζονται και συγκεντρώνονται οι απαιτήσεις των πελατών.

2. Αναγνώριση και επίλυση κινδύνων: Αξιολογούνται όλες οι πιθανές λύσεις και αναγνωρίζονται οι κίνδυνοι που συνδέονται με αυτές τις λύσεις και επιλύονται μέσω της καλύτερης δυνατής στρατηγικής, για να επιλεγεί η καλύτερη δυνατή λύση.

3. Υλοποίηση και δοκιμή νέας έκδοσης : Αναπτύσσεται η νέα έκδοση της εφαρμογής, βάσει των απαιτήσεων που συλλέχθηκαν στην προηγούμενη επανάληψη, με όσο το δυνατόν καλύτερη λύση και επαληθεύεται η ορθότητα των αλλαγών.

4. Ανασκόπηση και σχεδιασμός για την επόμενη φάση : Οι πελάτες αξιολογούν την μέχρι τώρα ανεπτυγμένη έκδοση του λογισμικού, προκειμένου να δουν αν τους ικανοποιεί, η επανάληψη τελειώνει και αν κριθεί απαραίτητο ξεκινάει η φαση σχεδιασμού για την επόμενη επανάληψη

Ένα από τα σημαντικότερα χαρακτηριστικά του μοντέλου Spiral είναι η διαρκής αξιολόγηση και αναθεώρηση των κινδύνων κατά τη διάρκεια της ανάπτυξης. Ενθαρρύνεται η επικοινωνία μεταξύ πελάτη και ομάδων ανάπτυξης, βοηθώντας στη βελτίωση της κατανόησης των απαιτήσεων, επιτρέποντας στις ομάδες να προβλέπουν πιθανά προβλήματα και να λαμβάνουν μέτρα πρόληψης, μειώνοντας έτσι τον κίνδυνο αποτυχίας του έργου συνολικά. Για αυτό το λόγο χρησιμοποιείται συχνότερα σε πιο μεγάλα και σύνθετα έργα αναπτυξης λογισμικού καθώς επιτρέπει μια πιο ευέλικτη και προσαρμοστική προσέγγιση δεκτική στις αλλαγές, με υψηλοτέρα ίσως επίπεδα κινδύνου, αλλά από την άλλη υψηλότερες απαιτήσεις ποιότητας .

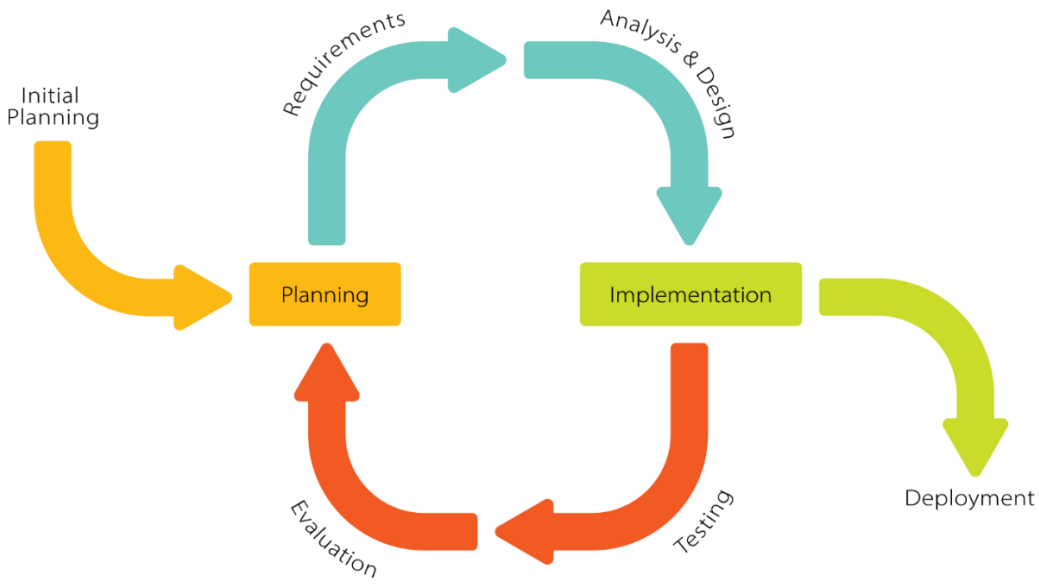
Από την άλλη, το μοντέλο spiral θεωρείται από τα πιο περίπλοκα μοντέλα, με σημαντικότερη να είναι η δυσκολία διαχείρισης χρόνου και πόρων λόγω του αγνώστου αριθμού φάσεων στις αρχές του έργου, που μπορεί να αποβεί δαπανηρό τελικά και ιδίως για μικρότερα έργα. Το μοντέλο αυτό βασίζεται έντονα στην ανάλυση του ρίσκου, γεγονός που απαιτεί πολύ καλούς ειδικούς, προκειμένου να μην αποτύχει το συνολικό έργο, οπότε εξαρτάται και από την εκάστοτε ομάδα [30].

Συνοψίζοντας, το μοντέλο Spiral είναι ένα πιο ευέλικτο και προσαρμοστικό μοντέλο ανάπτυξης λογισμικού που μπορεί να χρησιμοποιηθεί για έργα με υψηλό επίπεδο αβεβαιότητας ή κινδύνου. Ενθαρρύνει την επικοινωνία, τη συνεργασία και την έγκαιρη λήψη αποφάσεων. Το μοντέλο αυτό μπορεί να είναι πιο ακριβό και χρονοβόρο από άλλα μοντέλα ανάπτυξης λογισμικού, αλλά μπορεί να βοηθήσει στη μείωση του κινδύνου αποτυχίας του έργου και στη βελτίωση της ποιότητας του τελικού προϊόντος. Περιλαμβάνει τα πιο γνωστά μοντέλα του SDLC, όπως τα μοντέλα Waterfall, Spiral και Iterative.

### 2.3.5 Iterative Model

Το επαναληπτικό μοντέλο θυμίζει αρκετά το μοντέλο Agile, αφού και τα δυο χρησιμοποιούν επαναληπτικά sprints. Κάθε επανάληψη sprint περιέχει τα στάδια Planning, Design, Development και Testing που σε γενικές γραμμές και σε αυτό το μοντέλο συμβαίνουν οι ενέργειες που έχουν ήδη αναφερθεί. Το μοντέλο Iterative εστιάζει σε μια πιο απλοποιημένη προσέγγιση στην αρχή της υλοποίησης, η οποία σταδιακά αποκτά μεγαλύτερη πολυπλοκότητα και βελτιώνεται επαναληπτικά μέχρι το πλήρες σύστημα να είναι έτοιμο για ανάπτυξη [31][32].

Έτσι, στην αρχή δεν εφαρμόζεται όλο το σύνολο των απαιτήσεων, αλλά ένα κομμάτι αυτών, ξεκινώντας με τον καθορισμό και την εφαρμογή κάποιων από τις απαιτήσεις. Έπειτα, ελέγχονται για προσδιορισμό άλλων απαιτήσεων που θα προκύψουν , οι οποίες προστίθενται στην εφαρμογή σε κάθε επανάληψη και σταδιακά δημιουργείται μια πλήρως αναπτυγμένη εκδοχή του λογισμικού.




*Σχήμα 6. Iterative Model (https://en.wikipedia.org/wiki/Iterative_and_incremental_development)*

**Πλεονεκτήματα :**

- **Εύκολη διαχείριση :** Οι προγραμματιστές μπορούν να κάνουν πιο εύκολα αλλαγές αφού η διαδικασία αναπτυξης είναι διαιρεμένη σε μικρότερα πιο διαχειρίσιμα κομμάτια και έτσι μειώνεται και το κόστος των αλλαγών που μπορεί να προκύψουν.

- **Ευελιξία:** Το μοντέλο αυτό είναι εξαιρετικά ευέλικτο, δίνοντας τη δυνατότητα προσαρμογής των προγραμματιστών στις αλλαγές που μπορεί να προκύψουν από τους χρήστες κατά τη διάρκεια της ανάπτυξης, οπότε είναι μια καλή επιλογή για περίπλοκα έργα ή για έργα με πιο απροσδιόριστες απαιτήσεις.

- **Επαναπροσδιορισμός:** Η δυνατότητα επανεκτίμησης των απαιτήσεων και του σχεδιασμού μετά από κάθε επανάληψη βοηθάει στη βελτίωση του τελικού προϊόντος.

- **Διαρκής βελτίωση:** Η δυνατότητα ενσωμάτωσης αναδυόμενων απαιτήσεων στις επόμενες επαναλήψεις οδηγεί σε διαρκή βελτίωση του λογισμικού.

**Μειονεκτήματα :**

- **Δυσκολία στον έλεγχο του χρονοδιαγράμματος:** Η ανεπίσημη φύση του μοντέλου μπορεί να καθυστερήσει τον προγραμματισμένο χρόνο παράδοσης, καθώς η συνεχής επανάληψη μπορεί να παρατείνει τη διάρκεια της ανάπτυξης.

- **Πιθανότητα προστασίας στο σφάλμα:** Κάθε επανάληψη μπορεί να εισάγει νέα σφάλματα στο λογισμικό, ενώ τα προβλήματα που αναδύονται μετά από κάθε επανάληψη μπορεί να είναι δύσκολο να εντοπιστούν. Λόγω του ότι δεν συγκεντρώνονται όλες οι απαιτήσεις εξαρχής μπορεί να προκύψουν ζητήματα αρχιτεκτονικής ή σχεδιασμού και να καθυστερήσουν ή να κάνουν αβέβαιο το τέλος του Project.

- **Αύξηση του κόστους:** Η επαναληπτική φύση του μοντέλου μπορεί να αυξήσει το κόστος της ανάπτυξης λόγω της ανάγκης για συνεχή αναθεώρηση και επανεκτίμηση, αλλά και απαιτούνται υψηλοί πόροι για την ανάλυση κίνδυνου.

Συνοψίζοντας, το Iterative Model έχει εφαρμογή σε έργα που οι απαιτήσεις δεν είναι, γενικώς ή από την αρχή, ξεκάθαρες ή συνεχώς αλλάζουν κατά την διάρκεια της αναπτυξης και που έχουν ανάγκη από ευελιξία σε τροποποιήσεις. Είναι κατάλληλο για έργα όπου υπάρχει ανάγκη συχνής αναθεώρησης και αξιολόγησης της προόδου και συμμετοχής από τους ενδιαφερομένους. Σε γενικές γραμμές, το επαναληπτικό μοντέλο είναι κατάλληλο για περιπτώσεις όπου η αβεβαιότητα, οι αλλαγές και η δυναμικότητα είναι κυρίαρχες, και όπου η συνεχής προσαρμογή και βελτίωση είναι ουσιώδεις για την επίτευξη των στόχων του έργου.

# *Κεφάλαιο 3: Σχεδιασμός και ανάπτυξη συστήματος*

Το τρίτο κεφάλαιο αποτελεί τον πυρήνα της εργασίας, όπου θα εξεταστούν εκτενώς οι βασικές φάσεις του σχεδιασμού και της ανάπτυξης του συστήματος. Κατά τη διάρκεια αυτών των φάσεων, οι σχεδιαστικές αποφάσεις και οι τεχνικές επιλογές λαμβάνουν χώρα, προσαρμόζοντας τις απαιτήσεις σε μια λειτουργική αρχιτεκτονική και μετατρέποντάς τις σε πραγματικότητα. Οι πληροφορίες προκειμένου να δομηθεί το κεφάλαιο 3 αντλήθηκαν από την αναφορά [33].

Το κεφάλαιο ξεκινά με την περιγραφή του συστήματος, όπου γίνεται η ανάλυση των ζητούμενων στόχων και του τι θα κάνει η εφαρμογή καθώς και μια ανάλυση των ρόλων των χρηστών στο σύστημα.

Στη συνέχεια, ακολουθεί η καταγραφή των λειτουργικών και μη λειτουργικών απαιτήσεων που πρέπει να πληροί το σύστημα. Η εμβάθυνση σε αυτό το στάδιο βοηθά στο να διασφαλιστεί ότι οι ανάγκες των χρηστών και του περιβάλλοντος λαμβάνονται υπόψιν. Αυτή η φάση επικεντρώνεται στον σχεδιασμό της δομής και των συνιστωσών του συστήματος, προκειμένου να επιτευχθεί αποδοτική και αξιόπιστη λειτουργία.

Επίσης, θα γίνει αναφορά στις περιπτώσεις χρήσης του συστήματος όπου θα αναλυθούν όλα τα σενάρια διεξοδικά για κάθε χρήστη μέσω των αντίστοιχων πινάκων.

Ένα σημαντικό εργαλείο στον σχεδιασμό του συστήματος αποτελεί το σχεσιακό διάγραμμα βάσης δεδομένων (ERD), το οποίο παρουσιάζει τη δομή της βάσης δεδομένων και τις σχέσεις μεταξύ των διαφόρων οντοτήτων. Μέσω αυτού του διαγράμματος, επιτυγχάνεται η βελτιστοποίηση της αποθήκευσης και διαχείρισης των δεδομένων, αλλά και η εύκολη κατανόηση της βάσης σε οπτική μορφή.

Τέλος, θα γίνει αναφορά στις τεχνολογίες που επιλέχτηκαν για την εφαρμογή, αλλά και στις δυνατότητες και τα πλεονέκτημα που προσφέρουν. Αυτό το κεφάλαιο αποτελεί το σημαντικό βήμα προς την πραγματοποίηση των απαιτήσεων, προετοιμάζοντας το έδαφος για την περαιτέρω ανάπτυξη, υλοποίηση και δοκιμή του συστήματος.

## *3.1 Περιγραφή Συστήματος, Χρήστες (System Analysis,User Roles)*

Η παρούσα εφαρμογή αναπτύχθηκε με στόχο να παρέχει μια σειρά λειτουργιών για τη διαχείριση έργων σύμφωνα με την διαδικασία SDLC που αναφέρθηκε παραπάνω αλλά και του προσωπικού που το απαρτίζουν, με έμφαση στην απλότητα και την χρηστικότητα. Η ανάπτυξη τέτοιων λειτουργιών απαιτεί να εντοπιστούν και να εξεταστούν οι υπάρχουσες ανάγκες και να οριστούν οι διαφορετικοί ρόλοι των χρηστών, η συμπεριφορά τους σε διαφορετικές περιπτώσεις χρήσης και οι απαιτήσεις όλου του συστήματος που θα πρέπει να ικανοποιούνται τελικά.

Η εφαρμογή χωρίζεται, χρονολογικά αλλά και χωρικά σε τέσσερις μεγάλες κατηγορίες: Επαλήθευση Χρηστών, Καταχώρηση Πληροφοριών, Τροποποιήσεις, Προβολή Δεδομένων.

Η επαλήθευση των χρηστών αναφέρεται σε όλες τις ενέργειες που κάνει ο χρήστης προκειμένου να συνδεθεί με κάποιο ρόλο στην εφαρμογή και να του δοθεί πρόσβαση και δικαιώματα χρήσης σε ορισμένες από τις λειτουργίες.

Η καταχώρηση πληροφοριών αναφέρεται σε όλες εκείνες τις λειτουργίες με τις οποίες ο χρήστης μπορεί να αρχικοποιήσει οποιαδήποτε πληροφορία στο σύστημα, οποιαδήποτε εισροή και είναι ίσως η πιο σημαντική κατηγορία της συγκεκριμένης εφαρμογής. Έχει δοθεί μεγάλη προσοχή στην υλοποίηση αυτού του κομματιού, προκειμένου οι αναγκαίες πληροφορίες να συλλέγονται σωστά και να αποθηκεύονται στην συνέχεια στο σύστημα. Κυρίος υπεύθυνος της κατηγορίας Καταχώρησης Πληροφοριών είναι ο διαχειριστής, αφού το μεγαλύτερο μέρος όλων των δεδομένων που παρέχονται στην εφαρμογή καταχωρούνται από τον ίδιο, από τα στοιχεία του κάθε πρότζεκτ μέχρι και τα προσωπικά στοιχεία των υπαλλήλων.

Η κατηγορία της τροποποίησης αναφέρεται στις λειτουργίες στις οποίες γίνεται επεξεργασία ή αλλαγή ήδη υπαρχόντων δεδομένων. Βασικές λειτουργίες είναι η επεξεργασία υπαλλήλου και η παρακολούθηση της προόδου που θα αναλυθεί και παρακάτω.

Η κατηγορία της προβολής δεδομένων έχει να κάνει με όλες τις λειτουργίες οι οποίες εξυπηρετούν τον χρήστη ως προς την προβολή όλων αυτών των στοιχείων, που δόθηκαν στο σύστημα, με οργανωμένο και ανάλογο για την χρήση τρόπο.

Αυτές οι τέσσερις κατηγορίες μπορούν να συνδυαστούν και ο συνδυασμός αυτών αφορά την σύνθεση των Project με Employees και την παρακολούθηση αυτών προκειμένου να δημιουργηθεί μια πλατφόρμα που θα βοηθήσει τον χρήστη να κατανοήσει τη διαδικασία Ανάπτυξης Λογισμικού (SDLC) στην πράξη και να παρέχει δυνατότητες διαχείρισης και προβολής πληροφοριών προς όλους τους εμπλεκομένους στην ανάληψη ενός προγραμματιστικού έργου με διαφορετικές απαιτήσεις και διαφορετικά χρονικά πλαίσια.

**Ρολόι Χρηστών Συστήματος**

Το πρώτο βήμα στην σχεδίαση της εφαρμογής, είναι η αναγνώριση των χρηστών που προβλέπεται να χρησιμοποιήσουν την εφαρμογή, καθώς και οι λειτουργίες που αναμένονται να εξυπηρετήσουν κάθε χρήστη. Τρεις διαφορετικοί ρόλοι χρηστών ορίζονται : διαχειριστής (admin), υπεύθυνοι έργου (managers) και υπάλληλοι (employees) .

Όπως σε κάθε υπηρεσία, έτσι και στην παρούσα εφαρμογή, ο χρήστης με το ρόλο του διαχειριστή, έχει πρόσβαση σχεδόν σε κάθε λειτουργία του συστήματος, σε σχέση με άλλους χρήστες με

διαφορετικούς ρόλους, που οι λειτουργίες τους περιορίζονται. Ο αριθμός των διαχειριστών χρηστών περιορίζεται σε έναν για λόγους ασφαλείας και απλότητας. Κυρίες λειτουργίες του διαχειριστή είναι η αρχικοποίηση και η παρακολούθηση της ομαλής διεξαγωγής των έργων, της προσθήκης εργαζομένων στο σύστημα και η επίβλεψη των πληρωμών τους. Να σημειωθεί ότι μόνο σε έναν μικρό αριθμό λειτουργιών δεν έχει πρόσβαση ο διαχειριστής, καθώς δεν τον αφορούν έμμεσα, αλλά θα αναλυθούν όλες οι λειτουργίες εκτενέστερα στην συνέχεια.

Οι υπεύθυνοι έργων (managers) αποτελούν ξεχωριστό κομμάτι και ταυτόχρονα μέρος των υπαλλήλων και διαφοροποιούνται σε σχέση με αυτούς, καθώς επιλέγονται από τους διαχειριστές κατά την διάρκεια της αρχικοποίησης ως υπεύθυνοι συγκεκριμένων έργων. Ως εκ τούτου διαφοροποιούνται και οι δυνατότητες που τους παρέχονται, με σημαντικότερη αυτή της παρακολούθησης και της ανατροφοδότησης της πορείας του έργου στον διαχειριστή και της επιλογής κατάλληλων υπαλλήλων για το εκάστοτε έργο.

Οι υπάλληλοι (employees) είναι άτομα που απασχολούνται από την επιχείρηση και συνεπώς και από την πλατφόρμα και έχουν προσληφθεί και εκχωρηθεί στο σύστημα από τον διαχειριστή. Όπως αναφέρθηκε παραπάνω, μερικοί από αυτούς ενδεχομένως να έχουν και δεύτερο ρόλο, αυτόν του manager, αλλά στην περίπτωση που μόνο συμμετέχουν σε project, η πρόσβαση τους περιορίζεται στην προβολή των λεπτομερειών των project στα οποία συμμετέχουν και των πληρωμών που έχουν πραγματοποιηθεί η θα πραγματοποιηθούν μελλοντικά και τους αφορούν.

Είναι σημαντικό να σημειωθεί ότι αυτές οι λειτουργίες αποτελούν απλά τη βάση για την ανάπτυξη περαιτέρω χαρακτηριστικών και προσθηκών στο μέλλον, ώστε να ανταποκρίνονται στις εξελισσόμενες ανάγκες των χρηστών και της αγοράς και να προσφέρουν ακόμη περισσότερη ευελιξία και αποδοτικότητα.

## *3.2 Απαιτήσεις και Λειτουργίες Συστήματος*

Το κεφάλαιο των Απαιτήσεων Συστήματος αποτελεί θεμέλιο για τον σχεδιασμό και την ανάπτυξη κάθε πληροφοριακού συστήματος καθώς ορίζει πώς το πληροφοριακό σύστημα πρέπει να λειτουργεί, επιτυγχάνοντας τον στόχο του και ικανοποιώντας τις ανάγκες των χρηστών. Εδώ αναδεικνύονται όλες οι λειτουργίες που πρέπει το σύστημα να περιλαμβάνει για κάθε χρήστη και που οδηγούν σε μια πλατφόρμα που απλοποιεί την διαδικασία της ανάπτυξης και αναλύονται όλες οι περιπτώσεις χρήσης για κάθε πιθανό σενάριο χρήσης της εφαρμογής.

### <u>Σύνδεση Χρήστη στην πλατφόρμα</u>

Η σύνδεση του χρήστη στην πλατφόρμα, γίνεται με την χρήση των διαπιστευτηρίων του, όπως σε κάθε σύγχρονη εφαρμογή, δηλαδή το email του χρήστη (Email) και τον κωδικό πρόσβασης (Password). Στο κομμάτι της σύνδεσης, μαζί με τα διαπιστευτήρια, ο χρήστης θα πρέπει να επιλέξει και τον ρόλο, που θέλει να πάρει κατά την σύνδεση του στο σύστημα μέσω των επιλογών στην καρτέλα σύνδεσης. Ο Admin όπως έχει ήδη αναφερθεί είναι ήδη καταχωρημένος εξ ορισμού από το σύστημα. Προκειμένου οι σύνδεση των υπαλλήλων η των managers να είναι επιτυχής, ο Admin θα πρέπει να έχει καταχωρήσει τους αντίστοιχους υπαλλήλους στο παρελθόν. Όπως έχει επίσης αναφερθεί, η ύπαρξη του manager είναι και αυτή άμεσα συνδεδεμένη με τον διαχειριστή της πλατφόρμας, αφού και σε αυτή την περίπτωση ο Admin είναι υπεύθυνος για τον ορισμό και την δημιουργία manager. Επομένως το σύστημα δεν έχει και σελίδα εγγραφής (Register page) για

καινούργιους χρήστες αφού όλα τα employees accounts δημιουργούνται και παρέχονται ήδη από τον Admin για κάθε χρήστη αυτόματα με την εισαγωγή του στο σύστημα.

## Καταχώρηση Υπάλληλων

Η καταχώρηση υπαλλήλων στο σύστημα, είναι η πρωτη σημαντικότερη λειτουργία που πρέπει να προσφέρει η εφαρμογή και αφορά τον Admin και μόνο. Αφού επιλέξει την καταχώρηση ενός υπαλλήλου, ο Admin συμπληρώνει τα απαραίτητα στοιχεία που ζητούνται από το σύστημα , όπως το όνομα του, τον κωδικό πρόσβασης του, τα μισθοδοτικά του στοιχεία κλπ. Τα στοιχεία αυτά θα χρησιμοποιηθούν σε διαφορετικά σημεία της εφαρμογής και με το πέρας της καταχώρησης ενός υπαλλήλου, δημιουργείται ο λογαριασμός του. Πλέον και έπειτα από συνεννόηση με τον διαχειριστή, για να ενημερωθεί για τον κωδικό πρόσβασης, ο υπάλληλος έχει την δυνατότητα να συνδεθεί στην πλατφόρμα με δικαιώματα υπαλλήλου. Επίσης με την νέα καταχώρηση του υπαλλήλου στο σύστημα, αυτός θα είναι ορατός πλέον στην σελίδα των υπαλλήλων και θα είναι διαθέσιμος να του οριστούν καθήκοντα σε ένα Project, διαδικασία που θα αναλυθεί παρακάτω.

