ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Web Εφαρμογής για Διαχείριση Εξόδων** **Expenses Management Web Application Development**
Ονοματεπώνυμο Φοιτητή	**Αντωνία Βάθη**
Πατρώνυμο	**Νικήτας Βάθης**
Αριθμός Μητρώου	**ΜΠΠΛ18005**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περίληψη

Η παρούσα μεταπτυχιακή διατριβή πραγματεύεται την δημιουργία μιας διαδικτυακής εφαρμογής για την διαχείριση και έγκριση εξόδων, στα πλαίσια των παροχών που προσφέρουν οι επιχειρήσεις προς τους εργαζομένους.

Για την υλοποίηση της εφαρμογής, προσδιορίστηκαν τρία στάδια ανάπτυξης: το Στάδιο Αξιολόγησης Ιδεών, το Στάδιο Σχεδιασμού και το Στάδιο Υλοποίησης.

Στο Στάδιο Αξιολόγησης Ιδεών, προσεγγίζουμε και αξιολογούμε την ιδέα πάνω στην οποία θα βασιστεί η εφαρμογή, κάνοντας μια έρευνα αγοράς για παρόμοιες εφαρμογές και προσδιορίζοντας την λειτουργικότητα της εφαρμογής.

Στο Στάδιο Σχεδιασμού, σχεδιάζουμε σε μεγαλύτερη ανάλυση τις λειτουργίες που θα έχει η εφαρμογή, χρησιμοποιώντας ροές για τα σενάρια χρήσης και διαγράμματα. Στην συνέχεια, προσεγγίζουμε την μορφή που θα έχουν οι διαφορετικές οθόνες της εφαρμογής (Διεπαφή Χρήστη). Το Στάδιο Σχεδιασμού ολοκληρώνεται με τον τεχνικό σχεδιασμό, στα πλαίσια του οποίου ορίζουμε την αρχιτεκτονική της εφαρμογής, καθώς επίσης και τα εργαλεία και τις τεχνολογίες που θα χρησιμοποιηθούν.

Στο Στάδιο Υλοποίησης παρουσιάζουμε την τεχνική υλοποίηση της εφαρμογής και τον κώδικα με τον οποίο δημιουργήσαμε την εφαρμογή.

Η μεταπτυχιακή διατριβή ολοκληρώνεται με την παρουσίαση της εφαρμογής, μελετώντας τα βασικά σενάρια χρήσης.

Abstract

This paper examines the implementation of a web application that manages the expenses performed by employees, as part of the benefits provided by the company.

The implementation of the application is divided into three parts: the Ideas Assessment Part, the Design Part and the Development Part.

The Ideas Assessment Part, evaluates the idea based on which the application will be designed and developed, taking into account similar products and defining the application’s functionality.

In the Design Part, we examine the functionalities in more detail, using various tools such as Use Case Scenarios Flows and Diagrams, and we design the User Interface (UI). Lastly, we approach the technical architecture, the tools and the technology that will be used for the application’s development.

The last part is the Development Part, in which we provide the technical implementation and the code used for the creation of the application.

Last but not least, we present the created application, using the main use case scenarios.

Contents

1 Εισαγωγή

Είναι γεγονός ότι από την πανδημία και έπειτα, οι επιχειρήσεις καλούνται να αντιμετωπίσουν ακόμα περισσότερες προκλήσεις. Μια από τις κυριότερες αυτών, είναι η ανάγκη για «ανέπαφη» διεκπεραίωση των καθημερινών εργασιών και διαδικασιών. Η εργασία από το σπίτι ομοίως συνέβαλε στην αναγκαιότητα αυτή. Στα πλαίσια αυτά, θελήσαμε μέσω της συγκεκριμένης μεταπτυχιακής διατριβής να μελετήσουμε και αναλύσουμε μια επιχειρησιακή διαδικασία και να αναπτύξουμε μια εφαρμογή με σκοπό την αυτοματοποίησή της.

Το γεγονός ότι στην αγορά υπάρχουν ήδη αρκετά συστήματα διαχείρισης επιχειρησιακών πόρων που εξυπηρετούν τις δραστηριότητες μιας επιχείρησης στο σύνολό τους, αποτελεί πλεονέκτημα για τις μεγάλες εταιρίες, για τις μικρομεσαίες όμως επιχειρήσεις είναι δύσκολο να αξιοποιηθούν προγράμματα και εφαρμογές τέτοιας εμβέλειας, λόγω κόστους. Στην ελληνική αγορά οι εταιρίες ανάπτυξης συστημάτων επιχειρησιακών πόρων έχουν επικεντρωθεί κατά βάση στην εμπορική και στην λογιστική διαχείριση και αυτοματοποίηση των διαδικασιών. Με γνώμονα τα παραπάνω, αποφασίσαμε να επικεντρωθούμε σε διαδικασίες που θα μπορούσαν να είναι ανεξάρτητες από τις υπόλοιπες λειτουργίες της επιχείρησης, και εντοπίζονται σε επιχειρήσεις ανεξαρτήτου μεγέθους, περιορίζοντας έτσι ένα μεγάλο φάσμα δραστηριοτήτων.

Με αφορμή το γεγονός ότι όλο και περισσότερες επιχειρήσεις, στα πλαίσια παροχών προς τους εργαζομένους, καλύπτουν ένα μέρος των εξόδων τους για τρόφιμα, βενζίνη κλπ., καταλήξαμε στην ιδέα ανάπτυξης μιας εφαρμογής για διαχείριση και έγκριση των εξόδων αυτών.

Στα πλαίσια αυτά, μελετήθηκε σε θεωρητικό υπόβαθρο η διαδικασία έγκρισης των εξόδων σε μια εταιρία. Με βάση προϊόντα που ήδη υπάρχουν στην αγορά, προσδιορίσαμε τις λειτουργίες που θα μπορούσε να έχει η συγκεκριμένη εφαρμογή, με σκοπό πάντα να περιοριστεί το κόστος της εφαρμογής και να απλοποιηθούν όσο περισσότερο γίνεται οι διαδικασίες.

Κατόπιν θεωρητικού προσδιορισμού των λειτουργιών, σχεδιάσαμε πιο αναλυτικά τις λειτουργίες αυτές, χρησιμοποιώντας διάφορα εργαλεία, για παράδειγμα ροές περιγραφής των διαφορετικών σεναρίων χρήσης και διαγράμματα. Επιπλέον σχεδιάστηκε το οπτικό τμήμα της εφαρμογής, ο σχεδιασμός δηλαδή των οθονών της εφαρμογής. Με βάση τα παραπάνω, καταλήξαμε στον τεχνικό σχεδιασμό και αρχιτεκτονική της εφαρμογής.

Στο τελευταίο στάδιο ανάπτυξης, υλοποιήσαμε την εφαρμογή. Η εφαρμογή είναι διαδικτυακή εφαρμογή, δηλαδή ο χρήστης έχει πρόσβαση και την χρησιμοποιεί μέσω προγραμμάτων πλοήγησης. Το λειτουργικό κομμάτι έχει υλοποιηθεί σε γλώσσα προγραμματισμού JAVA, χρησιμοποιώντας την πλατφόρμα IntelliJ, ενώ η διεπαφή χρήστη σχεδιάστηκε με React JS, χρησιμοποιώντας ως πλατφόρμα το Visual Studio Code. Για την αρχιτεκτονική, βασιστήκαμε στην τεχνολογία των ‘Microservices’ (7.4.1), και στην περίπτωση μας στην τεχνολογία SpringBoot της JAVA. Ο λόγος που δομήθηκε έτσι η εφαρμογή είναι το γεγονός ότι με αυτό τον τρόπο η βάση δεδομένων και ο τρόπος διαχείρισης των δεδομένων σε λειτουργικό επίπεδο είναι ανεξάρτητο με τρόπο πρόσβασης του χρήστη στην εφαρμογή. Αυτό μας δίνει την δυνατότητα στο μέλλον να επεκτείνουμε την εφαρμογή και σε άλλες συσκευές (κινητό, ταμπλετ κλπ.) χωρίς να χρειαστεί να αλλάξουμε το λειτουργικό υπόβαθρο.

Η μεταπτυχιακή διατριβή ολοκληρώνεται με της παρουσίασης της εφαρμογής, τόσο σε οπτικό επίπεδο (μέσω εικόνων), όσο και σε λειτουργικό (παρουσιάζοντας τα βασικά σενάρια χρήσης).

2 Στάδια Υλοποίησης

Η υλοποίηση της εφαρμογής χωρίστηκε και τρία βασικά στάδια (7.1.1):

- Το Στάδιο Αξιολόγησης Ιδεών
- Το Στάδιο Σχεδιασμού
- Το Στάδιο Υλοποίησης

2.1 Στάδιο Αξιολόγησης Ιδεών

Το Στάδιο Αξιολόγησης Ιδεών περιλαμβάνει την Εύρεση της Ιδέας, την Έρευνα Αγοράς προκειμένου να εντοπίσουμε παρόμοια προϊόντα, και τον Προσδιορισμό των Λειτουργιών της Εφαρμογής (7.1.1).

Όπως αναφέρθηκε στην Εισαγωγή, κατόπιν διερεύνησης και μελέτης της αγοράς και των συνθηκών καταλήξαμε στην ιδέα ανάπτυξης μιας εφαρμογής για την διαχείριση και έγκριση εξόδων υπαλλήλων, στα πλαίσια παροχών των επιχειρήσεων προς τους εργαζομένους.

Έχοντας ως αφετηρία την παραπάνω ιδέα, προχωρήσαμε σε Έρευνα Αγοράς και συγκεκριμένα στην εύρεση παρόμοιων προϊόντων / εφαρμογών και στον προσδιορισμό της αγοράς που θα απευθυνθεί ή δικιά μας εφαρμογή.

Για την εύρεση παρόμοιων προϊόντων / εφαρμογών, έγινε έρευνα στο διαδίκτυο με σκοπό να βρεθούν τα πιο γνωστά προϊόντα που κυκλοφορούν αυτή την στιγμή στην αγορά και αφορούν την διαχείριση εταιρικών εξόδων. Συγκεκριμένα, βρέθηκαν 8 ιστοσελίδες (7.2.1,7.2.2,7.2.3,7.2.4,7.2.5,7.2.6,7.2.7, 7.2.8) που αξιολογούν τέτοιου είδος λογισμικά.W Από αυτά τα λογισμικά συγκεντρώσαμε αυτά που αναφέρονται σε τουλάχιστον δύο ιστοσελίδες από τις οχτώ.

Τα λογισμικά που καταλήξαμε είναι τα εξής:

- Expensify: Αναφορά σε 8 / 8 ιστοσελίδες (7.2.1,7.2.2,7.2.3,7.2.4,7.2.5,7.2.6,7.2.7, 7.2.8)
- SAP Concur: Αναφορά σε 6 / 8 ιστοσελίδες (7.2.1, 7.2.3, 7.2.4, 7.2.5, 7.2.7, 7.2.8)
- Zoho Expenses: Αναφορά σε 4 / 8 ιστοσελίδες (7.2.3, 7.2.4, 7.2.5, 7.2.7)
- Rydoo: Αναφορά σε 3 / 8 ιστοσελίδες (7.2.3, 7.2.4, 7.2.7)
- Everlance: Αναφορά σε 2 / 8 ιστοσελίδες (7.2.1, 7.2.6)

Το επόμενο βήμα είναι να βρούμε τις λειτουργίες που προσφέρει η κάθε μια από τις παραπάνω εφαρμογές. Για τον σκοπό αυτό, χρησιμοποιήθηκαν συγκεκριμένες πηγές, ώστε να υπάρχει κοινή αφετηρία για κάθε εφαρμογή. Οι πηγές που χρησιμοποιήθηκαν είναι οι παρακάτω:

- Getapp
- Saasworthy
- G2

Στις επόμενες ενότητες παρουσιάζουμε τις λειτουργίες ανά εφαρμογή, καθώς επίσης και το τμήμα αγοράς και οι λειτουργίες τις δικιά μας εφαρμογή.

2.1.1 Expensify

Το Expensify (7.2.9, 7.2.107.2.11) είναι εφαρμογή για την διαχείριση και κατηγοριοποίηση αποδείξεων. Επιτρέπει την αυτοματοποιημένη καταχώρηση εξόδων για έγκριση και αξιολόγηση.

Η αγορά που καλύπτει το προϊόν είναι κυρίως:

- Ελεύθεροι επαγγελματίες
- Μικρές / Μεσαίες / Μεγάλες επιχειρήσεις

Οι πλατφόρμες που υποστηρίζει η εφαρμογή είναι οι εξής:

- Web
- Android
- iPhone / iPad

Για τεχνική υποστήριξη της εφαρμογής, υπάρχουν οι παρακάτω τρόποι:

- Υποστήριξη μέσω chat
- Υποστήριξη μέσω ηλεκτρονικού ταχυδρομείου
- Υποστήριξη μέσω τηλεφώνου
- Γραφείο υποστήριξης
- Φόρουμ
- Εγχειρίδια

Οι κύριες λειτουργίες του προϊόντος είναι:

- Χρηματοοικονομικές Αναφορές
- Διαχείριση Χρόνου και Εξόδων
- Αναφορές Εξόδων
- Διαχείριση Ταξιδιών

Τα χαρακτηριστικά της εφαρμογής είναι:

- Παρακολούθηση Χρόνου & Εξόδων
- Παρακολούθηση Μιλιών
- Παρακολούθηση Εξόδων Αυτοκινήτου
- Εγκρίσεις Εξόδων
- Εγκρίσεις σε πολλά επίπεδα – ροή εγκρίσεων
- Διαχείριση Αποζημιώσεων
- Διαχείριση Αποδείξεων
- Διαχείριση Φόρων
- Ανέβασμα Αποδείξεων Ψηφιακά
- Αναφορές σε διαφορετικά νομίσματα
- Αναφορές εξόδων
- Διεπαφή με τραπεζικά συστήματα

- Διεπαφή με εξωτερικά συστήματα
- Διασταύρωση Καταχωρήσεων
- Αυτοματοποιημένες Ειδοποιήσεις
- Παραμετροποίηση Χρηστών, Ρολών και Προσβάσεων

2.1.2 SAP Concur

Το SAP Concur (7.2.127.2.137.2.14) είναι το κορυφαίο λογισμικό για διαχείριση ταξιδιών, εξόδων και τιμολογίων. Σκοπός του λογισμικού είναι η απλοποιημένη και αυτοματοποιημένη χρήση του από υπαλλήλους σε όλα τα στάδια ενός ταξιδιού. Επίσης, υποστηρίζει την διαχείριση εξόδων σε όλα τα στάδια.

Η αγορά που καλύπτει το προϊόν είναι κυρίως:

- Ελεύθεροι επαγγελματίες
- Μικρές / Μεσαίες / Μεγάλες επιχειρήσεις

Οι πλατφόρμες που υποστηρίζει η εφαρμογή είναι οι εξής:

- Web
- Android
- iPhone / iPad

Για τεχνική υποστήριξη της εφαρμογής, υπάρχουν οι παρακάτω τρόποι:

- Υποστήριξη μέσω chat
- Υποστήριξη μέσω ηλεκτρονικού ταχυδρομείου
- Υποστήριξη μέσω τηλεφώνου
- Γραφείο υποστήριξης
- Εγχειρίδια

Οι κύριες λειτουργίες του προϊόντος είναι:

- Καταχώρηση και Διαχείριση Λογιστικών Κινήσεων
- Αναφορές Εξόδων
- Διαχείριση Ταξιδιών
- Διαχείριση Εξόδων

Τα χαρακτηριστικά της εφαρμογής είναι:

- Παρακολούθηση Εξόδων
- Παρακολούθηση Μιλιών
- Εγκρίσεις Εξόδων
- Εγκρίσεις σε πολλά επίπεδα – ροή εγκρίσεων
- Διαχείριση Αποζημιώσεων
- Διαχείριση Αποδείξεων

- Διαχείριση Δρομολογίων
- Ανέβασμα Αποδείξεων Ψηφιακά
- Αναφορές σε διαφορετικά νομίσματα
- Αναφορές εξόδων
- Διεπαφή με τραπεζικά συστήματα
- Διεπαφή με εξωτερικά συστήματα
- Καταχώρηση Κρατήσεων σε Ξενοδοχειακές Μονάδες κλπ.
- Ενημερώσεις για Δρομολογίων και Ταξιδιών
- Αυτοματοποιημένες Ειδοποιήσεις
- Παραμετροποίηση Χρηστών, Ρολών και Προσβάσεων

2.1.3 Zoho Expenses

Το Zoho Expenses (7.2.157.2.167.2.17) είναι εφαρμογή που επιτρέπει την διαχείριση επαγγελματικών ταξιδιών και εξόδων από επιχειρήσεις.

Η αγορά που καλύπτει το προϊόν είναι κυρίως:

- Ελεύθεροι επαγγελματίες
- Μικρές / Μεσαίες / Μεγάλες επιχειρήσεις

Οι πλατφόρμες που υποστηρίζει:

- Web
- Android
- iPhone / iPad

Για τεχνική υποστήριξη της εφαρμογής, υπάρχουν οι παρακάτω τρόποι:

- Υποστήριξη μέσω chat
- Υποστήριξη μέσω ηλεκτρονικού ταχυδρομείου
- Υποστήριξη μέσω τηλεφώνου
- Γραφείο υποστήριξης
- Φόρουμ
- Εγχειρίδια

Οι κύριες λειτουργίες του προϊόντος είναι:

- Αναφορές Εξόδων
- Διαχείριση Ταξιδιών
- Τα χαρακτηριστικά της εφαρμογής είναι:
- Παρακολούθηση Εξόδων & Χρόνου
- Παρακολούθηση Μιλιών
- Διαχείριση Εξόδων & Αποδείξων

- Διαχείριση Φόρων
- Διαχείριση Αποζημιώσεων
- Εγκρίσεις σε πολλά επίπεδα – ροή εγκρίσεων
- Ανέβασμα Αποδείξεων Ψηφιακά
- Αναφορές σε διαφορετικά νομίσματα
- Διεπαφή με τραπεζικά συστήματα
- Διεπαφή με τρίτα συστήματα
- Διασταύρωση Καταχωρήσεων

2.1.4 Rydoo

Το Rydoo (7.2.187.2.19) είναι ένα από τα κορυφαία λογισμικά για την διαχείριση εταιρικών εξόδων και ταξιδιών, μέσω αυτοματοποιημένων διαδικασιών.

Η αγορά που καλύπτει το προϊόν είναι κυρίως:

- Μικρές / Μεσαίες / Μεγάλες επιχειρήσεις

Οι πλατφόρμες που υποστηρίζει:

- Web
- Android
- iPhone / iPad

Για τεχνική υποστήριξη της εφαρμογής, υπάρχουν οι παρακάτω τρόποι:

- Υποστήριξη μέσω chat
- Υποστήριξη μέσω ηλεκτρονικού ταχυδρομείου
- Υποστήριξη μέσω τηλεφώνου
- Γραφείο υποστήριξης
- Φόρουμ
- Εγχειρίδια
- Υποστήριξη 24/7

Οι κύριες λειτουργίες του προϊόντος είναι:

- Λογιστικές Κινήσεις
- Αναφορές Εξόδων
- Διαχείριση Ταξιδιών

Τα χαρακτηριστικά της εφαρμογής είναι:

- Διαχείριση Αποδείξεων Ψηφιακά
- Διαχείριση Μιλιών
- Αναφορές Εξόδων & άλλα
- Κατηγοριοποίηση Εξόδων

- Μετατροπές σε άλλα νομίσματα
- Διεπαφή με Τραπεζικά Συστήματα
- Διεπαφή με Επιχειρησιακά Εργαλεία
- Αυτοματοποιημένες Ειδοποιήσεις
- Ροή Εγκρίσεων
- Παραμετροποίηση Χρηστών, Ρολών και Προσβάσεων

2.1.5 Everlance

Το Everlance (7.2.207.2.21) είναι λογισμικό για την διαχείριση μιλιών και εξόδων.
Η αγορά που καλύπτει το προϊόν είναι κυρίως:

- Ελεύθεροι επαγγελματίες
- Μικρές / Μεσαίες / Μεγάλες επιχειρήσεις

Οι πλατφόρμες που υποστηρίζει:

- Web
- Android
- iPhone / iPad

Για τεχνική υποστήριξη της εφαρμογής, υπάρχουν οι παρακάτω τρόποι:

- Υποστήριξη μέσω chat
- Υποστήριξη μέσω ηλεκτρονικού ταχυδρομείου
- Υποστήριξη μέσω τηλεφώνου
- Γραφείο υποστήριξης
- Φόρουμ
- Εγχειρίδια

Οι κύριες λειτουργίες του προϊόντος είναι:

- Διαχείριση Οχημάτων
- Διαχείριση Χρόνου & Εξόδων
- Αναφορές Εξόδων
- Ημερολόγιο
- Τα χαρακτηριστικά της εφαρμογής είναι:
- Διαχείριση Καυσίμων
- Διαχείριση Μιλιών
- Διαχείριση Μηνυμάτων
- Διαχείριση Αποδείξεων
- Διαχείριση Εγκριτικής Ροή

- Διαχείριση Διαδρομών
- Διαχείριση Αποζημιώσεων
- Πολλαπλά Νομίσματα

Παρακολούθηση Εξόδων

- Παρακολούθηση Δραστηριοτήτων

2.1.6 Τμήμα Αγοράς & Βασικές Λειτουργίες Εφαρμογής

Όπως αναφέραμε στην εισαγωγή, το προϊόν μας απευθύνεται σε επιχειρήσεις που η χρήση ενός συστήματος διαχείρισης επιχειρησιακών πόρων για την διαχείριση & έγκριση εξόδων δεν θα ήταν εφικτή λόγω κόστους. Συνεπώς, το τμήμα αγοράς στο οποίο θα απευθυνθεί η δικιά μας εφαρμογή είναι:

- Μικρές / Μεσαίες Επιχειρήσεις

Οι λειτουργίες της εφαρμογής μας θα είναι οι παρακάτω:

- Διαχείριση Εγκρίσεων: Ο κάθε χρήστης θα έχει έναν εγκριτή εξόδων (όπου χρειάζεται). Αυτό θα ορίζεται από την παραμετροποίηση. Ο κάθε χρήστης θα μπορεί να έχει τον πολύ έναν εγκριτή. Ένας εγκριτής μπορεί να είναι υπεύθυνος για τις εγκρίσεις εξόδων πολλών χρηστών. Ένας χρήστης μπορεί να είναι και εγκριτής και να έχει και εγκριτή.
- Εγκρίσεις Εξόδων: Ο εγκριτής εξόδων μπορεί είτε να αποδεχτεί ένα έξοδο είτε να απορρίψει ένα έξοδο. Σε κάθε περίπτωση, θα στέλνεται μήνυμα στον χρήστη που καταχώρησε το έξοδο για να τον ενημερώσει για την αποδοχή/απόρριψη.
- Παρακολούθηση Εξόδων: Ο χρήστης θα μπορεί να δει τα έξοδα που έχει καταχωρήσει και την κατάσταση τους. Επιπλέον θα μπορεί να δει τα έξοδα που έχει αποδεχτεί/απορρίψει.
- Ανέβασμα Αποδείξεων Ψηφιακά: Στην καρτέλα εξόδου ο χρήστης θα πρέπει να ανεβάζει φωτογραφία του εξόδου, προκειμένου να το αξιολογήσει ο εγκριτής.
- Παραμετροποίηση Χρηστών, Ρολών και Προσβάσεων: Ο κάθε χρήστης θα παραμετροποιείται έτσι ώστε να ορίζονται οι οθόνες που θα μπορεί να δει, και ποιον εγκριτή θα έχει.

2.2 Σχεδιασμός Εφαρμογής

Ο σχεδιασμός αποτελείται από τις εξής ενότητες (7.1.1):

- Ονομασία & Λογότυπο
- Διεπαφή Χρήστη
- Περιγραφή Σεναρίων Χρήσης (Use Case Description)
- Διαγράμματα Χρήσης (Use Case Diagram)
- Τεχνικός Σχεδιασμός

2.2.1 Λογότυπο & Ονομασία

Το λογότυπο της εφαρμογής σχεδιάστηκε χρησιμοποιώντας την παρακάτω εφαρμογή:
Adobe Create Logo.
Η εφαρμογή ονομάζεται EBEM, και τα λογότυπα που θα χρησιμοποιηθούν είναι τα παρακάτω:

Εικόνα 1: Λογότυπο

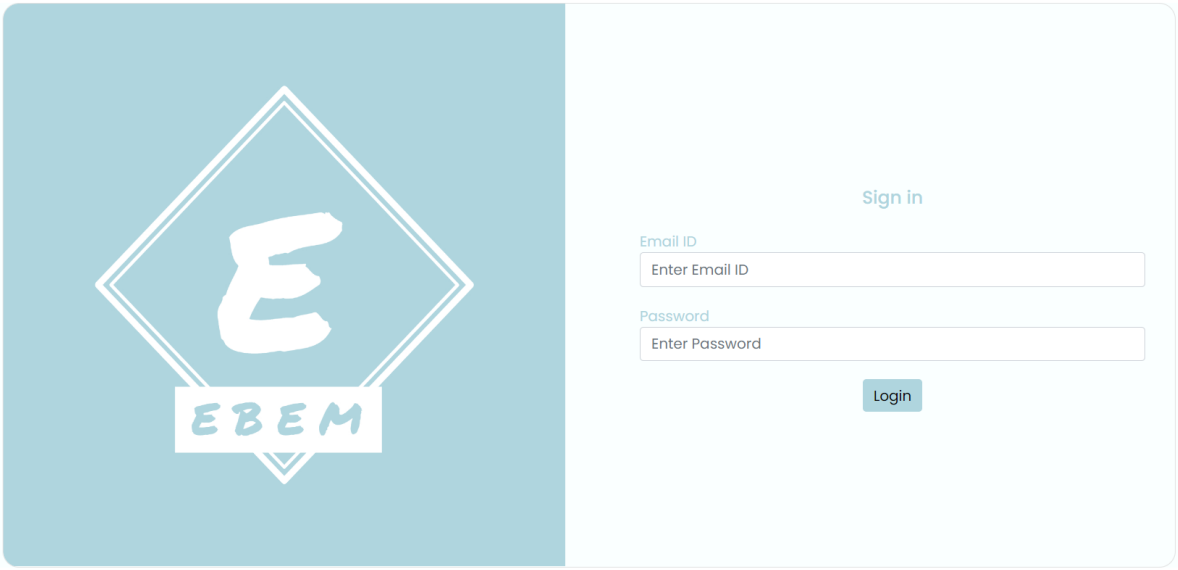
Εικόνα 2: Απλό Λογότυπο

2.2.2 Διεπαφή Χρήστη

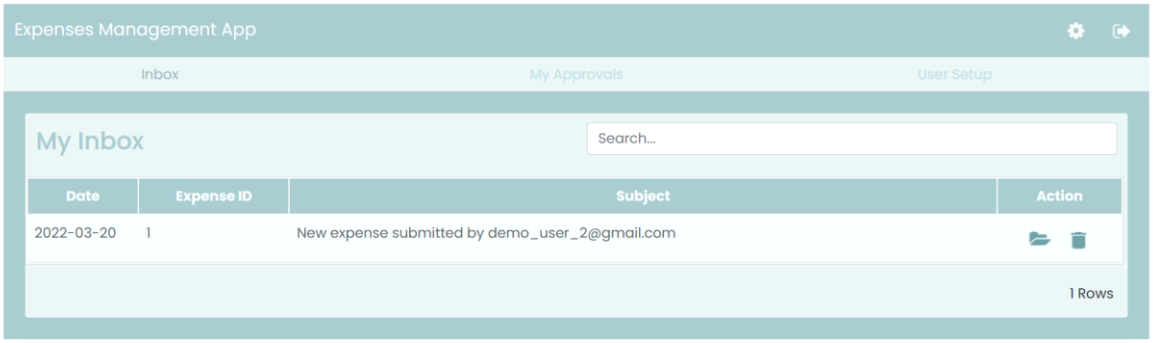
Η Διεπαφή Χρήστη αφορά τον σχεδιασμό των οθονών της εφαρμογής.
Η εφαρμογή θα έχει συνολικά οχτώ διαφορετικές οθόνες, συγκεκριμένα:

- Οθόνη Εισόδου Χρήστη
- Οθόνη Μηνυμάτων
- Οθόνη Εξόδων
- Οθόνη Εγκρίσεων
- Οθόνη Χρηστών
- Οθόνη Καρτέλας Εξόδου
- Οθόνη Καρτέλας Χρήστη
- Οθόνη Ρυθμίσεων