## Καταχώρηση Project

Η επόμενη πιο σημαντική λειτουργία σε μια τέτοια εφαρμογή και ουσιαστικά και ο κύριος λόγος δημιουργίας της ιδίας της εφαρμογής είναι προφανώς, η καταχώρηση νέων πρότζεκτ στην πλατφόρμα. Κάθε πρότζεκτ πρέπει να είναι σωστά ορισμένο ως προς τις φάσεις του και τους εργαζομένους που απασχολούνται σε αυτό. Κατά την αρχικοποίηση του πρότζεκτ ο χρήστης πρέπει να είναι σε θέση να επιλέγει τις φάσεις που το απαρτίζουν, να ορίζει ποτέ ξεκινούν, τον προϋπολογισμό τους και ενδεχομένως κάποια tasks. Ο τρόπος που ο χρήστης επιλεγεί τις φάσεις θα πρέπει να είναι τέτοιος, που να είναι σε θέση να επιλέξει το μοντέλο SDLC που τον εξυπηρετεί να χρησιμοποιήσει. Αυτό σημαίνει να μπορεί να ορίσει επικαλυπτόμενες ημερομηνίες στις φάσεις, η επαναλαμβανόμενα σετ φάσεων, όπως θα χρειαζόταν για παράδειγμα στο μοντέλο Spiral. Οι φάσεις θα πρέπει να είναι σύμφωνες χρονικά με τα χρονικά όρια του Project συνολικά και τα budget τους να μην υπερβαίνουν το προβλεπόμενο budget όλου του έργου. Για να είναι λειτουργική η εφαρμογή, ο Manager θα πρέπει να μπορεί να επιλέγει και να ορίζει υπάλληλους για κάθε φαση. Σημασία στην επιλογή έχουν και οι διαθέσιμοι πόροι αλλά και οι διαθέσιμες ημερομηνίες των φάσεων του εκάστοτε Project οπότε η εφαρμογή θα πρέπει να προσφέρει με όσο το δυνατόν πιο ξεκάθαρο τρόπο την όλη διαδικασία. Για αυτό το λόγο, η αρχικοποίηση ενός project στην συγκεκριμένη εφαρμογή έχει χωριστεί σε τρεις ειδικότερες φάσεις, μοιράζοντας την διαδικασία σε τρεις κατηγορίες ανάλογα με το τι πραγματεύεται η καθεμιά και είναι οι εξής : Προσθήκη Project ( 1η φαση), Προσθήκη Φάσεων και Ορισμός Manager ( 2η φαση) και Προσθήκη Υπάλληλου για κάθε φαση (3η Φαση). Κάθε φαση της τιμής status που ορίζει την τρέχουσα κατάσταση του Project, η πλατφόρμα προβάλλει στον εκάστοτε χρήστη τις ανάλογες ενέργειες που μπορεί να εκτελέσει την προκείμενη στιγμή. Η τρέχουσα κατάσταση δεν αναφέρεται μόνο στην αρχικοποίηση του Project, όσο και μετά το πέρας αυτής, όπως θα αναλυθεί παρακάτω και αφορά το status κάθε Project. Παρακάτω θα αναλυθούν οι απαιτήσεις που πρέπει να πληροί η κάθε φάση αρχικοποίησης, ώστε όλες μαζί να ικανοποιούν συνολικά τη δομή ενός νέου Project βασιζόμενες στις ανάγκες και τις απαιτήσεις μιας πραγματικής εταιρείας Software.

## 1η Φαση Αρχικοποίησης – Προσθήκη Project

Τορ πρώτο κομμάτι της αρχικοποίησης του Project, αφορά αποκλειστικά και μόνο τον διαχειριστή, αφού μόνο αυτός έχει πρόσβαση σε αυτή την λειτουργία και έχει να κάνει με τον καθορισμό των βασικότερων στοιχείων του Project. Στοιχεία όπως το όνομα του Project, η διάρκεια του (σε μήνες) και ο συνολικός του προϋπολογισμός, έπειτα από την επιλογή "Add New Project", συμπληρώνονται από τον διαχειριστή, για να εκχωρήσει στο σύστημα μια πρώτη "εικόνα" με τα βασικά για αρχή, δεδομένα που θα χρειαστούν για να δομήσουν ολόκληρο το Project στη δεύτερη φάση αρχικοποίησης.

Η λειτουργία αυτή είναι στοχευμένα απλή στην χρήση της αφού ο διαχειριστής δηλώνει μόνο τις πολύ βασικές πληροφορίες και δεν απαιτεί ιδιαίτερη προσοχή. Με την ολοκλήρωση της πρώτης φάσης, οι δοθείσες πληροφορίες καταγράφονται στο σύστημα και ο διαχειριστής μπορεί πλέον να δει το συγκεκριμένο Project στην λίστα με τα συνολικά Projects. Να σημειωθεί επίσης πως αφού ολοκληρωθεί η 1η φαση, τότε ενεργοποιείται μέσω της πλατφόρμας, η επιλογή αρχικοποίησης της 2ης φάσης που θα αναλυθεί παρακάτω.

## 2η Φαση Αρχικοποίησης – Προσθήκη Φάσεων και Manager

Το δευτερο κομμάτι της αρχικοποίησης του Project, αφορά επίσης αποκλειστικά τον διαχειριστή, αφού μόνο αυτός έχει πρόσβαση σε αυτή την λειτουργία και έχει να κάνει με τον καθορισμό των φάσεων του εκάστοτε Project και την επιλογή Manager. Η λειτουργία αυτή είναι πιο περίπλοκη, αφού σε αυτή ορίζονται ο αριθμός των φάσεων, η επιλογή του κατάλληλου SDLC μοντέλου, η καταχώρηση των απαραίτητων πληροφοριών για κάθε φάση που επιλεχθεί αλλά και του κατάλληλου manager για το έργο. Για αυτό τον λόγο και για να επιτευχθεί η απλότητα και σε αυτή τη φαση, η αρχικοποίηση έχει χωριστεί σε τρία στάδια. Στο πρώτο στάδιο γίνεται η επιλογή των φάσεων, στο δευτερο στάδιο γίνεται η αρχικοποίηση των δεδομένων των φάσεων και στο τρίτο στάδιο γίνεται η επιλογή του manager.

### *1) Επιλογή Φάσεων*

Με την έναρξη της δεύτερης φάσης, αρχικά ο διαχειριστής καλείται να επιλέξει τις απαραίτητες φάσεις που έχουν οριστεί για την ανάληψη και ολοκλήρωση του εκάστοτε Project. Η εφαρμογή του δίνει την δυνατότητα, είτε να επιλέξει από τις ήδη υπάρχουσες φάσεις, που αφορούν τις φάσεις του μοντέλου waterfall, είτε να προσθέσουν τις δικές τους custom φάσεις. Η επιλογή που παρέχεται στον διαχειριστή να προσθέτει την δικιά του φάση εξυπηρετεί την προφανή ανάγκη επιλογής κάποιου άλλου μοντέλου SDLC, όπως για παράδειγμα του μοντέλου Spiral η του μοντέλου Agile, οπότε και οι ήδη υπάρχουσες από το σύστημα φάσεις να μην τον εξυπηρετούν. Ο διαχειριστής μπορεί να δώσει οποιοδήποτε όνομα θέλει στην φάση και να την προσθέσει στην λίστα των επιλεγμένων φάσεων, είτε μαζί με κάποιες από τις ήδη υπάρχουσες φάσεις είτε αποκλειστικά και μόνο με δικές του φάσεις. Δίνεται η ελευθερία πλήρους επιλογής, καθώς αυτό το σημείο για κάθε project μπορεί να είναι πολύ διαφορετικό από ένα άλλο, οπότε η λογική των προεπιλεγμένων

φάσεων, ή συγκεκριμένου μοντέλου, δεν θα ήταν ούτε χρηστική ούτε ευέλικτη. Ο μόνος περιορισμός που συναντά ο διαχειριστής είναι η απαίτηση από το σύστημα ο αριθμός των φάσεων να είναι μεγαλύτερος από τέσσερις και αυτό γιατί ανάληψη έργου με λιγότερες από 4 φάσεις δεν υφίσταται στις σημερινές απαίτησης ανάπτυξης λογισμικού.

### *2) Αρχικοποίηση Φάσεων*

Αμέσως μετά την επιλογή των φάσεων, ο διαχειριστής καλείται να αρχικοποιήσει με δεδομένα την κάθε φάση ξεχωριστά. Εξίσου απαιτητική είναι και αυτή η λειτουργία, καθώς η λεπτομερής και μελετημένη καταχώρηση στοιχείων στις φάσεις απαιτείται για την ορθή διεξαγωγή του Project συνολικά. Οποιοδήποτε λάθος στην καταχώρηση θα έχει σίγουρο αντίκτυπο στην συνέχιση της αρχικοποίησης του Project και τελικά στη λανθασμένη διαχείριση αυτού. Ο λόγος που αυτή η διαδικασία πρέπει να είναι ακριβής ως προς την υλοποίηση της και ως προς το περιεχόμενο της, είναι επειδή σε αυτό το στάδιο ορίζονται η διάρκεια (σε μήνες), το budget και τα tasks αλλά και η ημερομηνία έναρξης της κάθε φάσης.

Η διάρκεια ορίζεται ως ένα υποσύνολο της συνολικής διάρκειας του Project που είχε οριστεί στο πρώτο στάδιο αρχικοποίησης και η ελάχιστη διάρκεια κάθε φάσης είναι ένας μήνας. Να σημειωθεί ότι ο πρώτος περιορισμός που προκύπτει από το σύστημα είναι η κατάλληλη δήλωση μηνών σε κάθε φάση έτσι ώστε η συνολική διάρκεια όλων των φάσεων συνολικά, να είναι σύμφωνη με την ορισμένη διάρκεια του Project.

Ο δεύτερος περιορισμός προκύπτει από το budget που ορίζεται και αυτό ως ένα υποσύνολο του συνολικού προϋπολογισμού του Project που είχε οριστεί στο πρώτο στάδιο αρχικοποίησης. Ελάχιστο budget ανά φάση δεν υπάρχει, αλλά και σε αυτήν την περίπτωση ο συνολικός αριθμός όλων των δοθέντων συγκεντρωτικά budget κάθε φάσης πρέπει να είναι σύμφωνος με τον προϋπολογισμό που είχε οριστεί στο πρότζεκτ στην πρώτη φάση. Να σημειωθεί πως η εφαρμογή ενημερώνει σε πραγματικό χρόνο κατάλληλα τον διαχειριστή, τόσο για το αν έχει υπερβεί την συνολική διάρκεια όσο και τον συνολικό προϋπολογισμό αλλά και για τις εναπομείνασες φάσεις και προϋπολογισμό, προκειμένου να συμφωνούν με τα πρώτα αρχικά δεδομένα του Project.

Όσον αφορά την ημερομηνία έναρξης, είναι ίσως το πιο σημαντικό στοιχείο και αν συνδυαστεί και με τα tasks με συγκεκριμένο τρόπο μπορούν να διαρθρώσουν και να προσομοιώσουν οποιοδήποτε μοντέλο SDLC. Ο διαχειριστής ορίζει την ημερομηνία έναρξης για κάθε φάση, και σε συνδυασμό επίσης με την διάρκεια της φάσης, ορίζεται και η τελική ημερομηνία. Οι ημερομηνίες μεταξύ τους μπορούν είτε να επικαλύπτονται, είτε να τελειώνει η μια και να ξεκινάει η άλλη, ή να τρέχει κάποιο σετ φάσεων παράλληλα και γενικώς υπάρχει ελευθερία στην επιλογή και στην διάρθρωση του Project. Ειδικά όμως εξαιτίας αυτής της ελευθερίας, είναι πολύ εύκολο ο διαχειριστής να μην καθορίσει σωστά τις φάσεις, οπότε στην προκείμενη, η ελευθερία δημιουργεί απαιτήσεις και αυστηρά καθορισμένες φάσεις. Ο τρόπος που θα οριστούν οι ημερομηνίες έναρξης θα καθορίσει το μοντέλο που θα χρησιμοποιηθεί για την ανάπτυξη της εφαρμογής.

Τα tasks αναφέρονται στον αριθμό των tasks που πρέπει να εκτελεστούν για τον τερματισμό κάθε φάσης και εν συνεχεία και του πρότζεκτ και στην ουσία υποδηλώνουν την πρόοδο κάθε φάσης. Να σημειωθεί πως αν ο διαχειριστής θέλει να ακολουθήσει το μοντέλο Agile ή Spiral παραδείγματος χάρη, ο αριθμός των tasks θα δήλωνε τον αριθμό των φάσεων της

μιας επανάληψης στο Agile ή τον αριθμό των φάσεων κάθε σπείρας στο Spiral, προκειμένου να μπορούν να αποδοθούν όλες οι τεχνικές λεπτομέρειες αναλόγως μοντέλου επιλογής.

***3) Επιλογή Manager***

Αφού λοιπόν ο διαχειριστής τελειώσει και με το δευτερο στάδιο της δεύτερης φάσης αρχικοποίησης, καλείται να ορίσει έναν υπάλληλο ως manager του Project. Η ανάγκη ύπαρξης manager σε ένα Project προκύπτει από το γεγονός ότι το Project θα είναι πιο λειτουργικό και θα απελευθερώνει τον Admin από κάποια καθήκοντα όπως την επιλογή των κατάλληλων υπαλλήλων και την επίβλεψη της προόδου του Project. Ούτως η άλλως και σήμερα, οι τωρινές απαιτήσεις των εταιρειών προβλέπουν την θέση του manager και μάλιστα ως πολύ σημαντική, καθώς επεκτείνεται η ιεραρχία, μοιράζονται οι ευθύνες και τα Project γίνονται πιο διαχειρίσιμα. Η επιλογή του manager πέραν δεξιοτήτων εξαρτάται και από τα μισθολογικά του στοιχεία. Για αυτό το λόγο η εφαρμογή προβάλλει μια λίστα με όσους υπάλληλους είναι διαθέσιμοι και δίνει την δυνατότητα επιλογής του πιο κατάλληλου υπαλλήλου. Η πλατφόρμα σε αυτό το σημείο έχει περιορίσει την λίστα με τους υπάλληλους, έτσι ώστε να προβάλλονται μόνο όσοι έχουν προσληφθεί σαν “full time”. Για κάθε υπάλληλο η λίστα δείχνει το όνομα του καθώς και τη μηνιαία αμοιβή του. Αναλόγως την επιλογή του, ο διαχειριστής ενημερώνεται σε πραγματικό χρόνο για τη συνολική αμοιβή του manager για ολόκληρο το Project, προκειμένου να έχει μια εικόνα του υπομέρους του budget που θα μοιραστεί στους υπάλληλους και του υπομέρους budget που θα διατεθεί στο manager, με την συγκεκριμένη επιλογή υπαλλήλου για την αρμόδια θέση. Να σημειωθεί πως, αφού γίνει η επιλογή, η τελική αμοιβή του manager, θα αφαιρεθεί ισόποσα από τους προϋπολογισμούς των φάσεων που έχουν οριστεί.

Με την ολοκλήρωση της δεύτερης φάσης, οι δοθείσες πληροφορίες καταγράφονται στο σύστημα και ο διαχειριστής μπορεί πλέον να δει το ενημερωμένο Project στην λίστα με τα συνολικά Projects. Αφού ολοκληρωθεί η 2η φαση, τότε ενεργοποιείται μέσω της πλατφόρμας, η επιλογή αρχικοποίησης της 3ης φάσης που θα αναλυθεί παρακάτω. Με το πέρας όμως της διαδικασίας αυτής μια άλλη σημαντική λειτουργία είναι διατεθειμένη προς τους υπάλληλους της πλατφόρμας, όπως είχε αναφερθεί και παραπάνω. Σε αυτή την περίπτωση αναφερόμαστε σε αυτούς τους υπάλληλους που έχουν και δικαιώματα manager σε κάποιο Project, εκτός της ιδιότητας τους ως υπάλληλοι. Πλέον αυτοί οι υπάλληλοι μπορούν να συνδεθούν στην πλατφόρμα με τον ρόλο του Manager, δίνοντας τους κάποιες παραπάνω λειτουργίες όσον αφορά το Project.

## 3η Φαση Αρχικοποίησης – Επιλογή Υπαλλήλων για το Project

Όπως αναφέρθηκε παραπάνω, η χρήση manager είναι αναγκαία για να λειτουργήσει το σύστημα με βάση τα σημερινά δεδομένα. Για αυτό το λόγο, τα τρίτο κομμάτι της αρχικοποίησης του Project, το όποιο έχει να κάνει με τον καθορισμό υπαλλήλων στις φάσεις, σε αντίθεση με τις προηγούμενες δυο φάσεις, αφορά αποκλειστικά και μόνο το μάνατζερ. Με βάση τον τρόπο που κατασκευάστηκε το σύστημα, η υποχρέωση-καθήκον της επιλογής προσωπικού για το εκάστοτε Project βαραίνει αποκλειστικά και μόνο τον manager, αφού μονός αυτός έχει πρόσβαση σε αυτή τη λειτουργία. Η λειτουργία αυτή είναι θεωρητικά απλή στην εκτέλεση της και με το πέρας της τερματίζεται συνολικά όλη η διαδικασία αρχικοποίησης. Να σημειωθεί επίσης πως δημιουργούνται και οι

ανάλογες πληρωμές από το σύστημα για τους εκάστοτε υπαλλήλους που επιλέγονται. Ο τρόπος επιλογής θα πρέπει να είναι απλός και να συμφωνεί με τρόπο λειτουργίας του όλου Project. Οι περιορισμοί που υπάρχουν σε αυτή τη φάση, έχουν να κάνουν με τα ποσά πληρωμών αλλά και την ημερομηνία διεξαγωγής της εκάστοτε φάσης.

Οι πληρωμές των υπαλλήλων ορίζονται ως υποσύνολα του συνολικού προϋπολογισμού της εκάστοτε φάσης που είχε οριστεί στο δευτερο στάδιο αρχικοποίησης. Το σύνολο των πληρωμών όλων των επιλεγμένων υπαλλήλων για κάθε φάση πρέπει να είναι σύμφωνο με το προϋπολογισμό που είχε οριστεί στο δεύτερο στάδιο αρχικοποίησης.

Αναφορικά με την ημερομηνία διεξαγωγής της φάσης, το σύστημα έχει δημιουργηθεί με τέτοιο τρόπο ώστε να εξασφαλίσει στον manager, την επιλογή μόνο των υπαλλήλων που δεν εργάζονται σε κάποιο άλλο Project κατά την διάρκεια διεξαγωγής της συγκεκριμένης φάσης, προκειμένου να μην υπάρξουν επιπλοκές.

Η εφαρμογή ενημερώνει σε πραγματικό χρόνο κατάλληλα τον manager, για το συνολικό ποσό πληρωμών, προκειμένου να συμφωνεί και να είναι στα αποδεκτά όρια του προϋπολογισμού της φάσης, όπως ορίστηκε στο δεύτερο στάδιο αρχικοποίησης.

Με την εκτέλεση αυτού του σταδίου, το Project ολοκληρώνεται, ως προς την αρχικοποίηση του και οι εμπλεκόμενοι υπάλληλοι καθώς και ο Admin μπορούν πλέον να δουν το Project στην λίστα με τα Project τους.

## Κατάργηση κατά την Αρχικοποίηση

Αφού αναλύθηκαν τα στάδια αρχικοποίησης ενός πρότζεκτ και ο τρόπος χρήσης τους, γίνεται αντιληπτό, πως λόγω της πολυπλοκότητας κάποιων σημείων στα στάδια αυτά, άρα και ενδεχομένου λάθους, ο Admin πρέπει να έχει τη δυνατότητα κατάργησης ενός Project, σε οποιοδήποτε φαση και αν βρίσκεται, προκειμένου να μπορέσει να το αρχικοποιήσει και να το διορθώσει με τις σωστές πληροφορίες ξανά από την αρχή. Αυτή η λειτουργία αφορά επίσης και τον manager, εφόσον το Project βρίσκεται στο τρίτο στάδιο αρχικοποίησης. Αυτή η λειτουργία προλαμβάνει αστοχίες που μπορεί να προέκυψαν με την καταχώρηση λανθασμένων πληροφοριών και κρίνεται απαραίτητη για την εύρυθμη λειτουργία της πλατφόρμας συνολικά. Τη λειτουργία αυτή ο εκάστοτε χρήστης μπορεί να τη βρει στην λίστα με τα Projects, σαν action button για κάθε Project ξεχωριστά και αν επιλεχθεί η κατάργηση, ο χρήστης μέσω ενός modal επιβεβαιώνει την επιλογή του. Με το πέρας αυτής της λειτουργίας, το Project και όλες οι πληροφορίες, όπως οι φάσεις, οι υπάλληλοι κλπ., διαγράφονται ολοκληρωτικά από την βάση δεδομένων και από τις λίστες με τα Project των χρηστών. Αναφορικά με την διαγραφή, να σημειωθεί πως αυτή η λειτουργία είναι η μόνη που κάνει πλήρη διαγραφή πληροφοριών από το σύστημα και δεν θα ξαναεμφανιστεί ανάλογη λειτουργία στην υπόλοιπη εφαρμογή.

## Τερματισμός κατά την Εκτέλεση

Άλλη μια λειτουργία που είναι απαραίτητη στην εφαρμογή είναι και αυτή που τερματίζει το Project. Δυνατότητα τερματισμού του Project δίνεται μόνο στο Admin και ο τερματισμός μπορεί να αναφέρεται είτε στην επιτυχή ολοκλήρωση του Project, είτε στον πρόωρο τερματισμό του, γεγονός που το καθιστά πλέον μη ενεργό, αρά και μη διαχειρίσιμο. Με το πέρας αυτής της λειτουργίας, το

status του Project τίθεται με την τιμή 4, γεγονός που δηλώνει ότι στην λίστα με τα Projects, το συγκεκριμένο Project θα έχει την δυνατότητα πλέον μόνο να προβληθεί όπως αναφέρθηκε παραπάνω.

## Προβολή όλων των Project

Όπως αναφέρθηκε παραπάνω τα Project θα πρέπει να εμφανίζονται συγκεντρωτικά στο μενού του εκάστοτε χρήστη. Με την μορφή λίστας ο εκάστοτε χρήστης θα μπορεί να δει όλα τα Project η να αναζητήσει ένα συγκεκριμένο Project, προκειμένου να μπορεί να έχει μια συνολική εικόνα των τρεχόντων η των ολοκληρωμένων Projects. Ανάλογα με τον ρόλο που έχει συνδεθεί ο χρήστης θα μπορεί να δει συγκεκριμένα Projects στην λίστα του. Σαν Admin, ο χρήστης θα μπορεί να δει μια λίστα με όλα τα Projects που έχουν εκχωρηθεί στην εφαρμογή, είτε αυτά είναι στην φάση της αρχικοποίησης, είτε αυτά είναι ενεργά, είτε είναι ανενεργά. Η λίστα αυτή δεν θα είναι ίδια για τον Manager και τον υπάλληλο, αφού θα περιέχει μόνο όσα Projects, στα οποία ο χρήστης έχει οριστεί ως manager ή ως υπάλληλος αντίστοιχα. Η λίστα περιέχει πληροφορίες για το κάθε Project, όπως το όνομα του, τη διάρκεια του, τον συνολικό προϋπολογισμό, τα συνολικά tasks των φάσεων καθώς και ένα πεδίο actions. Το πεδίο actions σχετίζεται με τις ενέργειες που μπορεί ο χρήστης να κάνει την δεδομένη για το Project στιγμή, και συνδέεται άμεσα με τα status του.

Το status είναι μια μεταβλητή που ορίζεται στο Project αυτόματα στην καταχώρηση του κατά την πρώτη φάση αρχικοποίησης και αφορά την τρέχουσα κατάσταση του Project. Η χρήση της μεταβλητής από την εφαρμογή υπάρχει καθαρά για σκοπούς χειρισμού και όχι προβολής, οπότε και δεν είναι άμεσα εμφανής στο χρήστη, πάρα μόνο από το πεδίο actions και το τι αυτό εμφανίζει. Το status από 1 έως και 2, αναφέρεται στη δεύτερη και τρίτη φάση αρχικοποίησης, αντίστοιχα. Το status με τιμή 3, υποδηλώνει ότι το Project εκτελείται και το status με τιμή 4, υποδηλώνει ότι το Project έχει τερματίσει. Έτσι όταν το Project status είναι ένα ( 1), το πεδίο tasks της λίστας είναι κενό (αφού αρχικοποιείται στην επόμενη φάση) και στο πεδίο actions εμφανίζονται δύο επιλογές, αυτή της έναρξης της δεύτερης φάσης αρχικοποίησης και αυτή της κατάργησης του Project. Όταν το Project status έχει την τιμή δύο (2), πλέον το πεδίο tasks της λίστας θα δείχνει το συνολικό αριθμό των tasks του Project και στο πεδίο action μαζί με την επιλογή κατάργησης που αναφέρθηκε νωρίτερα, πλέον θα εμφανίζεται η επιλογή έναρξης της τρίτης φάσης αρχικοποίησης αντί αυτής της δεύτερης. Η επιλογή αυτή όμως θα είναι δυνατή μόνο ως προς τον manager, σύμφωνα με τις απαιτήσεις της εφαρμογής, ενώ στην αντίστοιχη λίστα του Admin η λειτουργία αυτή θα είναι απενεργοποιημένη (not clickable). Όταν το Project status έχει την τιμή τρία (3), το πεδίο tasks θα δείχνει, ανάλογα το progress του Project, τις εκτελεσμένες φάσεις σε σχέση με τις συνολικές φάσεις και στο πεδίο actions για τον Admin θα εμφανίζονται οι επιλογές προβολής και τερματισμού του Project, ενώ στις αντίστοιχες λίστες του manager και του υπαλλήλου θα εμφανίζεται μόνο η επιλογή προβολής του Project. Τέλος, όταν το Project status έχει την τιμή 4 (4), το πεδίο tasks θα δείχνει όλα τα tasks τερματισμένα, ή ημιτελή σε περίπτωση προώρου τερματισμού και στο πεδίο action θα εμφανίζεται μόνο η προβολή του Project για όλους πλέον τους χρήστες της εφαρμογής.

## Προβολή όλων των υπαλλήλων

Με παρόμοιο τρόπο θα γίνεται και η προβολή των υπαλλήλων στον Admin. Με την μορφή λίστας ο Admin θα μπορεί να δει όλους τους υπαλλήλους η να αναζητήσει έναν συγκεκριμένο υπάλληλο, προκειμένου να έχει μια συνολική και γρήγορη εικόνα των ενεργών ή μη υπαλλήλων της

πλατφόρμας. Η λίστα πρέπει να προβάλει πληροφορίες για τον κάθε υπάλληλο, όπως το ονοματεπώνυμο του, το email του, την ημερομηνία γέννησης του και τα μισθολογικά του στοιχεία. Όπως και στην λίστα των Projects, έτσι και στην λίστα των υπαλλήλων περιέχεται και ένα πεδίο actions, το όποιο σχετίζεται με τις λειτουργίες που μπορεί ο διαχειριστής να επιλέξει σχετικά με τον εκάστοτε υπάλληλο. Οι λειτουργίες εξ ορισμού είναι τρεις, με πρώτη την προβολή του υπαλλήλου, δεύτερη την τροποποίηση των στοιχείων του και τρίτη την απόλυση-απαλλαγή του υπαλλήλου. Αν ο υπάλληλος είναι μη ενεργός στο σύστημα, τότε το πεδίο actions περιέχει μόνο την προβολή του.

## Επεξεργασία Υπαλλήλων

Όπως αναφέρθηκε παραπάνω, στην λίστα των υπαλλήλων, στο πεδίο actions εμφανίζεται η επιλογή επεξεργασίας του εκάστοτε υπαλλήλου. Επειδή όμως η συγκεκριμένη λειτουργία αφορά εκτός του διαχειριστή και τον υπάλληλο, το σύστημα παρέχει την λειτουργία επεξεργασίας και στο μενού του υπαλλήλου. Στο σύστημα το όποιο έχει αναπτυχθεί, ο χρήστης θα δύναται να αλλάξει τις καταχωρημένες πληροφορίες του υπαλλήλου. Με την πάροδο του χρόνου, στοιχεία όπως ο μισθός ή το συμβόλαιο, ενδέχεται να χρειάζεται να αλλάξουν, αλλά και για διόρθωση λαθών κατά την αρχικοποίηση, όπως επίσης και ο κωδικός πρόσβασης, ενδεχομένως για λόγους ασφαλείας, οπότε και η τροποποίηση στοιχείων πρέπει να προσφέρεται σαν λειτουργία. Επίσης από την πλευρά του υπαλλήλου ως χρήστη, αυτός θα έχει τη δυνατότητα να τροποποιήσει τα προσωπικά του στοιχεία, αλλά και τον κωδικό πρόσβασης του. Πιο συγκεκριμένα, στο κομμάτι της επεξεργασίας, οι δυνατότητες που θα έχει ο Admin θα είναι μόνο να αλλάξει τα μισθολογικά στοιχεία του εκάστοτε υπαλλήλου, ενώ ο υπάλληλος δεν θα μπορεί να τροποποιήσει τα σχετικά αυτά στοιχεία, πάρα μόνο όπως είναι λογικό, μόνο τα πεδία με τα στοιχεία του και τον κωδικό πρόσβασης. Αυτή η λειτουργία δίνει στο σύστημα μια πιο ολοκληρωμένη λύση όσον αφορά τη διαχείριση προσωπικού και κατά συνέπεια των Projects, καθώς οι αλλαγές στα μισθολογικά του στοιχεία, επηρεάζουν τις πληρωμές του και το ρόλο που μπορεί να πάρει στα Projects, αφού για παράδειγμα, όπως είχε αναφερθεί, με την αλλαγή ενός υπαλλήλου σε full time, θα μπορεί να αναλαμβάνει χρέη manager.

## Προεπισκόπηση Υπαλλήλου

Για λογούς χρηστικότητας, μια ακόμα λειτουργία που προστέθηκε στην εφαρμογή είναι η προεπισκόπηση ενός υπαλλήλου. Οι πληρωμές για κάθε υπάλληλο, που όπως αναφέρθηκε νωρίτερα, δημιουργούνται με την ολοκλήρωση της τρίτης φάσης, πρέπει να είναι προσβάσιμες προς προβολή από το σύστημα. Η παρακολούθηση των πληρωμών ενός υπαλλήλου, τόσο από τον διαχειριστή όσο και από τον ίδιο τον υπάλληλο, κρίνεται αναγκαία, καθώς θα παρέχεται και στους δύο, μια καθαρή εικόνα όλων των πληρωμών και όλων των Projects που ο υπάλληλος έχει απασχοληθεί συγκεντρωτικά.

Ο διαχειριστής μέσω της λίστας και από το πεδίο actions που αναφέρθηκε ήδη, θα μπορεί να επιλέξει την ενέργεια προεπισκόπησης, ενώ ο υπάλληλος, θα μπορεί να κάνει την αντίστοιχη επιλογή από την αρχική του σελίδα. Μαζί με τα στοιχεία του υπαλλήλου προβάλλεται και μια λίστα με τις αντίστοιχες πληρωμές, όπου ο χρήστης αρχικά βλέπει τις βασικές λεπτομέρειες, όπως το status της πληρωμής, το συνολικό ποσό πληρωμής αλλά και τις ώρες απασχόλησης του, ενώ στην συνέχεια μπορεί να επιλέξει να δει περισσότερες λεπτομέρειες τις συγκεκριμένης πληρωμής, όπως

την ημερομηνία έναρξης και λήξης της φάσης της πληρωμής, το όνομα του Project κλπ. Να σημειωθεί ότι ο εκάστοτε χρήστης μπορεί να κάνει αναζήτηση οποιασδήποτε πληρωμής, με βάση το όνομα του Project η του τύπου της φάσης αυτής της πληρωμής.

## Αποδέσμευση Υπαλλήλου

Όπως αναλύθηκε η λειτουργία του τερματισμού ενός Project, έτσι και για τον υπάλληλο έχει δοθεί μια παρόμοια λειτουργία προς το διαχειριστή. Ο διαχειριστής έχει τη δυνατότητα οποιαδήποτε στιγμή να αποδεσμεύσει τον υπάλληλο από τα καθήκοντα του, ορίζοντας τον ως πλέον μη ενεργό στο σύστημα. Όταν ένας υπάλληλος έχει οριστεί ως μη ενεργός, σημαίνει είτε την παραίτηση του είτε την απόλυση του και εν συνεχεία δεν θα είναι διαθέσιμος για επιλογή και ανάληψη κάποιου Project κατά την αρχικοποίηση του από τους εκάστοτε managers. Από την λίστα των υπαλλήλων, στο πεδίο actions ο χρήστης θα μπορεί να επιλέξει την συγκεκριμένη ενέργεια και εφόσον την επιλέξει θα επιβεβαιώνει την επιλογή μέσω του modal που προβάλλεται. Να σημειωθεί πως για λόγους ιστορικού και πρόσβασης σε ήδη καταχωρημένες πληροφορίες του, ο υπάλληλος δεν διαγράφεται εντελώς από το σύστημα. Με το πέρας αυτής της λειτουργίας, το state του υπαλλήλου ανανεώνεται και πλέον στην λίστα των εργαζομένων, μπορεί να γίνει μόνο προβολή των ήδη υπαρχόντων στοιχείων του.

## Παρακολούθηση Project

Για να θεωρηθεί πετυχημένη η εκάστοτε εφαρμογή που παράγεται, κάθε σύγχρονη εταιρεία που αναλαμβάνει την διαχείριση Projects, θα πρέπει να δίνει στον χρήστη την δυνατότητα να παρακολουθεί και να ενημερώνεται λεπτομερώς για το Project στο όποιο εργάζεται. Η παρακολούθηση των Projects, είναι ίσως η πιο σημαντική λειτουργία μιας τέτοιας εφαρμογής, αφού όλες αυτές οι πληροφορίες που συλλέχθηκαν κατά τα στάδια αρχικοποίησης προβάλλονται συγκεντρωτικά και με ευκολία στο χρήστη. Η λειτουργία αφορά όλους τους χρήστες του συστήματος, σύμφωνα πάντα με την λίστα των Projects που αφορούν τον καθένα και προκειμένου να είναι σε θέση να κατανοήσουν πλήρως την φύση του Project, ιδανική κρίθηκε η χρήση ανάλογων διαγραμμάτων. Μέσω των διαγραμμάτων δόθηκε ιδιαίτερη έμφαση στη χρονική ανάλυση του Project, αλλά και στο πως είναι δομημένες οι φάσεις και οι υπάλληλοι σε αυτό.

### *Hierarchy Tree*

Το πρώτο διάγραμμα που παρέχεται στον χρήστη είναι ένα διάγραμμα τύπου Hierarchy Tree, όπου χωρίζει και αναλύει όλο το Project σε τρία επίπεδα. Το πρώτο επίπεδο αναφέρεται στον πρώτο κόμβο του διαγράμματος που είναι το ίδιο το Project και πιο συγκεκριμένα το όνομα του και επεκτείνεται στους κόμβους του δευτέρου επιπέδου. Οι κόμβοι του δευτέρου επιπέδου αναφέρονται στον τύπο των φάσεων του συγκεκριμένου Project και κάθε φάση στην συνέχεια επεκτείνεται στους αντίστοιχους υπαλλήλους της, με το ονοματεπώνυμο τους και αυτοί παρόμοια, σαν κόμβοι του τρίτου επιπέδου. Παράλληλα η εφαρμογή παρέχει στον χρήστη την δυνατότητα να επιλέξει οποιονδήποτε από αυτούς τους κόμβους, έτσι ώστε αν θέλει να μπορεί να δει περισσότερες λεπτομέρειες σχετικά με το αντικείμενο του κόμβου. Αν ο χρήστης για παράδειγμα επιλέξει τον πρώτο κόμβο, οι πληροφορίες που θα του προβάλλονται θα είναι αναφορικά με το συγκεκριμένο Project,

όπως το budget, ο manager αλλά και το ποσοστό ολοκλήρωσης του. Αν επιλέξει κάποιον από τους κόμβους του δευτέρου επιπέδου, θα του προβληθούν οι αντίστοιχες πληροφορίες της φάσης που επέλεξε. Αν τέλος επιλέξει κάποιον από τους κόμβους του τρίτου επιπέδου, αυτοί θα προβάλλουν τις αντίστοιχες πληροφορίες του υπαλλήλου μαζί με μια γρήγορη επισκόπηση της πληρωμής του για την εκάστοτε φάση. Συνολικά, το συγκεκριμένο διάγραμμα είναι πολύ χρήσιμο για όποιον το παρακολουθεί, αφού μπορεί άμεσα να δει ποιοι εργαζόμενοι απασχολούνται σε κάθε φάση, πόσες είναι οι φάσεις και πόσοι είναι συνολικά οι εργαζόμενοι.

***Gannt Diagram***

Το δεύτερο διάγραμμα που παρέχεται έχει την μορφή του γνωστού διαγράμματος Gannt. Η εφαρμογή παρέχει στο χρήστη ένα διάγραμμα με βάση τις ημερομηνίες διεξαγωγής των εκάστοτε φάσεων του Project. Έτσι για κάθε φάση, το διάγραμμα δείχνει μια μπάρα με αρχή την ημερομηνία έναρξης και τέλος την ημερομηνία λήξης. Με τον τρόπο επίσης που κατασκευάστηκε το διάγραμμα Gannt, η κατάσταση των tasks της κάθε φάσης αντανακλούν χρωματικά στη μπάρα προβάλλοντας και την πρόοδο της εκάστοτε φάσης. Όπως είχε αναφερθεί, οι φάσεις μπορεί να ταυτίζονται με σπείρες η επαναλήψεις, με βάση το μοντέλο Spiral η Agile, οπότε το συγκεκριμένο διάγραμμα κάνει ακόμα πιο κατανοητό και βολικό τον τρόπο που λειτουργεί κάθε μοντέλο και κάνει πιο ξεκάθαρο το πότε λαμβάνει μέρος το κάθε “υποκομμάτι” του Project.

## Παρακολούθηση Προόδου

Χάρη στην παραπάνω λειτουργία και μέσω του περιβάλλοντος που έχει σχεδιαστεί για την παρακολούθηση των Projects, η εφαρμογή παρέχει την δυνατότητα στον χρήστη και στην συγκεκριμένη περίπτωση του manager, να παρακολουθεί και να σημειώνει την πρόοδο ενός Project. Ο σκοπός αυτής της λειτουργίας είναι να προσφέρει στο manager μεγαλύτερη διαδραστικότητα σε σχέση με την πλατφόρμα αλλά και σε σχέση με τις υποχρεώσεις που έχει ως υπεύθυνος έργου απέναντι στο ίδιο το έργο. Ο χρήστης πρέπει σε οποιαδήποτε φάση της υλοποίησης να μπορεί να ενημερώνει το status και την πρόοδο του έργου, προκειμένου και οι υπόλοιποι χρήστες να μπορούν να γνωρίζουν σε ποια φάση βρίσκονται αναφορικά με τα παραδοτέα και τα tasks. Μέσω του πρώτου διαγράμματος της παρακολούθησης Project, ο manager επιλέγει μια από τις φάσεις- κόμβους του δευτέρου επιπέδου και στις επιπλέον λεπτομέρειες που προβάλλονται τους εμφανίζεται και ένα κουμπί. Το κουμπί αυτό έχει να κάνει με τον τερματισμό ενός task και την έναρξη του επόμενου. Με την επιλογή του τερματισμού του εκάστοτε task, ξεκινάει το επόμενο ή τερματίζεται συνολικά η φάση, αν πρόκειται για το τελευταίο task. Από την αλλαγή αυτή προκύπτει ότι και το διάγραμμα Gannt που είχε αναφερθεί παραπάνω, θα ανανεωθεί αυτόματα και άμεσα, καθώς οποιαδήποτε αλλαγή task, θα επεκταθεί και στη μπάρα προόδου της συγκεκριμένης φάσης. Ενημερώνεται επίσης και η λίστα των Project, αφού ένα από τα πεδία της αναφέρεται στην πρόοδο του Project συνολικά. Γενικά, η χρήση πληροφοριών σχετικά με την πρόοδο του έργου είναι συχνή στην πλατφόρμα, αντανακλώντας έτσι, στην ανάγκη μιας λειτουργίας που χειρίζεται ή τερματίζει υποκομμάτια του έργου. Έτσι διατηρείται η συνέπεια των παραδοτέων και των χρονικών πλαισίων των φάσεων και η υλοποίηση του έργου είναι πιο διαχειρίσιμη όσον αφορά το χρόνο.

**Αποσύνδεση Χρήστη**

Για ευνόητους λόγους, έχει δοθεί στους χρήστες της πλατφόρμας και η δυνατότητα αποσύνδεσης. Η ανάγκη ύπαρξης αυτής της δυνατότητας προκύπτει από το γεγονός ότι η πλατφόρμα φιλοξενεί και συνδεδεμένους χρήστες. Ο συνδεδεμένος χρήστης σε οποιοδήποτε σημείο της εφαρμογής έχει την δυνατότητα είτε να αποσυνδεθεί εντελώς είτε να θέλει να εισέλθει με άλλο ρόλο από αυτό που ήδη έχει. Σε οποιαδήποτε περίπτωση, πρέπει το σύστημα να παρέχει και αυτή τη δυνατότητα.

## *3.3 Περιπτώσεις Χρήσης (Use Cases)*

Από την ανάλυση των απαιτήσεων του συστήματος προκύπτει ότι οι διαφορετικοί χρήστες της εφαρμογής για να εκτελέσουν μια λειτουργία, προχωρούν σε μια ακολουθία συγκεκριμένων ενεργειών και βημάτων. Αυτές οι ενέργειες ονομάζονται περιπτώσεις χρήσης και ορίζονται από μια λίστα ενεργειών ή γεγονότων που ορίζουν την αλληλεπίδραση μεταξύ ενός χρήστη και του συστήματος για την ολοκλήρωση μιας λειτουργίας.

Ακολουθούν δεκαπέντε (15) περιπτώσεις χρήσης της πλατφόρμας και παρουσιάζονται σε μορφή πινάκων που παρέχουν όλες τις απαραίτητες πληροφορίες για να γίνουν κατανοητές, να μοντελοποιηθούν και τελικά να εφαρμοστούν βοηθώντας στο σχεδιασμό και την ανάπτυξη της πλατφόρμας.

*Πίνακας 1. Admin Login Use Case*

| **Use Case (UC01) : Login σαν Διαχειριστής** | |
|---|---|
| Actor | Admin |
| Brief | Ο διαχειριστής μπορεί να συνδεθεί στην εφαρμογή και να χρησιμοποιήσει τις λειτουργίες της. |
| Προϋπόθεση | Να υπάρχει admin στο σύστημα ( ορίζεται με το σύστημα ) |
| Αποτέλεσμα | Ο χρήστης συνδέεται στην εφαρμογή και μπορεί να δει το dashboard. |
| Βασική Ροή | 1. Ο διαχειριστής βλέπει τη φόρμα login στην αρχική σελίδα<br>2. Ο διαχειριστής συμπληρώνει το email και το κωδικό του και επιλεγεί το ρόλο “admin”.<br>3. Ο χρήστης συνδέεται στην εφαρμογή και ανακατευθύνεται στο dashboard. |
| Εναλλακτική Ροή | Τα στοιχεία του διαχειριστή είναι λανθασμένα και μπορεί να προσπαθήσει να συνδεθεί ξανά με νέα στοιχεία.<br>Δεν έγινε επιλογή του ρόλου “admin”. |
| Εξαρτάται από | ---- |
| Εκτείνεται σε | UC02, UC03 |

*Πίνακας 2. Add New Employee Use Case*

| **Use Case (UC02) : Καταχώρηση Νέου Υπάλληλου** | |
|---|---|
| Actor | Admin |
| Brief | Ο διαχειριστής μπορεί να καταχωρήσει ένα νέο υπάλληλο στην πλατφόρμα. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως admin και να μην έχει καταχωρήσει κάποιον υπάλληλο με τα ίδια στοιχεία στο παρελθόν. |
| Αποτέλεσμα | Ο διαχειριστής καταχωρεί ένα νέο υπάλληλο στην εφαρμογή. |
| Βασική Ροή | 1. Ο διαχειριστής συνδέεται στην πλατφόρμα και κάνει επιλογή του Employees από το αρχικό Dashboard.<br>2. Ανακατευθύνεται στο μενού των Employees και επιλεγεί την Προσθήκη Νέου Υπαλλήλου.<br>3. Συμπληρώνει τα απαιτούμενα στοιχεία από το αναδυόμενο modal και επιλέγει το κουμπί “Τέλος”.<br>4. Ο διαχειριστής ανακατευθύνεται πάλι στο μενού Employees και μπορεί τώρα να δει το νέο υπάλληλο στη λίστα των Employees. |
| Εναλλακτική Ροή | Ο διαχειριστής επιλέγει “Ακύρωση” κατά την διάρκεια της εκχώρησης. |
| Εξαρτάται από | UC01 |
| Εκτείνεται σε | UC08, UC11, UC12, UC13 |

*Πίνακας 3. Add New Project Use Case (Phase 1)*

| **Use Case (UC03) : Καταχώρηση Νέου Project (1η φαση)** | |
|---|---|
| Actor | Admin |
| Brief | Ο διαχειριστής μπορεί να καταχωρήσει ένα νέο project στην πλατφόρμα. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως admin και να μην έχει καταχωρήσει project με τα ίδια στοιχεία στο παρελθόν. |
| Αποτέλεσμα | Ο διαχειριστής καταχωρεί ένα νέο project (1η φαση) στην εφαρμογή και ενεργοποιείται το επόμενο στάδιο αρχικοποίησης. |

| | |
|---|---|
| Βασική Ροή | 1. Ο διαχειριστής συνδέεται στην πλατφόρμα και κάνει επιλογή του Projects από το αρχικό Dashboard.<br>2. Ανακατευθύνεται στο μενού των Projects και επιλέγει την Προσθήκη Νέου Project.<br>3. Συμπληρώνει τα απαιτούμενα στοιχεία από το αναδυόμενο modal και επιλέγει “Εκχώρηση”<br>4. Ο διαχειριστής ανακατευθύνεται στο μενού των Projects και μπορεί τώρα να δει το νέο project στη λίστα των Projects. |
| Εναλλακτική Ροή | Ο χρήστης επιλέγει “Ακύρωση” κατά τη διάρκεια της εκχώρησης. |
| Εξαρτάται από | UC01 |
| Εκτείνεται σε | UC04, UC07 |

*Πίνακας 4.Add Phases & Manager to Project Use Case (Phase 2)*

| **Use Case (UC04) : Καταχώρηση Φάσεων και Manager σε Project (2η φαση)** | |
|---|---|
| Actor | Admin |
| Brief | Ο admin μπορεί να ορίσει manager και φάσεις για ένα συγκεκριμένο project κατά την διάρκεια της αρχικοποίησης. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως admin.<br>Ο admin να έχει καταχωρήσει υπάλληλους στο σύστημα και να έχει ολοκληρώσει το πρώτο μέρος της αρχικοποίησης του συγκεκριμένου project. |
| Αποτέλεσμα | Ο χρήστης καταχωρεί τις απαιτούμενες φάσεις και κάποιον υπάλληλο ως manager του συγκεκριμένου πρότζεκτ (2η φαση) και ενεργοποιείται το επόμενο στάδιο αρχικοποίησης. |
| Βασική Ροή | 1. Ο διαχειριστής συνδέεται στην πλατφόρμα και κάνει επιλογή του Projects από το αρχικό Dashboard.<br>2. Ανακατευθύνεται στο μενού με τη λίστα όλων των project και αναζητεί ένα συγκεκριμένο.<br>3. Επιλεγεί το ανάλογο κουμπί αρχικοποίησης (2ης φάσης) και αναδύεται ένα modal.<br>4. Επιλέγει από την λίστα που εμφανίζεται, τις φάσεις του project. Μπορεί να δημιουργήσει και custom φάσεις σε αυτό το σημείο, αν θέλει και επιλεγεί συνέχεια.<br>5. Αρχικοποιεί με πληροφορίες τη κάθε φαση και επιλεγεί το κουμπί “Συνέχεια”.<br>6. Ο διαχειριστής διαλέγει το μάνατζερ από μια λίστα υπαλλήλων και επιλεγεί “Τέλος”. |
| Εναλλακτική Ροή | Ο χρήστης επιλεγεί “Ακύρωση” οποιαδήποτε στιγμή της διαδικασίας αρχικοποίησης των φάσεων. |
| Εξαρτήται από | UC03, UC02 |
| Εκτείνεται σε | UC05, UC06, UC07 |

*Πίνακας 5. Manager Login Use Case*

| Use Case (UC05): Login σαν Manager | |
|---|---|
| Actor | Manager |
| Brief | Ο μάνατζερ μπορεί να συνδεθεί στην εφαρμογή και να χρησιμοποιήσει τις λειτουργίες της. |
| Προϋπόθεση | Ο συγκεκριμένος υπάλληλος να έχει καθήκοντα και δικαιώματα μάνατζερ σε ένα τουλάχιστον Project. |
| Αποτέλεσμα | Ο manager συνδέεται στην εφαρμογή και μπορεί να δει τα project στα οποία συμμετέχει. |
| Βασική Ροή | 1. Ο χρήστης βλέπει την φόρμα login στην αρχική σελίδα<br>2. Ο χρήστης αρχικά συμπληρώνει το email και το κωδικό του και στην συνέχεια επιλέγει τον ρόλο του “manager”.<br>3. Ο manager επιλέγει το κουμπί “Login”, συνδέεται στην εφαρμογή και ανακατευθύνεται στη λίστα με τα projects που συμμετέχει. |
| Εναλλακτική Ροή | Τα στοιχεία του manager είναι λανθασμένα και μπορεί να προσπαθήσει να συνδεθεί ξανά με νέα στοιχεία.<br>Δεν έγινε επιλογή του ρόλου “manager”.<br>Ο συγκεκριμένος υπάλληλος δεν είναι υπεύθυνος σε κανένα project και έτσι η λίστα των projects είναι κενή. |
| Εξαρτάται από | UC04 |
| Εκτείνεται σε | UC06 |

*Πίνακας 6. Add Employees in Project (Phase 3)*

| Use Case (UC06) : Καταχώρηση Υπάλληλων σε Φάσεις ( 3η φαση) | |
|---|---|
| Actor | Manager |
| Brief | Ο manager μπορεί να καταχωρήσει υπαλλήλους σε κάθε φάση ενός project. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως manager.<br>Ο admin να έχει καταχωρήσει υπαλλήλους στο σύστημα καθώς και να έχει ολοκληρώσει το μέρος της αρχικοποίησης που τον αφορά για το συγκεκριμένο project.<br>Να έχει οριστεί ο συγκεκριμένος manager ως υπεύθυνος του συγκεκριμένου project από τον admin. |
| Αποτέλεσμα | Ο manager καταχωρεί έναν η περισσοτέρους υπαλλήλους σε κάθε φάση του project (3η φάση) και τερματίζει η αρχικοποίηση του συγκεκριμένου Project. |

| | |
|---|---|
| Βασική Ροή | 1. Ο manager συνδέεται στην πλατφόρμα και ανακατευθύνεται στην λίστα με τα projects που συμμετέχει.<br>2. Βλέπει την λίστα με όλα τα projects και αναζητεί ένα συγκεκριμένο.<br>3. Επιλέγει το ανάλογο κουμπί αρχικοποίησης (3ης φάσης) και αναδύεται ένα modal.<br>4. Ο manager επιλέγει, από την λίστα που εμφανίζεται, τους υπάλληλους για την συγκεκριμένη φαση και επιλέγει το κουμπί “Συνέχεια”.<br>5. Η διαδικασία επαναλαμβάνεται τόσες φορές, όσες και οι φάσεις του project και ο manager επιλέγει το κουμπί “Τέλος”. |
| Εναλλακτική Ροή | Ο manager δεν μπορεί να επιλέξει υπάλληλους γιατί δεν υπάρχουν διαθέσιμοι για το χρονικό διάστημα της εκάστοτε φάσης. |
| Εξαρτάται από | UC05, UC04 |
| Εκτείνεται σε | UC07, UC09, UC14 |