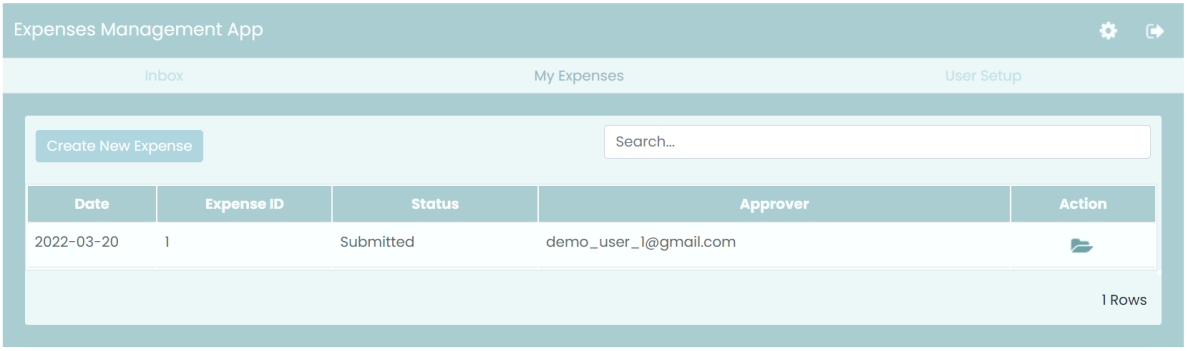
Για τα εντοπισμό του κατάλληλου χρώματος στις οθόνες χρησιμοποιήθηκε η εφαρμογή: Image Color Picker.
Η μορφή της κάθε οθόνης θα είναι:



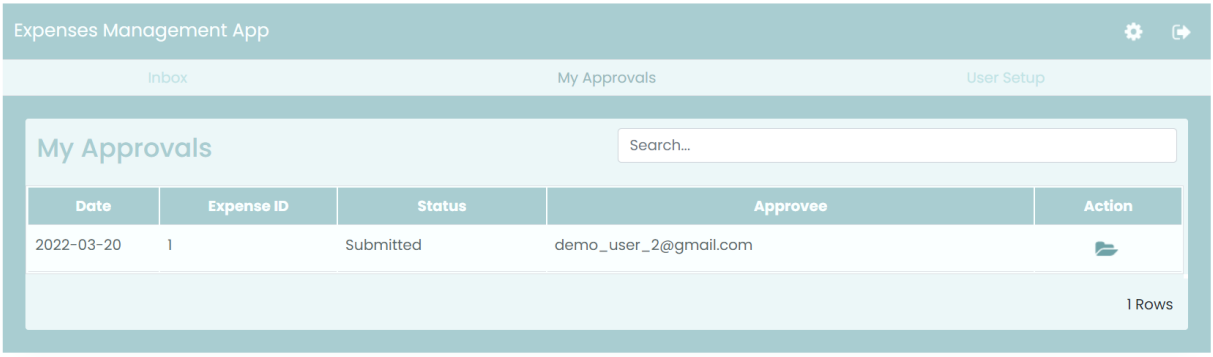
Εικόνα 3: Είσοδος Χρήστη



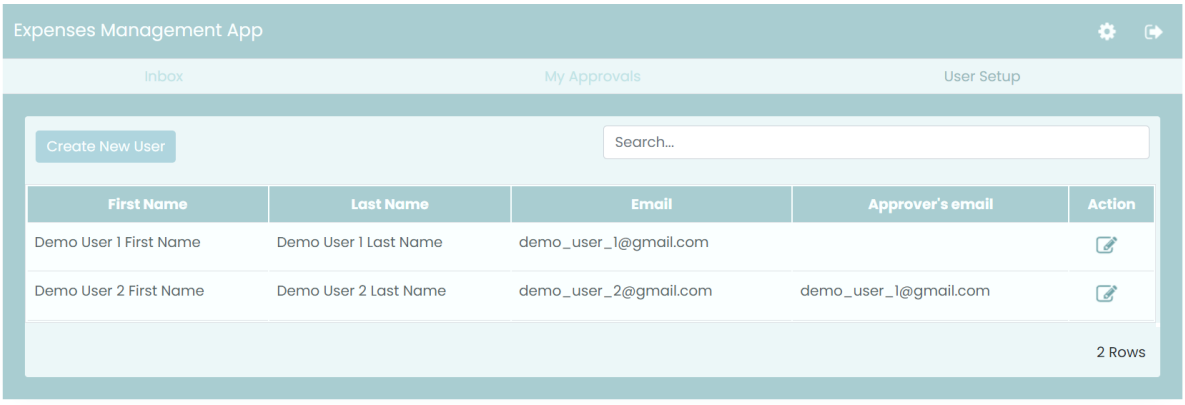
Εικόνα 4: Μηνύματα



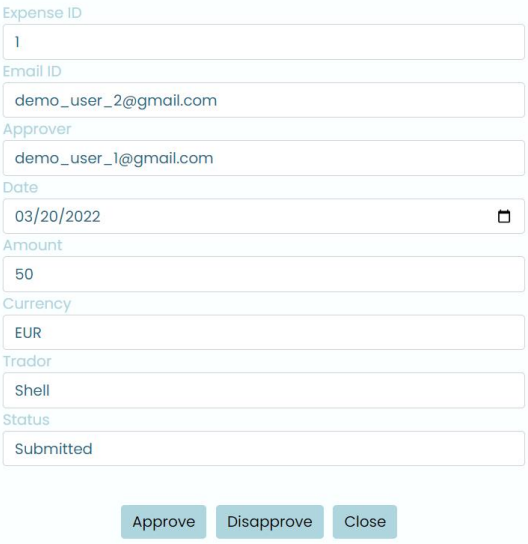
Εικόνα 5: Τα έξοδά μου



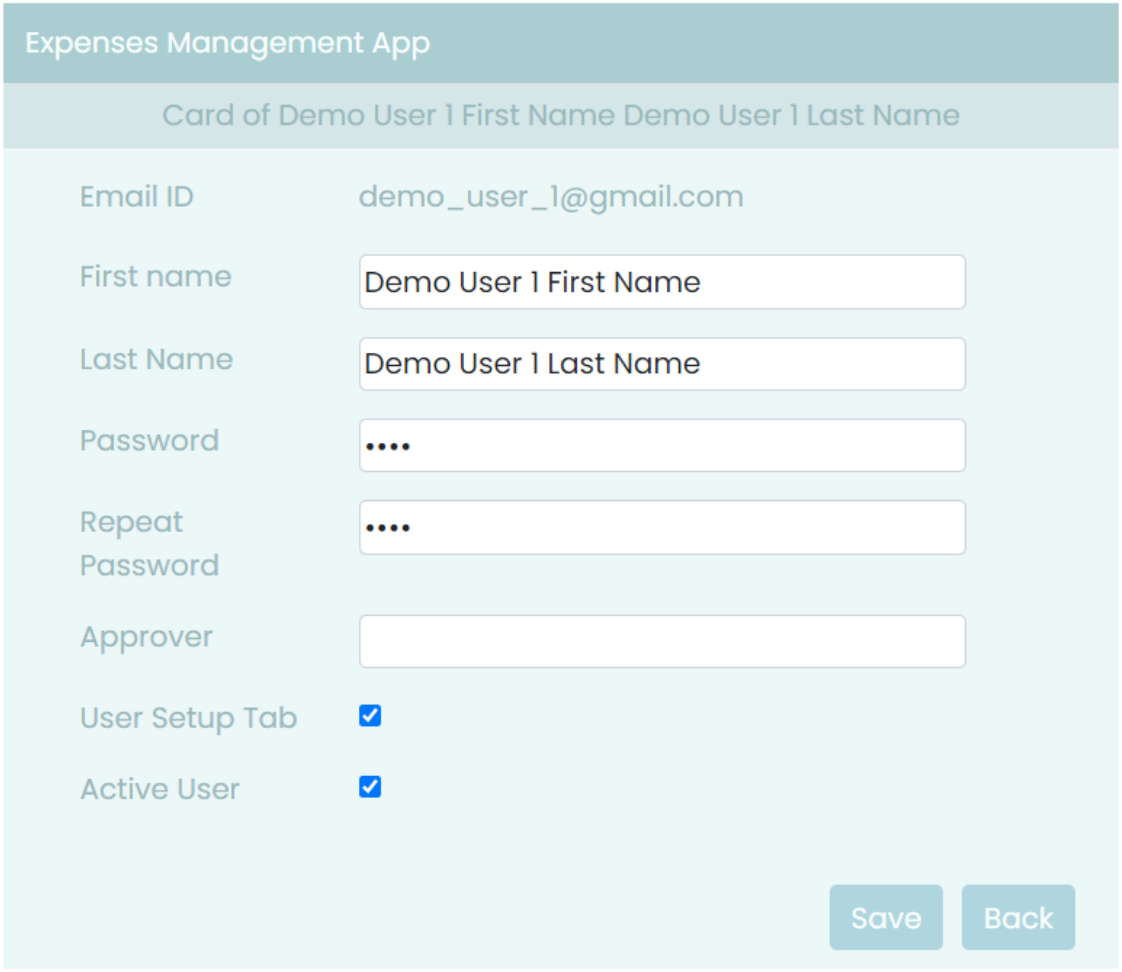
Εικόνα 6: Οι εγκρίσεις μου



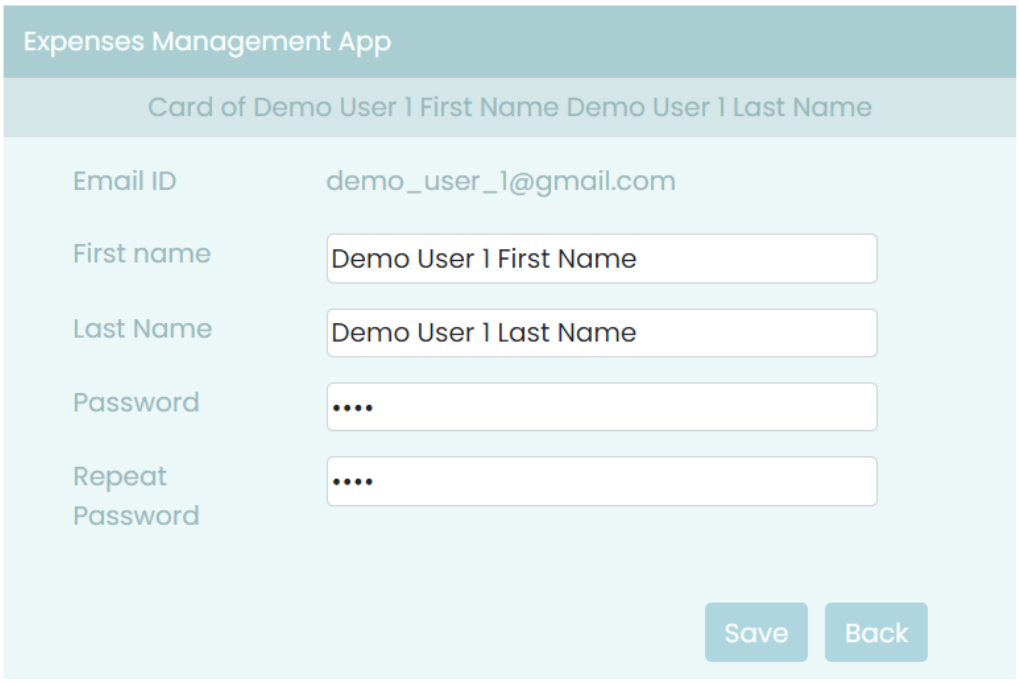
Εικόνα 7: Παραμετροποίηση Χρήστη



Εικόνα 8: Καρτέλα Εξόδου



Εικόνα 9: Καρτέλα Χρήστη



Εικόνα 10: Ρυθμίσεις

2.2.3 Περιγραφή Σεναρίων Χρήσης (Use Case Description)

Η περιγραφή σεναρίων χρήσης (7.3.1) απεικονίζει τα βήματα που πρέπει να πραγματοποιηθούν από τον χρήστη προκειμένου να ολοκληρώσει μια επιτυχημένη συναλλαγή / κίνηση στην εφαρμογή.

Το σενάριο χρήσης ξεκινάει με την ενέργεια ενός χρήστη ή του συστήματος (actor) και το κάθε βήμα χαρακτηρίζεται από την παροχή τιμής στον χρήστη / σύστημα (actor). Ο σκοπός είναι η επιτυχημένη χρήση του συστήματος από τον χρήστη.

Τα ελάχιστα στοιχεία που πρέπει να έχει η περιγραφή σεναρίου χρήστης:

- Τίτλο.
- Μέλη (Actors): είναι τα άτομα ή συστήματα που αλληλοεπιδρούν στο σενάριο χρήσης. Χωρίζονται στις εξής κατηγορίες:
    - Πρωταρχικός (Primary): το μέλος που ξεκινάει την χρήση του συστήματος.
    - Δευτερεύων (Secondary): το μέλος που έχει αλληλεπίδραση στην χρήση του συστήματος.
    - Αφανής (Off-stage): το μέλος που δεν αλληλοεπιδρά απευθείας στο σενάριο χρήσης αλλά η συμμετοχή του είναι απαραίτητη για επιχειρηματικούς λόγους.

- Προαπαιτούμενα: οι συνθήκες που θα πρέπει να ισχύουν προκειμένου να ξεκινήσει το σενάριο.
- Απαιτούμενα μετά την ολοκλήρωση της χρήσης: οι συνθήκες που θα πρέπει να ισχύουν όταν το σενάριο χρήσης θα έχει ολοκληρωθεί. Η επιτυχία του σεναρίου δηλώνει ότι η χρήση του συστήματος από τον χρήστη ήταν επιτυχημένη και ο χρήστης έλαβε το αποτέλεσμα που χρειαζόταν, ενώ η αποτυχία δηλώνει ότι το σενάριο δεν ολοκληρώθηκε με επιτυχία.
- Ροή: τα βήματα και η διεπαφή του χρήστη και του συστήματος. Διακρίνεται στις εξής κατηγορίες:
    - Πρωταρχική ροή (primary path): η ροή με την οποία θα έχουμε επιτυχία σεναρίου.
    - Εναλλακτική ροή (alternate path): η ροή που δεν χρησιμοποιείται τόσο συχνά αλλά οδηγεί σε επιτυχία σεναρίου.
    - Ροή Εξαίρεσης (exception path): η ροή που θα οδηγήσει σε αποτυχία σεναρίου.
- ID Σεναρίου Χρήστης: μοναδική ταυτοποίηση του σεναρίου χρήσης.
- Περιγραφή: η περιγραφή του σεναρίου χρήσης.
- Δημιουργός: ο συγγραφέας του σεναρίου χρήσης.
- Ημερομηνία Δημιουργίας και Ιστορικό Αλλαγών.
- Προτεραιότητα Σεναρίου Χρήσης.
- Συχνότητα Χρήσης.

Τα σενάρια χρήσης της εφαρμογής EBEM μπορεί να έχουν πρωταρχικό μέλος τον υπάλληλο ή τον εγκριτή. Παρακάτω θα προσεγγίσουμε τα σενάρια χρήσης ανά πρωταρχικό μέλος.

Τα σταθερά στοιχεία του σεναρίου χρήσης:

Σενάριο Χρήσης Εφαρμογής EBEM	
ID	1
Τίτλος	Σενάριο Χρήσης Διαχείρισης Εξόδων
Περιγραφή	Περιλαμβάνει το σενάριο καταχώρησης εξόδου από τον υπάλληλο και την έγκριση/απόρριψη του εξόδου αυτού από τον εγκριτή.
Δημιουργός	Αντωνία Βάθη
Ημερομηνία Δημιουργίας	25.09.2022
Προτεραιότητα	Υψηλή
Χρήση	Υψηλή
Μέλη	Υπάλληλος (Πρωταρχικός, Δευτερεύων)
	Εγκριτής (Πρωταρχικός, Δευτερεύων)
	Σύστημα (Δευτερεύων)
Ροές	Πρωταρχική
	Εναλλακτική
	Εξαίρεσης

Προαπαιτούμενα	Ο υπάλληλος έχει εγκριτή στην παραμετροποίηση χρήστη
	Ο εγκριτής έχει ανατεθεί σε υπαλλήλους στην παραμετροποίηση χρήστη
Αποτέλεσμα	Επιτυχία
	Αποτυχία

Table 1: Σενάριο Χρήσης Εφαρμογής EBEM

Στην τυπική ροή με πρωταρχικό μέλος τον Υπάλληλο, έχουμε τα παρακάτω μέλη:

- Πρωταρχικό Μέλος: Υπάλληλος
- Δευτερεύων Μέλος: Εγκριτής, Σύστημα

Πρωταρχικό Μέλος: Υπάλληλος	Δευτερεύων Μέλος: Σύστημα / Εγκριτής
1. Ο υπάλληλος εισάγει το URI στο πρόγραμμα περιήγησης.	2. Το σύστημα εμφανίζει την σελίδα εισόδου.
2. Ο υπάλληλος εισάγει τα στοιχεία εισόδου.	3. Το σύστημα ελέγχει ότι τα στοιχεία εξόδου είναι σωστά.
	4. Το σύστημα επιβεβαιώνει ότι τα στοιχεία εξόδου είναι σωστά.
	5. Το σύστημα εμφανίζει την σελίδα μηνυμάτων.
6. Ο υπάλληλος πατάει την επικεφαλίδα 'My Expenses'.	7. Το σύστημα εμφανίζει την σελίδα εξόδων.
7. Ο υπάλληλος επιλέγει δημιουργία νέου εξόδου.	8. Το σύστημα εμφανίζει στον χρήστη την καρτέλα δημιουργίας εξόδου.
9. Ο υπάλληλος εισάγει τα στοιχεία και επιλέγει 'Submit' για να καταχωρήσει το έξοδο.	10. Το σύστημα ελέγχει ότι όλα τα στοιχεία έχουν συμπληρωθεί.

	11. Το σύστημα επιβεβαιώνει την συμπλήρωση όλων των στοιχείων.
	12. Το σύστημα αποθηκεύει το έξοδο και στέλνει μήνυμα στον εγκριτή.
	13. Ο εγκριτής εγκρίνει/απορρίπτει το έξοδο.
14. ◌ Το σενάριο τελειώνει με επιτυχία.	

Table 2: Πρωταρχικό Σενάριο Χρήσης ΕΒΕΜ 1 Πρωταρχικό Μέλος Υπάλληλος

Το εναλλακτικό σενάριο χρήσης με πρωταρχικό μέλος τον Υπάλληλο:

Πρωταρχικό Μέλος: Υπάλληλος	Δευτερεύων Μέλος: Σύστημα / Εγκριτής
7α. Ο υπάλληλος επιλέγει να ανοίξει υπάρχον μη καταχωρημένο έξοδο.	8. Το σύστημα εμφανίζει την καρτέλα εξόδου.
9. Ο υπάλληλος καταχωρεί το έξοδο	10. Το σύστημα ελέγχει ότι όλα τα στοιχεία έχου συμπληρωθεί.
	11. Το σύστημα επιβεβαιώνει την συμπλήρωση όλων των στοιχείων.
	12. Το σύστημα αποθηκεύει το έξοδο και στέλνει μήνυμα στον εγκριτή.
	13. Ο εγκριτής εγκρίνει/απορρίπτει το έξοδο.
14. ◌Το σενάριο τελειώνει με επιτυχία.	

Table 3: Εναλλακτικό Σενάριο Χρήσης ΕΒΕΜ 1 Πρωταρχικό Μέλος Υπάλληλος

Έχουμε δύο ροές εξαίρεσης για το σενάριο χρήσης 1 με πρωταρχικό μέλος τον Υπάλληλο:

Πρωταρχικό Μέλος: Υπάλληλος	Δευτερεύων Μέλος: Σύστημα / Εγκριτής
	4α. Το **σύστημα** δεν επιβεβαιώνει ότι τα στοιχεία εισόδου είναι σωστά.
	5. Το **σύστημα** εμφανίζει σφάλμα.
2. ◌Το σενάριο τελειώνει με αποτυχία.	

Table 4: Ροή Εξαίρεσης 1 Πρωταρχικό Μέλος Υπάλληλος

Πρωταρχικό Μέλος: Υπάλληλος	Δευτερεύων Μέλος: Σύστημα / Εγκριτής
	11α. Το σύστημα δεν επιβεβαιώνει την συμπλήρωση των στοιχείων.

	12. Το σύστημα εμφανίζει σφάλμα.
13. ◌Το σενάριο τελειώνει με αποτυχία.	

Table 5: Ροή Εξαίρεσης 2 Πρωταρχικό Μέλος Υπάλληλος

Στην τυπική ροή με πρωταρχικό μέλος τον Εγκριτή, έχουμε τα παρακάτω μέλη:

- Πρωταρχικό Μέλος: Εγκριτής
- Δευτερεύων Μέλος: Σύστημα

Πρωταρχικό Μέλος: Εγκριτής	Δευτερεύων Μέλος: Σύστημα
1. Ο εγκριτής εισάγει το URI στο πρόγραμμα πλοήγησης	2. Το σύστημα εμφανίζει την σελίδα εισόδου.
2. Ο εγκριτής εισάγει τα στοιχεία εισόδου.	3. Το σύστημα ελέγχει ότι τα στοιχεία εξόδου είναι σωστά.
	4. Το σύστημα επιβεβαιώνει ότι τα στοιχεία εξόδου είναι σωστά.
	5. Το σύστημα εμφανίζει την σελίδα μηνυμάτων.
6. Ο εγκριτής πατάει την επικεφαλίδα 'My Approvals'.	7. Το σύστημα εμφανίζει την σελίδα εγκρίσεων.
7. Ο εγκριτής επιλέγει το άνοιγμα εξόδου.	8. Το σύστημα εμφανίζει στον χρήστη την καρτέλα εξόδου.
9. Ο εγκριτής εγκρίνει / απορρίπτει το έξοδο.	
10. ◌Το σενάριο τελειώνει με επιτυχία.	

Table 6: Πρωταρχικό Σενάριο Χρήσης ΕΒΕΜ 1 Πρωταρχικό Μέλος Εγκριτής

Το εναλλακτικό σενάριο χρήσης με πρωταρχικό μέλος τον Εγκριτή:

Πρωταρχικό Μέλος: Εγκριτής	Δευτερεύων Μέλος: Σύστημα
6α. Ο εγκριτής παραμένει στην επικεφαλίδα 'Inbox'.	
7. Ο εγκριτής επιλέγει το άνοιγμα εξόδου από συγκεκριμένο μήνυμα.	8. Το σύστημα εμφανίζει στον χρήστη την καρτέλα εξόδου.
9. Ο εγκριτής εγκρίνει / απορρίπτει το έξοδο.	
10. ◌Το σενάριο τελειώνει με επιτυχία.	

Table 7: Εναλλακτικό Σενάριο Χρήσης ΕΒΕΜ 1 Πρωταρχικό Μέλος Εγκριτής

Έχουμε μία ροή εξαίρεσης για το σενάριο χρήσης 1 με πρωταρχικό μέλος τον Εγκριτή:

Πρωταρχικό Μέλος: Εγκριτής	Δευτερεύων Μέλος: Σύστημα
	4α. Το σύστημα δεν επιβεβαιώνει ότι τα στοιχεία εισόδου είναι σωστά.

	5. Το σύστημα εμφανίζει σφάλμα.
2. ◌Το σενάριο τελειώνει με αποτυχία.	

Table 8: Ροή Εξαίρεσης 1 Πρωταρχικό Μέλος Εγκριτής

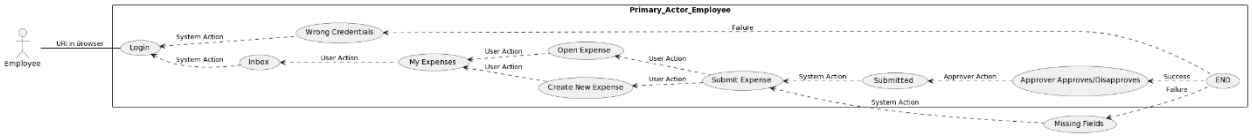
2.2.4 Διαγράμματα Χρήσης (Use Case Diagram)

Με τα διαγράμματα χρήσης απεικονίζουμε τα σενάρια χρήσης σε διαγράμματα. Για τον σκοπό αυτό, χρησιμοποιήθηκε η εφαρμογή Plant UML .

Έχουμε δύο διαγράμματα χρήσης:

- Διάγραμμα Χρήσης με Πρωταρχικό Μέλος τον Υπάλληλο
- Διάγραμμα Χρήσης με Πρωταρχικό Μέλος τον Εγκριτή

Το Διάγραμμα Χρήσης με Πρωταρχικό Μέλος τον Υπάλληλο:



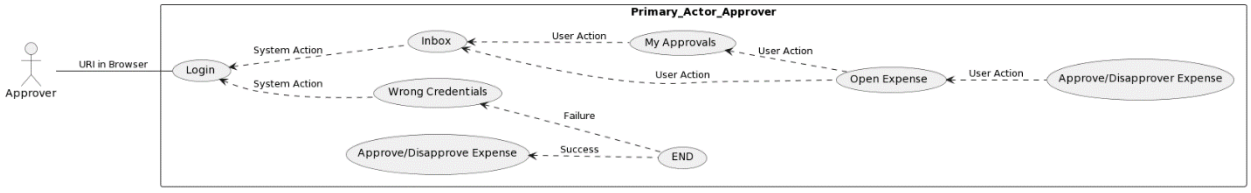
Διάγραμμα 1: Πρωταρχικό Μέλος τον Υπάλληλο

Ο κώδικας UML που χρησιμοποιήθηκε για το διάγραμμα:

Code 1: Κώδικας UML Πρωταρχικό Μέλος Υπάλληλος

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor Employee
rectangle Primary_Actor_Employee {
Employee -- (Login): URI in Browser
(Inbox)
(Inbox)-up.>(Login): System Action
(My Expenses)-up.>(Inbox): User Action
(Create New Expense)-up.>(My Expenses): User Action
(Submit Expense)-up.>(Create New Expense): User Action
(Submitted)-up.>(Submit Expense): System Action
(Approver Approves/Disapproves)-up.>(Submitted): Approver Action
(END)-up.>(Approver Approves/Disapproves): Success
(Wrong Credentials)-up.>(Login): System Action
(END)-up.>(Wrong Credentials): Failure
(Open Expense)-up.>(My Expenses): User Action
(Submit Expense)-up.>(Open Expense): User Action
}
(Missing Fields)-up.>(Submit Expense): System Action
(END)-up.>(Missing Fields): Failure
@enduml
```

Το Διάγραμμα Χρήσης με Πρωταρχικό Μέλος τον Εγκριτή:



Διάγραμμα 2: Πρωταρχικό Μέλος τον Εγκριτή

Ο κώδικας UML που χρησιμοποιήθηκε για το διάγραμμα:

Code 2: Κώδικας UML Πρωταρχικό Μέλος Εγκριτής

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor Approver
rectangle Primary_Actor_Approver {
Approver -- (Login): URI in Browser
(Inbox)
(Inbox)-up.>(Login): System Action
(My Approvals)-up.>(Inbox): User Action
(Open Expense)-up.>(My Approvals): User Action
(Approve/Disapprover Expense)-up.>(Open Expense): User Action
(END)-up.>(Approve/Disapprove Expense): Success
(Open Expense)-up.>(Inbox): User Action
(Wrong Credentials)-up.>(Login): System Action
(END)-up.>(Wrong Credentials): Failure
}
@enduml
```

2.3 Τεχνικός Σχεδιασμός

Κατά τον τεχνικό σχεδιασμό (7.1.1):

- Προσδιορίζουμε την αρχιτεκτονική της εφαρμογής και τις τεχνολογίες που θα χρησιμοποιήσουμε.
- Προσδιορίζουμε τα εργαλεία / εφαρμογές / πλατφόρμες που θα χρησιμοποιηθούν για την ανάπτυξη της εφαρμογής.
- Σχεδιάζουμε την δομή της Βάσης Δεδομένων .
- Σχεδιάζουμε την δομή του προγραμματιστικού τμήματος που αφορά λειτουργικότητα του προϊόντος και την διαχείριση της Βάσης Δεδομένων.
- Σχεδιάζουμε την δομή του προγραμματιστικού τμήματος που αφορά την Διεπαφή Χρήστη.

2.3.1 Αρχιτεκτονική & Τεχνολογία

Η αρχιτεκτονική της εφαρμογής βασίζεται στην αρχιτεκτονική των microservices (7.4.1), και συγκεκριμένα την τεχνολογία Springboot (7.4.2) της JAVA. Προκειμένου να κατανοήσουμε την αρχιτεκτονική αυτή, θα πρέπει να γνωρίζουμε τι είναι το πρωτόκολλο των web services (7.4.3) και τα REST APIs (7.4.4). Επιπλέον θα πρέπει να έχουμε εικόνα για το μοντέλο σχεδιασμού MVC (Model – View – Controller) (7.4.5), με βάση το οποίο δομείται η τεχνολογία Springboot.

Τα web services (7.4.3) είναι πρωτόκολλο επικοινωνίας μεταξύ δύο συστημάτων που γίνεται μέσω υπηρεσίας ή server που διαχειρίζεται διάφορες κλήσεις μέσω διαδικτύου. Αφορά την επικοινωνία μεταξύ δύο συστημάτων: του consumer – καταναλωτή (αυτού που καλεί το άλλο σύστημα) και του producer – παραγωγού (αυτού που καλείται από το άλλο σύστημα). Ο παραγωγός εξάγει ένα web

service προκειμένου να το καταναλώσουν εξωτερικά συστήματα. Ο καταναλωτής στέλνει ένα request – μήνυμα στον παραγωγό, ο οποίος από μεριά του απαντάει στον καταναλωτή με ένα response – απάντηση. Το κάθε μήνυμα μπορεί να είναι σε μορφή HTML, JSON, XML ή εικόνας.