*Πίνακας 7. Cancel Project's Initialization Use Case*

| **Use Case (UC07): Κατάργηση Project** | |
|---|---|
| Actor | Admin, Manager |
| Brief | Ο χρήστης μπορεί να καταργήσει ένα συγκεκριμένο project που βρίσκεται ακόμα στην διαδικασία της αρχικοποίησης. Γίνεται ολική διαγραφή όλων των στοιχείων του project από το σύστημα |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Admin ή Manager.<br>Ο admin να έχει καταχωρήσει τουλάχιστον στην 1η φαση του το συγκεκριμένο πρότζεκτ.<br>Το συγκεκριμένο Project να βρίσκεται στην 3η φαση, για να μπορεί να το καταργήσει ο manager. |
| Αποτέλεσμα | Ο χρήστης καταργεί ένα συγκεκριμένο πρότζεκτ από την εφαρμογή. |

| | |
|---|---|
| Βασική Ροή | 1. Ο χρήστης συνδέεται στην πλατφόρμα και αν είναι ο admin επιλέγει από το αρχικό Dashboard το κουμπί Projects, αλλιώς στην αντίθετη περίπτωση αν είναι ο μάνατζερ ανακατευθύνεται στην λίστα με τα projects που συμμετέχει<br>2. Ο χρήστης βλέπει την λίστα με όλα τα project και αναζητεί ένα συγκεκριμένο.<br>3. Ο χρήστης επιλέγει το κουμπί κατάργησης του συγκεκριμένου Project και αναδύεται ένα modal.<br>4. Ο χρήστης επιβεβαιώνει με την επιλογή του και ανακατευθύνεται στο μενού Projects.<br>5. Ο χρήστης μπορεί τώρα να δει την ανανεωμένη λίστα των projects, χωρίς το συγκεκριμένο. |
| Εναλλακτική Ροή | Ο admin δεν βρίσκει το πρότζεκτ γιατί δεν υπάρχει στο σύστημα.<br>Ο manager δεν βρίσκει το Project γιατί δεν είναι ορισμένος ως υπεύθυνος του συγκεκριμένου έργου.<br>Ο χρήστης επιλέγει “Ακύρωση” κατά την προτροπή επιβεβαίωσης της επιλογής του. |
| Εξαρτάται από | UC01/UC05, UC03/UC04/UC06 |
| Εκτείνεται σε | ---- |

*Πίνακας 8. Employee Login Use Case*

| **Use Case (UC08): Login σαν Employee** | |
|---|---|
| Actor | Employee |
| Brief | Ο υπάλληλος μπορεί να συνδεθεί στην εφαρμογή και να χρησιμοποιήσει τις λειτουργίες της. |
| Προϋπόθεση | Να έχει καταχωρηθεί ο συγκεκριμένος υπάλληλος από τον admin στο σύστημα στο παρελθόν. |
| Αποτέλεσμα | Ο υπάλληλος συνδέεται στην εφαρμογή και μπορεί να δει τα project στα οποία συμμετέχει. |
| Βασική Ροή | 1. Ο υπάλληλος βλέπει την φόρμα login στην αρχική σελίδα<br>2. Ο υπάλληλος αρχικά συμπληρώνει το email και το κωδικό του και στην συνέχεια επιλέγει τον ρόλο του “employee”.<br>3. Ο υπάλληλος επιλέγει το κουμπί “Login”, συνδέεται στην εφαρμογή και ανακατευθύνεται στην λίστα με τα projects που συμμετέχει. |
| Εναλλακτική Ροή | Τα στοιχεία του υπαλλήλου είναι λανθασμένα και μπορεί να προσπαθήσει να συνδεθεί ξανά με νέα στοιχεία.<br>Δεν έγινε επιλογή του ρόλου “employee”. |
| Εξαρτάται από | UC02 |
| Εκτείνεται σε | ---- |

*Πίνακας 9. View Project Details Use Case*

| Use Case (UC09): Προεπισκόπηση ενός Project | |
|---|---|
| Actor | Admin, Manager, Employee |
| Brief | Ο χρήστης μπορεί να δει όλες τις σχετικές λεπτομέρειες για ένα συγκεκριμένο project. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος στο σύστημα.<br>Ο admin να έχει καταχωρήσει το συγκεκριμένο πρότζεκτ στο παρελθόν.<br>Ο admin και ο manager να έχουν τελειώσει με την διαδικασία αρχικοποίησης του project.<br>Αν ο χρήστης συνδεθεί σαν employee η manager, να συμμετέχει στο συγκεκριμένο project. |
| Αποτέλεσμα | Ο χρήστης βλέπει όλες τις πληροφορίες ενός project. |
| Βασική Ροή | 1. Ο χρήστης συνδέεται στην πλατφόρμα και ανακατευθύνεται στην λίστα με τα αντίστοιχα projects.<br>2. Ο χρήστης βλέπει την αντίστοιχη λίστα με τα πρότζεκτ και αναζητεί ένα συγκεκριμένο πρότζεκτ.<br>3. Ο χρήστης επιλέγει το αντίστοιχο κουμπί προβολής και αναδύεται ένα modal.<br>4. Ο χρήστης βλέπει τις πληροφορίες του project και αναλυτικά διαγράμματα αυτού. |
| Εναλλακτική Ροή | Ο χρήστης δεν μπορεί να επιλέξει ένα συγκεκριμένο project γιατί δεν έχει τελειώσει η αρχικοποίηση του.<br>Ο manager η ο υπάλληλος δεν μπορούν να βρουν το συγκεκριμένο project γιατί δεν έχουν λάβει μέρος σε αυτό |
| Εξαρτάται από | UC06, UC01/UC05/UC08 |
| Εκτείνεται σε | UC10 |

*Πίνακας 10. Change Task Use Case*

| Use Case (UC10) : Ολοκλήρωση ενός task μιας φάσης | |
|---|---|
| Actor | Manager |
| Brief | Ο manager μπορεί να αλλάξει την κατάσταση μιας φάσης επιβεβαιώνοντας την ολοκλήρωση ενός task της. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Manager και να μην έχει ολοκληρώσει όλα τα task της συγκεκριμένης φάσης.<br>Ο admin και ο manager να έχουν τελειώσει με την διαδικασία αρχικοποίησης του project. |
| Αποτέλεσμα | Ο manager τερματίζει το task μιας φάσης και είτε η φαση τερματίζει είτε συνεχίζει στο επόμενο task. |

| | |
|---|---|
| Βασική Ροή | 1. Ο manager συνδέεται στη πλατφόρμα και ανακατευθύνεται στην λίστα με τα projects που συμμετέχει.<br>2. Ο manager βλέπει τη λίστα με όλα τα projects, αναζητεί ένα συγκεκριμένο, επιλέγει το κουμπί “Προβολή Project” και αναδύεται ένα modal.<br>3. Ο manager επιλέγει μια φάση και στην συνέχεια με το κουμπί “Next” τερματίζει το παρόν task της φάσης ενός συγκεκριμένου project. |
| Εναλλακτική Ροή | Ο manager δεν μπορεί να επιλέξει ένα συγκεκριμένο project γιατί είτε δεν έχει τελειώσει είτε δεν έχει ξεκινήσει η αρχικοποίηση του, ή δεν υπάρχει. Ο manager δεν μπορεί να επιλέξει “Next” στο επόμενο task, επειδή όλα τα task τις συγκεκριμένης φάσης έχουν ολοκληρωθεί. |
| Εξαρτάται από | UC09 |
| Εκτείνεται σε | ---- |

*Πίνακας 11. View Employees Payments Use Case*

| **Use Case (UC11) : Προεπισκόπηση Πληρωμών ενός υπαλλήλου** | |
|---|---|
| Actor | Admin, Υπάλληλος |
| Brief | Ο χρήστης μπορεί να δει όλες τις πληρωμές ενός υπαλλήλου. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Admin ή Employee.<br>Ο Admin να έχει καταχωρήσει τον συγκεκριμένο υπάλληλο στο παρελθόν και ο υπάλληλος να έχει οριστεί σε κάποιο project στο παρελθόν. |
| Αποτέλεσμα | Ο χρήστης βλέπει όλες τις πληρωμές ή μια συγκεκριμένη για έναν υπάλληλο. |
| Βασική Ροή | 1. a) Ο χρήστης συνδέεται στην πλατφόρμα ως admin και από το αρχικό Dashboard κάνει επιλογή του Employees και ανακατευθύνεται στη λίστα με όλους τους employees του συστήματος.<br>b) Ο χρήστης συνδέεται στην πλατφόρμα ως employee και ανακατευθύνεται αυτόματα στο Employee μενού.<br>2. a) Ο admin αναζητεί έναν συγκεκριμένο υπάλληλο, επιλέγει του κουμπί “Προβολή” και αναδύεται ένα modal.<br>b) Ο υπάλληλος επιλέγει από το μενού του το κουμπί “Προβολή” και αναδύεται ένα modal.<br>3. Ο χρήστης μπορεί να δει τώρα την λίστα με όλες τις πληρωμές του υπαλλήλου (του ιδίου αν είναι logged in ως employee)<br>4. Ο χρήστης μπορεί να αναζητήσει μια συγκεκριμένη πληρωμή και να την δει λεπτομερώς. |

| | |
|---|---|
| Εναλλακτική Ροή | Ο admin δεν βρίσκει τον συγκεκριμένο υπάλληλο γιατί δεν υπάρχει στο σύστημα.<br>Ο admin η ο υπάλληλος, δεν μπορεί να δει πληρωμές για ένα συγκεκριμένο υπάλληλο η για τον ίδιο αντίστοιχα γιατί δεν υπάρχουν καταχωρημένες στο σύστημα. |
| Εξαρτάται από | UC01/UC08, UC06 |
| Εκτείνεται σε | ---- |

*Πίνακας 12. Edit Employee Use Case*

| Use Case (UC12): Επεξεργασία υπαλλήλου | |
|---|---|
| Actor | Employee |
| Brief | Ο υπάλληλος μπορεί να επεξεργαστεί τα ήδη καταχωρημένα στοιχεία του. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Employee και να έχει καταχωρηθεί από τον Admin στο παρελθόν. |
| Αποτέλεσμα | Ο υπάλληλος επεξεργάστηκε και καταχώρησε τα στοιχεία του στην πλατφόρμα. |
| Βασική Ροή | 1. Ο υπάλληλος συνδέεται στην πλατφόρμα ως employee και ανακατευθύνεται αυτόματα στο Employee μενού.<br>2. Ο υπάλληλος επιλέγει το κουμπί “Edit” και αναδύεται ένα modal.<br>3. Ο υπάλληλος συμπληρώνει τα στοιχεία προς αλλαγή.<br>4. Ο υπάλληλος επιλέγει το κουμπί “Τέλος” και ανακατευθύνεται στο μενού Employees. |
| Εναλλακτική Ροή | Ο υπάλληλος επιλέγει “Ακύρωση” κατά την διάρκεια της επεξεργασίας. |
| Εξαρτάται από | UC08 |
| Εκτείνεται σε | UC08 |

*Πίνακας 13. Discharge Employee Use Case*

| Use Case (UC13) : Αποδέσμευση υπαλλήλου | |
|---|---|
| Actor | Admin |
| Brief | Ο διαχειριστής μπορεί να αποδεσμεύσει έναν ενεργό υπάλληλο. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Admin και να έχει καταχωρήσει τον υπάλληλο στο παρελθόν. |
| Αποτέλεσμα | Ο διαχειριστής αποδεσμεύει έναν συγκεκριμένο υπάλληλο από την εφαρμογή. Ωστόσο δεν διαγράφεται από το σύστημα, πάρα μόνο το status του employee αλλάζει σε μη ενεργό. |

| | |
|---|---|
| Βασική Ροή | 1. Ο διαχειριστής συνδέεται στην πλατφόρμα ως admin και από το αρχικό Dashboard κάνει επιλογή του Employees και ανακατευθύνεται στην λίστα με όλους τους employees του συστήματος.<br>2. Ο διαχειριστής αναζητεί έναν συγκεκριμένο υπάλληλο.<br>3. Ο διαχειριστής επιλέγει το κουμπί “αποδέσμευση” του συγκεκριμένου υπαλλήλου και αναδύεται ένα modal.<br>4. Ο διαχειριστής επιβεβαιώνει με την επιλογή του και ανακατευθύνεται στο μενού Employees. |
| Εναλλακτική Ροή | Ο χρήστης επιλέγει “Ακύρωση” κατά την επιβεβαίωση της επιλογής του. |
| Εξαρτάται από | UC01, UC02 |
| Εκτείνεται σε | ---- |

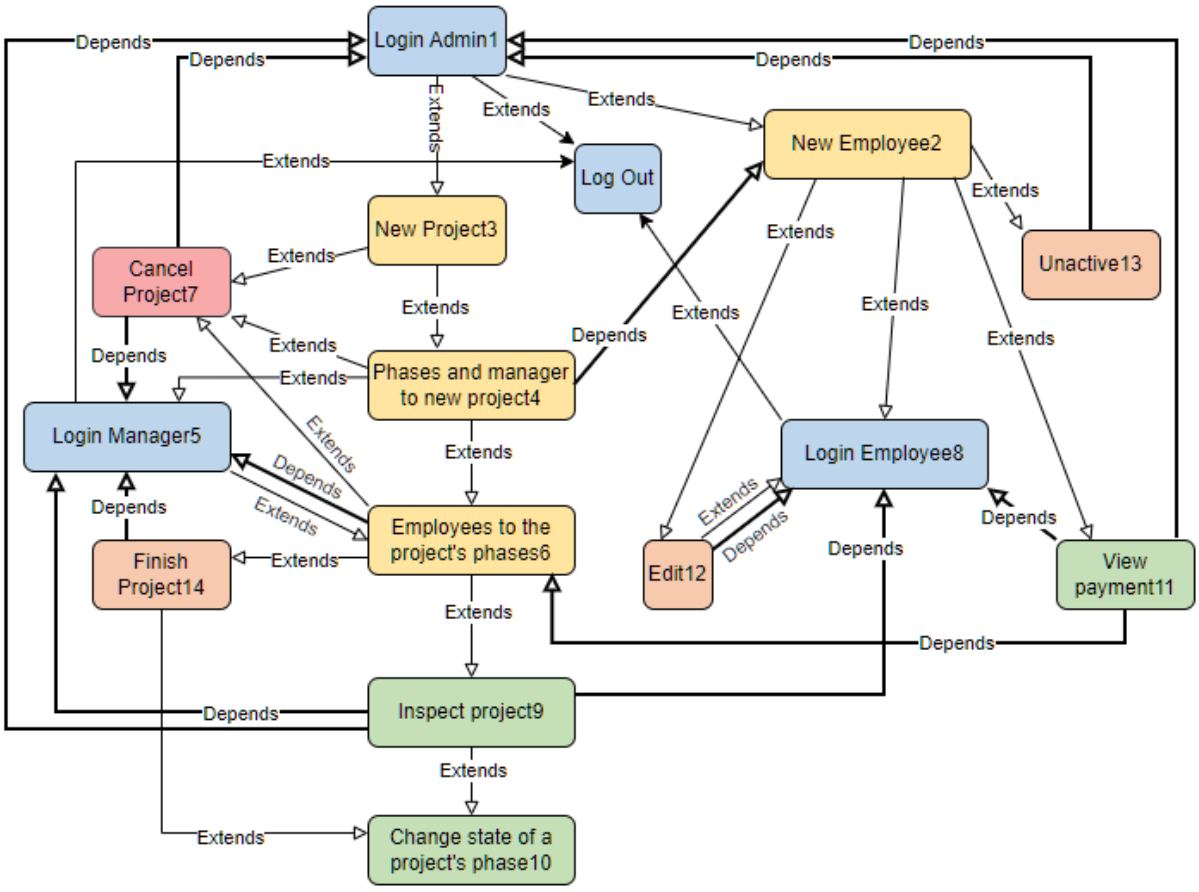
*Πίνακας 14. Terminate Project Use Case*

| **Use Case (UC14): Τερματισμός Project** | |
|---|---|
| Actor | Manager |
| Brief | Ο υπεύθυνος μπορεί να τερματίσει οριστικά η πρόωρα ένα συγκεκριμένο project από το σύστημα. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος ως Manager και να έχει αρχικοποιήσει με υπαλλήλους τις φάσεις του συγκεκριμένου project. |
| Αποτέλεσμα | Ο manager τερματίζει ένα συγκεκριμένο πρότζεκτ από την εφαρμογή. Ωστόσο δεν διαγράφεται από το σύστημα, πάρα μόνο το status του project αλλάζει σε μη ενεργό. |
| Βασική Ροή | 1. Ο manager συνδέεται στην πλατφόρμα και ανακατευθύνεται στην λίστα με τα projects που συμμετέχει.<br>2. Ο manager αναζητεί ένα συγκεκριμένο πρότζεκτ από την λίστα που εμφανίζεται, επιλέγει το κουμπί ”τερματισμού” του συγκεκριμένου πρότζεκτ και αναδύεται ένα modal.<br>3. Ο manager επιβεβαιώνει την επιλογή του και ανακατευθύνεται στο μενού Projects.<br>4. Ο manager μπορεί τώρα να δει την ανανεωμένη λίστα των projects. |
| Εναλλακτική Ροή | Ο χρήστης επιλέγει “Ακύρωση” κατά την προτροπή επιβεβαίωσης της επιλογής του. |
| Εξαρτάται από | UC05, UC06 |
| Εκτείνεται σε | UC10 |

*Πίνακας 15. Log Out Use Case*

| Use Case (UC15): Log Out Χρήστη | |
|---|---|
| Actor | Admin, Manager, Employee |
| Brief | Ο χρήστης μπορεί να αποσυνδεθεί ανά πάσα στιγμή από το σύστημα. |
| Προϋπόθεση | Ο χρήστης να είναι συνδεδεμένος, με κάποιο ρόλο στο σύστημα. |
| Αποτέλεσμα | Ο χρήστης αποσυνδέεται από την εφαρμογή. |
| Βασική Ροή | 1. Ο χρήστης επιλέγει το αντίστοιχο κουμπί αποσύνδεσης που βρίσκεται στην εκάστοτε σελίδα.<br>2. Ο χρήστης αποσυνδέεται από το σύστημα και μεταφέρεται στην σελίδα του login. |
| Εναλλακτική Ροή | ---- |
| Εξαρτάται από | UC01, UC05, UC08 |
| Εκτείνεται σε | ---- |

Παρακάτω έχει σχεδιαστεί ένα διάγραμμα προκειμένου να αναπαραστήσει τις εξαρτήσεις που αναλυθήκαν στα σενάρια χρήσης προηγουμένως για τις λειτουργίες, την έκταση που έχουν αυτές μεταξύ τους άλλα και το ποτέ διαδέχεται η μια την άλλη.

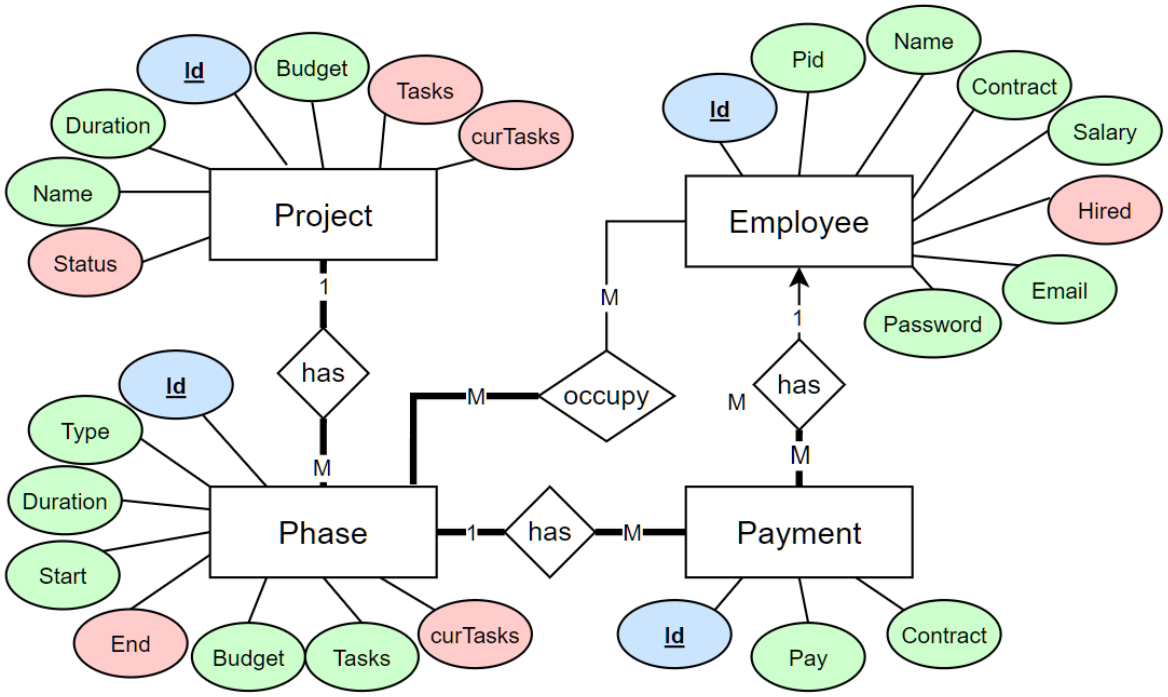



*Σχήμα 7. Διάγραμμα Εξαρτήσεων μεταξύ Λειτουργιών*

## *3.4 Βάση Δεδομένων*

Για την μοντελοποίηση των δεδομένων και της δομής μιας βάσης δεδομένων είναι ευρέως διαδεδομένη η χρήση του διαγράμματος οντοτήτων-συσχετίσεων (ER Diagram), καθώς αποτελεί σημαντική βοήθεια στην κατανόηση και στην αναπαράσταση των αναγκαίων μεταβλητών που αποθηκεύονται στο σύστημα. Το διάγραμμα ER χρησιμοποιείται συνήθως στο πρώτο στάδιο της διαδικασίας σχεδιασμού βάσης δεδομένων, κατά την ανάλυση των απαιτήσεων της βάσης δεδομένων και σκοπός του είναι να κατανοήσει τις ανάγκες των χρηστών και να δημιουργήσει ένα μοντέλο δεδομένων που να τις ικανοποιεί. Είναι ένα σημαντικό εργαλείο για τον σχεδιασμό βάσης δεδομένων επειδή παρέχει μια οπτική αναπαράσταση των δεδομένων. Αυτό βοηθά τους σχεδιαστές να κατανοήσουν καλύτερα τα δεδομένα και να δημιουργήσουν ένα μοντέλο που είναι αποτελεσματικό και αποδοτικό. Καθώς η εφαρμογή χρησιμοποιεί ένα σχεσιακό μοντέλο βάσης δεδομένων, το διάγραμμα ER θα δώσει μια πιο ξεκάθαρη εικόνα μεταξύ των οντοτήτων, των στοιχείων τους και των συσχετίσεων.

Προκειμένου η εφαρμογή να συλλεγεί και να χρησιμοποιεί τα δεδομένα που καταχωρεί ο χρήστης με ταχύτητα, αξιοπιστία και ευελιξία, η βάση δεδομένων δημιουργήθηκε με το MySQL. Το MySQL είναι ένα open source σύστημα, που χρησιμοποιεί τη γλώσσα SQL (Structured Query Language) για να επεμβαίνει στα δεδομένα της βάσης δεδομένων και χρησιμοποιείται σε ιστοσελίδες και σε εφαρμογές ηλεκτρονικού εμπορίου, διαχείρισης πελατών η υπαλλήλων και γενικότερα εφαρμογές με μεγάλο όγκο δεδομένων κλπ.




*Σχήμα 8. Διάγραμμα ER της βάσης δεδομένων της εφαρμογής (created with draw.io)*

Όπως φαίνεται από το Σχήμα [8] του διαγράμματος της βάσης δεδομένων, κάποια από τα χαρακτηριστικά των οντοτήτων, συμβολίζονται με διαφορετικό χρώμα. Κάθε χρώμα συμβολίζει μια διαφορετική κατηγορία χαρακτηριστικού. Τα attributes με ανοιχτό μπλε χρώμα συμβολίζουν τα primary keys κάθε οντότητας, τα attributes με πράσινο ανοιχτό χρώμα συμβολίζουν τα βασικά χαρακτηριστικά της οντότητας, τα attributes με ανοιχτό μωβ χρώμα συμβολίζουν τα foreign keys

και τα attributes με ανοιχτό κόκκινο χρώμα συμβολίζουν τα χαρακτηριστικά της οντότητας, που είναι δευτερεύοντα και βοηθητικά μόνο ως προς το σύστημα η δημιουργούνται από αυτό. Ως δευτερεύοντα ορίζονται τα χαρακτηριστικά τα οποία είτε προκύπτουν από διαδικασίες της πλατφόρμας και όχι από τον χρήστη, είτε δεν εμφανίζονται καθόλου στο χρήστη και είναι χρήσιμα μόνο για το σύστημα. Οι οντότητες της βάσης αναφέρονται παρακάτω μαζί με τα στοιχεία που τις αποτελούν καθώς και μία λεπτομερή επεξήγηση των συσχετίσεων τους.

### 3.4.1 Οντότητες

#### *1. Οντότητα «Projects»*

Ο πίνακας των projects αναφέρεται στα Projects που έχουν καταχωρηθεί σε όλη την διάρκεια χρήσης του συστήματος από την εκάστοτε επιχείρηση και αποτελείται από 7 χαρακτηριστικά τα οποία είναι :

- **id:** Ένα αναγνωριστικό για το project.
- **name:** Το όνομα του project.
- **duration:** Η διάρκεια του project σε μήνες.
- **budget:** Ο προϋπολογισμός του project.
- **status:** Η κατάσταση του project.
- **tasks:** Ο αριθμός των εργασιών στο Project.
- **curTasks:** Ο αριθμός των τρεχουσών εργασιών στο Project.

Το id του project είναι μοναδικό, αυξάνεται αυτόματα με κάθε καταχώρηση, είναι τύπου int και αποτελεί το primary key του πίνακα. Το name είναι τύπου varchar και τα υπόλοιπα χαρακτηριστικά είναι τύπου int. Το status, το tasks και το cur Tasks εμφανίζονται στο σχήμα με ανοιχτό κόκκινο χρώμα, καθώς ο χρήστης δεν εμπλέκεται άμεσα με την αρχικοποίηση αυτών των τιμών, και αυτές χρησιμοποιούνται μόνο από το σύστημα πάρα μόνο για την παρακολούθηση της κατάστασης του project και του τερματισμού.

#### *2. Οντότητα «Employees»*

Ο πίνακας employees αναφέρεται στους υπάλληλους που εργάζονται στην εκάστοτε επιχείρηση και που έχουν καταχωρηθεί στο σύστημα. Αποτελείται από 9 χαρακτηριστικά τα οποία είναι :

- **id:** Ένα αναγνωριστικό για τον υπάλληλο.
- **pid:** Ο αριθμός ταυτότητας του υπαλλήλου.
- **email:** Η διεύθυνση ηλεκτρονικού ταχυδρομείου του υπαλλήλου.
- **name:** Το όνομα του υπαλλήλου.

- **contract:** Ο τύπος σύμβασης του υπαλλήλου.
- **salary:** Ο μισθός του υπαλλήλου.
- **birth:** Η ημερομηνία γέννησης του υπαλλήλου.
- **password:** Ο κωδικός πρόσβασης του υπαλλήλου.
- **hired:** Δείκτης που δηλώνει αν είναι ενεργός ο υπάλληλος.

Το id του employee είναι μοναδικό, αυξάνεται αυτόματα με κάθε καταχώρηση, είναι τύπου int και αποτελεί το primary key του πίνακα. Το name, το pid, το password και το email είναι τύπου varchar και το birth, salary, hired και contract να είναι τύπου int, με το contract να παίρνει τις τιμές 4 και 8, ανάλογα με την σύμβαση τους. Το hired εμφανίζεται στο σχήμα με ανοιχτό κόκκινο χρώμα, καθώς η τιμή συμπληρώνεται αυτόματα με την αρχικοποίηση του υπαλλήλου και δεν σχετίζεται άμεσα με τον χρήστη, αλλά χρησιμοποιείται κυρίως από το σύστημα για να διαχειρίζεται τους ενεργούς και μη υπαλλήλους αναλόγως.

### *3. Οντότητα «Phases»*

Ο πίνακας phase αναφέρεται σε όλες τις ενεργές η μη φάσεις των project του συστήματος και αποτελείται από 9 χαρακτηριστικά, τα οποία είναι :

- **id:** Ένα μοναδικό αναγνωριστικό για τη φάση.
- **type:** Ο τύπος-όνομα της φάσης.
- **duration:** Η διάρκεια της φάσης σε μήνες.
- **start:** Η ημερομηνία έναρξης της φάσης.
- **end:** Η ημερομηνία λήξης της φάσης.
- **budget:** Ο προϋπολογισμός της φάσης.
- **tasks:** Ο αριθμός των εργασιών στη φάση.
- **curTasks:** Ο αριθμός των τρεχουσών εργασιών στη φάση.
- **project_id:** Το έργο στο οποίο ανήκει η φάση.

Το id της φάσης είναι μοναδικό, αυξάνεται αυτόματα με κάθε καταχώρηση, είναι τύπου int και αποτελεί το primary key του πίνακα. Το type είναι τύπου varchar, το duration, το curTasks και tasks είναι τύπου number, ενώ το start και το end είναι τύπου date. Το end και το curTasks εμφανίζονται στο σχήμα με ανοιχτό κόκκινο χρώμα, καθώς το end δεν υπολογίζεται άμεσα από τον χρήστη, αλλά από το start και το duration που δίνει αυτός, οπότε το end υπολογίζεται από το σύστημα. Το curTasks επίσης χρησιμοποιείται και αυτό κυρίως από το σύστημα, πάρα μόνο για την παρακολούθηση της κατάστασης της φάσης και του τερματισμού της. Το project_id είναι foreign key που αναφέρεται στο αναγνωριστικό κάποιου project και υποδεικνύει σε ποιο project ανήκει η φαση.

### *4. Οντότητα «Payments»*

Ο πίνακας payments αναφέρεται σε όλες τις πληρωμές του συστήματος και δηλώνει ποιοι υπάλληλοι έχουν πάρει μέρος σε ποια φάση. Αποτελείται από 5 χαρακτηριστικά, τα οποία είναι :

- **id:** Ένα μοναδικό αναγνωριστικό για την πληρωμή.
- **pay:** Το ποσό χρημάτων που καταβλήθηκε στον υπάλληλο.
- **contract:** Οι ώρες απασχόλησης του υπαλλήλου για την συγκεκριμένη φαση.
- **phase_id:** Η φάση για την οποία έγινε η πληρωμή.
- **employee_id:** Ο υπάλληλος που έλαβε την πληρωμή.

Το id της φάσης είναι μοναδικό, αυξάνεται αυτόματα με κάθε καταχώρηση, είναι τύπου int και αποτελεί το primary key του πίνακα. Το pay και το contract είναι τύπου int, αλλά το pay προκύπτει από το σύστημα και από την επιλογή του contract που θα κάνει ο manager, οπότε εμφανίζεται και αυτό με ανοιχτό κόκκινο χρώμα, σε σχέση με το contract. Το phase_id και το employee_id είναι τα foreign keys που αναφέρονται στα αναγνωριστικά κάποιας φάσης και κάποιου υπαλλήλου αντίστοιχα και υποδεικνύουν σε ποια φαση και σε ποιον υπάλληλο αναφέρεται η συγκεκριμένη πληρωμή.

## 3.4.2 Σχέσεις Οντοτήτων

Σε γενικές γραμμές, όπως φαίνεται και από το διάγραμμα, η βάση δεδομένων της εφαρμογής είναι σχετικά απλή, αποτελείται από τέσσερις οντότητες εκ των οποίων οι δύο μπορούν να είναι ανεξάρτητες και οι υπόλοιπες δύο προϋποθέτουν την ύπαρξη των πρώτων δύο. Αναλυτικότερα, η ύπαρξη ενός υπαλλήλου δεν προϋποθέτει την ύπαρξη ενός project, ούτε μιας φάσης ή μιας πληρωμής. Το γεγονός ότι η εφαρμογή χτίστηκε, με βάση να χωρίσει την αρχικοποίηση ενός project σε στάδια, κάνει θεωρητικά και την οντότητα project ανεξάρτητη από όλες τις υπόλοιπες οντότητες, αν και μέσω του συστήματος και του δευτέρου σταδίου αρχικοποίησης, η οντότητα project συνδέεται με την οντότητα phases. Η πιο σκουρόχρωμη γραμμή στο διάγραμμα υποδηλώνει την εξάρτηση μιας οντότητας από την άλλη, οπότε στην συγκεκριμένη περίπτωση, ένα project δεν είναι απαραίτητο να έχει φάσεις, αλλά μια φάση πρέπει οπωσδήποτε να έχει ένα project για να ορίζεται, καθώς δεν νοείται φάση χωρίς project. Η σχέση μεταξύ project και φάσης είναι σχέση “1 προς πολλά” (one-to-many), που σημαίνει πως ένα project μπορεί να έχει πολλές φάσεις, αλλά μια φάση μπορεί να έχει μόνο ένα project, για αυτό όπως αναφέρθηκε παραπάνω, γίνεται και η χρήση του foreign key project_id στο πίνακα phases.
Η οντότητα phase από την άλλη, σχετίζεται και με τις τρεις άλλες οντότητες της βάσης. Η πρώτη συσχέτιση είναι με την οντότητα project που αναφέρθηκε μόλις και η δεύτερη συσχέτιση έρχεται με τον υπάλληλο. Στην συγκεκριμένη σχέση, ούτε ο υπάλληλος εξαρτάται από την ύπαρξη φάσης, ούτε η φάση από την ύπαρξη υπαλλήλου, αν και σε αυτή την περίπτωση επίσης, μέσω του τρίτου σταδίου αρχικοποίησης πρακτικά κάθε φάση έχει τελικά υπάλληλους που είναι «occupied» σε

αυτήν. Η σχέση μεταξύ τους είναι μια σχέση “πολλά προς πολλά” (many-to-many), αφού ένας υπάλληλος μπορεί να παίρνει μέρος σε πολλές φάσεις και μια φάση μπορεί να έχει πολλούς υπάλληλους. Η τρίτη συσχέτιση έρχεται με την οντότητα payment. Σε αυτή την σχέση η ύπαρξη payment προϋποθέτει την ύπαρξη φάσης, ενώ η ύπαρξη φάσης θεωρητικά δεν προϋποθέτει την ύπαρξη payment (πρακτικά όμως δεν ισχύει). Η σχέση μεταξύ τους είναι σχέση “ένα προς πολλά” (one-to-many), που σημαίνει πως μια φαση μπορεί να έχει πολλές πληρωμές, ενώ μια πληρωμή πρέπει να έχει μια συγκεκριμένη φαση, δεδομένου ότι αναφέρεται σε κάποιον υπάλληλο, οπότε αυτόματα δημιουργείται και η συσχέτιση μεταξύ οντότητας payment και οντότητας employee. Σε αυτή τη σχέση η ύπαρξη πληρωμής δεν είναι απαραίτητη για την ύπαρξη υπαλλήλου, αφού ένας υπάλληλος μπορεί να μην έχει πάρει ποτέ μέρος σε κάποια φαση, αλλά η ύπαρξη πληρωμής προϋποθέτει την ύπαρξη υπαλλήλου. Η σχέση μεταξύ τους είναι επίσης «ένα προς πολλά», αφού ένας υπάλληλος μπορεί να έχει περισσότερες από μια πληρωμές, ενώ μια πληρωμή αναφέρεται σε ακριβώς έναν υπάλληλο (employee-id) της δεδομένης φάσης (phase-id)

## *3.5 Τεχνολογίες αναπτυξης εφαρμογής*

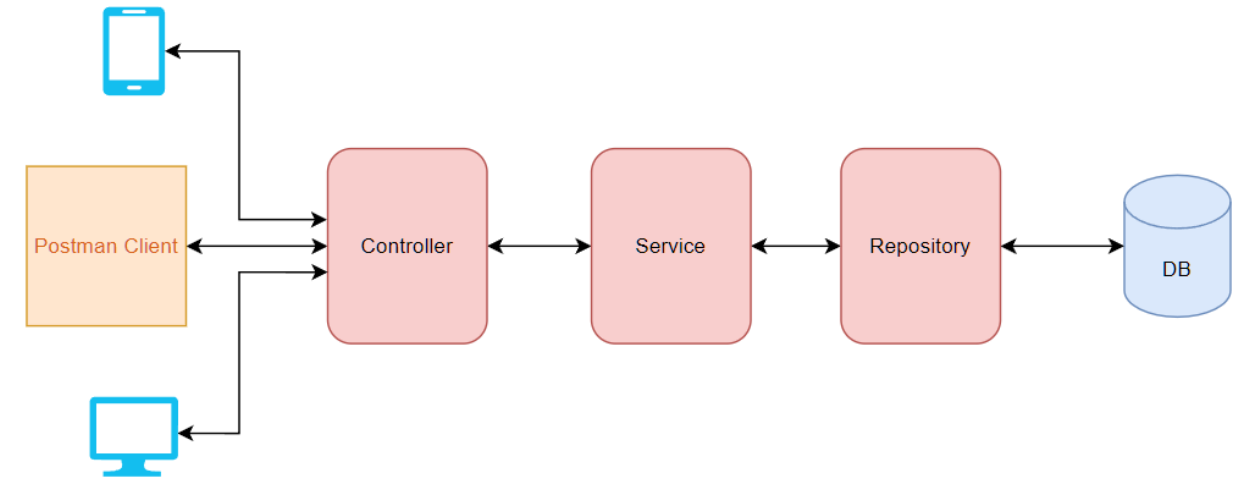
Για την βάση δεδομένων της εφαρμογής επιλέχθηκε όπως αναφέρθηκε και παραπάνω η χρήση της MySQL για λογούς απλότητας σταθερότητας και ευκολίας. Για το πίσω μέρος της εφαρμογής, ή αλλιώς backend, χρησιμοποιήθηκε το Spring Boot [34]. Το Spring Boot είναι ένα framework της Java, που είναι πολύ ευέλικτο και ισχυρό και που διευκολύνει το backend κομμάτι στην ανάπτυξη εφαρμογών. Η έντονη ενεργητικότητα που παρατηρείται, από τα μεγάλα έργα και τις επιχειρήσεις καθώς και η πληθώρα πλεονεκτημάτων που προσφέρει το framework αυτό, έκαναν την επιλογή του ευκολότερη και με την χρήση της δόθηκε και μεγαλύτερη αξία στην ίδια της εφαρμογή αλλά και στην προσπάθεια υλοποίησης της. Το Spring Boot είναι δημοφιλές από μικρές έως μεγάλες επιχειρήσεις και χρησιμοποιείται για web apps και εφαρμογές λογισμικού. Φημίζεται για τις λειτουργίες που προσφέρει και που επιταχύνουν τη διαδικασία ανάπτυξης, αλλά και για την εύκολη ρύθμιση της, παρέχοντας πολλές προεπιλεγμένες ρυθμίσεις, που εξοικονομούν χρόνο από τον επιπλέον κώδικα.

Παρέχονται πολλές αυτοματοποιημένες ενέργειες που κάνουν ευκολότερη την σύνδεση με την βάση δεδομένων αλλά και για την επεξεργασία των αιτημάτων και την ασφάλεια της εφαρμογής. Για την σύνδεση της MySQL βάσης δεδομένων με το Spring Boot, χρησιμοποιήθηκε η Spring Data JPA. Το framework αυτό της Spring διευκολύνει και εξοικονομεί πολύτιμο χρόνο στην ανάπτυξη της βάσης δεδομένων, των οντοτήτων και των συσχετίσεων αφού αυτοματοποιεί τις οντότητες, απλά σε κλάσεις, όπου για κάθε οντότητα της βάσης θα υπάρχει η αντίστοιχη κλάση.

Ως προς τη διάρθρωση του back end, η διάκριση γίνεται σε 4 μέρη. Ο πρώτος φάκελος models περιέχει τα classes των οντοτήτων, που όπως έχει ήδη αναλυθεί είναι τέσσερις, αρά και τέσσερα classes. Ο δεύτερος φάκελος repositories περιέχει τις κλάσεις με τα JPA repositories που παρέχουν εύκολη πρόσβαση στις αντίστοιχες οντότητες της βάσης δεδομένων. Ο τρίτος φάκελος services περιέχει όλες τις classes services και τις classes services implementations κάθε οντότητας. Ο φάκελος αυτός είναι υπεύθυνος να παρέχει όλες τις λειτουργίες της εφαρμογής για αυτό και συνήθως χρησιμοποιούνται τα repositories των models εδώ. Ο τελευταίος φάκελος controllers, περιέχει τις κλάσεις controller κάθε οντότητας και υλοποιούν την επιχειρησιακή λογική της

εφαρμογής. Οι controllers είναι υπεύθυνοι για την επεξεργασία των αιτημάτων από τους χρήστες και την επιστροφή των αποτελεσμάτων.

Με τις δεδομένες απαιτήσεις της εφαρμογής τα περισσότερα Api Requests ήταν τύπου Post και Get, οπότε η εφαρμογή ακολούθησε τις αρχές των RESTful Apis και τα δεδομένα που μεταφέρονται μεταξύ του πελάτη και του εξυπηρετητή έχουν την μορφή json. Για την άμεση και εύκολη δοκιμή της βάσης δεδομένων συνολικά, των υπηρεσιών και των controllers που κατασκευάστηκαν, χρησιμοποιήθηκε το Postman. Μια από τις βασικότερες λειτουργίες που χαρακτηρίζουν το Postman είναι η ευκολία της εκτέλεσης οποιουδήποτε τύπου HTTP Request σε οποιοδήποτε API, χωρίς την χρήση γραφικού περιβάλλοντος. Έτσι αναλόγως της απάντησης που θα επιστρέψει το εργαλείο, ο προγραμματιστής μπορεί να εντοπίσει συγκεκριμένα errors και να προλάβει αστοχίες στο back end.




*Σχήμα 9. Spring Data REST Architecture (created with draw.io)*

Η χρήση του framework Spring Boot απαιτούσε εξίσου μια εξελιγμένη τεχνολογία και γλώσσα προγραμματισμού, τόσο στο front end της εφαρμογής. Ένα από τα πιο δημοφιλή εργαλεία για την ανάπτυξη επαγγελματικών και διαδραστικών ιστοσελίδων και εφαρμογών είναι η React.js [35]. Η React είναι μια βιβλιοθήκη της Javascript που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη (UIs). Επιτρέπει τη σύνταξη JSX, δίνοντας την δυνατότητα χρήσης HTML και Javascript στον ίδιο κώδικα αλλά το μεγαλύτερο πλεονέκτημα είναι ότι χρησιμοποιεί components για να δομήσει το UI μιας εφαρμογής. Αυτά τα components μπορούν να είναι επαναχρησιμοποιούμενα σε διάφορα μέρη της εφαρμογής μειώνοντας το μέγεθος του κώδικα και της επανάληψης και το UI ανταποκρίνεται δυναμικά στις αλλαγές των components, χωρίς την ανάγκη για επαναφόρτωση της σελίδας.
Επίσης, χρησιμοποιήθηκε το εργαλείο διαχείρισης πακέτων npm (Node Package Manager) για την εγκατάσταση και την διαχείριση των απαραίτητων πακέτων που χρειάστηκε η εφαρμογή, όπως και της React βιβλιοθήκης. Για τα αιτήματα που δημιουργούνται στο front end χρησιμοποιήθηκε η βιβλιοθήκη Axios, γιατί μπορεί να χρησιμοποιεί ασύγχρονες λειτουργίες και να επεξεργαστεί σφάλματα η να ακυρώσει αιτήσεις αλλά και γιατί υποστηρίζει διάφορες μορφές δεδομένων όπως των JSON που μεταφέρουν τα Apis. Στο κομμάτι του front-end και της React χρησιμοποιήθηκε CSS για την μορφοποίηση των στιλιστικών παραμέτρων αλλά έγινε και αρκετή χρήση της βιβλιοθήκης Material UI.

# *Κεφάλαιο 4: Λειτουργίες και διεπαφή χρήστη (UI/UX)*

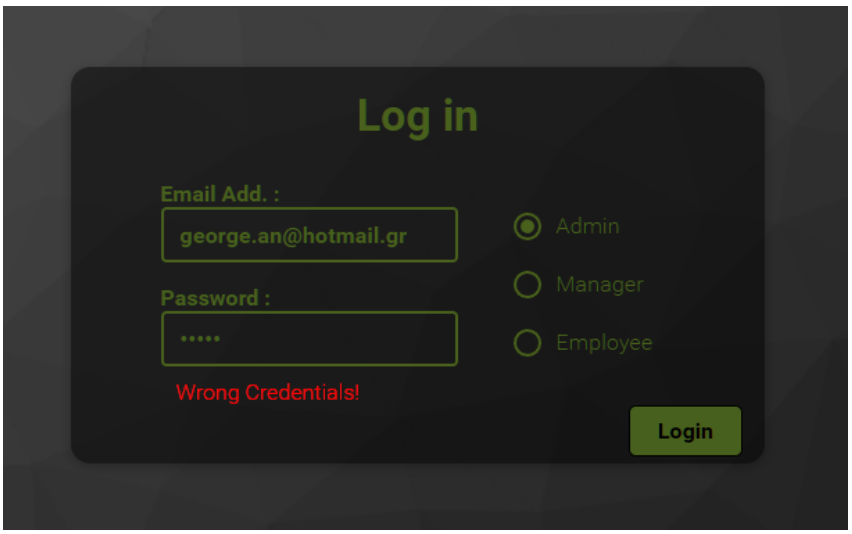
Στο κεφάλαιο αυτό θα περιγραφεί ο τρόπος λειτουργίας της εφαρμογής που αναπτύχθηκε στο πλαίσιο της διπλωματικής εργασίας. Θα γίνει εκτενής ανάλυση στο γραφικό περιβάλλον, το οποίο δημιουργήθηκε και θα αναδειχθούν όλες οι λειτουργίες των χρηστών. Κύρια στοιχεία του συγκεκριμένου κεφαλαίου θα είναι τα ενδεικτικά screenshots που θα περιλαμβάνονται, προκειμένου ο αναγνώστης να μπορέσει να κατανοήσει πλήρως όλες τις διαδικασίες και να ξεκαθαριστούν οποιαδήποτε κενά του έχουν δημιουργηθεί έως τώρα. Όλες οι λειτουργίες και οι ιδιαιτερότητες τους θα αποσαφηνιστούν και θα αναλυθούν και διαφορετικά σενάρια σε αυτές τις λειτουργίες.

Ξεκινώντας με την ανάλυση των απαιτήσεων του περιβάλλοντος του συστήματος, η εφαρμογή πρέπει να είναι όσο πιο απλή γίνεται στον χρήστη, να είναι καλοσχεδιασμένη και εύστοχη. Οι διάφορες λειτουργίες θα πρέπει να εμφανίζονται με τρόπο έξυπνο και λιτό, έτσι ώστε το περιβάλλον της εφαρμογής να μην είναι πολύπλοκο και να εξασφαλίζει την εύκολη κατανόηση των ενεργειών που πρέπει η μπορεί να κάνει ο χρήστης. Για αυτό τον λόγο, η πρώτη σελίδα που εμφανίζεται, που είναι η σύνδεση των χρηστών, είναι σχεδιασμένη όσο πιο απλά και ξεκάθαρα γίνεται. Επίσης και οι σελίδες που ακολουθούν μετά την σύνδεση του εκάστοτε χρήστη είναι σχεδιασμένες για να παρέχουν τις κατάλληλες πληροφορίες και ενέργειες και μπορούν να παρομοιαστούν με σελίδες τύπου dashboard.

Πέραν αυτών των σελίδων, θεωρήθηκε πως η χρήση modal components θα έδινε μια καλύτερη εμπειρία χρήσης και θα έκανε την εφαρμογή πιο οργανωμένη και πλοηγήσιμη, οπότε όλες οι υπόλοιπες λειτουργίες αναδύονται κάθε φορά με το αντίστοιχο modal.

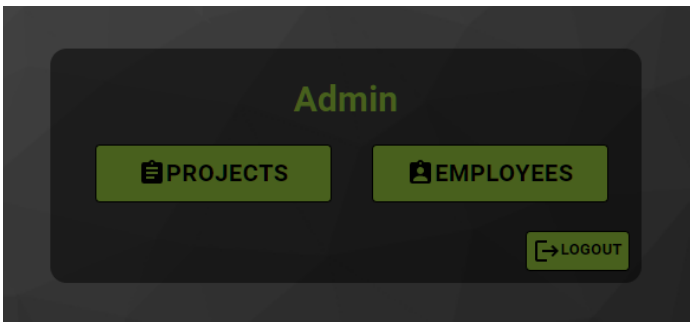
Αρχικά ο χρήστης με την έναρξη της χρήσης της εφαρμογής, βλέπει την αρχική σελίδα της, που είναι η σελίδα σύνδεσης. Εκεί ο χρήστης έχει την επιλογή να επιλέξει ρόλο χρήσης του συστήματος και με την μορφή radio buttons, εμφανίζονται οι επιλογές Admin, Manager, Employee.

Επίσης τα πεδία συμπλήρωσης εμφανίζονται με την μορφή text field και η εφαρμογή ενημερώνει κατάλληλα τον χρήστη με ένα μήνυμα σε κόκκινο χρώμα, σε περίπτωση λανθασμένων στοιχείων.