Το REST API (7.4.4) είναι αρχιτεκτονική web services, όπου η κλήση στο άλλο σύστημα γίνεται με βάση μια ενέργεια (POST, PUT, DELETE, GET). Η ενέργεια διαφοροποιεί την επίδραση που θα έχει η κλήση στον παραγωγό. Συγκεκριμένα, μια κλήση με ενέργεια:

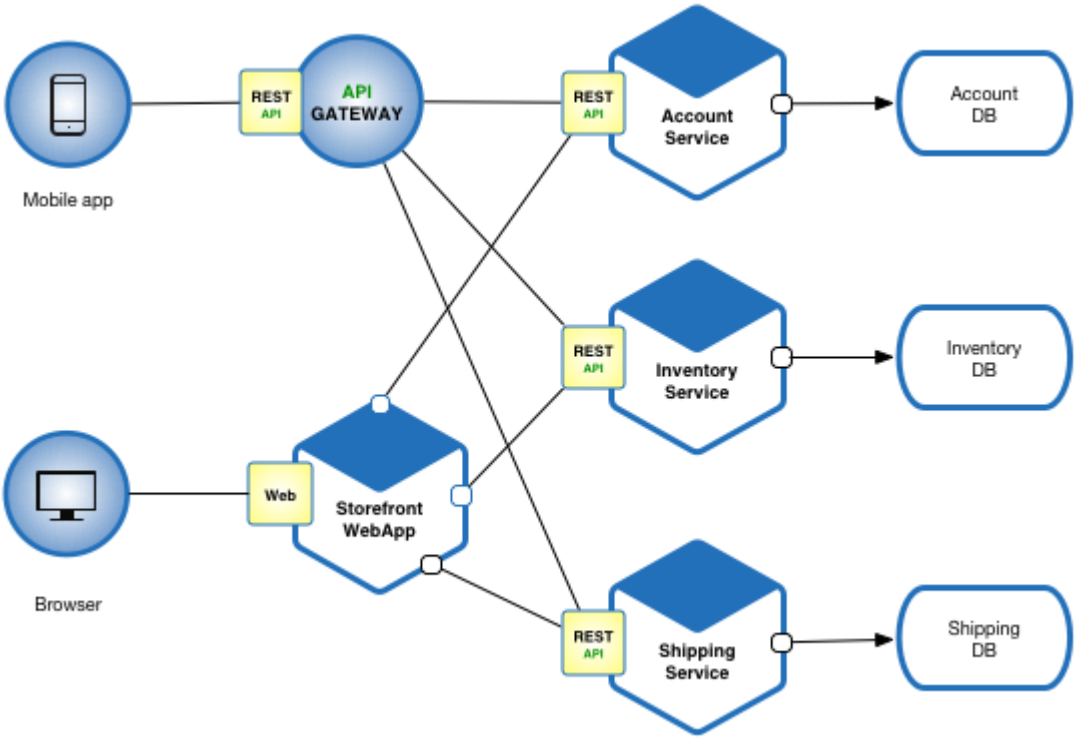
- POST: δημιουργεί (create) μια νέα εγγραφή σε πίνακα της βάσης δεδομένων του παραγωγού.
- PUT: ενημερώνει (update) υπάρχουσα εγγραφή σε πίνακα της βάσης δεδομένων του παραγωγού.
- DELETE: διαγράφει (delete) εγγραφή σε πίνακα της βάσης δεδομένων του παραγωγού.
- GET: διαβάζει (select) εγγραφή σε πίνακα της βάσης δεδομένων του παραγωγού.

Σε όλες τις περιπτώσεις η κλήση θα γίνει στο ίδιο URL, ενώ η δράση της κλήσης θα καθοριστεί από την ενέργεια (action).

Η αρχιτεκτονική των microservices (7.4.1) είναι μια μορφή αρχιτεκτονικής που δομεί ένα σύνολο υπηρεσιών.Η αρχιτεκτονική αυτή βασίζεται στο ότι μια εφαρμογή, θα έχει το ίδιο λειτουργικό υπόβαθρο και την ίδια βάση δεδομένων ανεξάρτητα από την συσκευή – πρόγραμμα που χρησιμοποιείται για να τρέξει. Η κάθε εφαρμογή χρησιμοποιεί REST APIs με σκοπό να λάβει την πληροφορία που χρειάζεται από μια υπηρεσία – service. Η υπηρεσία λαμβάνει τα δεδομένα από μια βάση δεδομένων. Σαν αποτέλεσμα, η βάση δεδομένων και η υπηρεσία είναι τα ίδια, ανεξάρτητα από την συσκευή/πρόγραμμα από τα οποία καλείται (εφαρμογή κινητού, πρόγραμμα περιήγησης κλπ.).

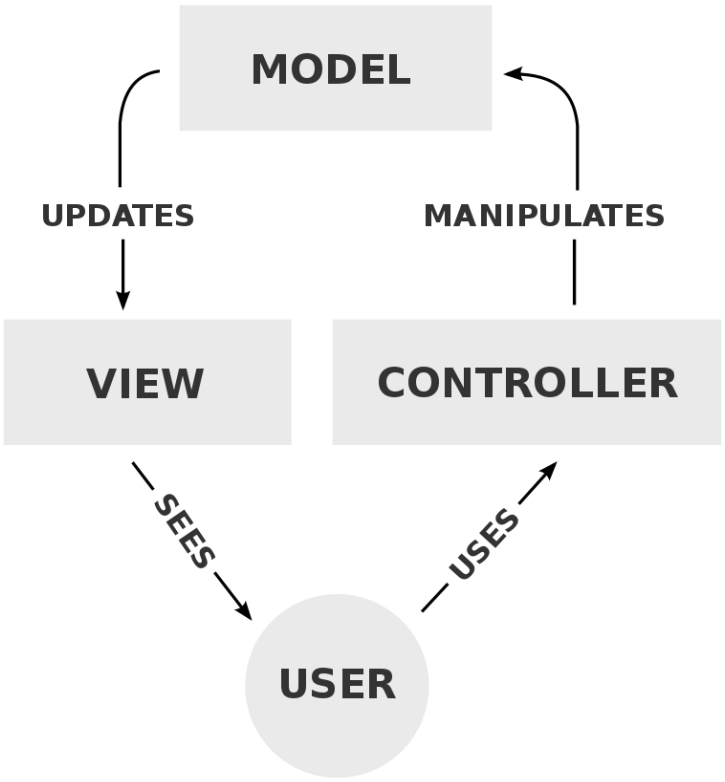
Αυτό μας δίνει την δυνατότητα να δημιουργήσουμε εφαρμογές που είναι εύκολα συντηρήσιμες και μπορούν να ελεγχθούν με ακρίβεια.

Η παρακάτω εικόνα είναι ένα παράδειγμα αρχιτεκτονικής microservices (7.4.1):



Εικόνα 11: Αρχιτεκτονική Microservices

Το μοντέλο σχεδιασμού Model – View – Controller (MVC) (7.4.5), αφορά των τρόπο που θα σχεδιαστεί – δομηθεί η διεπαφή μεταξύ συστημάτων. Ο τρόπος σχεδιασμού περιγράφεται από το παρακάτω σχήμα (7.4.5):



Εικόνα 12: MVC Model

Τα βασικά στοιχεία του MVC μοντέλου:

- Model: είναι το κεντρικό τμήμα του μοντέλου. Περιγράφει την δομή των δεδομένων ανεξάρτητα από την διεπαφή χρήστη που χρησιμοποιείται. Είναι το σημείο στο οποίο γίνεται η διαχείριση των δεδομένων, και περιγράφεται η λογική και οι κανόνες σε μια εφαρμογή.
- View: είναι η αναπαραγωγή των δεδομένων σε διαφορετικές μορφές. Τα δεδομένα λαμβάνονται από το τμήμα Model, και το τμήμα View είναι υπεύθυνο για την μορφή με βάση την οποία θα εμφανιστεί η πληροφορία στον χρήστη.
- Controller: Μέσω επικοινωνίας του χρήστη με το Model. Μέσω του Controller λαμβάνουμε την πληροφορία του χρήστη, την επεξεργαζόμαστε και στέλνουμε τα δεδομένα στο τμήμα Model.

Το Springboot (7.4.2) είναι framework της JAVA με βάση το οποίο μπορούμε να δημιουργήσουμε μια εφαρμογή με την αρχιτεκτονική των microservices. Τα microservices αυτά μπορούν να σχεδιαστούν εύκολα με βάση το μοντέλο MVC. Η δημιουργία της εφαρμογής γίνεται με βάση παραμετροποίηση χωρίς να χρειάζεται επιπλέον κώδικας.

Μέσω του Springboot:

- Φτιάχνουμε την δομή των δεδομένων στην βάση δεδομένων της εφαρμογής μας.

- Διαχειριζόμαστε τα δεδομένα αυτά.
- Κάνουμε expose ένα web service προκειμένου να το καλέσουν εξωτερικά συστήματα για να λάβουν / διαχειριστούν τα δεδομένα της βάσης.

2.3.2 Εργαλεία / Εφαρμογές / Πλατφόρμες

Τα εργαλεία-τεχνολογίες που θα χρησιμοποιηθούν για την δημιουργία-έλεγχο της εφαρμογής είναι τα παρακάτω:

- PostgreSQL: σύστημα διαχείρισης βάσης δεδομένων. Βασίζεται στην γλώσσα SQL (7.4.6, 7.4.7, 7.4.8).
- pgAdmin: εργαλείο για την διαχείριση της βάσης δεδομένων (7.4.97.4.10).
- IntelliJ IDEA: περιβάλλον στο οποίο θα προγραμματιστεί το λειτουργικό τμήμα της εφαρμογής σε JAVA, χρησιμοποιώντας την τεχνολογία Springboot για να εφαρμόσουμε την αρχιτεκτονική των Microservices (7.4.11).
- Visual Studio: περιβάλλον στο οποίο θα προγραμματιστεί η διεπαφή χρήστη της εφαρμογής σε React JS (7.4.127.4.137.4.14).
- Postman: Πλατφόρμα για τον έλεγχο του API της εφαρμογής (**Error! Reference source not found.**).
- Chrome Browser: Για την εκτέλεση της εφαρμογής.

2.3.3 Βάση Δεδομένων

Για την Βάση Δεδομένων θα χρησιμοποιηθεί το σύστημα διαχείρισης PostgreSQL (7.4.6, 7.4.7, 7.4.8), μέσω του pgAdmin (7.4.97.4.10). Η βάση δεδομένων θα ονομαστεί EBEM_Application.

Με βάση το μοντέλο MVC, οι διαφορετικές οντότητες θα δημιουργηθούν μέσω της τεχνολογίας Springboot, σε λειτουργικό επίπεδο, χρησιμοποιώντας το περιβάλλον IntelliJ (7.4.11).

Οι διαφορετικές οντότητες (πίνακες) είναι οι παρακάτω:

- Expense: απεικονίζει το έξοδο. Θα αποτελείται από τα παρακάτω πεδία:
    - ID (bigint)
    - Amount (double precision)
    - Approver (character varying)
    - Currency (character varying)
    - Date (date)
    - Email (character varying)
    - Image (text)
    - Status (character varying)
    - Trador (character varying)
- Inbox: απεικονίζει μήνυμα που λαμβάνει ο χρήστης για την εξέλιξη του εξόδου.
    - ID (bigint)

    - Date (date)
    - Email (character varying)
    - Expense_id (bigint)
    - Subject (character varying)
- Records: συνοπτική μορφή εξόδου.
    - ID (bigint)
    - Approver (character varying)
    - Date (character varying)
    - Email (character varying)
    - Status (character varying)
- Users: χρήστες στην εφαρμογή.
    - Email_id (character varying)
    - User_setup (character varying)
    - Approver (character varying)
    - First_name (character varying)
    - Is_active (character varying)
    - Last_name (character varying)
    - My_approvals (character varying)
    - My_expenses (character varying)
    - Password (character varying)

2.3.4 Προγραμματισμός Λειτουργικού Τμήματος

Για τον προγραμματισμό της λειτουργικότητας της εφαρμογής θα χρησιμοποιηθεί η γλώσσα προγραμματισμού JAVA. Η λειτουργικότητα θα σχεδιαστεί με βάση την αρχιτεκτονική των microservices, και συγκεκριμένα την τεχνολογία Springboot. Η δομή της εφαρμογής θα βασίζεται στο μοντέλο MVC.

Για την εφαρμογή θα δημιουργηθεί στο IntelliJ (7.4.11) το Project EBEM.

Στις ιδιότητες της εφαρμογής (resources->application.properties) θα οριστεί η βάση δεδομένων.

Το EBEM Project θα αποτελείται από τα παρακάτω τμήματα – packages:

- Model: Δημιουργία οντοτήτων της εφαρμογής:
    - Expense
    - Inbox
    - Records
    - User
- Repository: interface που κάνει extend το interface JpaRepository του spring framework, για το API:

    - ExpenseRepository
    - InboxRepository
    - RecordsRepository
    - UserRepository
- Exception: διαχείριση σφαλμάτων:
    - ExpenseNotFoundException
    - InboxNotFoundException
    - UserNotFoundException
- Controller: Η διαχείριση των διαφορετικών οντοτήτων της εφαρμογής:
    - ExpenseController
    - InboxController
    - RecordsController
    - UserController

2.3.5 Προγραμματισμός Διεπαφής Χρήστη

Η διεπαφή του χρήστη θα σχεδιαστεί χρησιμοποιώντας την βιβλιοθήκη React JS (7.4.13), μέσω του Visual Studio (7.4.127.4.14).

Στο Visual Studio (7.4.13), θα δημιουργηθεί το project EBEM και ο φάκελος react-frontend που θα περιλαμβάνει όλα τα συστατικά στοιχεία της Διεπαφής Χρήστη, συγκεκριμένα:

- Assets: φάκελος με όλες τις εικόνες & εικονίδια.
- Components: οι διάφορες οθόνες & συστατικά μέρη της διεπαφής, συγκεκριμένα:
    - ExpenseCardComponent.js: οθόνη της καρτέλας εξόδου.
    - HomeComponent.js: οθόνη της αρχικής σελίδας.
    - MySettingsComponent.js: οθόνη των ρυθμίσεων.
    - SignInComponent.js: οθόνη εισόδου στην εφαρμογή.
    - UserCardComponent.js: οθόνη της καρτέλας χρήστη.
    - SmartTableInboxComponent: σχεδιασμός του πίνακα με τα μηνύματα. Χρήστη στην αρχική σελίδα.
    - SmartTableMyApprovalsComponent: σχεδιασμός του πίνακα των διαφορετικών αιτήσεων έγκρισης εξόδου. Χρήση στην αρχική σελίδα.
    - SmartTableMyExpensesComponent: σχεδιασμός του πίνακα των διαφορετικών εξόδων. Χρήση στην αρχική σελίδα.
    - SmartTableUsersComponent: σχεδιασμός του πίνακα χρηστών. Χρήση στην αρχική σελίδα.
    - Index.js: σχεδιασμός του smart table σε javascript.
    - SmartTable.css: σχεδιασμός του smart table σε css. Χρησιμοποιείται από το index.js.

- Services: ορισμός των services, για την διασύνδεση της διεπαφής χρήστη με το λειτουργικό τμήμα της εφαρμογής. Η κάθε οντότητα έχει το δικό της service:
    - ExpenseService.js
    - InboxService.js
    - RecordsService.js
    - UserService.js

Στο βασικό javascript αρχείο App.js ορίζουμε τις διαφορετικές ροές (URIs) και τα αντίστοιχα components που καλούνται ανά URI. Η αρχική σελίδα της εφαρμογής θα έχει endpoint το /signin και θα αποτελεί μοναδική πηγή εισόδου στην εφαρμογή.

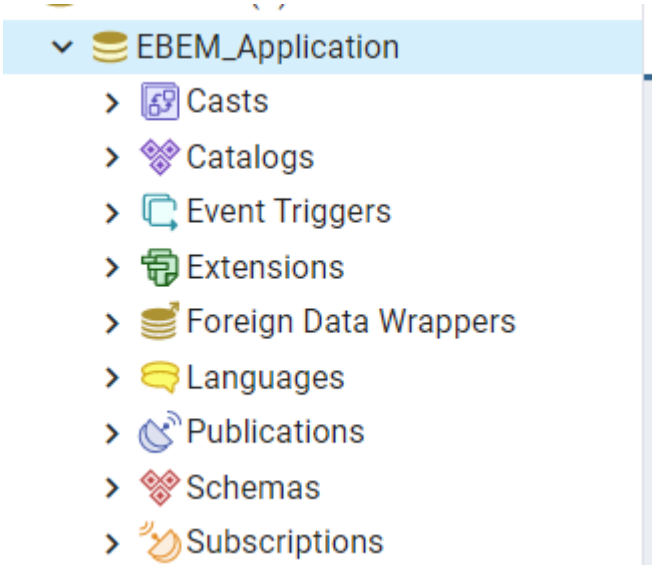
3 Υλοποίηση Εφαρμογής

Τα βήματα υλοποίησης της εφαρμογής, περιλαμβάνουν:

- Την δημιουργία της βάσης δεδομένων
- Τον προγραμματισμό του λειτουργικού ελέγχου & έλεγχο
- Τον προγραμματισμό της διεπαφής χρήστη

3.1 Βάση Δεδομένων

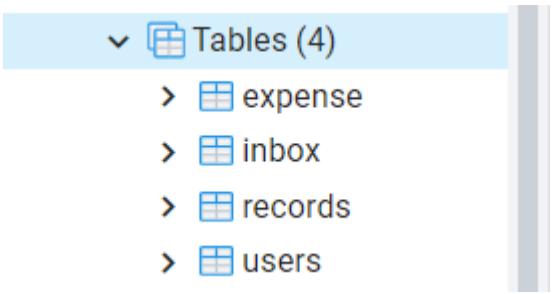
Μέσω του pgAdmin, δημιουργήσαμε βάση δεδομένων PostgreSQL, με όνομα EBEM_Application, όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 13: EBEM Βάση Δεδομένων

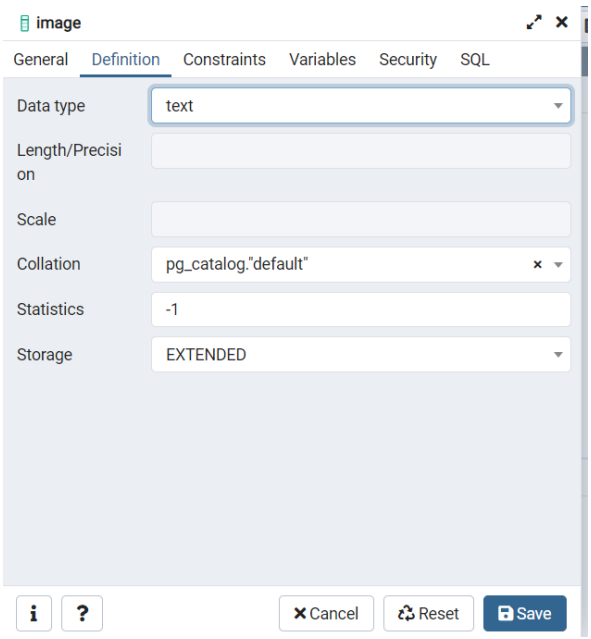
Στα πλαίσια της αρχιτεκτονικής των microservices, δεν χρειάζεται να δημιουργήσουμε κάποιον πίνακα στην βάση, αφού η δημιουργία των πινάκων θα γίνει κατά τον προγραμματισμό του λειτουργικού τμήματος της εφαρμογής, μέσω της πλατφόρμας IntelliJ.

Κατόπιν προγραμματισμού του λειτουργικού τμήματος και τρέξιμο της εφαρμογής στο IntelliJ, θα δημιουργηθούν οι παρακάτω πίνακες στην βάση δεδομένων:



Εικόνα 14: Πίνακες Βάσης Δεδομένων

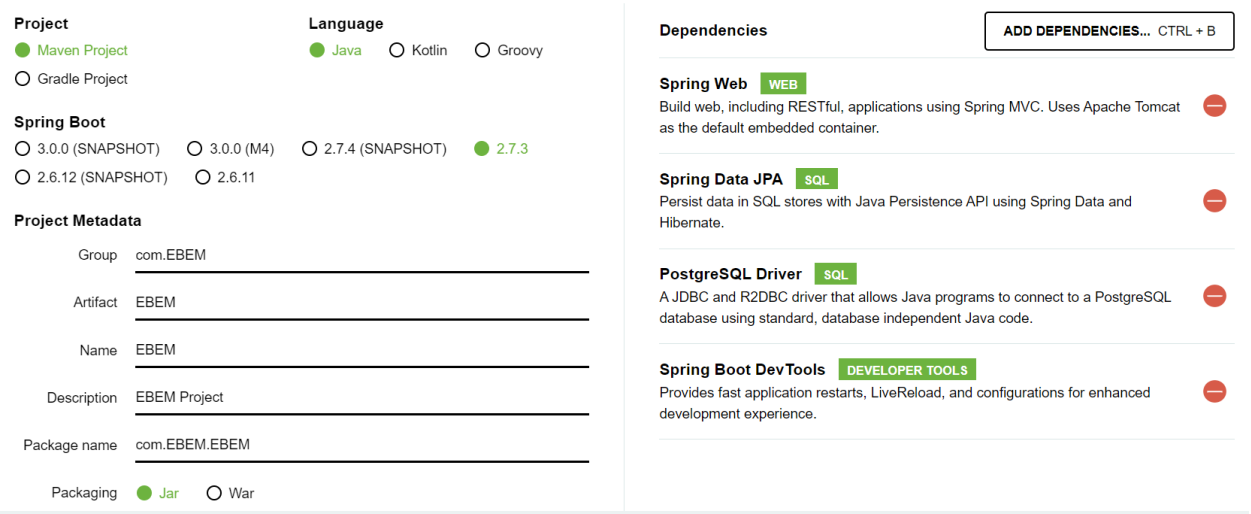
Η μόνη τροποποίηση που θα χρειαστεί να γίνει απευθείας στην βάση δεδομένων, είναι στο πεδίο image του πίνακα expense, το οποίο θα αλλάξει σε τύπο πεδίου text από character varying, προκειμένου να μπορεί να δεχθεί απεριόριστο αριθμό χαρακτήρων. Σε αυτό το πεδίο θα αποθηκεύουμε την φωτογραφία του εξόδου σε base64.



Εικόνα 15: Τύπος Πεδίου Images

3.2 Προγραμματισμός Λειτουργικού Τμήματος (Back-end)

Για την δημιουργία του project στο IntelliJ, αρχικά φτιάξαμε το παρακάτω Springboot project στο Start Spring Initializr:



Εικόνα 16: Springboot Project

Εισάγουμε το παραπάνω Project στην πλατφόρμα IntelliJ. Με αυτό τον τρόπο φτιάξαμε το Project EBEM, το οποίο περιλαμβάνει όλα τα dependencies που χρειάζονται για να εφαρμόσουμε την τεχνολογία Springboot.

Το επόμενο βήμα είναι να ορίσουμε τα στοιχεία της βάσης δεδομένων και τις αντίστοιχες παραμέτρους, στο application.properties, όπως φαίνεται παρακάτω.

Code 3: Application Properties IntelliJ

```
server.port=8085
spring.datasource.url=jdbc:postgresql://localhost:5432/EBEM_Application?useSSL=false
spring.datasource.username=postgres
spring.datasource.password=0721
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql=true
server.error.include-message=always
```

Στην συνέχεια θα δημιουργηθούν τα διαφορετικά packages, με βάση το μοντέλο MVC:

- Model

- Repository
- Exception
- Controller

Στο package Model, θα δημιουργήσουμε τις οντότητες – πίνακες της εφαρμογής. Για να το πετύχουμε αυτό δημιουργούμε τις παρακάτω κλάσεις:

- Expense
- Inbox
- User
- Records

Η κάθε κλάση είναι μια οντότητα που αντιπροσωπεύει έναν πίνακα στην βάση δεδομένων. Για κάθε οντότητα θα ορίσουμε τα πεδία του πίνακα, τον τύπο του πεδίου, καθώς επίσης και μεθόδους που σχετίζονται με την τιμή των συγκεκριμένων πεδίων (ορισμός τιμής – set, διάβασμα τιμής – get).

Ο κώδικας διαμορφώνεται ανά οντότητα ως εξής:

Code 4: Expense Class Code

```
package com.EBEM.EBEM.Model;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name = "expense")
public class Expense {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

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

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

    @Column(name = "date")
    private LocalDate date;

    @Column(name = "amount")
    private double amount;

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

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

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

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

    public Expense() {
    }

    public Expense(String email, String approver, LocalDate date, double amount, String currency, String trador, String
status, String image) {
        this.email = email;
        this.approver = approver;
        this.date = date;
        this.amount = amount;
```

```
        this.currency = currency;
        this.trador = trador;
        this.status = status;
        this.image = image;
    }

    public long getId() {
        return id;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getApprover() {
        return approver;
    }

    public void setApprover(String approver) {
        this.approver = approver;
    }

    public LocalDate getDate() {
        return date;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public String getCurrency() {
        return currency;
```

```
    }

    public void setCurrency(String currency) {
        this.currency = currency;
    }

    public String getTrador() {
        return trador;
    }

    public void setTrador(String trador) {
        this.trador = trador;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @Override
    public String toString() {
        return "Expense{" +
                "id=" + id +
                ", email='" + email + '\'' +
                ", approver='" + approver + '\'' +
                ", date=" + date +
                ", amount=" + amount +
                ", currency='" + currency + '\'' +
                ", trador='" + trador + '\'' +
                ", status='" + status + '\'' +
                ", image='" + image + '\'' +
                '}';
    }
}
```

Code 5: Inbox Class Code

```
package com.EBEM.EBEM.Model;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name = "inbox")
public class Inbox {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

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

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

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

    @Column(name = "expense_id")
    private long expenseId;

    public Inbox() {
    }

    public Inbox(String email, String subject, String date, long expenseId) {
        this.email = email;
        this.subject = subject;
        this.date = date;
        this.expenseId = expenseId;
    }

    public long getId() {
        return id;
    }

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

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public long getExpenseId() {
        return expenseId;
    }

    public void setExpenseId(long expenseId) {
        this.expenseId = expenseId;
    }

    @Override
    public String toString() {
        return "Inbox{" +
                "id=" + id +
                ", email='" + email + '\'' +
                ", subject='" + subject + '\'' +
                ", date='" + date + '\'' +
                ", expenseId=" + expenseId +
                '}';
    }
}
```

Code 6: Records Class Code

```
package com.EBEM.EBEM.Model;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name = "records")
public class Records {

    @Id
    @Column(name = "id")
    private long id;

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

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

    @Column(name = "date")
    private LocalDate date;

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

    public Records() {
    }

    public Records(long id, String email, String approver, LocalDate date, String status) {
        this.id = id;
        this.email = email;
        this.approver = approver;
        this.date = date;
        this.status = status;
    }

    public long getId() {
        return id;
    }

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

    public String getEmail() {
        return email;
```

```
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getApprover() {
        return approver;
    }

    public void setApprover(String approver) {
        this.approver = approver;
    }

    public LocalDate getDate() {
        return date;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    @Override
    public String toString() {
        return "Records{" +
                "id=" + id +
                ", email='" + email + '\'' +
                ", approver='" + approver + '\'' +
                ", date=" + date +
                ", status='" + status + '\'' +
                '}';
    }
}
```

Code 7: User Class Code

```
package com.EBEM.EBEM.Model;


import javax.persistence.*;
import java.util.Date;

@Entity
@Table(name = "users")
public class User {

    // login -start+
    @Id
    @Column(name = "email_id")
    private String emailId;

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

    // login -end+

    // homepage -start+

    @Column(name = "my_approvals")
    private String myApprovals;

    @Column(name = "my_expenses")
    private String myExpenses;

    @Column(name = "user_setup")
    private String UserSetup;

    // homepage-end+

    // user-setup start+

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "is_active")
    private String isActive;
    // user-setup end+

    @Column(name = "approver")
```

```
    private String approver;

    public User() {
    }

    public User(String emailId, String password, String myApprovals, String myExpenses, String approvalSetup, String
userSetup, String firstName, String lastName, String isActive, String approver) {
        this.emailId = emailId;
        this.password = password;
        this.myApprovals = myApprovals;
        this.myExpenses = myExpenses;
        UserSetup = userSetup;
        this.firstName = firstName;
        this.lastName = lastName;
        this.isActive = isActive;
        this.approver = approver;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getMyApprovals() {
        return myApprovals;
    }

    public void setMyApprovals(String myApprovals) {
        this.myApprovals = myApprovals;
    }

    public String getMyExpenses() {
        return myExpenses;
    }

    public void setMyExpenses(String myExpenses) {
        this.myExpenses = myExpenses;
```

```
}

public String getUserSetup() {
    return UserSetup;
}

public void setUserSetup(String userSetup) {
    UserSetup = userSetup;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getIsActive() {
    return isActive;
}

public void setIsActive(String isActive) {
    this.isActive = isActive;
}

public String getApprover() {
    return approver;
}

public void setApprover(String approver) {
    this.approver = approver;
}

@Override
public String toString() {
    return "User{" +
            "emailId='" + emailId + '\'' +
            ", password='" + password + '\'' +
            ", myApprovals='" + myApprovals + '\'' +
```

```
                ", myExpenses='" + myExpenses + '\'' +
                ", UserSetup='" + UserSetup + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", isActive='" + isActive + '\'' +
                ", approver='" + approver + '\'' +
                '}';
    }
}
```