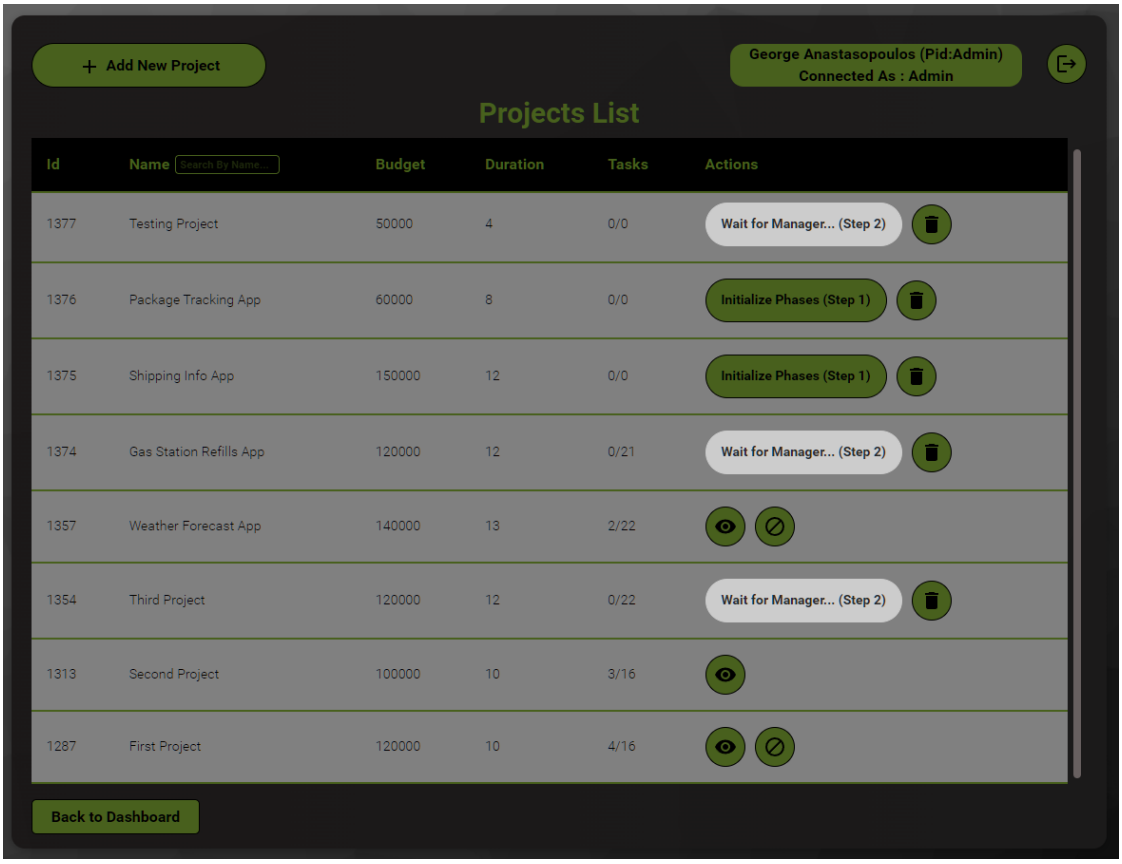

*Εικόνα 1. Σελίδα Σύνδεσης/ Αρχική Σελίδα*

Από το back end της εφαρμογής και των κατάλληλων Api requests, ελέγχεται η εγκυρότητα των στοιχείων του χρήστη και έπειτα από την επιτυχή σύνδεση του, ανάλογα με την ιδιότητα που έχει επιλέξει, μεταφέρεται και στην αντίστοιχη σελίδα. Λαμβάνοντας υπόψιν τις απαιτήσεις και τους περιορισμούς που αναλύθηκαν στο κεφάλαιο 3.2, οι αρχικές οθόνες της εκάστοτε ιδιότητας κατασκευάστηκαν με τρόπο τέτοιο ώστε να εμφανίζουν στον εκάστοτε χρήστη τις επιθυμητές πληροφορίες και ενέργειες. Σε γενικές γραμμές, οι σελίδες και των τριών χρηστών είναι παρόμοιες ως προς την δομή τους, καθώς οι λειτουργίες τους μοιάζουν η τουλάχιστον ο τρόπος εμφάνισης κάποιων από αυτές είναι παρόμοιος. Η μόνη διαφορά είναι πως στην περίπτωση του Admin, που έχει την επιλογή να δει και να επεξεργαστεί υπαλλήλους, μετά την επιτυχή σύνδεση μεταφέρεται σε μια σελίδα που έχει να επιλέξει αναμεσά στα projects και τους employees.




*Εικόνα 2. Υπομενού για τον Διαχειριστή*

Επιλέγοντας ένα από τα δυο, ο admin θα ανακατευθυνθεί στην αντίστοιχη αρχική σελίδα και συγκεκριμένα αν επιλέξει το κουμπί projects, θα μεταφερθεί στην σελίδα των projects, όπως γίνεται στην περίπτωση σύνδεσης των άλλων δυο ιδιοτήτων. Να σημειωθεί πως, ανάλογα την σύνδεση του χρήστη, η σελίδα των projects θα εμφανίζει μόνο τα projects που αφορούν τον καθένα, οπότε στην περίπτωση του admin θα εμφανίζονται όλα τα projects που έχουν εκτελεστεί στην εφαρμογή.



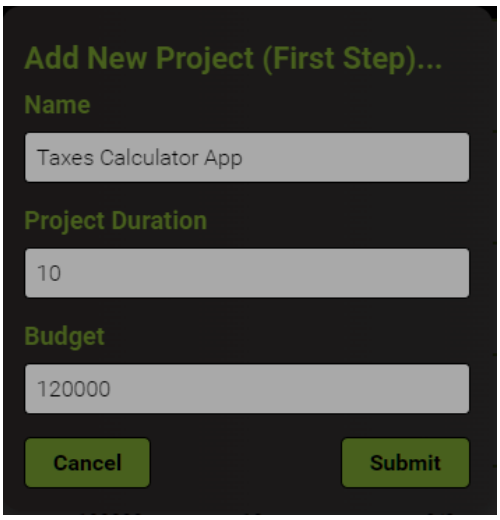

*Εικόνα 3. Αρχική σελίδα Διαχειριστή*

Για την αρχική σελίδα των projects, αναγκαία και χρήσιμη ήταν η προβολή αυτών σε μία αναλυτική λίστα. Έτσι με ένα Api get request, η εφαρμογή δέχεται όλα τα project που υπάρχουν στη βάση, οπότε η λίστα περιλαμβάνει τα πολύ βασικά στοιχεία μόνο, με πιο σημαντικό να είναι το πεδίο actions για το καθένα από αυτά. Το πεδίο actions περιλαμβάνει τις διαθέσιμες λειτουργίες που μπορεί ο χρήστης να επιλέξει, ανάλογα βέβαια με τον ρόλο σύνδεσης και κυρίως το εκάστοτε status του project. Στην περίπτωση της Εικόνας [3] , ο χρήστης έχει συνδεθεί ως Admin, οπότε έχει τη δυνατότητα της αρχικοποίησης των φάσεων, της κατάργησης, της διαγραφής και της παρακολούθησης του project, ενώ η δυνατότητα επιλογής υπαλλήλων εμφανίζεται ως disabled, καθώς είναι λειτουργία-υποχρέωση του manager. Ο χρήστης ενημερώνεται κατάλληλα από το πεδίο πάνω δεξιά της εικόνας, για τον ρόλο με τον οποίο έχει συνδεθεί στο σύστημα και ανά πάσα στιγμή ο χρήστης έχει την δυνατότητα να κάνει logout.

Στην περίπτωση του Admin, κάτω από την λίστα εμφανίζεται και ένα κουμπί Back to Dashboard, που δίνει την δυνατότητα στον admin να επιστρέψει στην αρχική του σελίδα, ενδεχομένως για να επιλέξει να μεταφερθεί στην αρχική σελίδα των employees, που μόνο αυτός έχει πρόσβαση.

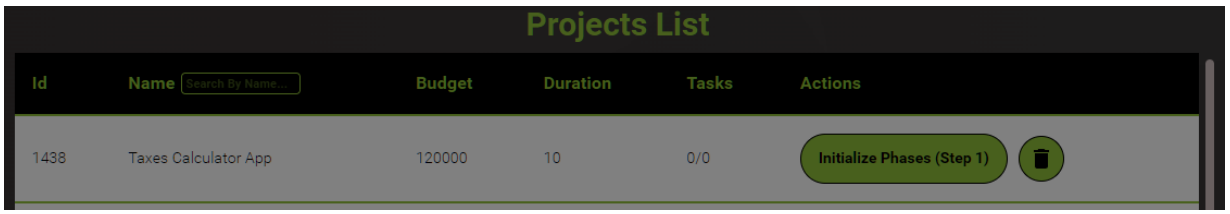
Επίσης για την προσθήκη νέου Project και την εκτέλεση του πρώτου σταδίου αρχικοποίησης του, έχει τοποθετηθεί το κατάλληλο custom button πάνω αριστερά από τη λίστα. Να σημειωθεί πως αυτό το κουμπί θα εμφανίζεται στην αρχική σελίδα μόνο του admin, καθώς μόνο αυτός είναι αρμόδιος για αυτήν τη λειτουργία, όπως προκύπτει από τις απαιτήσεις του συστήματος.

Με την επιλογή του κουμπιού αναδύεται ένα modal με την μορφή φόρμας, όπου ο admin πληκτρολογεί τα βασικά στοιχεία του project στα ανάλογα text fields.



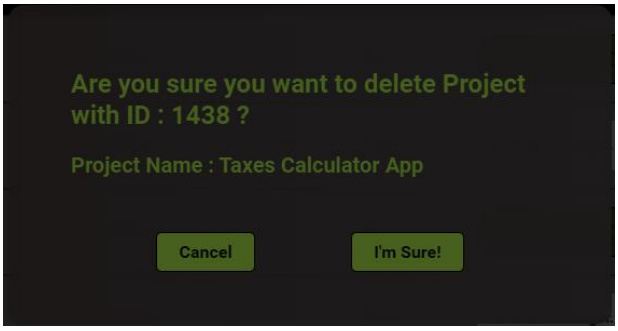

*Εικόνα 4. Φόρμα προσθήκης νέου project*

Με την ολοκλήρωση της συμπλήρωσης, ο χρήστης καλείται να επιλέξει το κουμπί submit για να αποθηκευτεί το project στη βάση δεδομένων και να ολοκληρώσει το πρώτο στάδιο αρχικοποίησης. Μόλις μεταφερθεί πάλι στην αρχική σελίδα των projects, η λίστα ανανεώνεται αυτόματα και πλέον είναι ορατό και το νέο project σε αυτήν. Να σημειωθεί πως το κουμπί submit είναι clickable μόνο εάν έχουν συμπληρωθεί τα στοιχεία σωστά και πως ο χρήστης έχει την δυνατότητα να ακυρώσει την διαδικασία με την επιλογή του κουμπιού cancel.



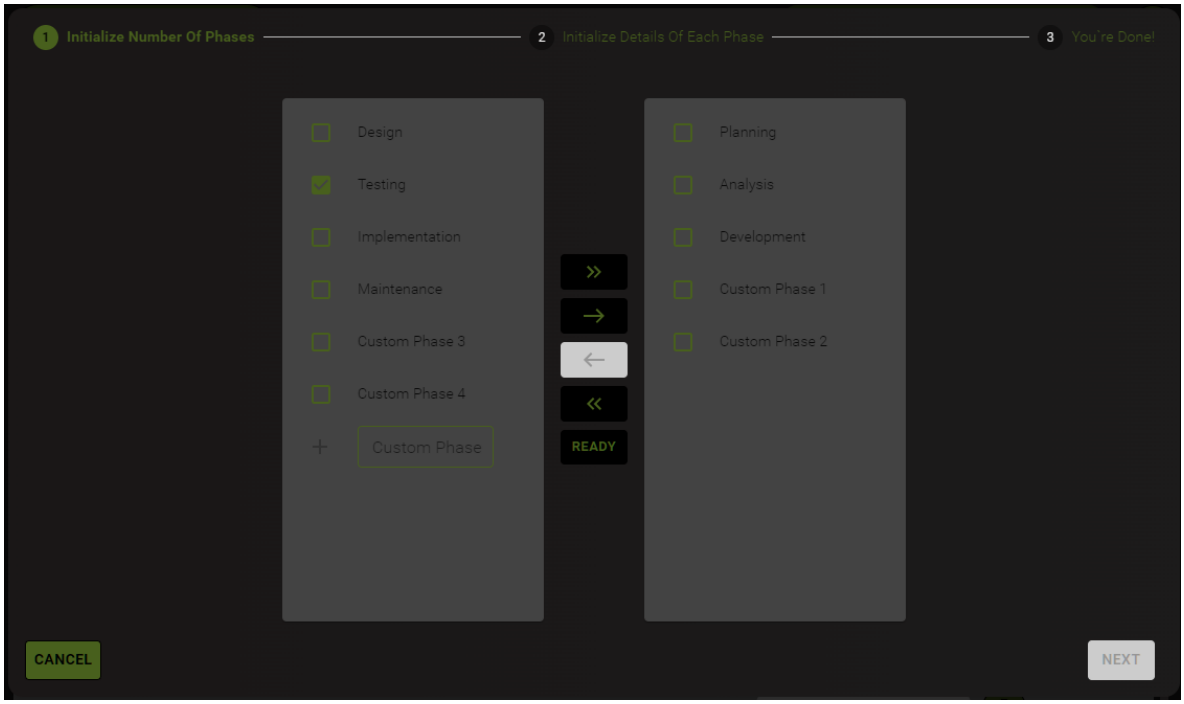

*Εικόνα 5. Ανανεωμένη με το νέο project λίστα*

Όπως φαίνεται στην Εικόνα [5], το project πλέον είναι εμφανές μαζί με τα υπόλοιπα και το πεδίο actions του, περιέχει δυο buttons. Με την επιλογή του δευτέρου button, ο admin μπορεί να διαγράψει το συγκεκριμένο project, μέσω του modal που εμφανίζεται και του αντίστοιχου Api delete request που γίνεται στο back end της εφαρμογής.




*Εικόνα 6. Modal διαγραφής Project*

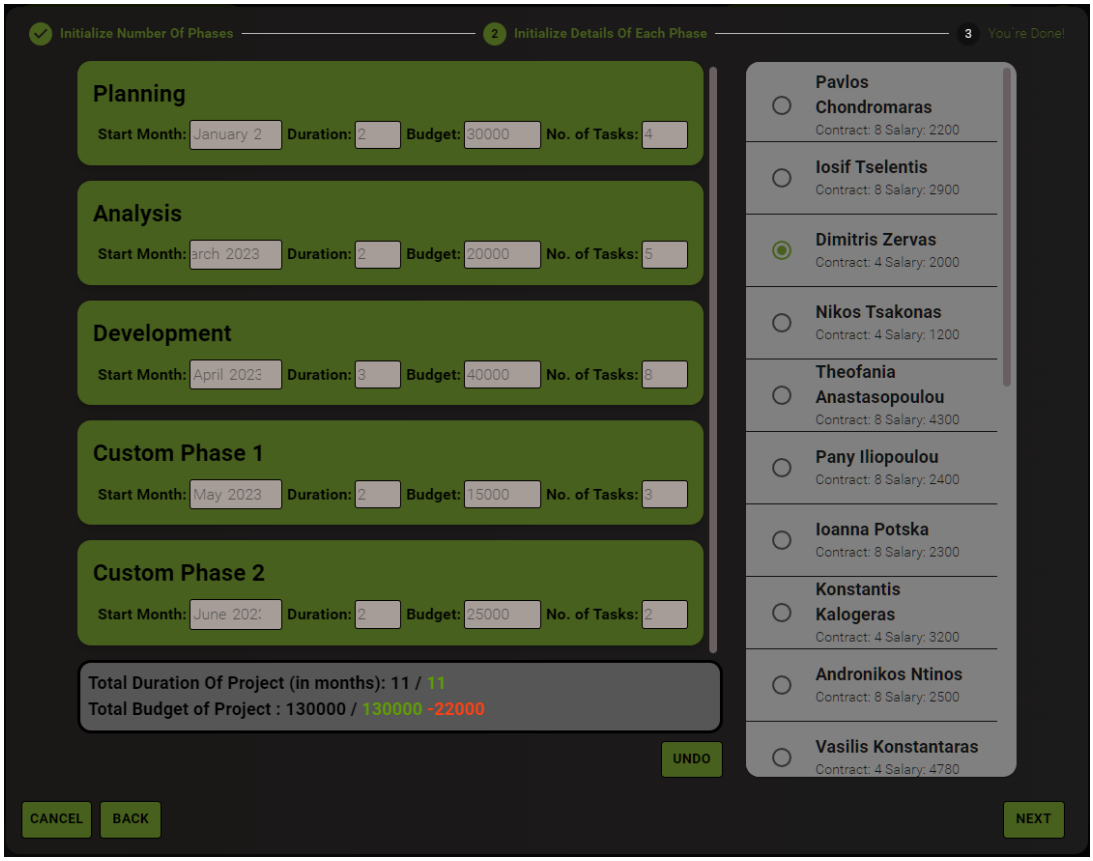
Με την επιλογή του πρώτου button, ο admin μπορεί να συνεχίσει με την αρχικοποίηση του project και να προσθέσει τις φάσεις, τις λεπτομέρειες τους και τον manager του έργου. Με τη χρήση παραθύρου modal, το πρώτο πράγμα που πρέπει να κάνει ο admin είναι να επιλέξει τις φάσεις που θέλει. Όπως έχει αναφερθεί στις απαιτήσεις του συστήματος, ο χρήστης μπορεί είτε να επιλέξει από τις ήδη προτεινόμενες φάσεις που παρέχει το σύστημα και ακολουθούν το μοντέλο waterfall του SDLC, είτε να προσθέσει και δικές του custom φάσεις στις ήδη υπάρχουσες. Χρησιμοποιήθηκε λοιπόν μια transfer list, όπου στην αριστερή λίστα είναι οι διαθέσιμες προς επιλογή φάσεις και στην δεξιά λίστα είναι οι επιλεγμένες φάσεις. Ο σχεδιασμός του γραφικού περιβάλλοντος με τέτοιο τρόπο, εξασφαλίζει την σαφήνεια της διαδικασίας και κάνει αυτονόητο άμεσα στο χρήστη τι πρέπει να κάνει, οπότε δεν μπορούν να δημιουργηθούν παρανοήσεις η να προκύψουν λάθη.




*Εικόνα 7. Επιλογή των Φάσεων του Project*

Παραδείγματος χάριν όπως φαίνεται στην Εικόνα [7], ο admin, πέραν των προκαθορισμένων φάσεων, έχει προσθέσει και τέσσερις custom φάσεις. Από αυτές τις φάσεις έχει επιλέξει τελικά τρεις φάσεις από τις προκαθορισμένες και δυο από τις custom φάσεις, όπως φαίνεται από τη δεξιά λίστα. Το button «Ready», αρχικά είναι disabled, καθώς το σύστημα έχει περιορίσει τον αριθμό των επιλεγμένων φάσεων σε περισσότερες από τέσσερις. Όταν ο χρήστης μπορεί και επιλέξει το button «Ready», η λίστα "κλειδώνει" και γίνεται enabled το button «Next», προκειμένου ο admin να συνεχίσει με την αρχικοποίηση των στοιχείων της κάθε φάσης από τις επιλεγμένες. Να σημειωθεί πως το button «ready», αλλάζει και γίνεται button «Edit», σε περίπτωση που ο χρήστης θέλει να αλλάξει κάτι πριν προχωρήσει στο επόμενο βήμα.

Αφού λοιπόν επιλέξει «Next», μεταφέρεται στο επόμενο στάδιο, όπου του προβάλλεται μια φόρμα, όπου για κάθε φαση του προηγουμένου σταδίου, περιέχει τα αντίστοιχα text fields, για την αρχικοποίηση των φάσεων. Τα text fields αφορούν το μήνα εκκίνησης, τη διάρκεια, τον προϋπολογισμό και τον αριθμό των tasks, της κάθε φάσης. Επίσης, το σύστημα ενημερώνει κατάλληλα τον χρήστη για το εάν οι πληροφορίες που έχει προσθέσει συμφωνούν με τις απαιτήσεις του project μέσω του ανάλογου πεδίου. Μόλις ο admin επιλέξει το button «Ready», κλειδώνουν τα πεδία συμπλήρωσης και εμφανίζεται δίπλα από την φόρμα μια λίστα με υπάλληλους. Από αυτή τη λίστα ο διαχειριστής καλείται να επιλέξει έναν από αυτούς για Manager του Project και μόλις το κάνει, ενεργοποιείται το button «Next», που εξαρχής ήταν disabled. Επίσης, το σύστημα ενημερώνει και για την συνολική πληρωμή του επιλεγμένου manager για το project, σε σχέση με το συνολικό προϋπολογισμό του, όπως φαίνεται και στην Εικόνα [8].

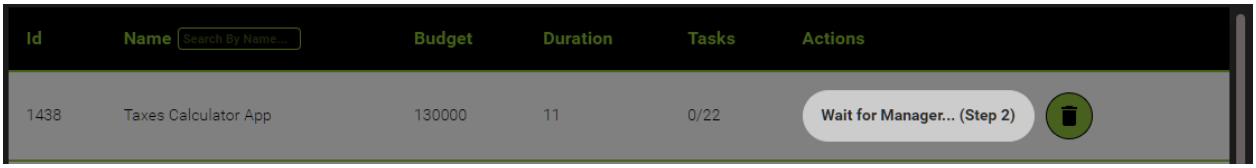



*Εικόνα 8. Μενού αρχικοποίησης φάσεων και επιλογή μάνατζερ*

Μόλις ο admin τελειώσει και με την επιλογή manager και πατήσει το button «Next», ουσιαστικά τερματίζει και η διαδικασία της δεύτερης φάσης αρχικοποίησης και ο χρήστης μεταφέρεται στη σελίδα του επομένου βήματος όπου προβάλλεται μια σύνοψη των πληροφοριών που καταχωρήθηκαν έως τώρα.

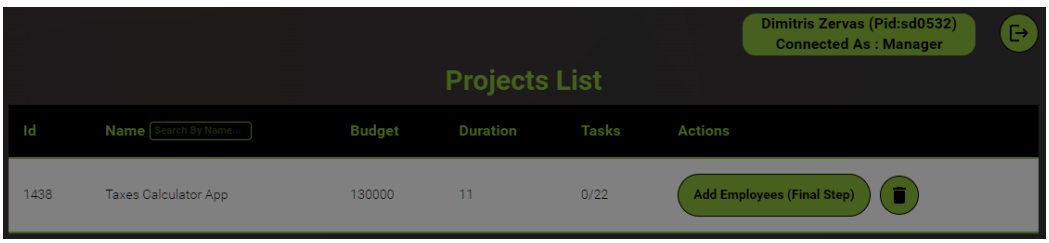

*Εικόνα 9. Επισκόπηση της 2ης φάσης, όπως αυτή έχει αρχικοποιήσει από τον διαχειριστή*

Με την επιλογή κλεισίματος της τελικής καρτέλας, κλείνει και το modal και ο admin μεταφέρεται στην αρχική σελίδα των Project όπου βλέπει το ενημερωμένο project πλέον στη λίστα με τα υπόλοιπα. Το πεδίο tasks ενημερώθηκε με τα συνολικά task των φάσεων όπως επίσης και το πεδίο actions. Στο πεδίο actions, πέραν του button διαγραφής, πλέον βρίσκεται και ένα disabled button που αναφέρεται στο τρίτο στάδιο αρχικοποίησης του project. Ο λόγος για τον οποίο φαίνεται ως disabled είναι επειδή αυτή η λειτουργία είναι διαθέσιμη μόνο για τον ορισμένο manager του project.



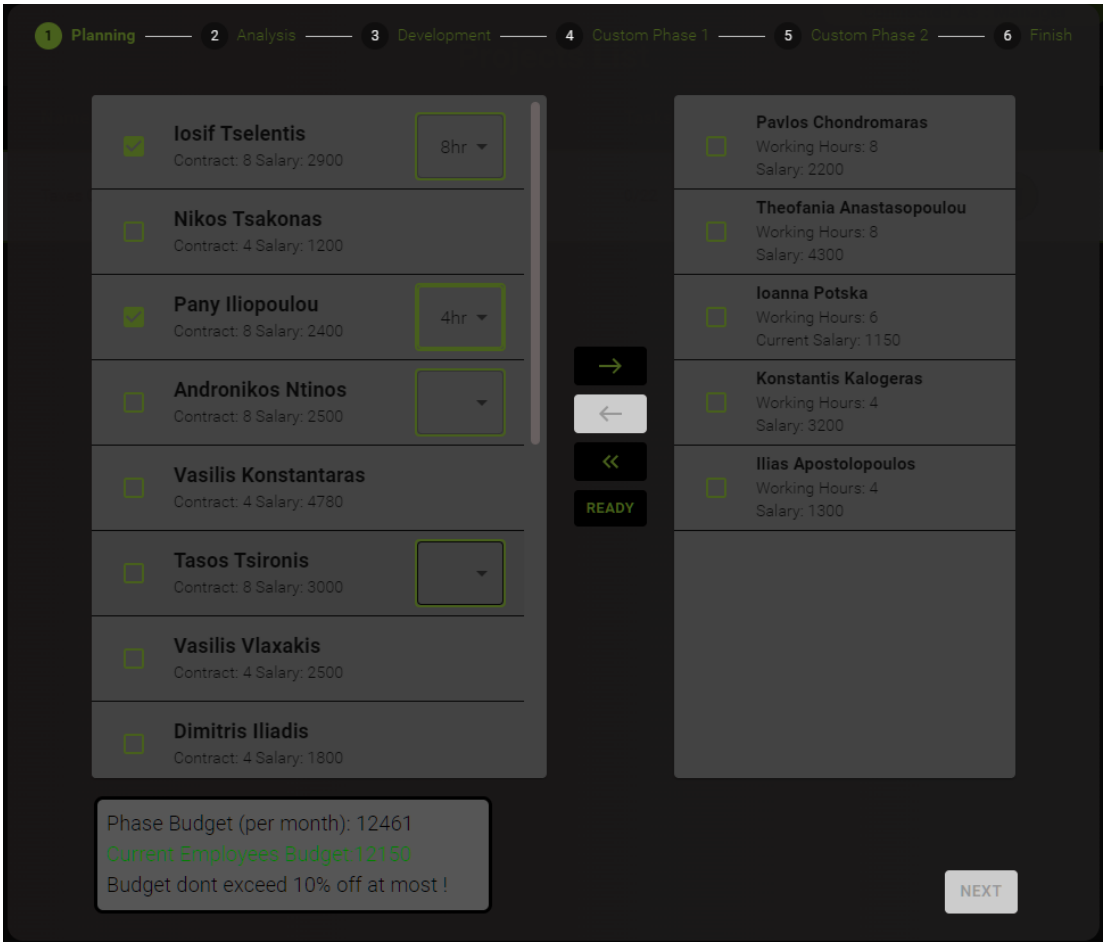

*Εικόνα 10. Μέρος της λίστας με το ενημερωμένο project*

Οπότε για να συνεχιστεί η διαδικασία αρχικοποίησης, θα πρέπει ο συγκεκριμένος υπάλληλος να κάνει login ως manager. Εκεί από την αρχική σελίδα και την λίστα των projects, ο Manager βλέπει το Project όπως φαίνεται στην παρακάτω εικόνα.




*Εικόνα 11. Το ίδιο project από την αρχική σελίδα του manager*

Με την επιλογή της ενέργειας αρχικοποίησης του τρίτου σταδίου του project, ο manager θα πρέπει να καταχωρήσει υπαλλήλους στο Project, μια διαδικασία πολύ σημαντική, οπότε η ανάγκη ευκολίας και απλότητας ήταν πρωταρχικού ρόλου. Για αυτό τον λόγο στο επόμενο modal, η διαδικασία χωρίστηκε σε στάδια και συγκεκριμένα σε τόσα, όσες και οι φάσεις, προκειμένου να διαιρεθεί σε πιο απλά και κατανοητά μέρη. Έτσι εμφανίζεται ένα modal στον manager, όπου παρατηρεί πάλι τη χρήση transfer list.

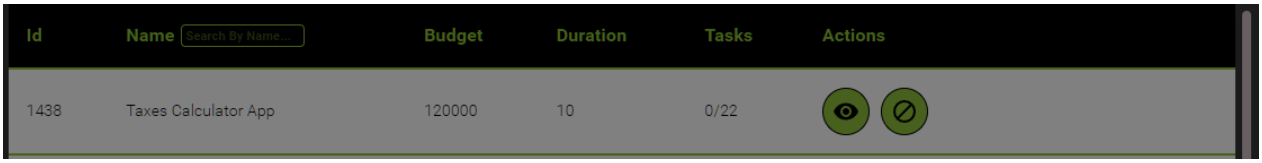



*Εικόνα 12. Επιλογή υπαλλήλων για την κάθε φαση*

Πιο συγκεκριμένα στην αριστερή λίστα εμφανίζονται οι διαθέσιμοι, για την χρονική περίοδο της φάσης, υπάλληλοι και στην δεξιά λίστα οι αντίστοιχοι επιλεγμένοι για αυτήν. Το σύστημα κρατά ενήμερο τον χρήστη σχετικά με τον προϋπολογισμό και τις πληρωμές των επιλεγμένων υπαλλήλων καθόλη την διάρκεια της διαδικασίας, για να κάνει πιο εύκολη την επιλογή των καταλληλότερων υπαλλήλων. Επίσης το σύστημα δίνει την επιλογή στον manager, να ορίσει υπαλλήλους από 8ωρους σε 4ωρους. Μόλις επιλέξει τους υπαλλήλους, επιλέγει το button «Ready» και αμέσως «Next» και εκτελείται ένα Api post request προς τη βάση δεδομένων για να αποθηκεύσει τους επιλεγμένους υπαλλήλους και συνεχίζει στην αρχικοποίηση των επομένων φάσεων ακριβώς με τον ίδιο τρόπο που παρουσιάστηκε μόλις.

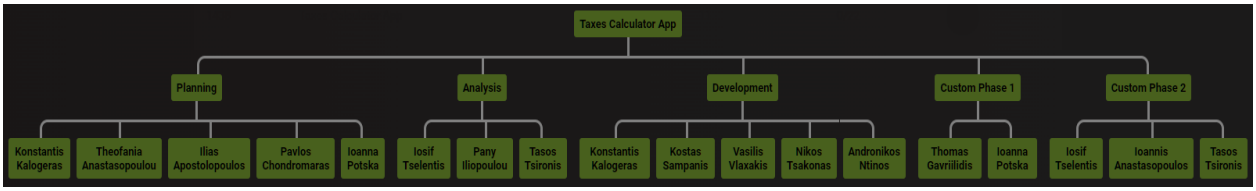
Μόλις τελειώσει με όλες τις φάσεις ο manager, τερματίζει το τρίτο και τελευταίο στάδιο αρχικοποίησης του Project και ενημερώνεται κατάλληλα η λίστα της αρχικής σελίδας με το ανανεωμένο Project. Πιο συγκεκριμένα, πλέον ο κάθε χρήστης, ανεξαίρετου ρόλου θα μπορεί να

παρακολουθήσει το Project από το ανανεωμένο πεδίο actions. Για τον admin μόνο , δίνεται και η δυνατότητα τερματισμού του project επίσης, από το δευτερο button που παρέχεται.



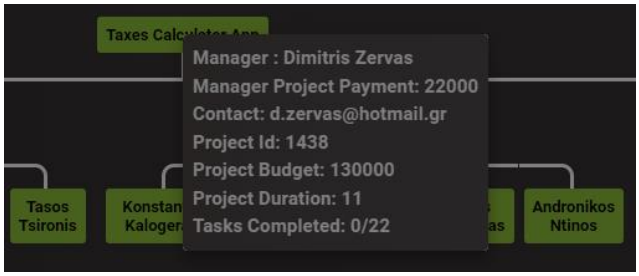

*Εικόνα 13. Τελική μορφή εμφάνισης του αρχικοποιημένου πλέον project στην λίστα*

Επιλέγοντας ο χρήστης την προβολή του project, πρέπει να μπορεί να δει συγκεντρωτικά και οργανωμένα όλες τις πληροφορίες οι οποίες συλλέχθηκαν. Το σύστημα δημιουργήθηκε με τέτοιο τρόπο ώστε, μέσω των διαγραμμάτων που χρησιμοποιήθηκαν, ο χρήστης να έχει μια πρακτική και γραφική εικόνα των φάσεων, των υπαλλήλων και των χρονικών περιθωρίων του project. Όπως και στις προηγούμενες περιπτώσεις, έτσι και εδώ εμφανίζεται ένα modal το οποίο περιέχει δύο διαγράμματα, ενός hierarchy tree και ενός διαγράμματος Gannt. Το συγκεκριμένο hierarchy tree κατασκευάστηκε ώστε να χρησιμοποιεί σαν κυρίους κόμβους, τον κόμβο project σε πρώτο επίπεδο, με παρακλάδια του τους κόμβους phases σε δεύτερο επίπεδο, και σε τρίτο επίπεδο τους employees που αφορούν την κάθε φάση.



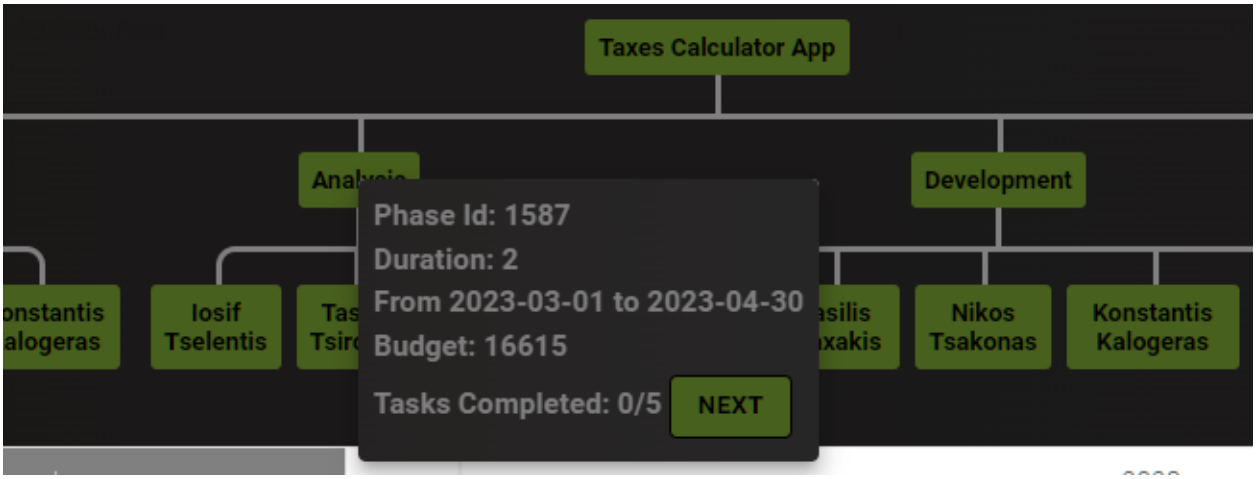

*Εικόνα 14. Hierarchy Tree*

Επίσης έχει δοθεί η δυνατότητα στον χρήστη να μπορεί να εμφανίζει τα στοιχεία κάθε κόμβου ή κάθε υπαλλήλου κάνοντας click πάνω στον αντίστοιχο κόμβο, όπως για παράδειγμα με τον πρώτο κόμβο στην Εικόνα [15].



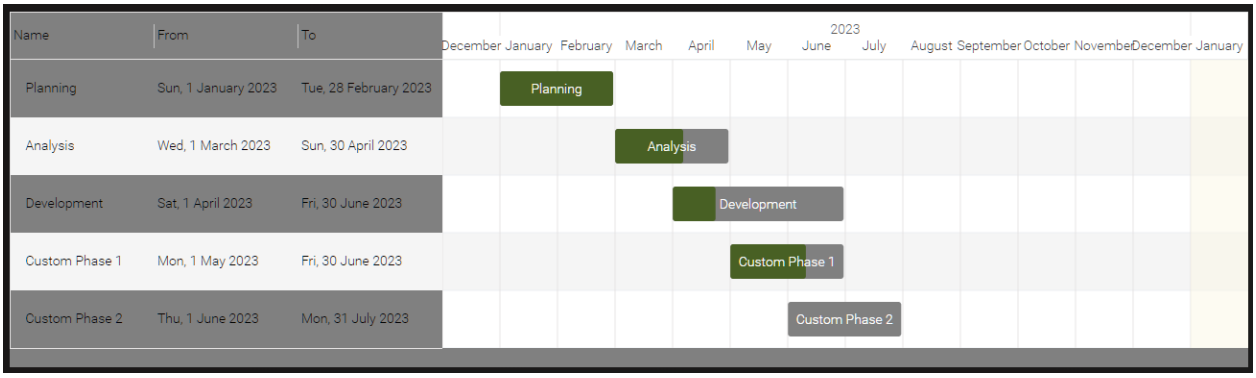

*Εικόνα 15. Λεπτομέρειες επιλεγμένου κόμβου*

Στην περίπτωση που η σύνδεση και η προβολή γίνονται από τον υπεύθυνο manager, το σύστημα δίνει την επιλογή σε αυτόν, κάνοντας click πάνω σε οποιονδήποτε κόμβο των φάσεων, να επιλέξει μέσω του κουμπιού next, αλλαγή του task και συνέχεια στο επόμενο, προσφέροντας μια διαδραστικότητα κατά την διαδικασία ανάπτυξης, καθώς ο manager θα ενημερώνει την πλατφόρμα για την πρόοδο κάθε φάσης μέσω των tasks της καθεμίας. Αξίζει να σημειωθεί, πως μετά την ολοκλήρωση όλων των tasks όλων των φάσεων, ολοκληρώνεται και το project συνολικά και αυτόματα ενημερώνεται και το status του ως ολοκληρωμένο.




*Εικόνα 16. Επιλεγμένος κόμβος με δυνατότητα διάδρασης με τα task*

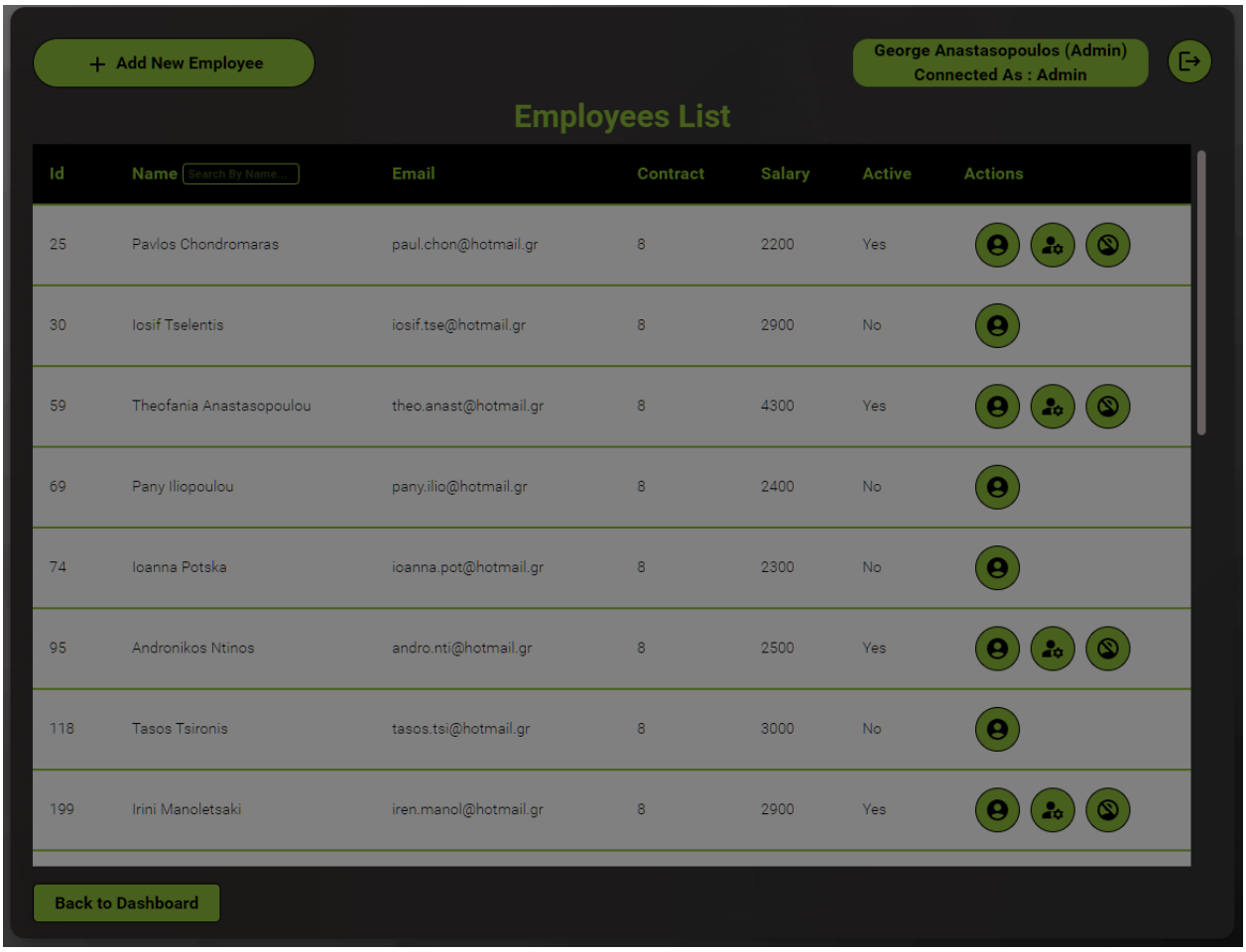
Η αλλαγή του task της φάσης είναι άμεσα αντιληπτή μέσω του διαγράμματος Gannt που παρέχεται ακριβώς κάτω από το hierarchy tree. Στην Εικόνα [17], φαίνονται όλες οι φάσεις και οι χρονικές τους απαιτήσεις όπως επίσης και η πρόοδος σε σχέση με τα tasks, με την μορφή bar μέσα στο διάγραμμα.




*Εικόνα 17. Gannt Diagram*

Όπως είχε αναφερθεί παραπάνω, ο διαχειριστής με την επιτυχή σύνδεση του μεταφέρεται σε μια σελίδα, οπού έχει να επιλέξει αναμεσά στα projects, που αναλύθηκε ήδη, αλλά και στους employees. Αν επιλέξει τους employees, θα ανακατευθυνθεί στην αρχική σελίδα των employees

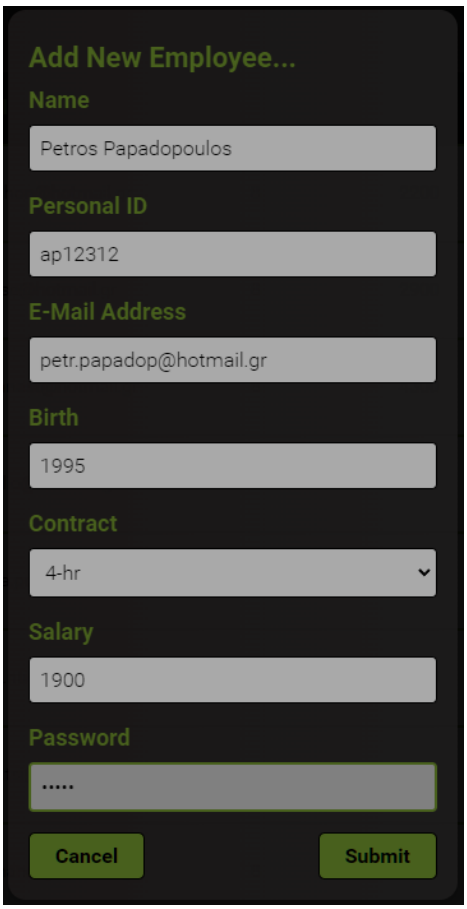
όπου εμφανίζεται μια λίστα με όλους τους υπαλλήλους, που έχουν απασχοληθεί από την εταιρεία, ενεργούς και μη.




*Εικόνα 18. Λίστα Υπαλλήλων*

Όπως και με την αρχική σελίδα των projects, έτσι και σε αυτή την αρχική, απαραίτητη ήταν η προβολή και η οργάνωση των υπαλλήλων σε μια αναλυτική λίστα. Η λίστα περιλαμβάνει τα απαραίτητα στοιχεία, αλλά και τα μισθολογικά, όπως επίσης και ένα πεδίο active και actions. Στο πεδίο actions περιέχονται τρία κουμπιά με την μορφή iconbutton και τα δυο πεδία αυτά σχετίζονται με το αν ο υπάλληλος είναι ενεργός η όχι την τωρινή στιγμή στο σύστημα. Και στις δύο περιπτώσεις, περιλαμβάνεται η λειτουργία της προβολής των λεπτομερειών του υπαλλήλου, ενώ αν αυτός είναι ενεργός, διαθέσιμες είναι και οι λειτουργίες της επεξεργασίας και της αποδέσμευσης του. Επίσης ο admin ενημερώνεται κατάλληλα από το πεδίο πάνω δεξιά της Εικόνας [18], για τον ρόλο με τον όποιο έχει συνδεθεί στο σύστημα και ανά πάσα στιγμή έχει τη δυνατότητα να κάνει logout, μέσω του αντίστοιχου iconbutton.

Για την προσθήκη νέου υπαλλήλου στο σύστημα, έχει τοποθετηθεί κατάλληλο custom button πάνω αριστερά από την λίστα και με την επιλογή του κουμπιού αναδύεται ένα modal με τη μορφή φόρμας, όπου ο admin μπορεί να πληκτρολογήσει τα απαραίτητα και απαιτούμενα στοιχεία του νέου υπαλλήλου στα ανάλογα textfields.



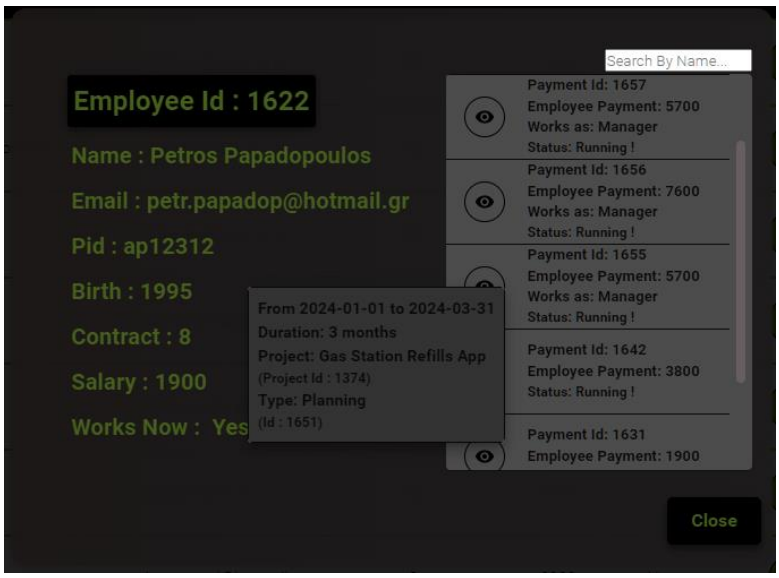

*Εικόνα 19. Φόρμα εκχώρησης νέου υπαλλήλου*

Με την ολοκλήρωση της συμπλήρωσης, ο admin καλείται να επιλέξει το κουμπί submit για να προσθέσει το νέο υπάλληλο στο σύστημα και να μεταφερθεί πάλι στην αρχική σελίδα των employees, όπου πλέον θα είναι ορατός και στην αρχική λίστα. Να σημειωθεί πως το κουμπί submit είναι clickable μόνο εάν έχουν συμπληρωθεί τα στοιχεία σωστά και πως ο χρήστης έχει την δυνατότητα να ακυρώσει τη διαδικασία με την επιλογή του κουμπιού cancel.


*Εικόνα 20. Εμφάνιση του εκχωρημένου υπαλλήλου στην λίστα υπαλλήλων*

Όπως φαίνεται στην Εικόνα [20], ο employee πλέον εμφανίζεται μαζί με τους υπολοίπους και το πεδίο actions του, περιέχει τα τρία icon buttons που αναφέρθηκαν παραπάνω.

Με την επιλογή του πρώτου iconbutton, ο admin μπορεί να δει όλες τις λεπτομέρειες του συγκεκριμένου υπαλλήλου μέσω του modal που εμφανίζεται. Στο modal, από την αριστερή πλευρά εμφανίζονται τα στοιχεία του υπαλλήλου και για την δεξιά πλευρά δημιουργήθηκε μια λίστα με όλες τις πληρωμές που έχει λάβει. Για την λίστα έχει τοποθετηθεί και ένα search textfield, όπου ο διαχειριστής μπορεί να αναζητήσει στις πληρωμές του με βάση το όνομα του project η με βάση τη φάση όπου, εξ ορισμού από το σύστημα, για κάθε φάση που έχει συμμετάσχει, αντιστοιχεί και μία πληρωμή. Επίσης, για κάθε πληρωμή, πέρα από τα βασικά της στοιχεία, όπως για παράδειγμα το αν εκτελούσε χρέη manager, ο admin μπορεί να επιλέξει το αντίστοιχο κουμπί προβολής επιπλέον πληροφοριών, προκειμένου να έχει καλύτερη εικόνα σχετικά με την συγκεκριμένη πληρωμή.



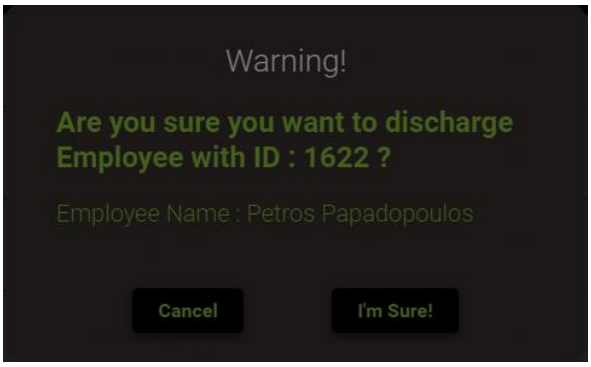

*Εικόνα 21. Modal προβολης υπαλληλου με επιλεγμενη πληρωμη*

Με την επιλογή του δευτέρου icon button, ο admin έχει τη δυνατότητα να τροποποιήσει τα στοιχεία ενός υπαλλήλου. Αυτή η λειτουργία εξυπηρετεί και την ανανέωση των προσωπικών στοιχείων του υπαλλήλου, αλλά και την αύξηση του μισθού του η την αλλαγή της σύμβασης του από part-time σε full-time και αντίστροφα. Δημιουργήθηκε λοιπόν μια φόρμα παρόμοια με αυτή της προσθήκης υπαλλήλου που αναφέρθηκε παραπάνω, με την μόνη προφανή αλλαγή ότι έρχεται συμπληρωμένη με τα στοιχεία που ελήφθησαν από το back end της εφαρμογής και του αντίστοιχου Api get request.


*Εικόνα 22. Modal επεξεργασίας υπαλλήλου*

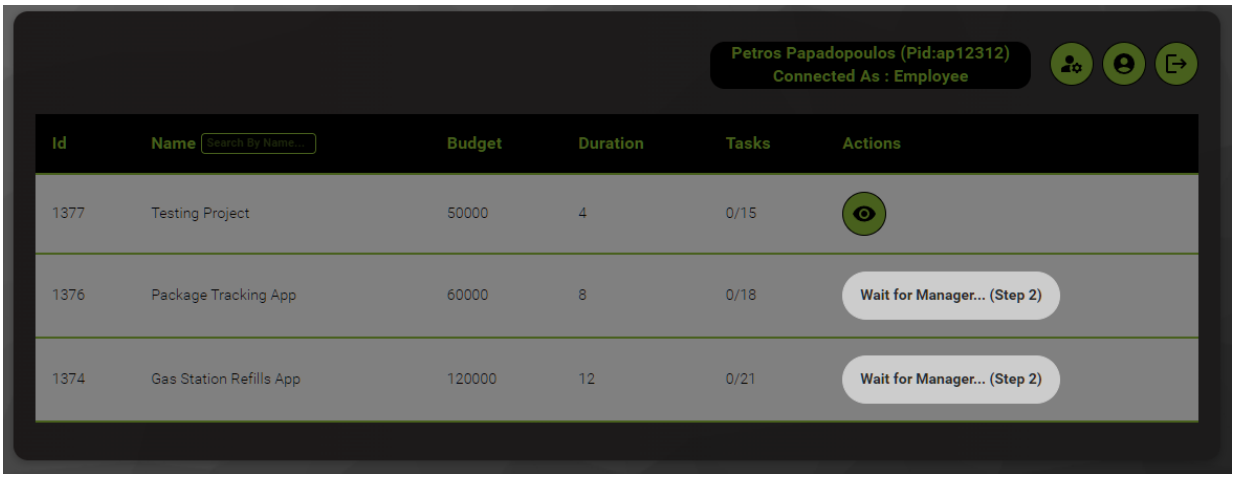
Μόλις ο διαχειριστής ολοκληρώσει την τροποποίηση των στοιχείων επιλέγει το κουμπί Edit και τα νέα δεδομένα του υπαλλήλου αποθηκεύονται εκ νέου στη βάση δεδομένων, αλλά μπορεί να πάσα στιγμή να ακυρώσει την τροποποίηση μέσω και ενός κουμπιού Cancel.