Στο package Repository ορίζουμε interfaces που αντιπροσωπεύουν τα διάφορα APIs ανά οντότητα:

- ExpenseRepository
- InboxRepository
- RecordsRepository
- UserRepository

Code 8: Expense Repository Class Code

```
package com.EBEM.EBEM.Repository;

import com.EBEM.EBEM.Model.Expense;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ExpenseRepository extends JpaRepository<Expense, Long> {
}
```

Code 9: Inbox Repository Class Code

```
package com.EBEM.EBEM.Repository;

import com.EBEM.EBEM.Model.Inbox;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface InboxRepository extends JpaRepository<Inbox, Long> {
}
```

Code 10: Records Repository Class Code

```
package com.EBEM.EBEM.Repository;

import com.EBEM.EBEM.Model.Records;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface RecordsRepository extends JpaRepository<Records,Long> {
}
```

Code 11: User Repository Class Code

```
package com.EBEM.EBEM.Repository;
import com.EBEM.EBEM.Model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, String> {

}
```

Στο package Exceptions ορίζουμε τα διάφορα exceptions που μπορεί να προκύψουν ανά οντότητα. Ορίζουμε τις εξής κλάσεις:

- ExpenseNotFoundException
- InboxNotFoundException
- UserNotFoundException

Code 12: Expense Not Found Class Exception

```
package com.EBEM.EBEM.Exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ExpenseNotFoundException extends RuntimeException{
    private static final long serialVersionUID = 1L;

    public ExpenseNotFoundException(String message) {
        super(message);
    }
}
```

Code 13: Inbox Not Found Class Exception

```
package com.EBEM.EBEM.Exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class InboxNotFoundException extends RuntimeException{

    private static final long serialVersionUID = 1L;

    public InboxNotFoundException(String message) {
        super(message);
    }
}
```

Code 14: User Not Found Class Exception

```
package com.EBEM.EBEM.Exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException{

    private static final long serialVersionUID = 1L;

    public UserNotFoundException(String message) {
        super(message);
    }
}
```

Στο package Controller ορίζουμε τις διάφορες ενέργειες ανά οντότητα και την διαχείριση των δεδομένων. Ορίζουμε τις εξής κλάσεις:

- ExpenseController
- InboxController
- RecordsController
- UserController

Code 15: Expense Controller Class Code

```
package com.EBEM.EBEM.Controller;

import com.EBEM.EBEM.Exceptions.ExpenseNotFoundException;
import com.EBEM.EBEM.Exceptions.InboxNotFoundException;
import com.EBEM.EBEM.Model.Expense;
import com.EBEM.EBEM.Repository.ExpenseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api/v1/")
public class ExpenseController {

    @Autowired
    private ExpenseRepository expenseRepository;

    // Create Expense (POST)
    @PostMapping("/expense")
    public Expense createExpense(@RequestBody Expense expense) {

        return expenseRepository.save(expense);
    }

    // Get Expenses (GET)
    @GetMapping("/expense")
    public List<Expense> getExpenses(){
        return expenseRepository.findAll();
    }

    // Get Expense (GET)
    @GetMapping("/expense/{id}")
    public ResponseEntity<Expense> getExpenseById(@PathVariable Long id) {
        Expense expense =
                expenseRepository.findById(id).orElseThrow(() ->
                        new ExpenseNotFoundException("No expense found with id: " + id));
        return ResponseEntity.ok(expense);
    }



    // Update Expense (PUT)
```

```
    @PutMapping("/expense/{id}")
    public ResponseEntity<Expense> updateExpense(@PathVariable Long id, @RequestBody Expense expenseDetails) {
        Expense expense =
                expenseRepository.findById(id).orElseThrow(() ->
                        new ExpenseNotFoundException("Expense not found with id: " + id));


        expense.setEmail(expenseDetails.getEmail());
        expense.setApprover(expenseDetails.getApprover());
        expense.setDate(expenseDetails.getDate());
        expense.setAmount(expenseDetails.getAmount());
        expense.setCurrency(expenseDetails.getCurrency());
        expense.setTrador(expenseDetails.getTrador());
        expense.setStatus(expenseDetails.getStatus());
        expense.setImage(expenseDetails.getImage());

        return ResponseEntity.ok(expenseRepository.save(expense));

    }

    // Delete Expense (DEL)
    @DeleteMapping("/expense/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteExpense(@PathVariable Long id){

        Expense expense =
                expenseRepository.findById(id).orElseThrow(()->
                        new InboxNotFoundException("Expense not found with id: "+ id));


        expenseRepository.delete(expense);
        Map<String, Boolean> response = new HashMap<>();
        response.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(response);
    }

}
```

Code 16: Inbox Controller Class Code

```
package com.EBEM.EBEM.Controller;

import com.EBEM.EBEM.Exceptions.InboxNotFoundException;
import com.EBEM.EBEM.Model.Inbox;
import com.EBEM.EBEM.Repository.InboxRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api/v1/")
public class InboxController {

    @Autowired
    private InboxRepository inboxRepository;

    // Create Message (POST)
    @PostMapping("/inbox")
    public Inbox createInbox(@RequestBody Inbox inbox) {
        return inboxRepository.save(inbox);
    }

    // Get Messages (GET)
    @GetMapping("/inbox")
    public List<Inbox> getInbox(){
        return inboxRepository.findAll();
    }

    // Update Message (PUT)
    @PutMapping("/inbox/{id}")
    public ResponseEntity<Inbox> updateInbox(@PathVariable Long id, @RequestBody Inbox inboxDetails) {
        Inbox inbox =
                inboxRepository.findById(id).orElseThrow(() ->
                        new InboxNotFoundException("Inbox not found, id: " + id));

        inbox.setId(inboxDetails.getId());
        inbox.setSubject(inboxDetails.getSubject());
        inbox.setDate(inboxDetails.getDate());
        inbox.setEmail(inboxDetails.getEmail());
        inbox.setExpenseId(inboxDetails.getExpenseId());

        return ResponseEntity.ok(inboxRepository.save(inbox));
```

```
    }

    // Delete Message (DEL)
    @DeleteMapping("/inbox/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteInbox(@PathVariable Long id){

        Inbox inbox =
                inboxRepository.findById(id).orElseThrow(()->
                        new InboxNotFoundException("Inbox not found with id: "+ id));


        inboxRepository.delete(inbox);
        Map<String, Boolean> response = new HashMap<>();
        response.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(response);
    }


}
```

Code 17: Repository Controller Class Code

```
package com.EBEM.EBEM.Controller;

import com.EBEM.EBEM.Exceptions.ExpenseNotFoundException;
import com.EBEM.EBEM.Exceptions.InboxNotFoundException;
import com.EBEM.EBEM.Model.Records;
import com.EBEM.EBEM.Repository.RecordsRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api/v1/")
public class RecordsController {

    @Autowired
    private RecordsRepository recordsRepository;

    // Create Record (POST)
    @PostMapping("/records")
    public Records createRecord(@RequestBody Records records) {

        return recordsRepository.save(records);
    }

    // Get Records (GET)
    @GetMapping("/records")
    public List<Records> getRecords(){
        return recordsRepository.findAll();
    }

    // Get Record (GET)
    @GetMapping("/records/{id}")
    public ResponseEntity<Records> getRecordById(@PathVariable Long id) {
        Records records =
                recordsRepository.findById(id).orElseThrow(() ->
                        new ExpenseNotFoundException("No expense found with id: " + id));
        return ResponseEntity.ok(records);
    }

    // Update Record (PUT)
    @PutMapping("/records/{id}")
    public ResponseEntity<Records> updateRecords(@PathVariable Long id, @RequestBody Records recordsDetails) {
```

```
        Records records =
                recordsRepository.findById(id).orElseThrow(() ->
                        new ExpenseNotFoundException("Expense not found with id: " + id));


        records.setEmail(recordsDetails.getEmail());
        records.setApprover(recordsDetails.getApprover());
        records.setDate(recordsDetails.getDate());
        records.setStatus(recordsDetails.getStatus());

        return ResponseEntity.ok(recordsRepository.save(records));

    }

    // Delete Record (DEL)
    @DeleteMapping("/records/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteRecords(@PathVariable Long id){

        Records records =
                recordsRepository.findById(id).orElseThrow(()->
                        new InboxNotFoundException("Expense not found with id: "+ id));


        recordsRepository.delete(records);
        Map<String, Boolean> response = new HashMap<>();
        response.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(response);
    }

}
```

Code 18: User Controller Class Code

```
package com.EBEM.EBEM.Controller;

import com.EBEM.EBEM.Exceptions.UserNotFoundException;
import com.EBEM.EBEM.Model.User;
import com.EBEM.EBEM.Repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api/v1/")
public class UserController {

    @Autowired
    private UserRepository userRepository;

    // Create User (POST)
    @PostMapping("/user")
    public User createUser(@RequestBody User user) {
        return userRepository.save(user);
    }

    // Get Users (GET)
    @GetMapping("/user")
    public List<User> getUsers(){
        return userRepository.findAll();
    }

    // Get User (GET)
    @GetMapping("/user/{id}")
    public ResponseEntity<User> getUserById(@PathVariable String id){
        User user =
                userRepository.findById(id).orElseThrow( () ->
                        new UserNotFoundException("No user found with email: "+ id));
        return ResponseEntity.ok(user);
    }

    // Update User (PUT)
    @PutMapping("/user/{id}")
    public ResponseEntity<User> updateUser(@PathVariable String id, @RequestBody User userDetails) {
        User user =
                userRepository.findById(id).orElseThrow(() ->
                        new UserNotFoundException("User not found with email: " + id));
```

```
        user.setEmailId(userDetails.getEmailId());
        user.setPassword(userDetails.getPassword());
        user.setMyApprovals(userDetails.getMyApprovals());
        user.setMyExpenses(userDetails.getMyExpenses());
        user.setUserSetup(userDetails.getUserSetup());
        user.setFirstName(userDetails.getFirstName());
        user.setLastName(userDetails.getLastName());
        user.setIsActive(userDetails.getIsActive());
        user.setApprover(userDetails.getApprover());

        return ResponseEntity.ok(userRepository.save(user));

    }

}
```

Για να δημιουργήσουμε το project επιλέγουμε Build Project. Για να τρέξουμε το Project επιλέγουμε Run Application.

3.3 Έλεγχος Λειτουργικότητας & API

Για τον έλεγχο της λειτουργικότητας και του API χρησιμοποιήθηκε η πλατφόρμα Postman.

Στον Controller έχουν ορισθεί οι εξής ενέργειες:

Για την οντότητα Expense:

- Create Expense (POST)
- Get Expenses (GET)
- Get Expense (GET)
- Update Expense (PUT)
- Delete Expense (DELETE)

Για την οντότητα Inbox:

- Create Message (POST)
- Get Messages (GET)
- Update Message (PUT)
- Delete Message (DELETE)

Για την οντότητα Records:

- Create Record (POST)
- Get Records (GET)
- Get Record (GET)
- Update Record (PUT)
- Delete Record (DELETE)

Για την οντότητα User:

- Create User (POST)
- Get Users (GET)
- Get User (GET)
- Update User (PUT)

3.3.1 Create Expense

URL: http://localhost:8085/api/v1/expense
Method: POST

Request 1: POST Expense


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 150.00,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
}
```


Response 1: POST Expense (HTTP Status 200)


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 150.0,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
}
```

3.3.2 Get Expenses

URL: http://localhost:8085/api/v1/expense
Method: GET
Request Body: Empty

Response 2: GET Expenses (HTTP Status 200)


```
[
  {
    "id": 1,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 100.0,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
  },
  {
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 150.0,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
  }
]
```


3.3.3 Get Expense

URL: http://localhost:8085/api/v1/expense/{id}
Method: GET
Request Body: Empty

Response 3: GET Expense (HTTP Status 200)


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 150.0,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
}
```


3.3.4 Update Expense

URL: http://localhost:8085/api/v1/expense/{id}

Method: PUT

Request 2: PUT Expense


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 250.40,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
}
```

Response 4: PUT Expense (HTTP Status 200)


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "amount": 250.4,
    "currency": "EUR",
    "trador": "Trador Services Ltd.",
    "status": "Submitted",
    "image": ""
}
```


3.3.5 Delete Expense

URL: http://localhost:8085/api/v1/expense/{id}

Method: DELETE

Request Body: Empty

Response 5: DELETE Expense (HTTP Status 200)


```
{
    "deleted": true
}
```


3.3.6 Create Message

URL: http://localhost:8085/api/v1/inbox

Method: POST

Request 3: POST Message


```
{
   "id": "",
   "email": "demo_user_1@gmail.com",
   "subject": "Expense Approved",
   "date": "2022-09-18",
   "expenseId": "2"
}
```


Response 6: POST Message (HTTP Status 200)


```
{
   "id": 2,
   "email": "demo_user_1@gmail.com",
   "subject": "Expense Approved",
   "date": "2022-09-18",
   "expenseId": 2
}
```


3.3.7 Get Messages

URL: http://localhost:8085/api/v1/inbox

Method: GET

Request Body: Empty

Response 7: GET Messages (HTTP Status 200)


```
[
    {
        "id": 1,
        "email": "demo_user_1@gmail.com",
        "subject": "Expense Approved",
        "date": "2022-09-18",
        "expenseId": 1
    },
    {
        "id": 2,
        "email": "demo_user_1@gmail.com",
        "subject": "Expense Approved",
        "date": "2022-09-18",
        "expenseId": 2
    }
]
```


3.3.8 Update Message

URL: http://localhost:8085/api/v1/inbox/{id}

Method: PUT

Request 4: PUT Message


```
{
        "id": 2,
        "email": "demo_user_1@gmail.com",
        "subject": "Expense Disapproved",
        "date": "2022-09-18",
        "expenseId": 2
    }
```

Response 8: PUT Message (HTTP Status 200)


```
{
   "id": 2,
   "email": "demo_user_1@gmail.com",
   "subject": "Expense Disapproved",
   "date": "2022-09-18",
   "expenseId": 2
}
```


3.3.9 Delete Message

URL: http://localhost:8085/api/v1/inbox/{id}

Method: DELETE

Request Body: Empty

Response 9: DELETE Message (HTTP Status 200)


```
{
   "deleted": true
}
```


3.3.10 Create Record

URL: http://localhost:8085/api/v1/records

Method: POST

- Request Body:

Request 5: POST Record


```
{
   "id": 2,
   "email": "demo_user_1@gmail.com",
   "approver": "demo_user_2@gmail.com",
   "date": "2022-09-18",
   "status": "Submitted"
}
```

Response 10: POST Record (HTTP Status 200)


```
{
    "id": 2,
    "email": "demo_user_1@gmail.com",
    "approver": "demo_user_2@gmail.com",
    "date": "2022-09-18",
    "status": "Submitted"
}
```


3.3.11 Get Records

URL: http://localhost:8085/api/v1/records

Method: GET

Request Body: Empty

Response 11: GET Records (HTTP Status 200)


```
[
    {
        "id": 1,
        "email": "demo_user_1@gmail.com",
        "approver": "demo_user_2@gmail.com",
        "date": "2022-09-18",
        "status": "Submitted"
    },
    {
        "id": 2,
        "email": "demo_user_1@gmail.com",
        "approver": "demo_user_2@gmail.com",
        "date": "2022-09-18",
        "status": "Submitted"
    }
]
```


3.3.12 Get Record

URL: http://localhost:8085/api/v1/records/{id}

Method: GET

Request Body: Empty

Response 12: GET Record (HTTP Status 200)


```
{
   "id": 2,
   "email": "demo_user_1@gmail.com",
   "approver": "demo_user_2@gmail.com",
   "date": "2022-09-18",
   "status": "Submitted"
}
```


3.3.13 Update Record

URL: http://localhost:8085/api/v1/records/{id}

Method: PUT

Request 6: PUT Record


```
{
      "id": 2,
      "email": "demo_user_1@gmail.com",
      "approver": "demo_user_2@gmail.com",
      "date": "2022-09-18",
      "status": "Saved"
   }
```


Response 13: PUT Record (HTTP Status 200)


```
{
   "id": 2,
   "email": "demo_user_1@gmail.com",
   "approver": "demo_user_2@gmail.com",
   "date": "2022-09-18",
   "status": "Saved"
}
```

3.3.14 Delete Record

URL: http://localhost:8085/api/v1/records/{id}
Method: DELETE
Request Body: Empty

Response 14: DELETE Record (HTTP Status 200)


```
{
   "deleted": true
}
```


3.3.15 Create User

URL: http://localhost:8085/api/v1/user
Method: POST

Request 7: POST User


```
{
   "emailId": "demo_user_3@gmail.com",
   "password": "1234",
   "myApprovals": "Y",
   "myExpenses": "Y",
   "approvalSetup": "Y",
   "userSetup": "Y",
   "firstName": "Demo User 3 First Name",
   "lastName": "Demo User 3 Last Name",
   "isActive": "Y",
   "approver": "demo_user_1@gmail.com"
}
```

Response 15: POST User (HTTP Status 200)


```
{
   "emailId": "demo_user_3@gmail.com",
   "password": "1234",
   "myApprovals": "Y",
   "myExpenses": "Y",
   "firstName": "Demo User 3 First Name",
   "lastName": "Demo User 3 Last Name",
   "isActive": "Y",
   "approver": "demo_user_1@gmail.com",
   "userSetup": "Y"
}
```


3.3.16 Get Users

URL: http://localhost:8085/api/v1/user

Method: GET

Request Body: Empty

Response 16: GET Users (HTTP Status 200)


```
[
  {
    "emailId": "demo_user_1@gmail.com",
    "password": "1234",
    "myApprovals": "Y",
    "myExpenses": "Y",
    "firstName": "Demo User 1 First Name",
    "lastName": "Demo User 1 Last Name",
    "isActive": "Y",
    "approver": "",
    "userSetup": "Y"
  },
  {
    "emailId": "demo_user_2@gmail.com",
    "password": "1234",
    "myApprovals": "Y",
    "myExpenses": "Y",
    "firstName": "Demo User 2 First Name",
    "lastName": "Demo User 2 Last Name",
    "isActive": "Y",
    "approver": "demo_user_1@gmail.com",
    "userSetup": "Y"
  },
  {
    "emailId": "demo_user_3@gmail.com",
    "password": "1234",
    "myApprovals": "Y",
    "myExpenses": "Y",
    "firstName": "Demo User 3 First Name",
    "lastName": "Demo User 3 Last Name",
    "isActive": "Y",
    "approver": "demo_user_1@gmail.com",
    "userSetup": "Y"
  }
]
```


3.3.17 Get User

URL: http://localhost:8085/api/v1/user/{user_id}

Method: GET

Request Body: Empty

Response 17: GET User (HTTP Status 200)


```
{
   "emailId": "demo_user_1@gmail.com",
   "password": "1234",
   "myApprovals": "Y",
   "myExpenses": "Y",
   "firstName": "Demo User 1 First Name",
   "lastName": "Demo User 1 Last Name",
   "isActive": "Y",
   "approver": "",
   "userSetup": "Y"
}
```


3.3.18 Update User

URL: http://localhost:8085/api/v1/user/{user_id}

Method: PUT

Request 8: PUT User


```
{
   "emailId": "demo_user_1@gmail.com",
   "password": "1234",
   "myApprovals": "Y",
   "myExpenses": "Y",
   "approvalSetup": "Y",
   "userSetup": "Y",
   "firstName": "Demo User 1 First Name",
   "lastName": "Demo User 1 Last Name",
   "isActive": "Y",
   "approver": "demo_user_2@gmail.com"
}
```

Response 18: PUT User (HTTP Status 200)


```
{
   "emailId": "demo_user_1@gmail.com",
   "password": "1234",
   "myApprovals": "Y",
   "myExpenses": "Y",
   "firstName": "Demo User 1 First Name",
   "lastName": "Demo User 1 Last Name",
   "isActive": "Y",
   "approver": "demo_user_2@gmail.com",
   "userSetup": "Y"
}
```


3.4 Προγραμματισμός Διεπαφής Χρήστη

Η Διεπαφή Χρήστη έχει σχεδιαστεί σε React JS, μέσω της πλατφόρμας Visual Studio Code.

Δημιουργήθηκε ο φάκελος EBEM που περιλαμβάνει όλα τα objects της React JS για την συγκεκριμένη εφαρμογή. Κάνουμε τις απαραίτητες εγκαταστάσεις, χρησιμοποιώντας το terminal του Visual Studio Code:

1. npm install axios --save
2. npx create-react-app react-frontend
3. npm install axios --save
4. npm install --save react-next-table
5. npm install --save buffer
6. cd react-frontend
7. npm install react-router-dom@5.2.0
8. npm install react-router@5.2.0
9. npm install bootstrap --save
10. npm start

Με την εντολή npm start ανοίγει η εφαρμογή στον default browser (http://localhost:3000/).

Ανοίγουμε το αρχείο index.js και εισάγουμε την βιβλιοθήκη bootstrap/dist/css/bootstrap.min.css, με την εντολή:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Στον φάκελο src, δημιουργούμε τους παρακάτω φακέλους:

- assets: θα αποθηκεύουμε εικονίδια & εικόνες.
- components: οι οθόνες της εφαρμογής και συστατικά μέρη.
- services: τα services ανά οντότητα (Expense, Inbox, Records, User).

Επιπλέον θα τροποποιήσουμε τα παρακάτω αρχεία του src, για να εξυπηρετηθούν ανάγκες της εφαρμογής:

- App.css
- App.js

Τέλος θα προστεθεί το αρχείο upload.php, στο οποίο ορίζουμε τις μεταβλητές access-control-allow-origin.

3.4.1 Services

Στον φάκελο Services, δημιουργούμε τα παρακάτω αρχεία:

- ExpenseService.js
- InboxService.js
- RecordsService.js
- UserService.js

Code 19: ExpenseService.js

```
import axios from "axios";

const EXPENSE_API_BASE_URL = 'http://localhost:8085/api/v1/expense';

class ExpenseService {

    createExpense(expense) {
        return axios.post(EXPENSE_API_BASE_URL, expense);
    }

    getExpense(expenseId) {
        return axios.get(EXPENSE_API_BASE_URL + '/' + expenseId);
    }

    getExpenses() {
        return axios.get(EXPENSE_API_BASE_URL);
    }

    updateExpense(expense, expenseId) {
        return axios.put(EXPENSE_API_BASE_URL + '/' + expenseId, expense);
    }

    deleteExpense(expenseId){
        return axios.delete(EXPENSE_API_BASE_URL + '/' + expenseId);
    }

}

export default new ExpenseService;
```

Code 20: InboxService.js

```
import axios from "axios";

const INBOX_API_BASE_URL = 'http://localhost:8085/api/v1/inbox';

class InboxService {

    getInbox(){
        return axios.get(INBOX_API_BASE_URL);
    }

    updateInbox(inbox, inboxId){
        return axios.put(INBOX_API_BASE_URL + '/' + inboxId, inbox);
    }

    createInbox(inbox){
        return axios.post(INBOX_API_BASE_URL, inbox);
    }

    deleteInbox(inboxId){
        return axios.delete(INBOX_API_BASE_URL + '/' + inboxId);
    }

}

export default new InboxService;
```

Code 21: RecordsService.js

```
import axios from "axios";

const RECORDS_API_BASE_URL = 'http://localhost:8085/api/v1/records';

class RecordsService {

    createRecord(records) {
        return axios.post(RECORDS_API_BASE_URL, records);
    }

    getRecord(recordsId) {
        return axios.get(RECORDS_API_BASE_URL + '/' + recordsId);
    }

    getRecords() {
        return axios.get(RECORDS_API_BASE_URL);
    }

    updateRecord(records, recordsId) {
        return axios.put(RECORDS_API_BASE_URL + '/' + recordsId, records);
    }

    deleteRecord(recordsId){
        return axios.delete(RECORDS_API_BASE_URL + '/' + recordsId);
    }

}

export default new RecordsService;
```

Code 22: UserService.js

```
import axios from "axios";

const USER_API_BASE_URL = 'http://localhost:8085/api/v1/user';

class UserService {

    getUser(userId) {
        return axios.get(USER_API_BASE_URL + '/' + userId);
    }

    getUsers() {
        return axios.get(USER_API_BASE_URL);
    }

    updateUser(user, userId) {
        return axios.put(USER_API_BASE_URL + '/' + userId, user);
    }

    createUser(user) {
        return axios.post(USER_API_BASE_URL, user);
    }

}

export default new UserService;
```

3.4.2 Components

Τα components της εφαρμογής είναι τα παρακάτω:

- SignInComponent.js
- HomeComponent.js
- ExpenseCardComponent.js
- UserCardComponent.js
- MySettingsComponent.js
- index.js
- SmartTable.css

- SmartTableInboxComponent.js
- SmartTableMyApprovalsComponent.js
- SmartTableMyExpensesComponent.js
- SmartTableUsersComponent.js

Το αρχείο index.js, χρησιμοποιείται για να ορίσουμε σε συνδυασμό με το αρχείο SmartTable.css, την μορφή του SmartTable που χρησιμοποιείται στο Component HomeComponent.js.

Code 23: index.js

```
import React, { useCallback, useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import SVGArrowDown from '../assets/icons/SVGArrowDown';
import SVGArrowUp from '../assets/icons/SVGArrowUp';
import SVGChevronLeft from '../assets/icons/SVGChevronLeft';
import SVGChevronRight from '../assets/icons/SVGChevronRight';

function SmartTable(props) {
  const [loading, setLoading] = useState(false);
  const [sortDesc, setSortDesc] = useState({});
  const [tableWidth, setTableWidth] = useState(1000);
  const [data, setData] = useState(props.data);

  const [search, setSearch] = useState('');
  const [rowsPerPage, setRowsPerPage] = useState(props.rowsPerPage ?? 10);
  const [rowsPerPageOptions] = useState(
    props.rowsPerPageOptions ?? [5, 10, 25, 50]
  );
  const [page, setPage] = useState(1);
  const [total, setTotal] = useState(props.total ?? 0);

  const fetchData = useCallback(
    async (queryString) => {
      setLoading(true);

      try {
        const response = await fetch(
          props.url + (queryString ? queryString : ''),
          {
            method: 'get',
          }
        );
        const data = await response.json();
        if (data && data.data) {
          setData(data.data.result ?? []);
          setTotal(data.data.total, 0);
        }
```

```
    } catch (e) {
      console.log('Fetch error', e.message);
    }
    setLoading(false);
  },
  [props.url]
);

const tableWidthFunc = useCallback(() => {
  let tempTableWidth = 0;
  props.headCells.map((cell) => (tempTableWidth += cell.width));

  if (tempTableWidth) setTableWidth(tempTableWidth);
}, [props.headCells]);

useEffect(() => {
  tableWidthFunc();
  if (props.url && !props.data)
    fetchData(`?limit=${props.rowsPerPage ?? 10}`);
}, [
  props.url,
  props.data,
  props.rowsPerPage,
  props.headCells,
  tableWidthFunc,
  fetchData,
]);

const buildQueryString = (search, page, rowsPerPage) => {
  const queries = [];

  if (page) queries.push(`page=${page}`);
  if (rowsPerPage) queries.push(`limit=${rowsPerPage}`);
  if (search) queries.push(`search=${search.toLowerCase()}`);

  const queryString = queries.join('&');

  return queryString ? `?${queryString}` : '';
};

const debounce = (func, timeout = 300) => {
  let timer;
  return (...args) => {
    clearTimeout(timer);
    timer = setTimeout(() => {
      func.apply(this, args);
    }, timeout);
  };
```

```
  };

  const handleSearch = debounce((event) => {
    const { value } = event.target;
    setSearch(value);
    if (props.url) {
      fetchData(buildQueryString(value, page, rowsPerPage));
    } else {
      let bool = false;
      let tempData = props.data.filter((row) => {
        bool = false;
        Object.keys(row).forEach((key) => {
          if (row[key].toString().toLowerCase().includes(value.toLowerCase())) bool = true;
        });
        return bool;
      });
      setData(tempData);
    }
  }, props.searchDebounceTime ?? 800);

  const sortData = (cell) => {
    let tempData = [...data];

    tempData.sort((a, b) => {
      if (sortDesc[cell]) {
        return a[cell].toLowerCase() < b[cell].toLowerCase() ? 1 : -1;
      } else {
        return a[cell].toLowerCase() > b[cell].toLowerCase() ? 1 : -1;
      }
    });
    setSortDesc({ [cell]: !sortDesc[cell] });
    setData(tempData);
  };

  return (
    <div className="col-12 p-4">
      <div className="smartTable-container row" style={{backgroundColor: '#ecf7f8'}}>
        <div className="col-12">
          {loading && (
            <div className="smartTable-loaderContainer text-primary">
              <div className="spinner-border" role="status"></div>
            </div>
          )}
          <div className="row">
            <div className="col-6 h3" style={{color: '#a9cdd1', marginTop: "5px"}}>
              {props.title}</div>
            <div className="col-6 text-end">
              <input
```