Με την επιλογή του τρίτου icon button, ο admin έχει την δυνατότητα να αποδεσμεύσει τον υπάλληλο από το σύστημα. Επί της ουσίας και όπως ήδη έχει αναφερθεί, ο υπάλληλος δεν διαγράφεται, αλλάζει μόνο το state του στην βάση δεδομένων μέσω του αντίστοιχου Api request. Με την χρήση ενός warning modal, ο admin ερωτάται για την απόφαση του και επιλέγει το κουμπί Sure, αλλιώς επιλέγει Cancel και ακυρώνει τη διαδικασία.




*Εικόνα 23. Διαγραφή υπαλλήλου*

Η τελευταία αρχική σελίδα του συστήματος είναι αυτή που ο χρήστης συνδέεται ως employee. Και σε αυτή την τελευταία περίπτωση, η αρχική κατασκευάστηκε με ίδιο τρόπο με τις προηγούμενες αρχικές, με τις μόνες αλλαγές να είναι στο περιεχόμενο που έχει πρόσβαση ο υπάλληλος. Έτσι λοιπόν στην λίστα που του εμφανίζεται, περιέχονται μόνο τα projects στα οποία έχει λάβει μέρος. Η μόνη λειτουργία που του είναι διαθέσιμη είναι μόνο η προβολή των project και των πληροφοριών και διαγραμμάτων τους. Επίσης η εφαρμογή έπρεπε να παρέχει τη δυνατότητα ελέγχου των πληρωμών και των στοιχείων από τον ίδιο τον εκάστοτε υπάλληλο, οπότε τοποθετήθηκε κατάλληλο icon button στο πάνω δεξιά μέρος της αρχικής σελίδας, όπως και ένα icon button δίπλα του που σχετίζεται με το logout. Το modal της προβολής των στοιχείων του και των πληρωμών του είναι ίδιο με το modal της προβολής υπαλλήλου που κατασκευάστηκε από την πλευρά του admin.




*Εικόνα 24. Αρχική σελίδα υπαλλήλου*

Το τρίτο icon button, αφορά την επεξεργασία των προσωπικών στοιχείων του υπαλλήλου από τον ίδιο, οπότε με τον ίδιο τρόπο με τον admin και ο υπάλληλος, μόλις επιλέξει αυτό το icon button, θα δει ένα modal με την ίδια φόρμα επεξεργασίας με του admin, αφού στην ουσία το μόνο που αλλάζει, είναι η πρόσβαση που έχει στα ως προς επεξεργασία text fields.


*Εικόνα 25. Modal επεξεργασίας στοιχείων υπαλλήλου*

Συνοψίζοντας, συνολικά το γραφικό περιβάλλον της εφαρμογής ακολούθησε μια λιτή σχεδίαση, προκειμένου να είναι εύκολα προσπελάσιμα και κατανοητά όλα τα στάδια και οι ενέργειες στον χρήστη. Η χρήση της σχεδιαστικής γραμμής των modals, επιτυγχάνει την καθαρότητα στις λειτουργίες, απομονώνει τις επιμέρους διαδικασίες που λαμβάνουν μέρος, ενώ η αρχική σελίδα υπάρχει πάντα στο προσκήνιο και παραμένει όσο πιο καθαρή και απλή γίνεται. Το logout button είναι άμεσα διαθέσιμο σε όλες τις σελίδες και συνολικά τα κουμπιά που τοποθετήθηκαν στην εφαρμογή είναι ευδιάκριτα και αλλάζουν κατάλληλα, κατά το hover η την κατάσταση ondisabled. Όπου κρίθηκε απαραίτητο σε πολλά σημεία της εφαρμογής τοποθετήθηκαν τα καταλληλά cancel buttons ενώ με την κατάσταση ondisabled κυρίως σε submit τύπου buttons, αποφεύχθηκε η πρόωρη υποβολή στοιχείων, οπότε και λειτουργούν σαν ένα στάδιο ελέγχου εγκυρότητας. Έλεγχος εγκυρότητας επίσης έχει προστεθεί στις φόρμες εισαγωγής μέσω των text fields. Οι αρχικές λίστες που δημιουργήθηκαν, από το 8$^{ο}$ αντικείμενο και έπειτα γίνονται scrollable, ενώ όπου θεωρήθηκε απαραίτητο τοποθετήθηκε και ένα text field αναζήτησης της εκάστοτε λίστας. Η εφαρμογή, όπου είναι αναγκαίο, έχει προνοήσει και προβάλει τα αντίστοιχα error messages, προλαμβάνοντας και επιδεικνύοντας λάθη στο χρήστη.

# Κεφάλαιο 5: Επίλογος

Σε αυτό το κεφάλαιο, θα συζητηθούν τα συμπεράσματα που βγήκαν από την μελέτη του SDLC, αλλά και σε σχέση με τη διαδικασία ανάπτυξης της εφαρμογής και πως αυτή εξελίχθηκε. Συμπληρωματικά θα γίνει αναφορά και στις μελλοντικές επεκτάσεις της εφαρμογής και πως θα μπορούσε να αναβαθμιστεί περαιτέρω.

## 5.1 Συμπεράσματα

Η συγκεκριμένη διπλωματική βασίστηκε από την μια, στη μελέτη του SDLC, ενώ από την άλλη, σε επίπεδο αναπτυξης, βασίστηκε στην κατασκευή ενός συστήματος το οποίο να προσαρμόζεται σε αυτό. Επίσης η ανάπτυξη της εφαρμογής, ακολούθησε και η ίδια μια μεθοδολογία βασισμένη στο SDLC. Το ενδιαφέρον δηλαδή με την συγκεκριμένη διπλωματική, ήταν ότι ταυτόχρονα αφορούσε τη μελέτη, την υιοθέτηση αλλά και την δημιουργία εφαρμογής για το θέμα που πραγματευόταν. Έπειτα λοιπόν από τη μελέτη και την κατανόηση των φάσεων και των μεθοδολογιών και με βάση αυτά, έπρεπε να γίνει επιλογή του κατάλληλου μοντέλου για την διαδικασία αναπτυξης της εφαρμογής. Η εφαρμογή λοιπόν, ακολούθησε μια μεθοδολογία που μπορεί να πει κάνεις ότι παρομοιάζει την μέθοδο Agile, αφού αναπτύχθηκε με μικρές επαναλήψεις, με κάθε επανάληψη να βελτιώνει την προηγουμένη, μέχρι να φτάσει στο τελικό επιθυμητό προϊόν.

Δεδομένου ότι οι λειτουργικές απαιτήσεις του συστήματος δεν ήταν γνωστές από την αρχή, το μοντέλο Agile, μέσω των επαναλήψεων, επέτρεπε σε κάθε νέα επανάληψη την ενημέρωση των αρχικών βασικών απαιτήσεων με νέες αλλά και τον επαναπροσδιορισμό τους σε περίπτωση

λάθους. Οπότε το γεγονός ότι οι απαιτήσεις θα άλλαζαν αλλά και το γεγονός ότι δεν υπήρχε ούτε συγκεκριμένος προϋπολογισμός αλλά και ούτε συγκεκριμένα χρονικά περιθώρια, έκαναν ιδανικό το έδαφος για την Agile.

Στην πρώτη φάση ανάπτυξης έγινε η ανάλυση και ο καθορισμός των βασικών και σημαντικών απαιτήσεων της εφαρμογής, έγινε η επιλογή των τεχνολογιών, ενώ το πρώτο sprint αφορούσε το σχεδιασμό της βασικής μορφής της εφαρμογής. Καθώς οι τεχνολογίες που επιλέχθηκαν ήταν νέες και προϋπέθεταν χρόνο για τη μάθηση τους, οι πρώτες επαναλήψεις όπως είναι φυσικό ήταν πιο χρονοβόρες και απαιτούσαν περισσότερη δοκιμή στο κομμάτι του κώδικα. Μάλλον, για να γίνει πιο σωστή διατύπωση, οι επαναλήψεις, όσο περνούσαν, είχαν όλο και πιο μικρή διάρκεια, αφού από τη συνεχή ανάπτυξη, η χρήση τους γινόταν πιο εύκολη.

Στις επόμενες φάσεις ανάπτυξης, η εφαρμογή αναβαθμιζόταν όλο και πιο πολύ, με περαιτέρω λειτουργίες να προστίθενται και να δοκιμάζονται. Έπειτα από τη σχεδίαση της βασικής μορφής, συνέχεια είχε η δημιουργία της σελίδας login και των αρχικών σελίδων των χρηστών που θα χρησιμοποιούσαν την εφαρμογή. Από αυτή τη λειτουργία δημιουργήθηκε η απαίτηση ενός μέσου προβολής των εργαζομένων και των project. Οπότε και η επόμενη επανάληψη αφορούσε τη δημιουργία κατάλληλων λιστών.

Στην συνέχεια, αφιερώθηκε αρκετός χρόνος στην προσθήκη δεδομένων στην εφαρμογή. Βήμα προς βήμα, σε μικρές επαναλήψεις, δημιουργήθηκαν και οι λειτουργίες προσθήκης υπαλλήλων και project στο σύστημά. Το στάδιό αυτό αποδείχτηκε απαιτητικό και πιο συγκεκριμένα, η λειτουργία προσθήκης νέων projects, καθώς έπρεπε να προσδιοριστεί συγκεκριμένος τρόπος που θα γινόταν αρχικοποίηση των Projects με φάσεις και στην συνέχεια οι φάσεις με υπαλλήλους. Χάρη στο γεγονός ότι η Agile λειτουργεί με δοκιμές σε κάθε επανάληψη και ότι αυτή η λειτουργία, που είναι και η πιο σημαντική αλλά και περίπλοκη, χωρίστηκε σε πολλές μικρές επαναλήψεις, έκανε την όλη διαδικασία πιο διαχειρίσιμη και μείωσε το συνολικό ρίσκο αποτυχίας της λειτουργίας εντελώς. Συχνά, κατά τη διάρκεια της αναπτυξης, παρατηρήθηκε η διαδοχή και η δημιουργία μιας νέας απαίτησης ή λειτουργίας αμέσως μετά την άλλη, δημιουργώντας μια ροή και μια συνέχεια στις επαναλήψεις.

Σημαντικό ρόλο στην όλη διαδικασία ανάπτυξης αλλά και υιοθέτησης του Agile, είχε και η επιλογή της React για το front end κομμάτι της εφαρμογής. Ο τρόπος που η React χειρίζεται τη δομή της γραφικής διεπαφής και της διαίρεσης της σε πολλά components και subcomponents, έκανε την προσθήκη λειτουργιών σε κάθε επανάληψη πολύ εύκολη και γρήγορη, χωρίς να δημιουργούνται ιδιαίτερες δυσκολίες και προβλήματα. Μια νέα ομάδα από components και μερικές μικρές αλλαγές στον ήδη υπάρχοντα κώδικα συνήθως ήταν αρκετά για να προστεθεί η νέα λειτουργία στην εφαρμογή, να δοκιμαστεί και μετά να συνεχίσει το έργο στο επόμενο “sprint”. Οπότε η Agile και η React συνδυάστηκαν με τέτοιο τρόπο, ώστε η ευελιξία της Agile να συμπληρώνει την οργανωμένη λογική της React. Στο back end κομμάτι επίσης της εφαρμογής, η Spring Boot λειτούργησε με τον ίδιο ακριβώς τρόπο. Οι απαιτήσεις δεν ήταν γνωστές εξαρχής, οπότε ούτε οι εισροές και οι εκροές του συστήματος. Η Spring Boot, έκανε πολύ εύκολη τη δημιουργία νέων οντοτήτων, νέων υπηρεσιών και νέων αιτημάτων ανά πάσα στιγμή και όποτε αυτό πρόκυπτε από τις απαιτήσεις του εκάστοτε “sprint”.

Συνολικά, η ανάπτυξη της εφαρμογής, αν και χωρίς συγκεκριμένες απαιτήσεις εξαρχής, χάρη στην διαχείριση των φάσεων και του χρόνου που υιοθέτησε από το μοντέλο Agile, κατάφερε αφενός να μειώσει το χρόνο των δοκιμών στο τέλος την ανάπτυξης αισθητά, αφού όποια δυσλειτουργία πρόκυπτε διορθωνόταν επί τόπου στο εκάστοτε στάδιό δοκιμής, αλλά και αφετέρου τελικά να ικανοποιεί τις ζητούμενες απαιτήσεις .

## *5.2 Μελλοντικές Επεκτάσεις*

Καθώς το πλαίσιο της διπλωματικής προέβλεπε μια βασική έκδοση ενός συστήματος διαχείρισης έργων, η ανάπτυξη λειτουργιών περιορίστηκε στις πολύ απαραίτητες, προκειμένου ο χρήστης να έχει μια συνοπτική εικόνα του τι είναι ικανό το SDLC και η εφαρμογή η ίδια να προσφέρουν σε αυτό και στον χρήστη και χωρίς παραπάνω λειτουργίες που θα περιέπλεκαν την εμπειρία χρήσης του.
Παρολαυτά, οι προοπτικές εξέλιξης και αναβάθμισης αυτής της εφαρμογής είναι πολλές. Πιο συγκεκριμένα, στη διαδικασία προσθήκης φάσεων έχει δοθεί επιλογή του waterfall model η και η προσθήκη custom φάσεων, προκειμένου να δημιουργεί το δικό του σχέδιο φάσεων ανάπτυξης. Ωστόσο μια αναβάθμιση σε αυτή την λειτουργία θα ήταν να δημιουργηθεί, μια αντίστοιχη επιλογή και για αλλά γνωστά μοντέλα, που να αρχικοποιεί, με τον αντίστοιχο για το κάθε μοντέλο τρόπο, τις φάσεις στο σύστημά. Αν και μέσω των custom φάσεων, ο χρήστης μπορεί κατά κάποιο και συγκεκριμένο τρόπο να επιλέξει Agile η Spiral η όποιο από τα μοντέλα αναλύθηκαν και καλύπτει και με τα τωρινά δεδομένα αυτή την ανάγκη, μια πιο ξεκάθαρη προσέγγιση αρχικοποίησης του κάθε μοντέλου, θα έκανε πιο εύκολη ακόμα την εμπειρία χρήσης του χρήστη.

Επιπροσθέτως, όπως έχει αναφερθεί και στις λειτουργίες της εφαρμογής στο υποκεφάλαιο 3.2, κατά τη διάρκεια επιλογής υπαλλήλων σε φάσεις, μπορεί να γίνει επιλογή 4ωρης απασχόλησης του υπαλλήλου για την διάρκεια της εκάστοτε φάσης. Μια αναβάθμιση μελλοντικά στο σύστημα θα ήταν η απασχόληση του υπαλλήλου ταυτοχρόνως σε δυο φάσεις ανάπτυξης για εκείνη την περίοδο, αντί της μιας που είναι τώρα, προκειμένου να καλύπτει το πλήρες ωράριο του συμβολαίου του.

Τέλος, παρόμοια με την αλλαγή των tasks για την εκάστοτέ φαση, που καλύπτεται ήδη από το σύστημά, μια αναβάθμιση θα ήταν αντίστοιχα η καταχώρηση παραδοτέων για κάθε υπάλληλο, για κάθε task της φάσης. Ακόμα η προσθήκη σχολίων, παρατηρήσεων και ανακοινώσεων για κάθε φάση ή για κάθε task από τους διαφορετικούς χρήστες του Project, θα έκανε την εφαρμογή να αποκτήσει ακόμα πιο διαδραστικό ύφος με τους χρήστες του.

Με αυτές τις επιπλέον προσθήκες, το σύστημα θα ήταν ακόμα πιο ολοκληρωμένο και θα μπορούσε να είναι το τέλειο εργαλείο οργάνωσης και διαχείρισης project για οποιαδήποτε μικρομεσαία επιχείρηση στον κλάδο.

# Βιβλιογραφία


[1] Shirley M. Radack (2009), The System Development Life Cycle (SDLC), ITL Bulletin, National Institute of Standards and Technology, Gaithersburg, MD, [online], https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=902622 (Accessed January 21, 2024)

[2] Andreja Velimirovic, “What is SDLC? Software Development Life Cycle Defined,” phoenixNAP Blog, Nov. 17, 2022. https://phoenixnap.com/blog/software-development-life-cycle (Accessed December 19, 2023)

[3] Kavita Sahu, Rajshree, Rajeev Kumar, “Risk Management Perspective in SDLC.”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCS), vol. 4, no. 3, pp. 1247-1251, March 2014

[4] Svitla Systems Inc, “System Development Life Cycle,” Svitla.com, Jan. 03, 2019. https://svitla.com/blog/system-development-life-cycle

[5] Noviana Pramitasari, Yova Ruldeviyani, Irvan Ramadhan Zarkasie, “Net impact implementation application development life-cycle management in banking sector,” Computer Science and Information Technologies, vol. 4, no. 2, pp. 169–182, July 2023, doi:10.11591/csit.v4i2.pp169-182.

[6] McMurtrey Mark, "A case study of the application of the systems development life cycle (sdlc) in 21st century health care: Something old, something new?." Journal of the Southern Association for Information Systems(JSAIS), vol. 1, no. 1, pp. 14-25, Winter 2013, doi:10.3998/jsais.11880084.0001.103

[7] Cicotti Giuseppe, "An evidence-based risk-oriented V-model methodology to develop ambient intelligent medical software." Journal of Reliable Intelligent Environments, vol. 3, no. 1, pp. 41-53, July 2017

[8] Carter Lemuria, Weerakkody Vishanth, Phillips Brandis, Dwivedi Yogesh, “Citizen Adoption of E-Government Services: Exploring Citizen Perceptions of Online Services in the US and UK.” Information Systems Management (ISM), vol 33, no. 2, pp. 124-140, February 2016, doi:10.1080/10580530.2016.1155948.

[9] Harrison Steve, Tzounis Antonis, Maglaras Leandros, Siewe Francois, Smith Richard, Janicke Helge, Helge “A Security Evaluation Framework for U.K. E-Government Services Agile Software Development.” International Journal of Network Security & Its Applications (IJNSA), vol 8, no. 2, pp. 51-69, March 2016, doi:10.5121/ijnsa.2016.8204.

[10] John Steer, Ashish Popli, “Building secure business applications at Microsoft.” Information Security Technical Report, vol 13, no. 2, pp. 105-110, May 2008, doi:10.1016/j.istr.2008.04.001.

[11] Sugiandi A, Kerlooza Yusrila, ”Competency Assessment Parameters for System Analyst Using System Development Life Cycle”. IOP Conference Series: Materials Science and Engineering (INCITEST), Bandung Indonesia, May 2018, vol 407, doi:10.1088/1757-899X/407/1/012143

[12] Yogeshwar Shastri, Rashida Hoda, Robert Amor, "Does the “Project Manager” Still Exist in Agile Software Development Projects?," 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), Hamilton, New Zealand, 2016, pp. 57-64, doi: 10.1109/APSEC.2016.019.

[13] Mark Preston, “7 Phases of the System Development Life Cycle Guide,” CloudDefense.AI, Sep. 14, 2023. https://www.clouddefense.ai/system-development-life-cycle/ (Accessed December 11, 2023)

[14] Irfan Ahmad Khan, Dipti Kumari, "The Role of Analysis Phase of SDLC for Small Scale Business Application-A Review." International Journal Of Humanities, Engineering, Science And Management (IJHESM), vol 2, no.1, pp. 63-75, June 2021

[15] Stuart Feldman, “Quality Assurance: Much More than Testing: Good QA is not only about technology, but also methods and approaches.”, vol. 3, no. 1, pp. 26-29, February 2005, doi:10.1145/1046931.1046943

[16] Kandl Susanne, Elshuber Martin, “A Formal Approach to System Integration Testing.” Proceedings of the Tenth European Dependable Computing Conference (EDCC 2014), April 2014, Available: https://arxiv.org/ftp/arxiv/papers/1404/1404.6743.pdf

[17] Ganesh Kushal, Mohapatra Sanjay, S.P. Anbuudayasankar, Sivakumar P., “User Acceptance Test.” Enterprise Resource Planning. Management for Professionals, pp. 123-127, June 2014, doi:10.1007/978-3-319-05927-3_9.

[18] Itti Hooda, Rajender Singh Chhillar, "Software test process, testing types and techniques." International Journal of Computer Applications (IJCA), vol 111, no. 13, pp. 10-14, February 2015, doi:10.5120/19597-1433

[19] Azzah A. Alghamdi, Mahmood Niazi, “Challenges of Secure Software Deployment: An Empirical Study”. 2022 26th International Conference on Evaluation and Assessment in Software Engineering (EASE '22), Gothenburg Sweden, June 2022, pp. 440–445. https://doi.org/10.1145/3530019.3531337

[20] Shray Khanna, Ashay Shah, Shubham Jain, Ramanathan L, "Software Maintenance: Challenges and Issues and Models for Reducing the Maintenance Cost." International Journal of Advanced Research in Computer Science (IJARCS), vol. 8, no. 3, pp. 1026-1032, April 2017, doi:10.26483/ijarcs.v8i3.3149

[21] PK. Ragunath, S. Velmourougan, P. Davachelvan, S. Kayalvizhi, R. Ravimohan, "Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC) ." International Journal of Computer Science and Network Security (IJCSNS), vol. 10, no. 1, pp. 112-119, January 2010

[22] AZM Ehtesham Chowdhury, Abhijit Bhowmik, Hasibul Hasan, Md Shamsur Rahim, "Analysis of the veracities of industry used software development life cycle methodologies." AIUB Journal of Science and Engineering (AJSE), vol. 16, no. 1, pp. 1-8, June 2017, doi:10.53799/ajse.v16i2.71.

[23] Kai Petersen, Clacs Wohlin, Dejan Baca, “The Waterfall Model in Large-Scale Development,” in Lecture Notes in Business Information Processing, Blekinge Institute of Technology, 2009, pp. 386–400, doi:10.1007/978-3-642-02152-7_29

[24] Halani Kenish Rajesh, Kavita Jhajharia, "A quantitative study of waterfall and agile methodologies with the perspective of project management." Contemporary Challenges for Agile Project Management. IGI Global, 2022, pp. 111-133, doi:10.4018/978-1-7998-7872-8.ch007

[25] Shravan Pargaonkar, “A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering”, International Journal of Scientific and Research Publications (IJSRP), vol. 13, no. 8, pp. 120-124, August 2023, doi:10.29322/IJSRP.13.08.2023.p14015

[26] Bishan Dayal Chauhan, Ajay Rana, Neeraj Kumar Sharma, "Impact of Development Methodology on Cost & Risk for Development Projects." 2017 6th International Conference

on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, September 2017, pp. 267-272, doi: 10.1109/ICRITO.2017.8342436

[27] Υπουργείο Ψηφιακού Μετασχηματισμού, «Βίβλος Ψηφιακού Μετασχηματισμού 2020-2025", Ιούνιος 2021, https://digitalstrategy.gov.gr/website/static/website/assets/uploads/digital_strategy.pdf

[28] Rajkumar, "V Model in Software Development Life Cycle," Software Testing Material, Apr. 18, 2016. https://www.softwaretestingmaterial.com/v-model-in-sdlc/

[29] Usman Adamu Sulaiman, Ogwueleka Francisca Nonyelum, "Application of the Systems Development Life Cycle (SDLC) in 21st Century Military Sector." Journal of Information Technology (IUP), vol. 14, no. 1, pp. 58-64, March 2018

[30] Seema Suresh Kute, Surabhi Deependra Thorat, " A Review on Various Software Development Life Cycle(SDLC) Models." International Journal of Research in Computer and Communication Technology (IJRCCT), vol. 3, no. 7, pp. 776-781, July 2014

[31] Nugroho Suryanto, Sigit Hadi Waluyo, Luqman Hakim, "Comparative analysis of software development methods between Parallel, V-Shaped and Iterative." International Journal of Computer Applications (IJCA), vol. 169, no. 11, pp. 7-11, July 2017, doi:10.5120/ijca2017914605

[32] Bittner Kurt, Spence Ian, "Managing Iterative Software Development Projects." Boston USA: Pearson Education, June 2006, ISBN: 0-321-26889-X

[33] Γιακουμάκης Εμμανουήλ, Διαμαντίδης Νικόλαος. " Τεχνολογία Λογισμικού", Αθήνα, Unibooks, Μάιος 2018, ISBN: 978-618-5304-41-6

[34] Dewailly Ludovic, "Building a RESTful web service with spring.", Birmingham, UK: Packt Publishing Ltd, 2015.

[35] Boduch Adam and Roy Derks, "React and React Native: A complete hands-on guide to modern web and mobile development with React.js.", Birmingham, UK: Packt Publishing Ltd, 2020.