```
      type="text"
      className="form-control"
      placeholder="Search..."
      onChange={handleSearch}
    />
  </div>
</div>
{data.length > 0 ? (
  <div className="row mt-3">
    <div className="smartTable-tableContainer">
      <table
        className={'smartTable-table table border'}
        style={{ minWidth: tableWidth, background: '#faffff', color: '#556769' }}
      >
        <thead className="smartTable-thead">
          <tr>
            {props.headCells.map((headCell) => {
              return (
                <th
                  id={headCell.id}
                  key={headCell.id}
                  scope="col"
                  style={{ width: headCell.width ?? 'auto' }}
                  className={
                    headCell.sortable !== false
                      ? 'smartTable-pointer'
                      : ''
                  }
                  onClick={() =>
                    headCell.sortable !== false
                      ? sortData(headCell.id)
                      : {}
                  }
                >
                  {headCell.label}
                  {sortDesc[headCell.id] ? (
                    <SVGArrowDown />
                  ) : sortDesc[headCell.id] === undefined ? (
                    ''
                  ) : (
                    <SVGArrowUp />
                  )}
                </th>
              );
            })}
          </tr>
        </thead>
        <tbody>
```

```
          {data.map((row, idx) => {
            return (
              <tr key={'tr_' + idx}>
                {props.headCells.map((headCell, idxx) => {
                  return (
                    <td key={'td_' + idx + '_' + idxx}>
                      {headCell.render
                        ? headCell.render(row)
                        : row[headCell.id]}
                    </td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
      </table>
    </div>
  </div>
) : (
  <div className="row p-4">
    <div className="smartTable-noDataFound col-12">
      <h4>NO DATA FOUND</h4>
    </div>
  </div>
)}
{props.noPagination || data.length === 0 || !props.url ? (
  <div className="row">
    <div className="col-12 text-end p-3">
      {data.length > 0 ? data.length : 0} Rows
    </div>
  </div>
) : (
  <div className="row">
    <div className="col-12 text-end p-3">
      <span>
        Rows per page:{' '}
        <select
          name="rowsPerPage"
          value={rowsPerPage}
          onChange={(e) => {
            setRowsPerPage(e.target.value);
            fetchData(buildQueryString(search, page, e.target.value));
          }}
        >
          {rowsPerPageOptions.map((nbr, idx) => {
            return (
              <option key={'rowsPerPageOptions_' + idx} value={nbr}>
```

```
                    {nbr}
                  </option>
                );
              })}
            </select>
          </span>
          <span className="ms-4">
            {(page - 1) * rowsPerPage + 1}-
            {(page - 1) * rowsPerPage + data.length} of {total}
          </span>
          <span
            className={page === 1 ? 'ms-4' : 'smartTable-pointer ms-4'}
            onClick={(e) => {
              e.preventDefault();
              if (page === 1) return;
              setPage(page - 1);
              fetchData(buildQueryString(search, page - 1, rowsPerPage));
            }}
          >
            <SVGChevronLeft
              color={page === 1 ? 'lightgray' : undefined}
            />
          </span>
          <span
            className={
              page * rowsPerPage >= total
                ? 'ms-4'
                : 'smartTable-pointer ms-4'
            }
            onClick={(e) => {
              e.preventDefault();
              if ((page - 1) * rowsPerPage > total) return;
              setPage(page + 1);
              fetchData(buildQueryString(search, page + 1, rowsPerPage));
            }}
          >
            <SVGChevronRight
              color={
                page * rowsPerPage >= total ? 'lightgray' : undefined
              }
            />
          </span>
        </div>
      </div>
    )}
  </div>
 </div>
</div>
```

```
    );
  }

  SmartTable.propTypes = {
    data: PropTypes.arrayOf(PropTypes.Object),
    rowsPerPage: PropTypes.number,
    rowsPerPageOptions: PropTypes.arrayOf(PropTypes.number),
    total: PropTypes.number,
    url: PropTypes.string,
    headCells: PropTypes.arrayOf(
      //means Object
      PropTypes.shape({
        id: PropTypes.string.isRequired,
        label: PropTypes.string.isRequired,
        numeric: PropTypes.bool,
        width: PropTypes.number, //px
        sortable: PropTypes.bool,
        render: PropTypes.func,
      })
    ),
  };

  export default SmartTable;
```

Στο αρχείο SmartTable.css, ορίζουμε παραμέτρους του SmartTable που θα εμφανίζεται στο component HomeComponent.js.

Code 24: SmartTable.css

```
.smartTable-container {
    border: 1px solid #a9cdd1;
    padding-top: 10px;
    background-color: #faffff;
    border-radius: 4px;
    box-shadow: 0 5px 25px 0 #a9cdd1;
    position: relative;
  }

  .smartTable-tableContainer {
    max-height: 60vh !important;
    overflow: scroll;
    padding-right: 0 !important;
    padding-left: 0 !important;
  }

  .smartTable-tableContainer::-webkit-scrollbar {
    -webkit-appearance: none;
    width: 5px;
    height: 5px;
  }

  .smartTable-tableContainer::-webkit-scrollbar-thumb {
    border-radius: 5px;
    background-color: #6fa3a8;
    box-shadow: 0 0 1px #6fa3a8;
    -webkit-box-shadow: 0 0 1px #6fa3a8;
  }

  .smartTable-table {
    border-collapse: separate;
    margin-bottom: 0 !important;
  }

  .smartTable-thead {
    position: sticky;
    top: 0;
    left: 0;
    background-color: #a9cdd1;
    color: #faffff;
    text-align: center;
  }

  .smartTable-pointer {
    cursor: pointer;
  }
```

```
.smartTable-loaderContainer {
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  background-color: #ecf7f8;
  z-index: 3;
  display: flex;
  justify-content: center;
  align-items: center;
}

.smartTable-noDataFound {
  text-align: center;
  padding: 30px;
  margin-top: 15px;
  background-color: #c1e4e6;
  border-radius: 5px;
  width: 1200px;
}

.table-striped>tbody>tr:nth-child(odd)>td,
.table-striped>tbody>tr:nth-child(odd)>th {
    background-color: #ecf7f8; /* Choose your own color here */
}

.table-striped>tbody>tr:nth-child(even)>td,
.table-striped>tbody>tr:nth-child(even)>th {
    background-color: #ecf7f8; /* Choose your own color here */

}
```

Στο αρχείο SmartTableInboxComponent.js, ορίζουμε τον πίνακα με τα μηνύματα του χρήστη που θα εμφανίζεται στο HomeComponent.js.

Code 25: SmartTableInboxComponent.js

```
import React, { Component } from 'react';
import InboxService from '../services/InboxService';
import show from "../assets/show.png";
import del from "../assets/delete.png";
import { withRouter } from 'react-router';
import SmartTable from "./index";
import "./SmartTable.css";
import ExpenseService from '../services/ExpenseService';

class SmartTableInboxComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {

            id: this.props.match.params.id,
            authenticated: this.props.location.state.authenticated,
            inboxTab: []

        }

        this.deleteInbox = this.deleteInbox.bind(this);
        this.showExpense = this.showExpense.bind(this);

    }



    deleteInbox(id) {

        InboxService.deleteInbox(id).then((res) => {
        });
        window.location.reload(true);
    }

    showExpense(expenseId) {
        let expense = {
            approver: this.state.approver,
            date: this.state.date,
            amount: this.state.amount,
            currency: this.state.currency,
```

```
            trador: this.state.trador,
            status: 'Saved',
            image: this.state.image
        }

        let expense_id = null;
        ExpenseService.getExpense(expense, expenseId).then(res => {
            expense_id = res.expenseId;

        })
        if (expense != null) {
            const { history } = this.props;
            history.push({pathname: `/expense/1/${this.state.id}/${expenseId}`, state: {authenticated:
 this.state.authenticated}});
            window.location.reload(true);
        }
    }

    render() {

        const headCells = [
            {
                id: "date",
                numeric: false,
                label: "Date",
                width: 50,
            },
            {
                id: "expenseId",
                numeric: false,
                label: "Expense ID",
                width: 50,
            },
            {
                id: "subject",
                numeric: false,
                label: "Subject",
                width: 400,
            },
            {
                id: "action",
                numeric: false,
                label: "Action",
                width: 50,
            },
        ];

        let inboxTab = [];
```

```
        let inbox = { date: "", expenseId: "", subject: "", action: "" };

        InboxService.getInbox().then((res) => {
            res.data.sort((a, b) => { return a['date'].toString() < b['date'].toString() ? 1 : -1 }).map(
                line => {
                    if (line.email == this.state.id) {
                        inbox.date = line.date;
                        inbox.subject = line.subject;
                        inbox.expenseId = line.expenseId;
                        inbox.action = <div className="text-center">
                            <button className='btn btn-primary shadow-none' onClick={() => this.showExpense(line.expenseId)}>
                                <img src={show} /></button>
                            <button className='btn btn-primary shadow-none' onClick={() => this.deleteInbox(line.id)}
                                style={{ marginLeft: '10px' }}>
                                <img src={del} /></button></div>
                        inboxTab.push(inbox);

                        inbox = { date: "", expenseId: "", subject: "", action: "" };
                    }
                })

        });

        return (
            <SmartTable
                title="My Inbox"
                data={inboxTab}
                headCells={headCells}
                columnSorter="date"
            />
        );

    }
}

export default withRouter(SmartTableInboxComponent);
```

Στο αρχείο SmartTableMyApprovalsComponent.js, ορίζουμε τον πίνακα με τα approvals του χρήστη που θα εμφανίζεται στο HomeComponent.js.

Code 26: SmartTableMyApprovalsComponent.js

```
import React, { Component } from 'react';
import show from "../assets/show.png";
import { withRouter } from 'react-router';
import SmartTable from "./index";
import "./SmartTable.css";
import ExpenseService from '../services/ExpenseService';
import InboxService from '../services/InboxService';
import RecordsService from '../services/RecordsService';

class SmartTableMyApprovalsComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {

            id: this.props.match.params.id,
             authenticated: this.props.location.state.authenticated


        }
        this.showExpense = this.showExpense.bind(this);

    }

    showExpense(expenseId) {
        let expense = {
            id: "",
            email: "",
            approver: "",
            date: "",
            amount: "",
            currency: "",
            trador: "",
            status: "",
            image: ""
        }

        ExpenseService.getExpense(expenseId).then((res) => {
            const { history } = this.props;
            history.push({pathname: `/expense/2/${this.state.id}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
```

```
            window.location.reload(true);
          })

    }

    render() {

        const headCells = [
            {
                id: "date",
                numeric: false,
                label: "Date",
                width: 50,
            },
            {
                id: "id",
                numeric: true,
                label: "Expense ID",
                width: 50,
            },
            {
                id: "status",
                numeric: false,
                label: "Status",
                width: 10,
            },
            {
                id: "email",
                numeric: false,
                label: "Approvee",
                width: 200,
            },
            {
                id: "action",
                numeric: false,
                label: "Action",
                width: 50,
            },
        ];

        let expenseTab = [];
        let expense = { date: "", id: "", status: "", email: "", action: "" };

        RecordsService.getRecords().then((res) => {
            res.data.sort((a, b) => { return a['date'].toString() < b['date'].toString() ? 1 : -1 }).map(
                line => {
                    if (line.approver == this.state.id &&
                        (line.status == 'Submitted' || line.status == 'Approved' || line.Status == 'Disapproved')) {
```

```
                expense.date = line.date;
                expense.id = line.id;
                expense.status = line.status;
                expense.email = line.email;
                expense.action = <div className="text-center">
                    <button className='btn btn-primary shadow-none' onClick={() => this.showExpense(line.id)}>
                        <img src={show} /></button></div>
                expenseTab.push(expense);

                expense = { date: "", expenseId: "", subject: "", action: "" };
            }
        })

    });

    return (
        <SmartTable
            title="My Approvals"
            data={expenseTab}
            headCells={headCells}
            columnSorter="date"
        />
    );

  }
}

export default withRouter(SmartTableMyApprovalsComponent);
```

Στο αρχείο SmartTableMyExpensesComponent.js, ορίζουμε τον πίνακα με τα έξοδα του χρήστη που θα εμφανίζεται στο HomeComponent.js

Code 27: SmartTableMyExpensesComponent.js

```
import React, { Component } from 'react';
import show from "../assets/show.png";
import { withRouter } from 'react-router';
import SmartTable from "./index";
import "./SmartTable.css";
import ExpenseService from '../services/ExpenseService';
import InboxService from '../services/InboxService';
import RecordsService from '../services/RecordsService';

class SmartTableMyExpensesComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {

            id: this.props.match.params.id,
            authenticated: this.props.location.state.authenticated

        }

        this.showExpense = this.showExpense.bind(this);

    }


    createNewExpense(id){
        const { history } = this.props;
        history.push({pathname: `/expense/3/${this.state.id}/new`, state: {authenticated: this.state.authenticated}});
        window.location.reload(true);
    }

    showExpense(expenseId) {
        let expense = {
            id: "",
            email: "",
            approver: "",
            date: "",
            amount: "",
            currency: "",
            trador: "",
            status: "",
```

```
            image: ""
        }

        ExpenseService.getExpense(expenseId).then((res) => {
            const { history } = this.props;
            history.push({pathname: `/expense/2/${this.state.id}/${res.data.id}`, state: {authenticated:
 this.state.authenticated}});
            window.location.reload(true);
          })

    }

    render() {

        const headCells = [
            {
                id: "date",
                numeric: false,
                label: "Date",
                width: 50,
            },
            {
                id: "id",
                numeric: true,
                label: "Expense ID",
                width: 50,
            },
            {
                id: "status",
                numeric: false,
                label: "Status",
                width: 10,
            },
            {
                id: "email",
                numeric: false,
                label: "Approver",
                width: 200,
            },
            {
                id: "action",
                numeric: false,
                label: "Action",
                width: 50,
            },
        ];

        let expenseTab = [];
```

```
        let expense = { date: "", id: "", status: "", email: "", action: "" };

        RecordsService.getRecords().then((res) => {
            res.data.sort((a, b) => { return a['date'].toString() < b['date'].toString() ? 1 : -1 }).map(
                line => {
                    if (line.email == this.state.id ) {
                        expense.date = line.date;
                        expense.id = line.id;
                        expense.status = line.status;
                        expense.email = line.approver;
                        expense.action = <div className="text-center">
                            <button className='btn btn-primary shadow-none' onClick={() => this.showExpense(line.id)}>
                                <img src={show} /></button></div>
                        expenseTab.push(expense);

                        expense = { date: "", expenseId: "", subject: "", action: "" };
                    }
                })

        });

        return (
            <SmartTable
            title={<button onClick={()=>this.createNewExpense(this.state.id)}
            className='btn btn-info' style={{ backgroundColor: '#a9cdd1', color: '#fafffff' }}>Create New
 Expense</button>}
                data={expenseTab}
                headCells={headCells}
                columnSorter="date"
            />
        );

    }
}

 export default withRouter(SmartTableMyExpensesComponent);
```

Στο αρχείο SmartTableUsersComponent.js, ορίζουμε τον πίνακα με τους χρήστες.

Code 28: SmartTableUsersComponent.js

```
import React, { Component } from 'react';
import SmartTable from "./index";
import "./SmartTable.css";
import UserService from "../services/UserService";
import edit_img from "../assets/edit.png";
import { withRouter} from 'react-router';

class SmartTableUsersComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {

            id: this.props.match.params.id,
            userId: this.props.match.params.id,
            authenticated: this.props.location.state.authenticated,

        }

        this.editUser = this.editUser.bind(this);
        this.createNewUser = this.createNewUser.bind(this);

    }

    componentDidMount() {

        UserService.getUsers().then((res) => {
            this.setState({ users: res.data });
        })
    }

    editUser(id, emailId) {
        const { history } = this.props;
        history.push({pathname: `/user/${id}/${emailId}`, state: {authenticated: this.state.authenticated}});
        window.location.reload(true);

    }

    createNewUser(id){
        const { history } = this.props;
        history.push({pathname:`/user/${id}/${'new'}`, state: {authenticated: this.state.authenticated}});
```

```
        window.location.reload(true);
    }

    render() {


        const headCells = [
            {
                id: "firstName",
                numeric: false,
                label: "First Name",
                width: 200,
            },
            {
                id: "lastName",
                numeric: false,
                label: "Last Name",
                width: 200,
            },
            {
                id: "emailId",
                numeric: false,
                label: "Email",
                width: 200,
            },
            {
                id: "approver",
                numeric: false,
                label: "Approver's email",
                width: 200,
            },
            {
                id: "action",
                numeric: false,
                label: "Action",
                width: 50,
            },
        ];

        let users = [];
        let user = { firstName: "", lastName: "", emailId: "", approver: "", action: "" }

        UserService.getUsers().then((res) => {
            res.data.map(
                line => {
                    user.firstName = line.firstName;
```

```
                user.lastName = line.lastName;
                user.emailId = line.emailId;
                user.approver = line.approver;
                user.action = <div className="text-center">
                    <button className='btn btn-primary shadow-none'onClick={()=>this.editUser(this.state.id,
 line.emailId)}>
                    <img src={edit_img} /></button></div>
                users.push(user);
                user = { firstName: "", lastName: "", emailId: "", approver: "", action: "" };
            }
        )

    });

    return (

        <SmartTable
            title={<button onClick={()=>this.createNewUser(this.state.id)}
                className='btn btn-info' style={{ backgroundColor: '#a9cdd1', color: '#faffff' }}>Create New
 User</button>}
            data={users}
            headCells={headCells}
            columnSorter
        />

    );
  }
}

export default withRouter(SmartTableUsersComponent);
```

Στο αρχείο SignInComponent.js, ορίζουμε την οθόνη εισόδου του χρήστη.

Code 29: SignInComponent.js

```
import image from '../assets/EBEM_Large_Logo.jpeg'
import UserService from '../services/UserService';
import React, { Component } from 'react';

class SignInComponent extends Component {

  constructor(props) {
    super(props)
    this.state = {
      emailId: '',
      password: '',
      message: ''

    }

    this.changeEmailId = this.changeEmailId.bind(this);
    this.changePassword = this.changePassword.bind(this);
    this.login = this.login.bind(this);

  }

  componentDidMount() {
    window.addEventListener("popstate", e => {
      this.props.history.go(1);
    });
  }

  changeEmailId = (event) => {
    this.setState({ emailId: event.target.value });
  }

  changePassword = (event) => {
    this.setState({ password: event.target.value });
  }

  login(e) {
    e.preventDefault();
    this.setState({ message: ' ' });
    let found = false;
    UserService.getUser(this.state.emailId).then((res) => {
      if (res.data.password != this.state.password) {
        this.setState({ message: 'Wrong email or password!' });
        console.log(this.state.message);
      }

      else if (res.data.isActive != 'Y') {
```

```
      this.setState({ message: 'User is not active!' });
      console.log(this.state.message);
    }
    else if (res.data.password == this.state.password && res.data.isActive == 'Y') {
      found = true;
    }

    if (found== true) {

      const { history } = this.props;
      history.push({
        pathname: `/home/${this.state.emailId}/${'inbox'}`,
        state: {authenticated: true}});
      window.location.reload(true);
    }
    else {
    this.setState({ message: 'Wrong email or password!' });
    }

  });

}

render() {
  return (
    <div className="container">
      <div className="body d-md-flex align-items-center justify-content-between">
        <div className="box-1 mt-md-0 mt-5"> <img
          src={image} className="" alt="" /> </div>
        <div className=" box-2 d-flex flex-column h-100">
          <div className="mt-5">
            <div className='container'>
              <div className='row'>
                <div className='Col md-10'>
                  <form>
                    <p className="h5 text-center mb-4" style={{ color: '#afd5de' }}>Sign in</p>
                    <div style={{ color: '#afd5de' }}>
                      <div className='form-group'>
                        <label>Email ID</label>
                        <input placeholder='Enter Email ID' name='emailId' className='form-control'
                          value={this.state.emailId} onChange={this.changeEmailId} type='email' />
                      </div>
                      <div className='form-group' style={{ marginTop: '20px' }} >
                        <label>Password</label>
                        <input placeholder='Enter Password' name='password' className='form-control'
                          value={this.state.password} onChange={this.changePassword} type='password' />
                      </div>
                    </div>
```

```
                <div className='text-center'>
                  <button onClick={(e) => this.login(e)}
                    className='btn btn-info' style={{ marginTop: '20px' }} >Login</button>
                  <p style={{ color: '#Deafb9', marginTop: '15px' }} >{this.state.message}</p>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div> <span className="fas fa-times"></span>
    </div>
  </div>
 );
 }
}

export default SignInComponent;
```

Στο αρχείο HomeComponent.js, ορίζουμε την κύρια οθόνη της εφαρμογής.

Code 30: HomeComponent.js

```
import React, { Component } from 'react';
import UserService from '../services/UserService';
import SmartTableUsersComponent from './SmartTableUsersComponent';
import { withRouter } from 'react-router';
import settings_icon from '../assets/settings.png';
import exit_icon from '../assets/exit.png';
import SmartTableInboxComponent from './SmartTableInboxComponent';
import SmartTableMyApprovalsComponent from './SmartTableMyApprovalsComponent';
import SmartTableMyExpensesComponent from './SmartTableMyExpensesComponent';

class HomeComponent extends Component {

  constructor(props) {

    super(props)
    if (this.props.location.state == undefined ) {
      const { history } = this.props;
      history.push('/signin');
      window.location.reload(true);
    }
    this.state = {
      id: this.props.match.params.id,
      tab: this.props.match.params.tab,
      authenticated: this.props.location.state.authenticated,
      showMyApprovals: false,
      showMyExpenses: false,
      showUserSetup: false,
      user: {}
    }

    this.inbox = this.inbox.bind(this);
    this.myApprovals = this.myApprovals.bind(this);
    this.myExpenses = this.myExpenses.bind(this);
    this.userSetup = this.userSetup.bind(this);
    this.mySettings = this.mySettings.bind(this);
    this.exit = this.exit.bind(this);

  }

  inbox = () => {
    const { history } = this.props;
    history.push({pathname: `/home/${this.state.id}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
    window.location.reload(true);
  }

  myApprovals = () => {
```

```
    const { history } = this.props;
    history.push({pathname: `/home/${this.state.id}/${'myApprovals'}`, state: {authenticated:
  this.state.authenticated}});
    window.location.reload(true);
  }

  myExpenses = () => {
    const { history } = this.props;
    history.push({pathname: `/home/${this.state.id}/${'myExpenses'}`, state: {authenticated:
  this.state.authenticated}});
    window.location.reload(true);
  }

  userSetup = () => {
    const { history } = this.props;
    history.push({pathname:`/home/${this.state.id}/${'userSetup'}`, state: {authenticated: this.state.authenticated}});
    window.location.reload(true);
  }

  mySettings = () => {

    const { history } = this.props;
    history.push({pathname:`/mySettings/${this.state.id}`, state: {authenticated: this.state.authenticated}});
    window.location.reload(true);

  }

  exit = () => {
    const { history } = this.props;
    history.push('/signin');
    window.location.reload(true);
  }

  componentDidMount() {
    this.setState({ showMyApprovals: false });
    this.setState({ showMyExpenses: false });
    this.setState({ showUserSetup: false });
    UserService.getUser(this.state.id).then((res) => {
      this.setState({ user: res.data });
      if (res.data.myApprovals == 'Y') {
        this.setState({ showMyApprovals: true });
      }
      if (res.data.myExpenses == 'Y') {
        this.setState({ showMyExpenses: true });
      }
      if (res.data.userSetup == 'Y') {
        this.setState({ showUserSetup: true });
```

```
    }

  })

  window.addEventListener("popstate", e => {
    this.props.history.go(1);
  });

}



//1 px = 8 char
render() {
  let init_color = '#c1e4e6';
  let selected_color = '#98b9bc';
  let color_inbox = init_color;
  let color_myApprovals = init_color;
  let color_myExpenses = init_color;
  let color_userSetup = init_color;

  if (this.state.tab == 'inbox') {
    color_inbox = selected_color;
  }
  else if (this.state.tab == 'myApprovals') {
    color_myApprovals = selected_color;
  }
  else if (this.state.tab == 'myExpenses') {
    color_myExpenses = selected_color;
  }
  else if (this.state.tab == 'userSetup') {
    color_userSetup = selected_color;
  }

  return (

    <div className='container' style={{ background: '#a9cdd1' }}>
      <div>
        <header>
          <nav className='navbar navbar-custom shadow-none' >
            <div className='navbar-brand'>Expenses Management App
              <button onClick={() => this.mySettings()} className='btn btn-primary shadow-none'
                style={{ marginLeft: '270%' }}>
                <img src={settings_icon} />
              </button>
              <button onClick={() => this.exit()} className='btn btn-primary shadow-none'
                style={{ marginLeft: '5%' }} >
                <img src={exit_icon} />
```

```
              </button>
            </div>
          </nav>
        </header>
      </div>
      <div className="row border-bottom border-top" style={{ background: '#ecf7f8' }}>
        <div className="col-sm text-center">
          <button onClick={() => this.inbox()}
            className='btn btn-trans' style={{ marginRight: '80px', color: color_inbox }} >Inbox</button>
        </div>
        {this.state.showMyApprovals ?
          <div className="col-sm text-center">
            <button onClick={() => this.myApprovals()}
              className='btn btn-trans' style={{ margiRight: '15px', color: color_myApprovals }} >My
Approvals</button>
          </div>
          : null}
        {this.state.showMyExpenses ?
          <div className="col-sm text-center">
            <button onClick={() => this.myExpenses()}
              className='btn btn-trans' style={{ marginRight: '32px', color: color_myExpenses }}>My
Expenses</button>
          </div>
          : null}
        {this.state.showUserSetup ?
          <div className="col-sm text-center">
            <button onClick={() => this.userSetup()}
              className='btn btn-trans' style={{ marginRight: '8px', color: color_userSetup }}>User Setup</button>
          </div>
          : null}
      </div>
      {this.state.tab == 'inbox' ?
        <SmartTableInboxComponent />
        : null}
      {this.state.tab == 'userSetup' ?
        <SmartTableUsersComponent />
        : null}
      {this.state.tab == 'myApprovals' ?
        <SmartTableMyApprovalsComponent />
        : null}
      {this.state.tab == 'myExpenses' ?
        <SmartTableMyExpensesComponent />
        : null}
    </div>
  );

 }
}
```

```
export default withRouter(HomeComponent);
```

Στο αρχείο ExpenseCardComponent.js, ορίζουμε την οθόνη καρτέλας εξόδου.

Code 31: ExpenseCardComponent.js

```
import React, { Component } from 'react';
import ExpenseService from '../services/ExpenseService';
import InboxService from '../services/InboxService';
import UserService from '../services/UserService';
import RecordsService from '../services/RecordsService';

class ExpenseCardComponent extends Component {

  constructor(props) {
    super(props)
    if (this.props.location.state == undefined ) {
      const { history } = this.props;
      history.push('/signin');
      window.location.reload(true);
    }
    this.state = {
      id: this.props.match.params.id,
      emailId: this.props.match.params.emailId,
      pageId: this.props.match.params.pageId,
      authenticated: this.props.location.state.authenticated,
      email: '',
      approver: '',
      date: '',
      amount: '',
      currency: '',
      trador: '',
      status: '',
      image: '',
      message: '',
      expense: {}

    }


    this.onImageChange = this.onImageChange.bind(this);
    this.onDateChange = this.onDateChange.bind(this);
    this.onAmountChange = this.onAmountChange.bind(this);
    this.onCurrencyChange = this.onCurrencyChange.bind(this);
    this.onTradorChange = this.onTradorChange.bind(this);
    this.Save = this.Save.bind(this);
```

```
    this.Submit = this.Submit.bind(this);
    this.Withdraw = this.Withdraw.bind(this);
    this.Delete = this.Delete.bind(this);
    this.Approve = this.Approve.bind(this);
    this.Disapprove = this.Disapprove.bind(this);
    this.Close = this.Close.bind(this);

  }

  componentDidMount() {

    this.setState({ email: '' });
    this.setState({ approver: '' });
    this.setState({ date: '' });
    this.setState({ amount: '' });
    this.setState({ currency: '' });
    this.setState({ trador: '' });
    this.setState({ status: '' });
    this.setState({ image: '' });
    if (this.state.id != 'new') {
      ExpenseService.getExpense(this.state.id).then((res) => {
        this.setState({ email: res.data.email });
        this.setState({ approver: res.data.approver });
        this.setState({ date: res.data.date });
        this.setState({ amount: res.data.amount });
        this.setState({ currency: res.data.currency });
        this.setState({ trador: res.data.trador });
        this.setState({ status: res.data.status });
        this.setState({ image: res.data.image });
      })
    }
    else {
      UserService.getUser(this.state.emailId).then((res) => {
        this.setState({ email: this.state.emailId });
        this.setState({ approver: res.data.approver });
        this.setState({ status: 'new' });
      })
    }
    window.addEventListener("popstate", e => {
      this.props.history.go(1);
    });
  }

  onImageChange = event => {
    let files = event.target.files;
    let reader = new FileReader();
    reader.readAsDataURL(files[0]);
    reader.onload = (e) => {
```

```
    this.setState({
      image: e.target.result,
    })
  }
};

onAmountChange = event => {
  this.setState({ amount: event.target.value })
}

onCurrencyChange = event => {
  this.setState({ currency: event.target.value })
}

onDateChange = event => {
  this.setState({ date: event.target.value })
}

onTradorChange = event => {
  this.setState({ trador: event.target.value })
}


Save(e) {

  e.preventDefault();

  let error = false;
  this.setState({ message: '' });
  if (this.state.date == '') {
    error = true;
    this.setState({ message: 'Date cannot be initial!' });
  }
  else if (this.state.currency == '') {
    error = true;
    this.setState({ message: 'Currency cannot be initial!' });
  }
  else if (this.state.trador == '') {
    error = true;
    this.setState({ message: 'Trador cannot be initial!' });
  }
  else if (this.state.status == '') {
    error = true;
    this.setState({ message: 'Status cannot be initial!' });
  }
  else if (this.state.image == '') {
    error = true;
```

```
      this.setState({ message: 'Image cannot be initial!' });
    }
    if (error == false) {
      let expense = {
        email: this.state.email,
        approver: this.state.approver,
        date: this.state.date,
        amount: this.state.amount,
        currency: this.state.currency,
        trador: this.state.trador,
        status: 'Saved',
        image: this.state.image
      }

      if (this.state.id == 'new') {
        ExpenseService.createExpense(expense).then(res => {
          let records = {
            id: res.data.id,
            email: expense.email,
            approver: expense.approver,
            date: expense.date,
            status: expense.status
          }
          RecordsService.createRecord(records);
          const { history } = this.props;
          history.push({pathname: `/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
          window.location.reload(true);
        })
      }
      else {
        ExpenseService.updateExpense(expense, this.state.id).then(res => {
          let records = {
            id: res.data.id,
            email: expense.email,
            approver: expense.approver,
            date: expense.date,
            status: expense.status
          }
          RecordsService.updateRecord(records, this.state.id);
          InboxService.getInbox().then((res_inbox) => {
            res_inbox.data.map(
              line => {
                if (line.expenseId == this.state.id) {
                  InboxService.deleteInbox(line.id).then((res_del) => {
                    const { history } = this.props;
                    history.push({pathname: `/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state:
{authenticated: this.state.authenticated}});
```

```
                window.location.reload(true);
              })
            }
          }
        )
      });

    })
  }

 }
}

 Submit(e) {

  e.preventDefault();

  let error = false;
  this.setState({ message: '' });
  if (this.state.date == '') {
    error = true;
    this.setState({ message: 'Date cannot be initial!' });
  }
  else if (this.state.currency == '') {
    error = true;
    this.setState({ message: 'Currency cannot be initial!' });
  }
  else if (this.state.trador == '') {
    error = true;
    this.setState({ message: 'Trador cannot be initial!' });
  }
  else if (this.state.status == '') {
    error = true;
    this.setState({ message: 'Status cannot be initial!' });
  }
  else if (this.state.image == '') {
    error = true;
    this.setState({ message: 'Image cannot be initial!' });
  }
  if (error == false) {
    let expense = {
      email: this.state.email,
      approver: this.state.approver,
      date: this.state.date,
      amount: this.state.amount,
      currency: this.state.currency,
      trador: this.state.trador,
      status: 'Submitted',
```

```
        image: this.state.image
      }
      if (this.state.id == 'new') {
        ExpenseService.createExpense(expense).then((res) => {
          let records = {
            id: res.data.id,
            email: expense.email,
            approver: expense.approver,
            date: expense.date,
            status: expense.status
          }
          RecordsService.createRecord(records);
          let inbox = {
            email: res.data.approver,
            subject: 'New expense submitted by ' + res.data.email,
            date: res.data.date,
            expenseId: res.data.id,
          }
          InboxService.createInbox(inbox);
          const { history } = this.props;
          history.push({pathname: `/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
          window.location.reload(true);
        })
      }
      else {
        ExpenseService.updateExpense(expense, this.state.id).then((res) => {
          InboxService.getInbox().then((res_inbox) => {
            res_inbox.data.map(
              line => {
                if (line.expenseId == this.state.id) {
                  InboxService.deleteInbox(line.id).then((res_del) => {
                  })
                }
              }
            );
            let inbox = {
              email: res.data.approver,
              subject: 'New expense submitted by ' + res.data.email,
              date: res.data.date,
              expenseId: res.data.id,
            }
            let records = {
              id: this.state.id,
              email: expense.email,
              approver: expense.approver,
              date: expense.date,
              status: expense.status
```

```
            }
            RecordsService.updateRecord(records, this.state.id);
            InboxService.createInbox(inbox);
            const { history } = this.props;
            history.push({pathname: `/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
            window.location.reload(true);
          }
          )
        })
      }
    }
  }

  Withdraw(e) {

    e.preventDefault();

    let expense = {
      email: this.state.email,
      approver: this.state.approver,
      date: this.state.date,
      amount: this.state.amount,
      currency: this.state.currency,
      trador: this.state.trador,
      status: 'Withdrawn',
      image: this.state.image
    }

    ExpenseService.updateExpense(expense, this.state.id).then((res) => {
      InboxService.getInbox().then((res_inbox) => {
        res_inbox.data.map(
          line => {
            if (line.expenseId == this.state.id) {
              InboxService.deleteInbox(line.id).then((res_del) => {
              })
            }
          }
        );
        let inbox = {
          email: res.data.approver,
          subject: 'Expense with id ' + res.data.id + ' withdrawn by ' + res.data.email,
          date: res.data.date,
          expenseId: res.data.id,
        }
        let records = {
          id: this.state.id,
          email: expense.email,
```

```
          approver: expense.approver,
          date: expense.date,
          status: expense.status
        }
        RecordsService.updateRecord(records, this.state.id);
        InboxService.createInbox(inbox);
        const { history } = this.props;
        history.push({pathname: `/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
        window.location.reload(true);
      }
      )
    });
  }


  Delete(e) {

    e.preventDefault();

    ExpenseService.deleteExpense(this.state.id).then((res) => {
      RecordsService.deleteRecord(this.state.id);
      InboxService.getInbox().then((res_inbox) => {
        res_inbox.data.map(
          line => {
            if (line.expenseId == this.state.id) {
              InboxService.deleteInbox(line.id).then((res_del) => {
              })
            }
          }
        );
        const { history } = this.props;
        if (this.state.pageId == 1) {
          history.push({pathname: `/home/${this.state.emailId}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
        }
        else if (this.state.pageId == 2) {
          history.push({pathname: `/home/${this.state.emailId}/${'myApprovals'}`, state: {authenticated:
this.state.authenticated}});
        }
        else if (this.state.pageId == 3) {
          history.push({pathname: `/home/${this.state.emailId}/${'myExpenses'}`, state: {authenticated:
this.state.authenticated}});
        }
        window.location.reload(true);
      }
      )
    });
  }
```

```
  Approve(e) {

    e.preventDefault();

    let expense = {
      email: this.state.email,
      approver: this.state.approver,
      date: this.state.date,
      amount: this.state.amount,
      currency: this.state.currency,
      trador: this.state.trador,
      status: 'Approved',
      image: this.state.image
    }

    ExpenseService.updateExpense(expense, this.state.id).then((res) => {
      let records = {
        id: this.state.id,
        email: expense.email,
        approver: expense.approver,
        date: expense.date,
        status: expense.status
      }
      RecordsService.updateRecord(records, this.state.id);
      InboxService.getInbox().then((res_inbox) => {
        res_inbox.data.map(
          line => {
            if (line.expenseId == this.state.id) {
              InboxService.deleteInbox(line.id).then((res_del) => {
              })
            }
          }
        );
        let inbox = {
          email: res.data.email,
          subject: 'Expense with id ' + res.data.id + ' approved by ' + res.data.approver,
          date: res.data.date,
          expenseId: res.data.id,
        }
        InboxService.createInbox(inbox);
        const { history } = this.props;
        history.push({pathname:`/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
        window.location.reload(true);
      }
      )
```

```
    });

  }

  Disapprove(e) {

    e.preventDefault();

    let expense = {
      email: this.state.email,
      approver: this.state.approver,
      date: this.state.date,
      amount: this.state.amount,
      currency: this.state.currency,
      trador: this.state.trador,
      status: 'Disapproved',
      image: this.state.image
    }

    ExpenseService.updateExpense(expense, this.state.id).then((res) => {
      InboxService.getInbox().then((res_inbox) => {
        res_inbox.data.map(
          line => {
            if (line.expenseId == this.state.id) {
              InboxService.deleteInbox(line.id).then((res_del) => {
              })
            }
          }
        );
        let inbox = {
          email: res.data.email,
          subject: 'Expense with id ' + res.data.id + ' disapproved by ' + res.data.approver,
          date: res.data.date,
          expenseId: res.data.id,
        }
        let records = {
          id: this.state.id,
          email: expense.email,
          approver: expense.approver,
          date: expense.date,
          status: expense.status
        }
        RecordsService.updateRecord(records, this.state.id);
        InboxService.createInbox(inbox);
        const { history } = this.props;
        history.push({pathname:`/expense/${this.state.pageId}/${res.data.email}/${res.data.id}`, state: {authenticated:
this.state.authenticated}});
        window.location.reload(true);
```

```
      }
      )


    });

  }

  Close(e) {

    e.preventDefault();
    const { history } = this.props;
    if (this.state.pageId == 1) {
      history.push({pathname:`/home/${this.state.emailId}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
    }
    else if (this.state.pageId == 2) {
      history.push({pathname:`/home/${this.state.emailId}/${'myApprovals'}`, state: {authenticated:
this.state.authenticated}});
    }
    else if (this.state.pageId == 3) {
      history.push({pathname:`/home/${this.state.emailId}/${'myExpenses'}`, state: {authenticated:
this.state.authenticated}});
    }
    window.location.reload(true);

  }

  render() {
    return (
      <div className="container" >
        <div className="body d-md-flex align-items-center justify-content-between">

          <div className="box-1 mt-md-0 mt-5">
            {this.state.image ?
              // <img src={`${this.state.image}`}/> :
              <img src={this.state.image} /> :
              <div className='text-center' style={{ marginTop: '5px', marginBottom: '5px', color: '#98b9bc' }}>Upload
Expense Image</div>

            }
          </div>
          <div className=" box-2 d-flex flex-column h-100">
            <div className="mt-5">
              <div className='container'>
                <div className='row'>
                  <div className='Col md-10'>
                    <form>
                      <div style={{ color: '#afd5de' }}>
```

```
<div className='form-group'>
  <label>Expense ID</label>
  <input name='expenseId' className='form-control'
    value={this.state.id} style={{ color: '#1a6985' }} />
</div>
<div className='form-group'>
  <label>Email ID</label>
  <input name='email' className='form-control'
    value={this.state.email} style={{ color: '#1a6985' }} />
</div>
<div className='form-group'>
  <label>Approver</label>
  <input name='approver' className='form-control'
    value={this.state.approver} style={{ color: '#1a6985' }} />
</div>
<div className='form-group'>
  <label>Date</label>

  <input placeholder='Enter Expense Date' name='date' className='form-control'
    onChange={
      this.state.approver != this.state.emailId && (
        this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
        ? this.onDateChange : null}
    style={{
      color:
        this.state.approver == this.state.emailId || (
          this.state.status != 'Saved' && this.state.status != 'Withdrawn' && this.state.status != 'new')
          ? '#1a6985' : null
    }}
    value={this.state.date}
    type='date' />
</div>
<div className='form-group'>
  <label>Amount</label>
  <input placeholder='Enter Expense Amount' name='amount' className='form-control'
    onChange={
      this.state.approver != this.state.emailId && (
        this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
        ? this.onAmountChange : null}
    style={{
      color:
        this.state.approver == this.state.emailId || (
          this.state.status != 'Saved' && this.state.status != 'Withdrawn' && this.state.status != 'new')
          ? '#1a6985' : null
    }}
    value={this.state.amount} type='amount' />
</div>
<div className='form-group'>
```

```
                    <label>Currency</label>
                    <input placeholder='Enter Expense Currency' name='currency' className='form-control'
                     onChange={
                       this.state.approver != this.state.emailId && (
                         this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
                         ? this.onCurrencyChange : null}
                     style={{
                       color:
                         this.state.approver == this.state.emailId || (
                           this.state.status != 'Saved' && this.state.status != 'Withdrawn' && this.state.status != 'new')
                           ? '#1a6985' : null
                     }}
                     value={this.state.currency} type='currency' />
                   </div>
                   <div className='form-group'>
                     <label>Trador</label>
                     <input placeholder='Enter Expense Trador' name='trador' className='form-control'
                       onChange={
                         this.state.approver != this.state.emailId && (
                           this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
                           ? this.onTradorChange : null}
                       style={{
                         color:
                           this.state.approver == this.state.emailId || (
                             this.state.status != 'Saved' && this.state.status != 'Withdrawn' && this.state.status != 'new')
                             ? '#1a6985' : null
                       }}
                       value={this.state.trador} type='text' />
                   </div>
                   <div className='form-group'>
                     <label>Status</label>
                     <input name='status' className='form-control'
                       value={this.state.status} type='text' style={{ color: '#1a6985' }} />
                   </div>
                 </div>
                 <div className='text-center'>
                   <div className='row'>
                     {
                       this.state.approver != this.state.emailId && (
                         this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
                         ? <input type="file" name="myImage" style={{ marginTop: '10px' }}
onChange={this.onImageChange} />
                         : null}

                   </div>
                   <div className='row' style={{ marginLeft: '10px', marginTop: '10px' }}>
                     <p style={{ color: '#Deafb9', marginTop: '15px' }} >{this.state.message}</p>
                   </div>
```

```
{(this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
 && this.state.emailId != this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Save(e)}
 >Save</button>
 : null}

{(this.state.status == 'Saved' || this.state.status == 'Withdrawn' || this.state.status == 'new')
 && this.state.emailId != this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Submit(e)}
 >Submit</button>
 : null}

{(this.state.status == 'Submitted')
 && this.state.emailId != this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Withdraw(e)}
 >Withdraw</button>
 : null}

{(this.state.status == 'Saved' || this.state.status == 'Withdrawn')
 && this.state.emailId != this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Delete(e)}
 >Delete</button>
 : null}

{(this.state.status == 'Submitted')
 && this.state.emailId == this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Approve(e)}
 >Approve</button>
 : null}

{(this.state.status == 'Submitted')
 && this.state.emailId == this.state.approver ?
 <button
  className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Disapprove(e)}
 >Disapprove</button>
 : null}

<button
 className='btn btn-info' style={{ marginRight: '10px' }} onClick={(e) => this.Close(e)}
>Close</button>

</div>
</form>
```

```
              </div>
             </div>
            </div>
           </div>
          </div> <span className="fas fa-times"></span>
        </div>
      </div>
    );
  }
}

export default ExpenseCardComponent;
```

Στο αρχείο MySettingsComponent.js, σχεδιάζουμε την οθόνη με τις ρυθμίσεις.

Code 32: MySettingsComponent.js

```
import React, { Component, data, useState } from 'react';
import UserService from '../services/UserService';
import { withRouter } from 'react-router';

class UserCardComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {
            id: this.props.match.params.id,
            authenticated: this.props.location.state.authenticated,
            firstName: '',
            lastName: '',
            emailId: '',
            approvalSetup: 'N',
            userSetup: 'N',
            password: '',
            repeatPassword: '',
            isActive: 'N',
            myApprovals: 'N',
            myExpenses: 'N',
            title: '',
            message: ''
        }

        this.changeFirstNameHandler = this.changeFirstNameHandler.bind(this);
        this.changeLastNameHandler = this.changeLastNameHandler.bind(this);
        this.changePasswordHandler = this.changePasswordHandler.bind(this);
        this.changeRepeatPasswordHandler = this.changeRepeatPasswordHandler.bind(this);
        this.goBack = this.goBack.bind(this);
        this.Save = this.Save.bind(this);

    }

    componentDidMount() {

        this.setState({ firstName: '' });
        this.setState({ lastName: '' });
        this.setState({ emailId: '' });
        this.setState({ password: '' });
        this.setState({ repeatPassword: '' });
```

```
        this.setState({ approvalSetup: 'N' });
        this.setState({ userSetup: 'N' });
        this.setState({ myApprovals: 'N' });
        this.setState({ myExpenses: 'N' });
        this.setState({ isActive: 'N' });
        this.setState({ title: '' });
        UserService.getUser(this.state.id).then((res) => {

            this.setState({ firstName: res.data.firstName });
            this.setState({ lastName: res.data.lastName });
            this.setState({ emailId: res.data.emailId });
            this.setState({ approvalSetup: res.data.approvalSetup });
            this.setState({ userSetup: res.data.userSetup });
            this.setState({ password: res.data.password });
            this.setState({ repeatPassword: res.data.password });
            this.setState({ myApprovals: res.data.myApprovals });
            this.setState({ myExpenses: res.data.myExpenses });
            this.setState({ isActive: res.data.isActive });
            this.setState({ title: 'Card of ' + res.data.firstName + ' ' + res.data.lastName });

        })

        window.addEventListener("popstate", e => {
            this.props.history.go(1);
         });
    }

    changeFirstNameHandler = (event) => {
        this.setState({ firstName: event.target.value });

    }

    changeLastNameHandler = (event) => {
        this.setState({ lastName: event.target.value });
    }

    changePasswordHandler = (event) => {
        this.setState({ password: event.target.value });
    }

    changeRepeatPasswordHandler = (event) => {
        this.setState({ repeatPassword: event.target.value });
    }

    goBack(e) {
        e.preventDefault();
        const { history } = this.props;
        history.push({pathname: `/home/${this.state.id}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
```

```
        window.location.reload(true);
    }

    Save(e) {

        e.preventDefault();

        let error = false;
        this.setState({ message: '' });
        if (this.state.firstName == '') {
            error = true;
            this.setState({ message: 'First Name cannot be initial!' });
        }
        else if (this.state.lastName == '') {
            error = true;
            this.setState({ message: 'Last Name cannot be initial!' });
        }
        else if (this.state.password == '') {
            error = true;
            this.setState({ message: 'Password cannot be initial!' });
        }
        else if (this.state.repeatPassword == '') {
            error = true;
            this.setState({ message: 'Repeat password cannot be initial!' });
        }
        else if (this.state.password != this.state.repeatPassword) {
            error = true;
            this.setState({ message: 'Password and Repeat Password do not match!' });
        }
        if (error == false) {
            let user = {
                emailId: this.state.emailId,
                password: this.state.password,
                myApprovals: this.state.myApprovals,
                myExpenses: this.state.myExpenses,
                approvalSetup: this.state.approvalSetup,
                firstName: this.state.firstName,
                lastName: this.state.lastName,
                isActive: this.state.isActive,
                userSetup: this.state.userSetup
            }
            UserService.updateUser(user, this.state.emailId).then(res => {
                const { history } = this.props;
                history.push({pathname: `/home/${this.state.id}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
                window.location.reload(true);
            })

        }
```

```
    }

    render() {

        return (
            <div className='container' style={{ backgroundColor: '#ecf7f8', width: '40%' }}>
                <div className='row' style={{ backgroundColor: '#a9cdd1' }}>
                    <div style={{ marginTop: '10px', marginBottom: '10px', color: '#fafffff' }}>
                        Expenses Management App

                    </div>
                </div>
                <div className='row border-bottom border-top' style={{ backgroundColor: '#d4e6e8' }}>
                    <div className='text-center' style={{ marginTop: '5px', marginBottom: '5px', color: '#98b9bc'
}}>{this.state.title}</div>
                </div>
                <div className='row' style={{ marginTop: '15px' }}>
                    <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                        Email ID
                    </div>
                    <div className='col-3' style={{ color: '#98b9bc' }}>
                        {this.state.id}
                    </div>
                </div>
                <div className='row' style={{ marginTop: '20px', marginTop: '20px' }}>
                    <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                        First name
                    </div>
                    <div className='col-7'>
                        <input value={this.state.firstName} name='firstName' className='form-control' type='text' style={{
padding: 2 }}
                            onChange={this.changeFirstNameHandler} />
                    </div>
                </div>
                <div className='row' style={{ marginTop: '15px' }}>
                    <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                        Last Name
                    </div>
                    <div className='col-7'>
                        <input name='lastName' value={this.state.lastName} className='form-control' type='text' style={{
padding: 2 }}
                            onChange={this.changeLastNameHandler} />
                    </div>
                </div>
                <div className='row' style={{ marginTop: '15px' }}>
                    <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                        Password
                    </div>
```

```
                <div className='col-7'>
                    <input name='password' value={this.state.password} className='form-control' type='password'
                        style={{ padding: 2 }}
                        onChange={this.changePasswordHandler} />
                </div>
            </div>
            <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                    Repeat Password
                </div>
                <div className='col-7'>
                    <input name='repeatPassword' value={this.state.repeatPassword} className='form-control'
                        type='password' style={{ padding: 2 }}
                        onChange={this.changeRepeatPasswordHandler} />
                </div>
            </div>
            <div className='row' style={{ marginLeft: '10px', marginTop: '10px' }}>
                <p style={{ color: '#Deafb9', marginTop: '15px' }} >{this.state.message}</p>
            </div>
            <div>
                <button onClick={(e) => this.Save(e)}
                    className='btn btn-info' style={{ color: '#faffff', marginLeft: '70%', marginBottom: '10px' }} >Save
                </button>
                <button onClick={(e) => this.goBack(e)}
                    className='btn btn-info' style={{ color: '#faffff', marginLeft: '10px', marginBottom: '10px' }} >Back
                </button>

            </div>
        </div>
    );

  }
}

export default withRouter(UserCardComponent);
```

Στο αρχείο UserCardComponent.js, ορίζουμε την οθόνη καρτέλας χρήστη.

Code 33: UserCardComponent.js

```
import React, { Component, data, useState } from 'react';
import UserService from '../services/UserService';
import { withRouter } from 'react-router';

class UserCardComponent extends Component {

    constructor(props) {
        super(props)
        if (this.props.location.state == undefined ) {
            const { history } = this.props;
            history.push('/signin');
            window.location.reload(true);
          }
        this.state = {
            id: this.props.match.params.id,
            user_id: this.props.match.params.emailId,
            authenticated: this.props.location.state.authenticated,
            firstName: '',
            lastName: '',
            emailId: '',
            // approvalSetup: 'N',
            userSetup: 'N',
            password: '',
            repeatPassword: '',
            approver: '',
            isApprover: '',
            isActive: 'N',
            myApprovals: 'N',
            myExpenses: 'N',
            title: 'Create New User',
            isCheckedAppSetup: false,
            isCheckedUserSetup: false,
            isCheckedActive: false,
            message: '',
            existedEmail: '',
            existedApprover: '',
            activeApprover: ''
        }

        this.changeFirstNameHandler = this.changeFirstNameHandler.bind(this);
        this.changeLastNameHandler = this.changeLastNameHandler.bind(this);
        this.changeEmailIdHandler = this.changeEmailIdHandler.bind(this);
        // this.changeApprovalSetupHandler = this.changeApprovalSetupHandler.bind(this);
        this.changeUserSetupHandler = this.changeUserSetupHandler.bind(this);
        this.changePasswordHandler = this.changePasswordHandler.bind(this);
        this.changeRepeatPasswordHandler = this.changeRepeatPasswordHandler.bind(this);
```

```
        this.changeActiveHandler = this.changeActiveHandler.bind(this);
        this.changeApproverHandler = this.changeApproverHandler.bind(this);
        this.goBack = this.goBack.bind(this);
        this.Save = this.Save.bind(this);



    }

    componentDidMount() {

        this.setState({ firstName: '' });
        this.setState({ lastName: '' });
        this.setState({ emailId: '' });
        this.setState({ password: '' });
        this.setState({ repeatPassword: '' });
        this.setState({ approver: '' });
        this.setState({ isApprover: '' });
        // this.setState({ approvalSetup: 'N' });
        this.setState({ userSetup: 'N' });
        this.setState({ myApprovals: 'N' });
        this.setState({ myExpenses: 'N' });
        this.setState({ isActive: 'N' });
        this.setState({ title: 'Create New User' });
        this.setState({ isCheckedUserSetup: false });
        this.setState({ isCheckedAppSetup: false });
        if (this.state.user_id != 'new') {
            UserService.getUser(this.state.user_id).then((res) => {

                this.setState({ firstName: res.data.firstName });
                this.setState({ lastName: res.data.lastName });
                this.setState({ emailId: res.data.emailId });
                // this.setState({ approvalSetup: res.data.approvalSetup });
                this.setState({ userSetup: res.data.userSetup });
                this.setState({ password: res.data.password });
                this.setState({ repeatPassword: res.data.password });
                this.setState({ myApprovals: res.data.myApprovals });
                this.setState({ myExpenses: res.data.myExpenses });
                this.setState({ isActive: res.data.isActive });
                this.setState({ approver: res.data.approver });
                this.setState({ existedApprover: res.data.approver });
                if (res.data.approver != null && res.data.approver != ''){
                    UserService.getUser(res.data.approver).then((res) => {
                        this.setState({ activeApprover: res.data.isActive });
                    })
                }
                this.setState({ title: 'Card of ' + res.data.firstName + ' ' + res.data.lastName });
                // if (res.data.approvalSetup == 'Y') {
```

```
            //     this.setState({ isCheckedAppSetup: true });
            // }
            if (res.data.userSetup == 'Y') {
                this.setState({ isCheckedUserSetup: true });
            }
            if (res.data.isActive == 'Y') {
                this.setState({ isCheckedActive: true });
            }

        })
    }

    window.addEventListener("popstate", e => {
        this.props.history.go(1);
    });
}

changeFirstNameHandler = (event) => {
    this.setState({ firstName: event.target.value });

}

changeLastNameHandler = (event) => {
    this.setState({ lastName: event.target.value });
}

changeEmailIdHandler = (event) => {
    this.setState({ emailId: event.target.value });
    this.setState({ existedEmail: '' });
    UserService.getUser(event.target.value).then((res) => {
        this.setState({ existedEmail: res.data.emailId });
    })
}

changeApproverHandler = (event) => {
    this.setState({ approver: event.target.value });
    this.setState({ existedApprover: '' });
    this.setState({ activeApprover: '' });
    this.setState({ myExpenses: 'N' });
    UserService.getUser(event.target.value).then((res) => {
        this.setState({ existedApprover: res.data.emailId });
        this.setState({ activeApprover: res.data.isActive });

    })
    if (event.target.value != '') {
        this.setState({ myExpenses: 'Y' });
    }
}
```

```
changePasswordHandler = (event) => {
    this.setState({ password: event.target.value });
}

changeRepeatPasswordHandler = (event) => {
    this.setState({ repeatPassword: event.target.value });
}
// changeApprovalSetupHandler = (event) => {
//     this.setState({ approvalSetup: 'N' });
//     if (this.state.isCheckedAppSetup == false) {
//         this.setState({ approvalSetup: 'Y' });
//     }
//     this.setState({ isCheckedAppSetup: !this.state.isCheckedAppSetup });
// }
changeUserSetupHandler = (event) => {
    this.setState({ userSetup: 'N' });
    if (this.state.isCheckedUserSetup == false) {
        this.setState({ userSetup: 'Y' });
    }
    this.setState({ isCheckedUserSetup: !this.state.isCheckedUserSetup });
}

changeActiveHandler = (event) => {
    this.setState({ isActive: 'N' });
    this.setState({ isApprover: '' });
    if (this.state.isCheckedActive == false) {
        this.setState({ isActive: 'Y' })
    }

    this.setState({ isCheckedActive: !this.state.isCheckedActive });

    if (this.state.user_id != 'new') {
        UserService.getUsers().then((res) => {
            res.data.map(line => {
                if (line.approver == this.state.user_id) {
                    this.setState({ isApprover: true });
                    console.log(line);
                }
            })
        })
    }




}
```

```
goBack = (e) => {
    const { history } = this.props;
    history.push({pathname: `/home/${this.state.id}/${'userSetup'}`, state: {authenticated: this.state.authenticated}});
    window.location.reload(true);
}

Save(e) {

    e.preventDefault();

    let error = false;
    this.setState({ message: '' });
    if (this.state.firstName == '') {
        error = true;
        this.setState({ message: 'First Name cannot be initial!' });
    }
    else if (this.state.lastName == '') {
        error = true;
        this.setState({ message: 'Last Name cannot be initial!' });
    }
    else if (this.state.emailId == '') {
        error = true;
        this.setState({ message: 'Email ID cannot be initial!' });
    }
    else if (this.state.password == '') {
        error = true;
        this.setState({ message: 'Password cannot be initial!' });
    }
    else if (this.state.repeatPassword == '') {
        error = true;
        this.setState({ message: 'Repeat password cannot be initial!' });
    }
    else if (this.state.password != this.state.repeatPassword) {
        error = true;
        this.setState({ message: 'Password and Repeat Password do not match!' });
    }
    else if (this.state.existedEmail == this.state.emailId) {
        error = true;
        this.setState({ message: 'Email ID already exists!' });
    }
    else if (this.state.existedApprover != this.state.approver && this.state.approver != null && this.state.approver != '') {
       console.log(this.state.existedApprover);
       console.log(this.state.approver);

        error = true;
        this.setState({ message: 'Approver does not exist!' });
    }
    else if (this.state.activeApprover != 'Y' && this.state.approver != null && this.state.approver != '') {
```

```
            error = true;
            this.setState({ message: 'Approver is not active!' });
        }
        else if (this.state.isActive == 'N' && this.state.isApprover == true) {
            error = true;
            this.setState({ message: 'User is approver, cannot be inactive!' });
        }
        if (error == false) {
            let user = {
                emailId: this.state.emailId,
                password: this.state.password,
                myApprovals: this.state.myApprovals,
                myExpenses: this.state.myExpenses,
                // approvalSetup: this.state.approvalSetup,
                firstName: this.state.firstName,
                lastName: this.state.lastName,
                isActive: this.state.isActive,
                userSetup: this.state.userSetup,
                approver: this.state.approver
            }

            // if (this.state.approvalSetup = 'Y' && this.state.approver != '') {
            //     UserService.getUser(this.state.approver).then((res) => {
            //         let approverUser = res.data;
            //         approverUser.myApprovals = 'Y';
            //         UserService.updateUser(approverUser, approverUser.emailId).then(res => { })
            //     })

            // }

            if (this.state.user_id == 'new') {
                UserService.createUser(user).then(res => {
                    if (this.state.userSetup == 'Y') {
                        const { history } = this.props;
                        history.push({pathname: `/home/${this.state.id}/${'userSetup'}`, state: {authenticated:
this.state.authenticated}});
                        window.location.reload(true);
                    }
                    if (this.state.userSetup == 'N') {
                        const { history } = this.props;
                        history.push({pathname: `/home/${this.state.id}/${'inbox'}`, state: {authenticated:
this.state.authenticated}});
                        window.location.reload(true);
                    }

                })
            }
            else {
```

```
            UserService.updateUser(user, this.state.emailId).then(res => {
                let path_id = this.state.id;
                if (this.state.id == this.state.user_id) {
                    path_id = this.state.emailId;
                }
                if (this.state.userSetup == 'Y') {
                    const { history } = this.props;
                    history.push({pathname: `/home/${path_id}/${'userSetup'}`, state: {authenticated:
this.state.authenticated}});
                    window.location.reload(true);
                }
                if (this.state.userSetup == 'N') {
                    const { history } = this.props;
                    history.push({pathname: `/home/${path_id}/${'inbox'}`, state: {authenticated: this.state.authenticated}});
                    window.location.reload(true);
                }
            })

        }

    }
  }

  render() {

    return (
      <div className='container' style={{ backgroundColor: '#ecf7f8', width: '40%' }}>
        <div className='row' style={{ backgroundColor: '#a9cdd1' }}>
          <div style={{ marginTop: '10px', marginBottom: '10px', color: '#faffff' }}>
            Expenses Management App
          </div>
        </div>
        <div className='row border-bottom border-top' style={{ backgroundColor: '#d4e6e8' }}>
          <div className='text-center' style={{ marginTop: '5px', marginBottom: '5px', color: '#98b9bc'
}}>{this.state.title}</div>
        </div>
        <div className='row' style={{ marginTop: '15px' }}>
          <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
            Email ID
          </div>
          {this.state.user_id == 'new' ?
            <div className='col-7'>
              <input value={this.state.emailId} name='emailId' className='form-control' type='email' style={{
padding: 2 }}
                onChange={this.changeEmailIdHandler} />
            </div>
            :
            <div className='col-3' style={{ color: '#98b9bc' }}>
```

```
                    {this.state.user_id}
                  </div>
                }
              </div>
              <div className='row' style={{ marginTop: '20px', marginTop: '20px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                  First name
                </div>
                <div className='col-7'>
                  <input value={this.state.firstName} name='firstName' className='form-control' type='text' style={{
padding: 2 }}
                    onChange={this.changeFirstNameHandler} />
                </div>
              </div>
              <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                  Last Name
                </div>
                <div className='col-7'>
                  <input name='lastName' value={this.state.lastName} className='form-control' type='text' style={{
padding: 2 }}
                    onChange={this.changeLastNameHandler} />
                </div>
              </div>
              <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                  Password
                </div>
                <div className='col-7'>
                  <input name='password' value={this.state.password} className='form-control' type='password'
                    style={{ padding: 2 }}
                    onChange={this.changePasswordHandler} />
                </div>
              </div>
              <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                  Repeat Password
                </div>
                <div className='col-7'>
                  <input name='repeatPassword' value={this.state.repeatPassword} className='form-control'
                    type='password' style={{ padding: 2 }}
                    onChange={this.changeRepeatPasswordHandler} />
                </div>
              </div>
              <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                  Approver
                </div>
```

```
                <div className='col-7'>
                    <input name='approver' value={this.state.approver} className='form-control' type='text' style={{
padding: 2 }}
                        onChange={this.changeApproverHandler} />
                </div>
            </div>
            {/* <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                    Approval Setup Tab
                </div>
                <div className='col'>
                    <input type="checkbox" checked={this.state.isCheckedAppSetup}
onChange={this.changeApprovalSetupHandler} />
                </div>
            </div> */}
            <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                    User Setup Tab
                </div>
                <div className='col'>
                    <input type="checkbox" checked={this.state.isCheckedUserSetup}
onChange={this.changeUserSetupHandler} />
                </div>
            </div>
            <div className='row' style={{ marginTop: '15px' }}>
                <div className='col-3' style={{ marginLeft: '30px', color: '#98b9bc' }}>
                    Active User
                </div>
                <div className='col'>
                    <input type="checkbox" checked={this.state.isCheckedActive} onChange={this.changeActiveHandler} />
                </div>
            </div>
            <div className='row' style={{ marginLeft: '10px', marginTop: '10px' }}>
                <p style={{ color: '#Deafb9', marginTop: '15px' }} >{this.state.message}</p>
            </div>

            <button onClick={(e) => this.Save(e)}
                className='btn btn-info' style={{ color: '#faffff', marginLeft: '75%', marginBottom: '10px' }} >Save</button>
            <button onClick={(e) => this.goBack(e)} className='btn btn-info' style={{ color: '#faffff', marginLeft: '10px',
marginBottom: '10px' }} >Back</button>

        </div>
    );

  }
}

export default withRouter(UserCardComponent);
```

Στο αρχείο App.css κάνουμε κάποιες προσθήκες με σκοπό να εξυπηρετηθούν ανάγκες τις εφαρμογής.

Code 34: App.css

```
/*SignInAdditions - START+*/
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&display=swap');

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif
}

.container {
  margin: 50px auto;
  background-color: #faffff;
}

.body {
  position: relative;
  width: auto;
  height: auto;
  margin: auto;
  border: 1px solid #dddd;
  border-radius: 18px;
  overflow: hidden;
  box-shadow: #afd5de
}

.box-1 img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  margin-right: 45px;
}

.box-2 {
  padding: 20px;
  margin-left: 50px;
}

.box-1,
.box-2 {
  width: 50%
}
```

```
.h-1 {
  font-size: 24px;
  font-weight: 700
}

.text-muted {
  font-size: 14px
}

.container .box {
  width: 100px;
  height: 100px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  border: 2px solid transparent;
  text-decoration: none;
  color: #afd5de
}

.box:active,
.box:visited {
  border: 2px solid #afd5de
}

.box:hover {
  border: 2px solid #afd5de
}

.btn.btn-primary {
  background-color: transparent;
  color: #ee82ee;
  border: 0px;
  padding: 0;
  font-size: 14px
}

.btn.btn-primary .fas.fa-chevron-right {
  font-size: 12px
}

.footer .p-color {
  color: #ee82ee
}

.footer.text-muted {
  font-size: 10px
```

```
}

.fas.fa-times {
  position: absolute;
  top: 20px;
  right: 20px;
  height: 20px;
  width: 20px;
  background-color: #faffff;
  font-size: 18px;
  display: none;
  /*SignInAdditions*/
  align-items: center;
  justify-content: center
}

.fas.fa-times:hover {
  color: #ff0000
}

@media (max-width:767px) {
  body {
    padding: 10px
  }

  .body {
    width: 100%;
    height: 100%
  }

  .box-1 {
    width: 100%
  }

  .box-2 {
    width: 100%;
    height: 450px
  }
}

@media (max-width:767px) {
  body {
    padding: 10px
  }

  .body {
    width: 100%;
    height: 100%
```

```
  }

}

/*SignInAdditions - END+*/

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }

  to {
    transform: rotate(360deg);
  }
}

/*SignInAdditions - START+*/
.btn.btn-info {
```

```
  background-color: #afd5de !important;
 }

 .btn.btn-trans{

  background-color: none;
  box-shadow: none;
  outline: none;
  border-color: none;
 }

 .md-form label.active {
  color: #afd5de !important;
 }

 .md-form input:focus {
  border-bottom: 1px solid #afd5de !important;
  box-shadow: 0 1px 0 0 #afd5de !important;
 }

 /*SignInAdditions - END+*/

 .navbar-custom {
  background-color: #a9cdd1 !important;
  color: #faffff !important;
  border: none;
  box-shadow: none;
 }

 .btn.btn-info {
  border: none !important;
  box-shadow: none !important;
 }
```

Στο αρχείο App.js ορίζουμε τις διαδρομές (paths) της εφαρμογής.

Code 35: App.js

```
import './App.css';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import SignInComponent from './components/SignInComponent';
import HomeComponent from './components/HomeComponent';
import UserCardComponent from './components/UserCardComponent';
import React from 'react';
import MySettingsComponent from './components/MySettingsComponent';
import ExpenseCardComponent from './components/ExpenseCardComponent';

function App() {

  return (
    <Router>
      <Switch>
        <Route path='/signin' exact component={SignInComponent}/>
        <Route path='/home/:id/:tab' component={HomeComponent}/>
        <Route path='/user/:id/:emailId' component={UserCardComponent}  />
        <Route path='/mySettings/:id' component={MySettingsComponent} />
        <Route path='/expense/:pageId/:emailId/:id' component={ExpenseCardComponent} />

      </Switch>
    </Router>
  );

}

export default App;
```

Στο αρχείο upload.php ορίζουμε τιμές για τις μεταβλητές Access-Control-Allow-Origin.

Code 36: upload.php

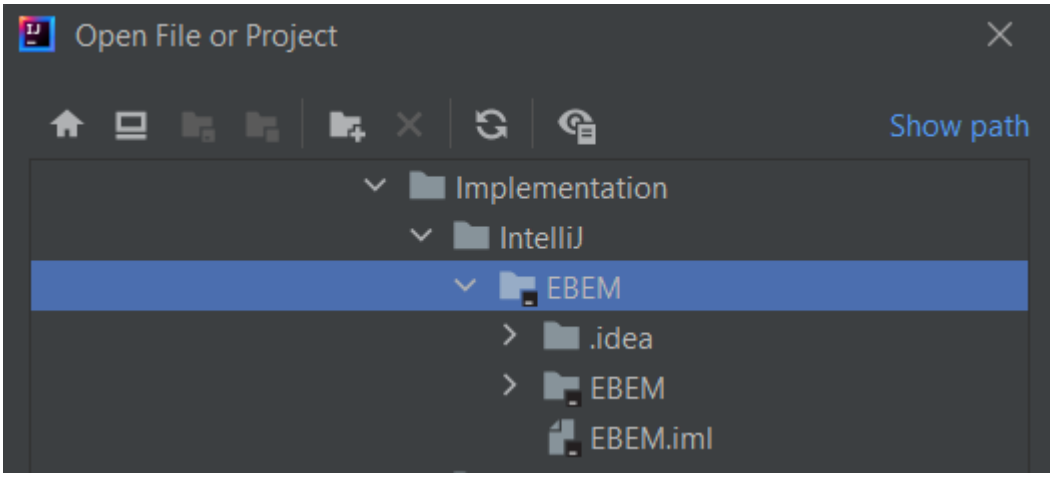
```
<? php

header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
header("Access-Control-Allow-Credentials: true");
header("Access-Control-Max-Age: 86400"); ?>
```

4 Τρέξιμο Εφαρμογής

Προκειμένου να τρέξουμε την εφαρμογή χρειαζόμαστε τα παρακάτω εργαλεία:

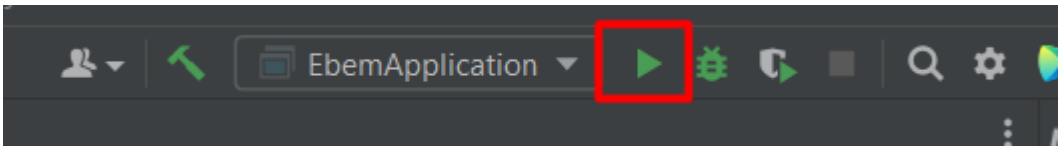
- Την πλατφόρμα IntelliJ.
- Την πλατφόρμα Visual Studio Code..
- Πρόγραμμα περιήγησης

Στο intelliJ, επιλέγουμε open και ανοίγουμε τον φάκελο Implementation -> IntelliJ -> EBEM.



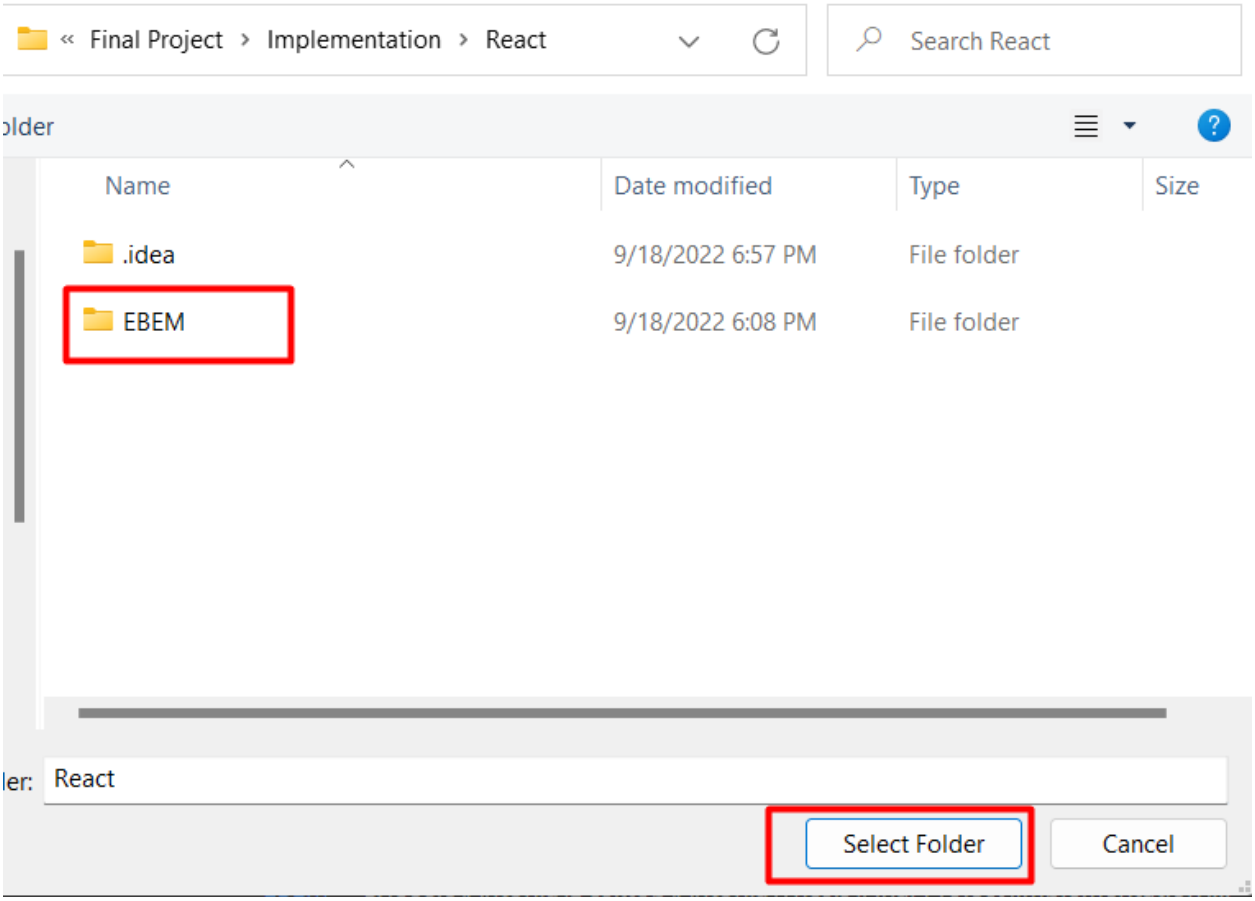
Εικόνα 17: IntelliJ άνοιγμα φακέλου

Επιλέγουμε Run Application.



Εικόνα 18: IntelliJ Run Application

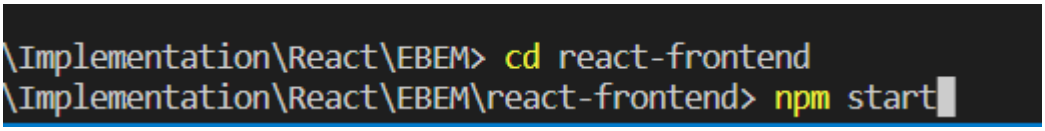
Στο Visual Studio Code επιλέγουμε τον φάκελο Implementation->React->EBEM.



Εικόνα 19: React Άνοιγμα Φακέλου

Επιλέγουμε Terminal -> New Terminal και εκτελούμε τις παρακάτω εντολές:

- cd react-frontend
- npm start

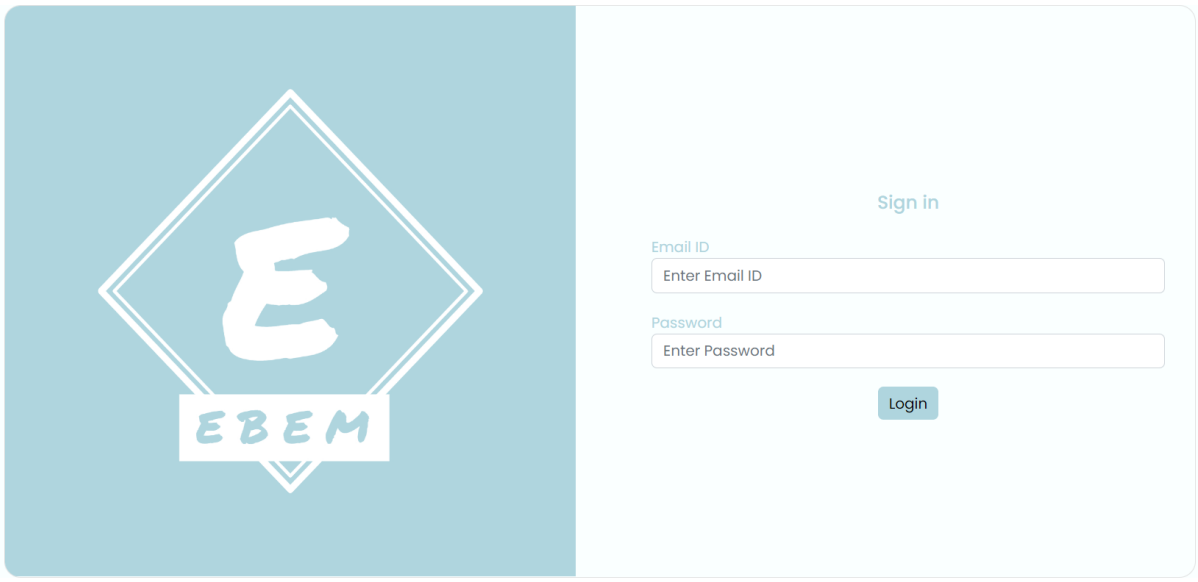


Εικόνα 20: React Terminal

Στον browser βάζουμε την διεύθυνση http://localhost:3000/signin για να εισέλθουμε στην εφαρμογή.

5 Παρουσίαση Εφαρμογής

Για να εισέλθουμε στην εφαρμογή εισάγουμε το URL http://localhost:3000/signin. Η αρχική οθόνη θα εμφανιστεί:

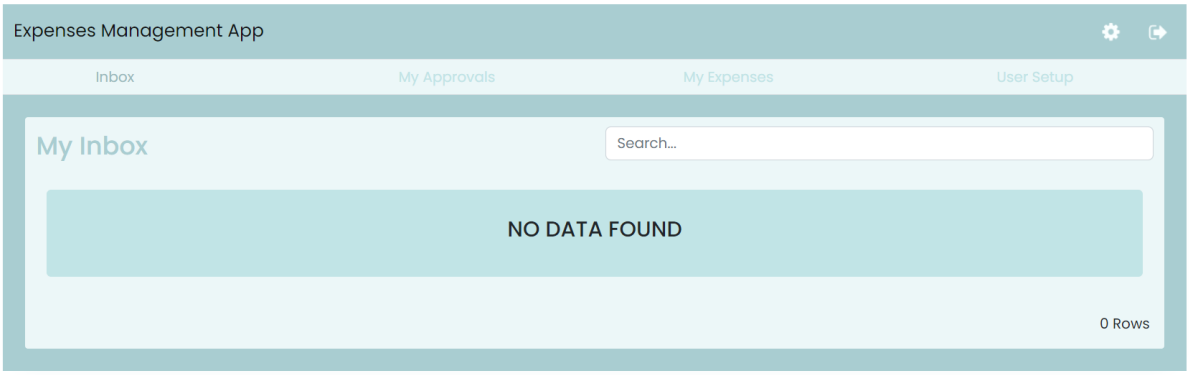


Εικόνα 21: Παρουσίαση Εφαρμογής Είσοδος

Εισάγουμε τα στοιχεία χρήστη μας. Σε περίπτωση που δεν είναι έγκυρα θα εμφανιστεί το παρακάτω μήνυμα:

Εικόνα 22: Wrong Email or Password

Με την επιτυχημένη είσοδο στην εφαρμογή θα οδηγηθούμε στην αρχική οθόνη:



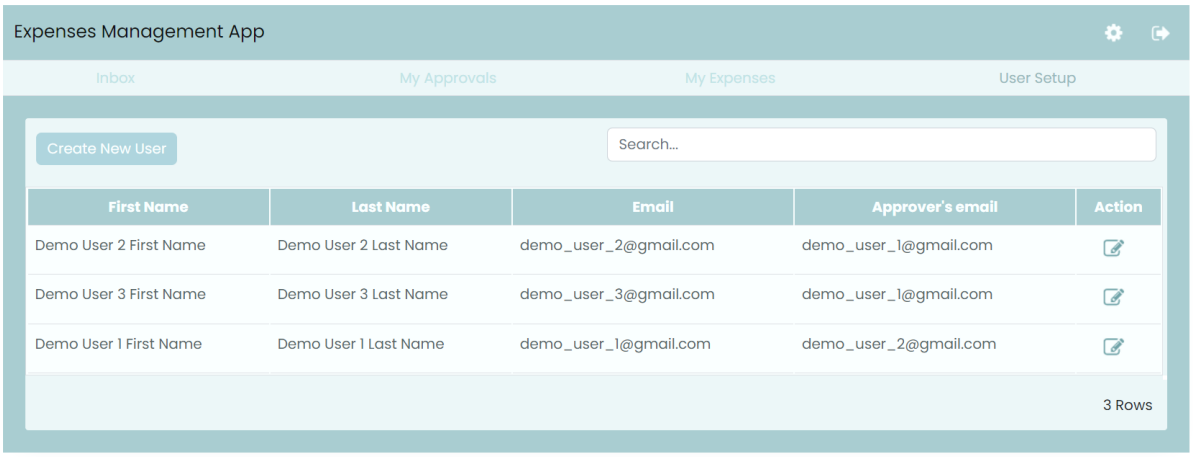
Εικόνα 23: Παρουσίαση Εφαρμογής Αρχική Οθόνη

Η αρχική οθόνη αποτελείται από τις παρακάτω υποκατηγορίες:

- Inbox: αποτελείται από τα μηνύματα που έχει ο χρήστης. Αν δεν έχει κανένα μήνυμα εμφανίζεται το λεκτικό NO DATA FOUND.
- My Approvals: εμφανίζεται μόνο αν ο χρήστης είναι approval σε τουλάχιστον ένα άλλον χρήστη του συστήματος. Σε αυτό το σημείο εμφανίζονται όλα τα σχετικά έξοδα που θα πρέπει να εγκρίνει ή έχει εγκρίνει ο χρήστης.

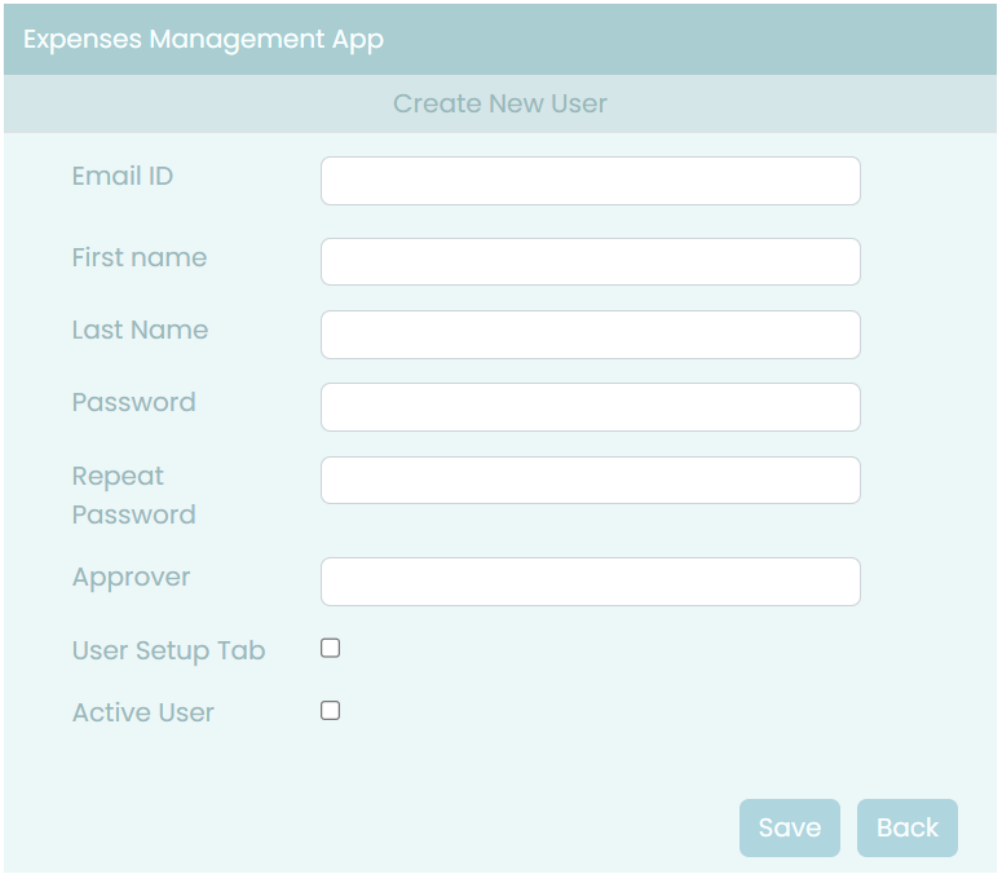
- My Expenses: εμφανίζεται μόνο αν ο χρήστης έχει approval στην παραμετροποίηση χρήστη. Σε αυτό το σημείο είναι όλα τα έξοδα του χρήστη.
- User Setup: εμφανίζεται μόνο αν είναι επιλεγμένο το αντίστοιχο πεδίο στην παραμετροποίηση χρήστη. Φαίνονται όλοι οι χρήστες του συστήματος.

Στην οθόνη User Setup, εμφανίζονται όλοι οι χρήστες του συστήματος.



Εικόνα 24: Παρουσίαση Εφαρμογής User Setup

Για να δημιουργήσουμε νέο χρήστη επιλέγουμε το Create New User. Θα ανοίξει η καρτέλα χρήστη.



Εικόνα 25: Παρουσίαση εφαρμογής New User

Η καρτέλα χρήστη αποτελείται από τα παρακάτω πεδία:

- Email ID: εισάγουμε το email id του χρήστη. Αυτό το πεδίο από την στιγμή που θα γίνει save ο χρήστης δεν μπορεί να αλλάξει. Υποχρεωτικό πεδίο.
- First Name: εισάγουμε το όνομα του χρήστη. Υποχρεωτικό πεδίο.
- Last Name: εισάγουμε το επίθετο του χρήστη. Υποχρεωτικό πεδίο.
- Password: ο κωδικός εισόδου του χρήστη. Υποχρεωτικό πεδίο.
- Repeat Password: επιβεβαίωση κωδικού χρήστη. Υποχρεωτικό πεδίο.
- Approver: εισάγουμε τον χρήστη που θα εγκρίνει τα έξοδα του νέου χρήστη, αν υπάρχει.
- User Setup Tab: το επιλέγουμε αν ο χρήστης θα έχει πρόσβαση στην υποκατηγορία User Setup της αρχικής οθόνης.

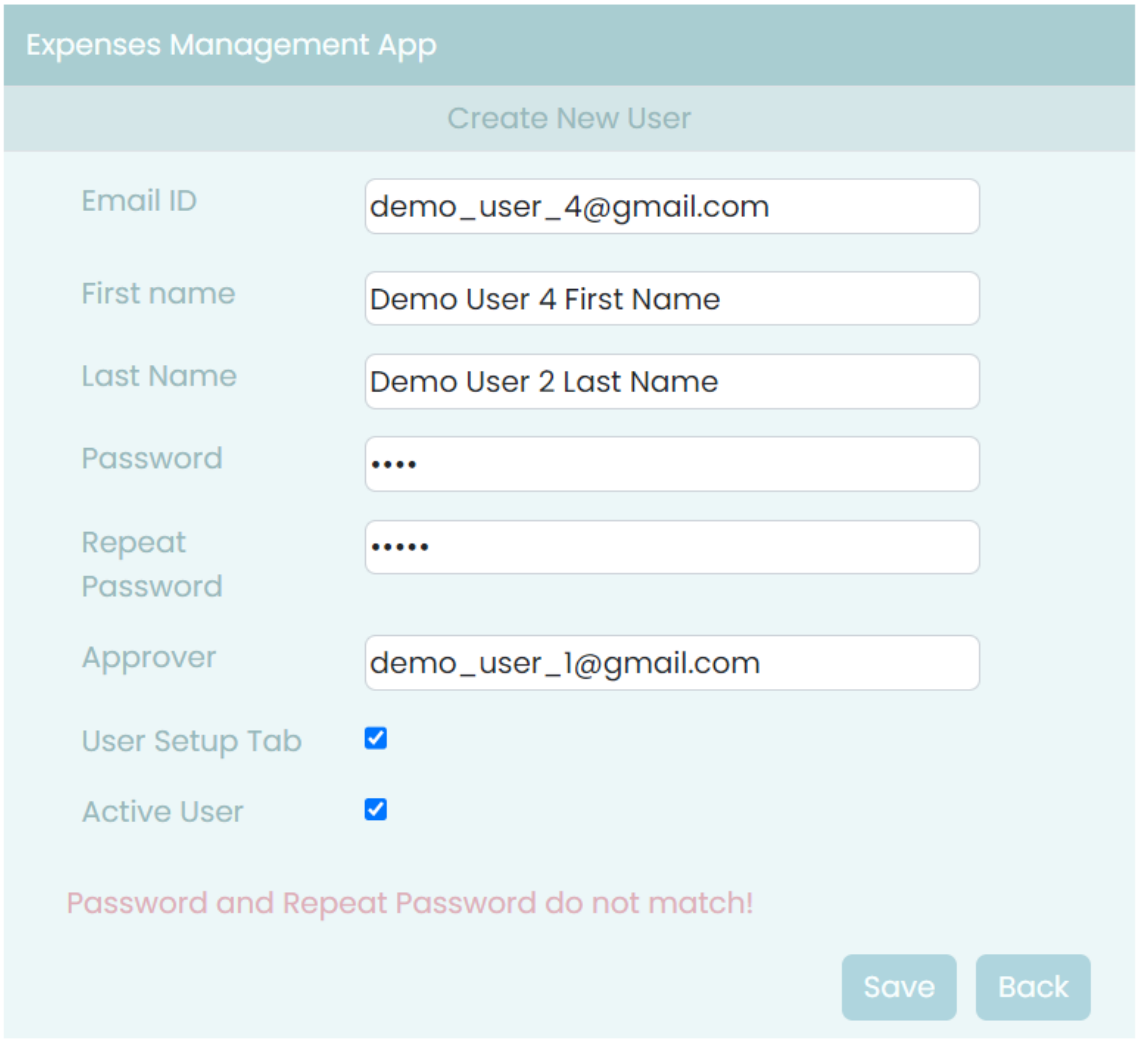
- Active User: ορίζουμε αν ο χρήστης είναι ενεργός η όχι. Σε περίπτωση που απενεργοποιήσουμε έναν χρήστη, το σύστημα ελέγχει αν είναι approver σε άλλο ενεργό χρήστη. Αν είναι ο χρήστης δεν μπορεί να απενεργοποιηθεί.

Επιλέγοντας Save, γίνονται οι έλεγχοι και αποθηκεύεται ο χρήστης. Αν επιλέξουμε Back ο χρήστης δεν αποθηκεύεται.

Οι έλεγχοι του συστήματος είναι οι παρακάτω:

- Έλεγχος ότι έχει συμπληρωθεί το First Name. Αν δεν έχει συμπληρωθεί βγαίνει μήνυμα: “First Name cannot be initial!”.
- Έλεγχος ότι έχει συμπληρωθεί το Last Name. Αν δεν έχει συμπληρωθεί βγαίνει μήνυμα: “Last Name cannot be initial!”.
- Έλεγχος ότι έχει συμπληρωθεί το Email ID. Αν δεν έχει συμπληρωθεί βγαίνει μήνυμα: “Email ID cannot be initial!”.
- Έλεγχος ότι έχει συμπληρωθεί το Password. Αν δεν έχει συμπληρωθεί βγαίνει μήνυμα: “Password cannot be initial!”.
- Έλεγχος ότι έχει συμπληρωθεί το Repeat Password. Αν δεν έχει συμπληρωθεί βγαίνει μήνυμα: “Repeat password cannot be initial!”.
- Έλεγχος ότι το Password και το Repeat Password είναι ίδια. Αν δεν είναι βγαίνει μήνυμα: “ Password and Repeat Password do not match!”.
- Έλεγχος ότι το Email ID δεν υπάρχει ήδη. Αν υπάρχει βγαίνει μήνυμα: “ Email ID already exists!".”
- Στην περίπτωση που έχει συμπληρωθεί Approver, ελέγχουμε αν υπάρχει ο χρήστης. Αν δεν υπάρχει βγαίνει μήνυμα: “ Approver does not exist!”.
- Στην περίπτωση που έχει συμπληρωθεί Approver, ελέγχουμε ότι ο χρήστης είναι ενεργός. Αν δεν είναι βγαίνει μήνυμα: “Approver is not active!”.

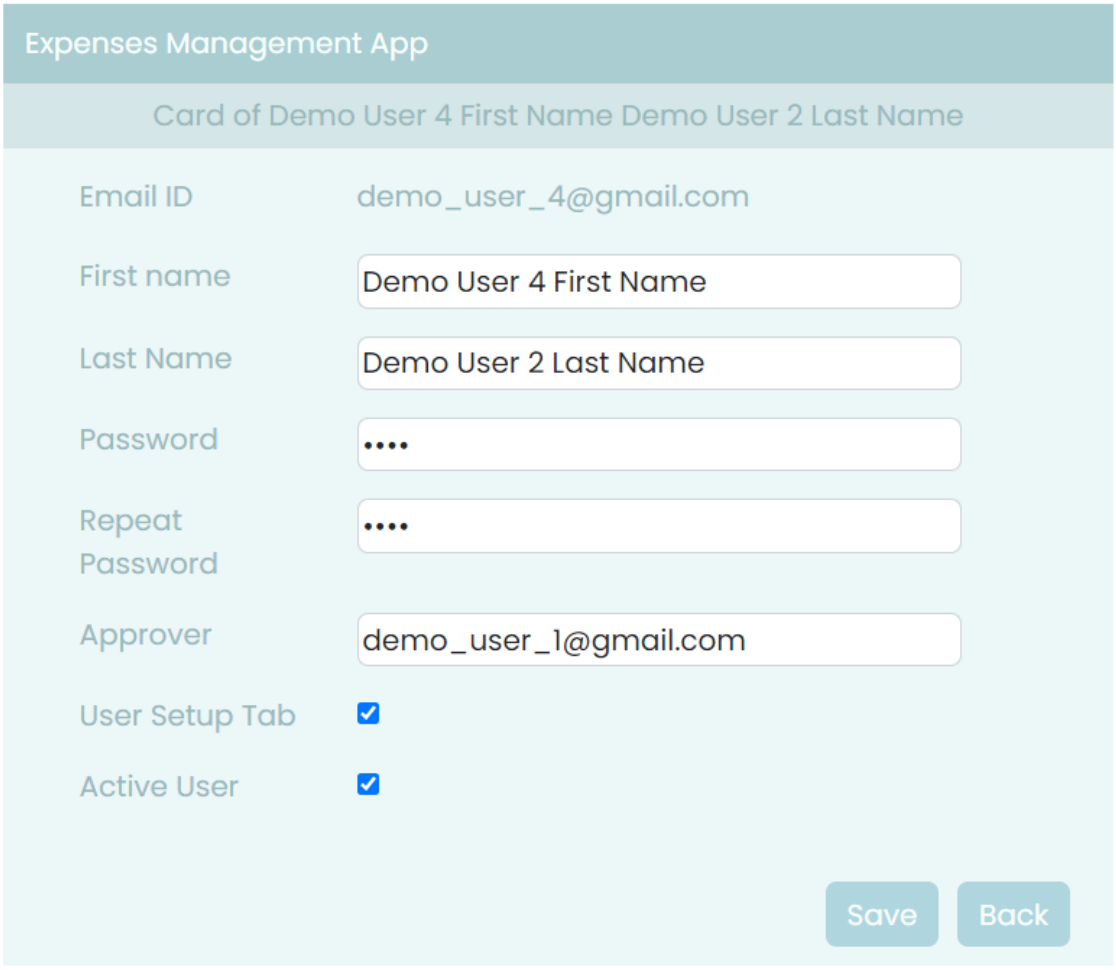
Το σφάλμα εμφανίζεται στο παρακάτω σημείο της οθόνης:



Εικόνα 26: Παρουσίαση Εφαρμογής Σφάλμα User Card

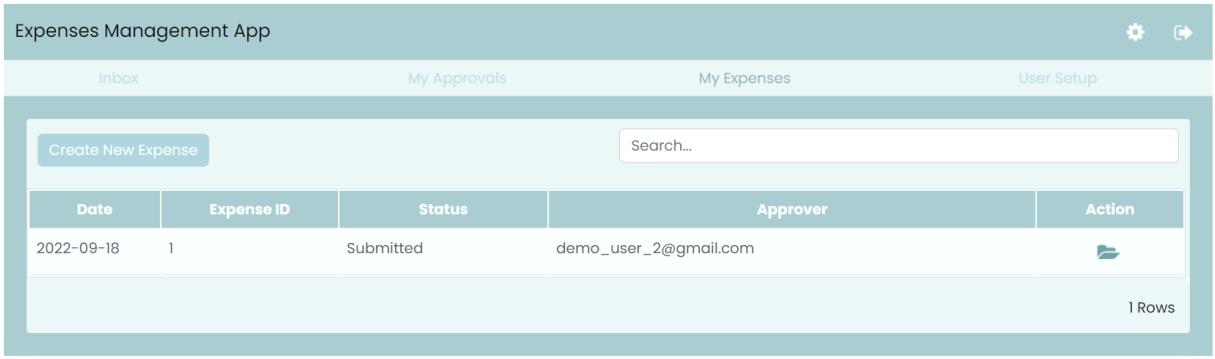
Αποθηκεύοντας τον χρήστη επιστρέφουμε στην αρχική οθόνη.

Για να ανοίξουμε την καρτέλα ενός χρήστη επιλέγουμε το εικονίδιο .



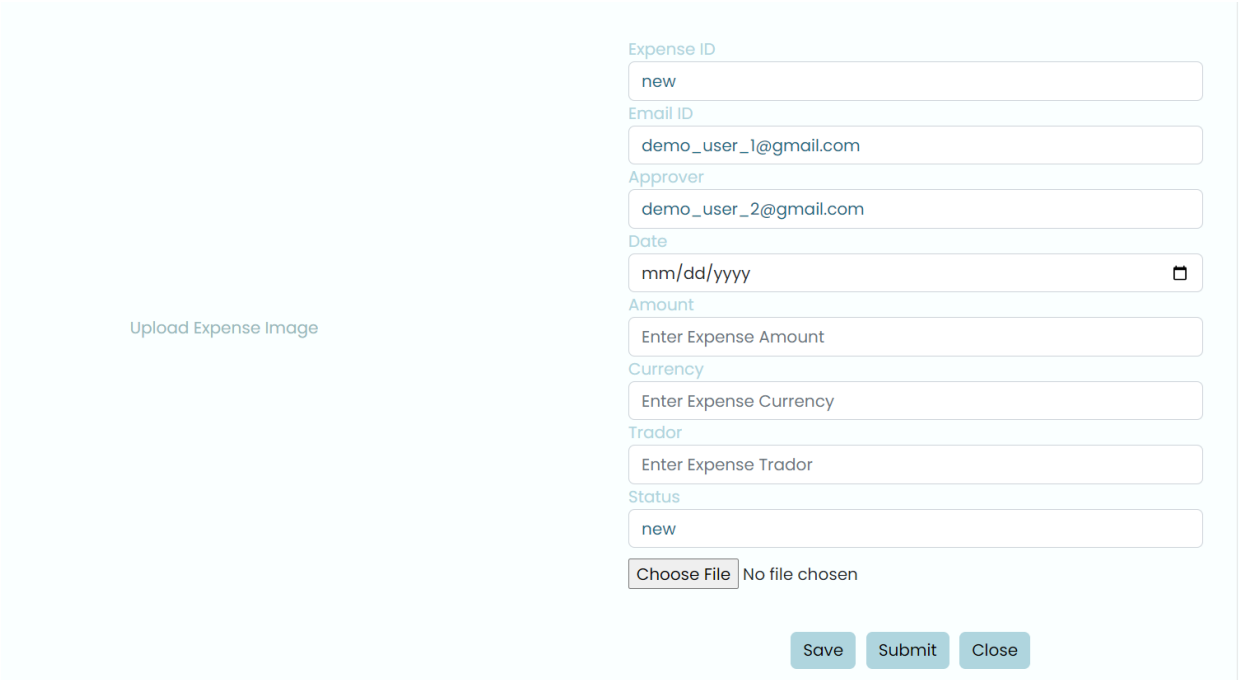
Εικόνα 27: Παρουσίαση Εφαρμογής User Card

Στην υποκατηγορία My Expenses, εμφανίζονται τα έξοδα μας και μπορούμε να καταχωρήσουμε και νέα έξοδα.



Εικόνα 28: Παρουσίαση Εφαρμογής My Expenses

Για να δημιουργήσουμε νέο έξοδο επιλέγουμε Create New Expense. Θα ανοίξει η καρτέλα εξόδου.



Εικόνα 29: Παρουσίαση Εφαρμογής Δημιουργία Εξόδου

Η καρτέλα εξόδου αποτελείται από τα παρακάτω πεδία:

- Expense ID: μοναδικός αριθμός ταυτοποίησης εξόδου. Συμπληρώνεται με την αποθήκευση του εξόδου από το σύστημα. Δεν συμπληρώνεται από τον χρήστη.
- Email ID: το email του χρήστη. Δεν συμπληρώνεται από τον χρήστη.

- Approver: ο approver που έχει ορισθεί στην παραμετροποίηση χρήστη. Δεν συμπληρώνεται από τον χρήστη.
- Date: Ημερομηνία εξόδου. Υποχρεωτικό πεδίο.
- Amount: Το ποσό του εξόδου. Υποχρεωτικό πεδίο.
- Currency: Το νόμισμα. Υποχρεωτικό πεδίο.
- Trador: Ο προμηθευτής του εξόδου. Υποχρεωτικό πεδίο.
- Status: Η κατάσταση του εξόδου. Συμπληρώνεται αυτόματα από το σύστημα. Το status του εξόδου μπορεί να είναι:
    - New: έξοδο που δεν έχει αποθηκευτεί ακόμα.
    - Submitted: καταχωρημένο έξοδο.
    - Withdrawn: έξοδο που έχει αποσυρθεί από την καταχώρηση.
    - Approved: έξοδο που έχει αποδεχτεί ο εγκριτής.
    - Disapproved: έξοδο που δεν έχει αποδεχτεί ο εγκριτής.
- Choose File: επιλέγουμε την εικόνα του εξόδου. Υποχρεωτικό πεδίο.

Επιλέγοντας Save, το έξοδο αποθηκεύεται χωρίς να σταλεί στον εγκριτή.

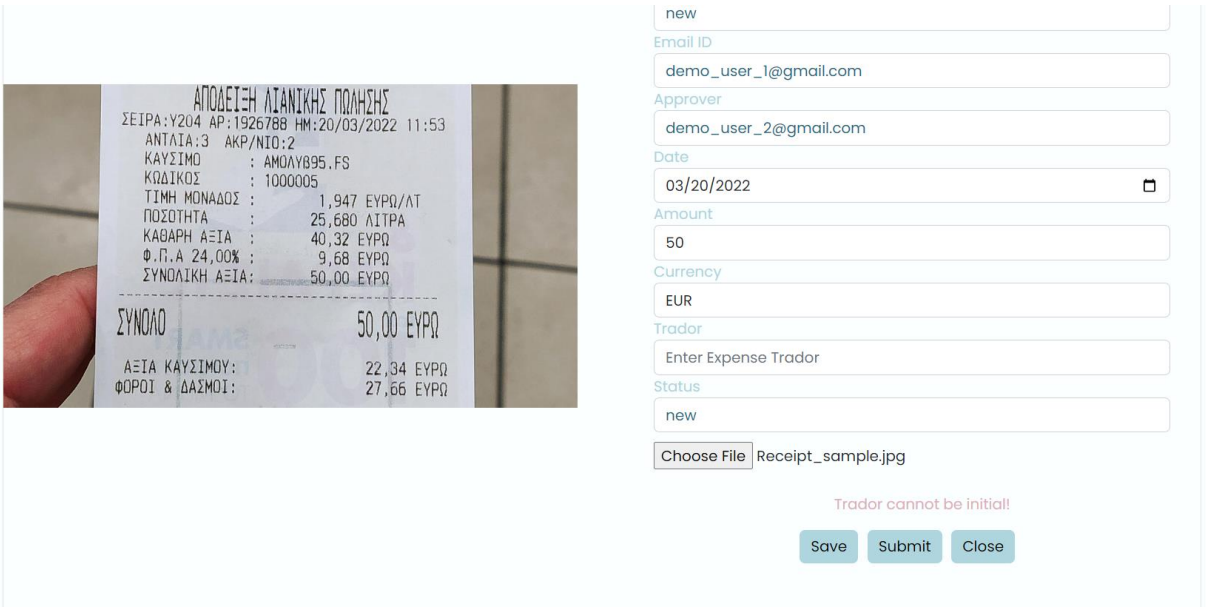
Επιλέγοντας Submit, το έξοδο αποθηκεύεται και στέλνεται στον εγκριτή για έγκριση. Το submit δημιουργεί μήνυμα στο inbox του εγκριτή.

Επιλέγοντας Close, το έξοδο δεν αποθηκεύεται.

Κατά την αποθήκευση γίνονται οι παρακάτω έλεγχοι:

- Η ημερομηνία έχει συμπληρωθεί. Αν δεν έχει συμπληρωθεί βγαίνει σφάλμα: “Date cannot be initial!”.
- Το νόμισμα έχει συμπληρωθεί. Αν δεν έχει συμπληρωθεί βγαίνει σφάλμα: “Currency cannot be initial!".
- Ο προμηθευτής έχει συμπληρωθεί. Αν δεν έχει συμπληρωθεί βγαίνει σφάλμα: “Trador cannot be initial!”.
- Το ποσό έχει συμπληρωθεί. Αν δεν έχει συμπληρωθεί βγαίνει σφάλμα: “Amount cannot be initial!".
- Έχει εισαχθεί εικόνα. Αν δεν έχει εισαχθεί βγαίνει σφάλμα: “Image cannot be initial!”.

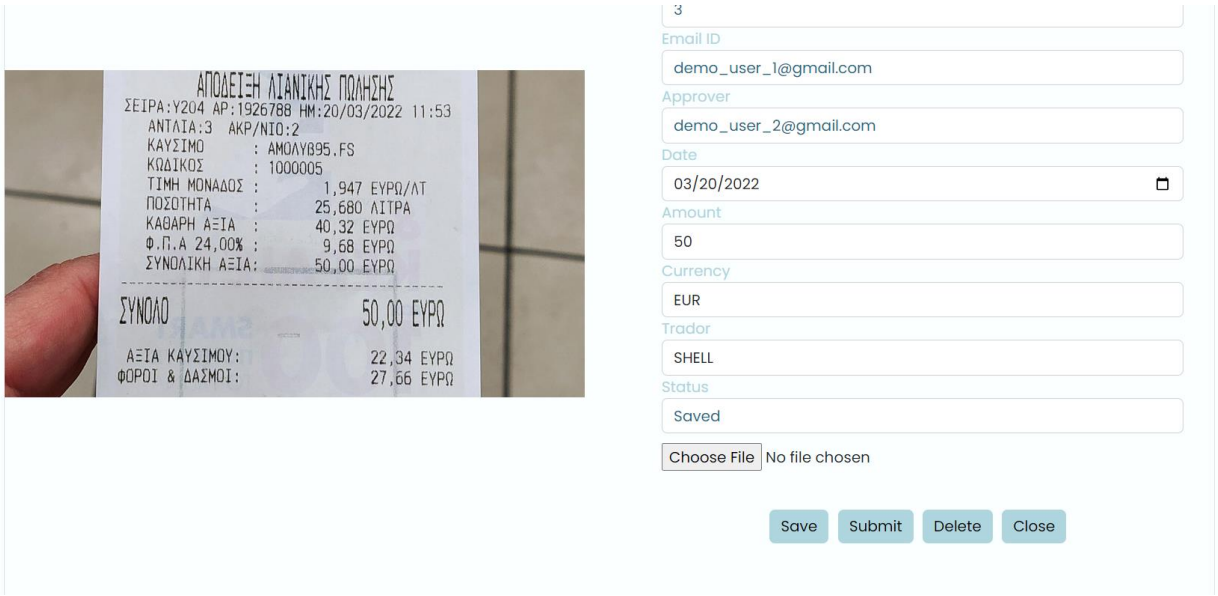
Το μήνυμα εμφανίζεται στο παρακάτω σημείο:



Εικόνα 30: Παρουσίαση Εφαρμογής Καρτέλα Εξόδου Σφάλμα

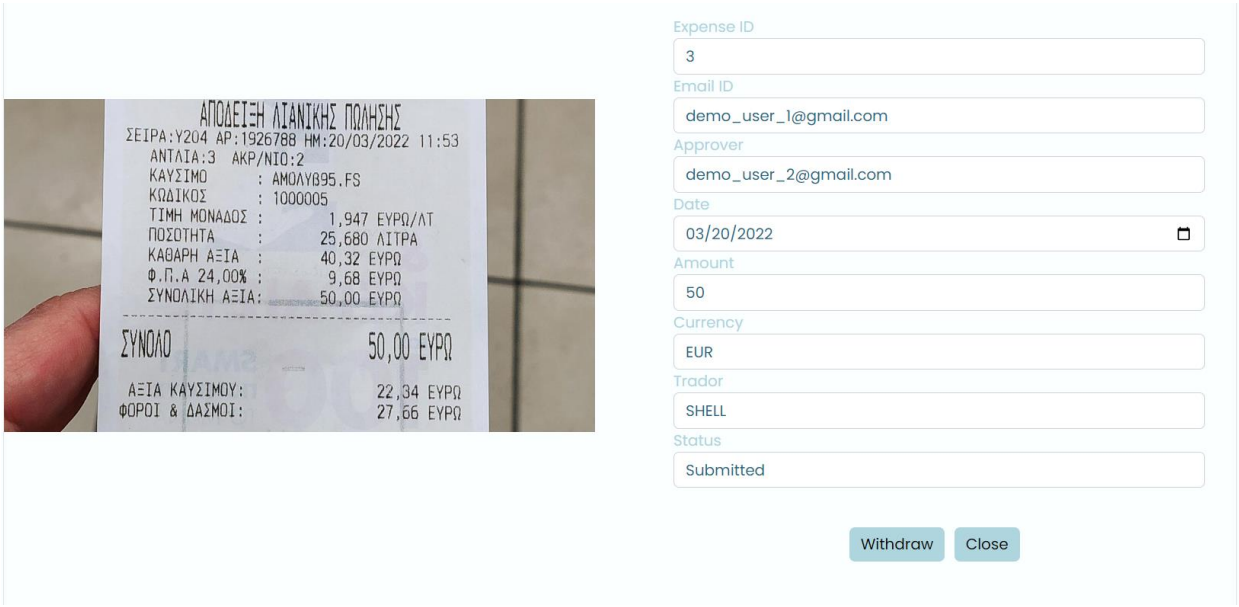
Με την αποθήκευση του εξόδου, το status το εξόδου γίνεται Saved. Στην καρτέλα εξόδου εμφανίζονται οι επιλογές:

- Save: σε περίπτωση που θέλουμε να κάνουμε αλλαγές στο έξοδο.
- Submit: για την καταχώρηση του εξόδου.
- Delete: για διαγραφή εξόδου.
- Close: για να κλείσουμε την καρτέλα εξόδου.



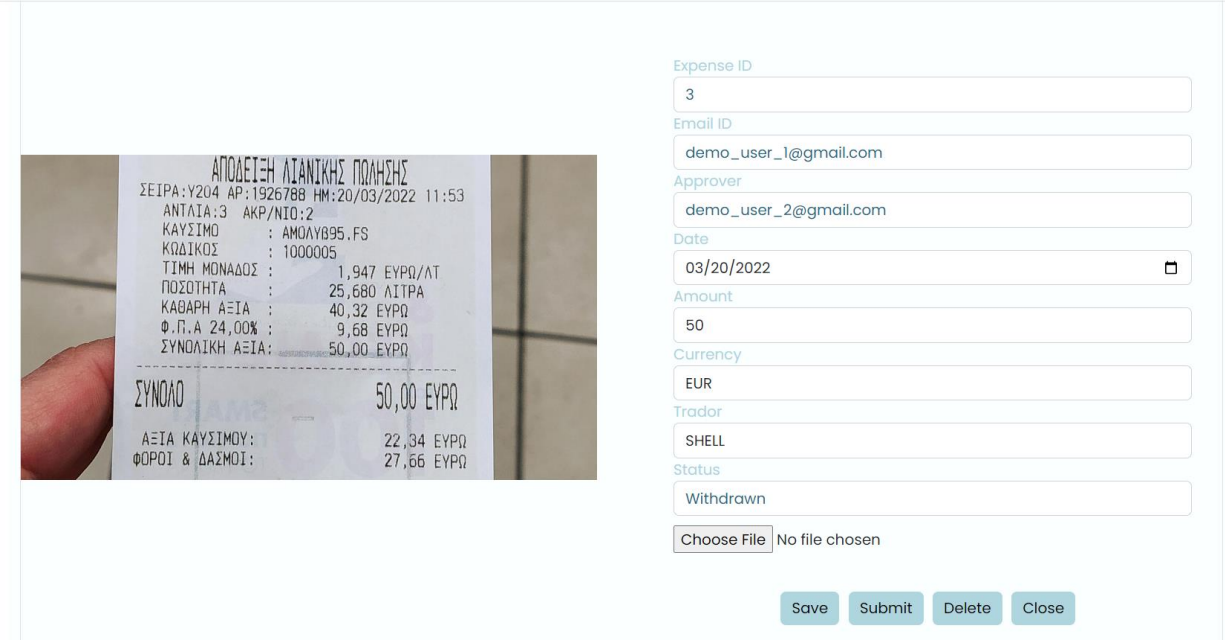
Εικόνα 31: Saved Expense

Με την καταχώρηση του εξόδου, το έξοδο αλλάζει σε status Submitted και στέλνεται μήνυμα στο inbox του approver για καταχώρηση.



Εικόνα 32: Submitted Expense

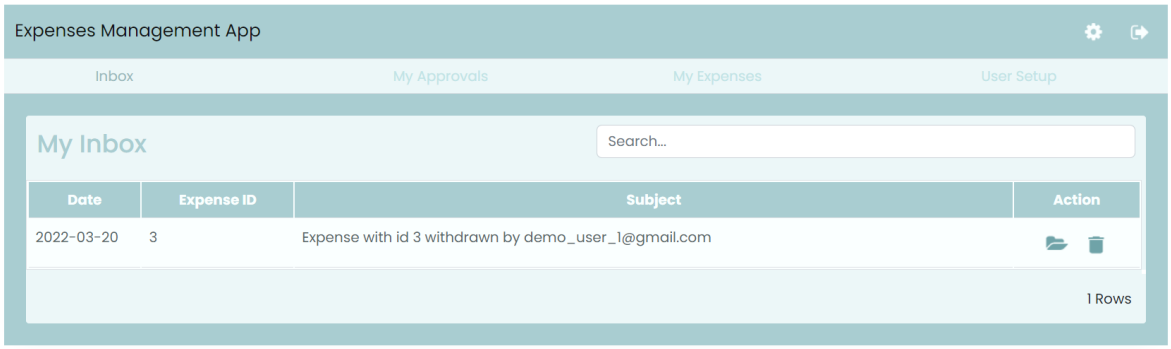
Ο χρήστης δεν μπορεί να κάνει αλλαγές σε Submitted έξοδο. Για να κάνει αλλαγές θα πρέπει να επιλέξει Withdraw. Με την επιλογή Withdraw αποσύρουμε το έξοδο. Το έξοδο δεν θα εμφανίζεται στην λίστα εξόδων προς έγκριση του εγκριτή.



Εικόνα 33: Withrdrawn Expense

Το έξοδο έχει την ίδια συμπεριφορά με αυτή του εξόδου που είναι σε κατάσταση saved. Οι επιλογές είναι Save, Submit, Delete, Close.

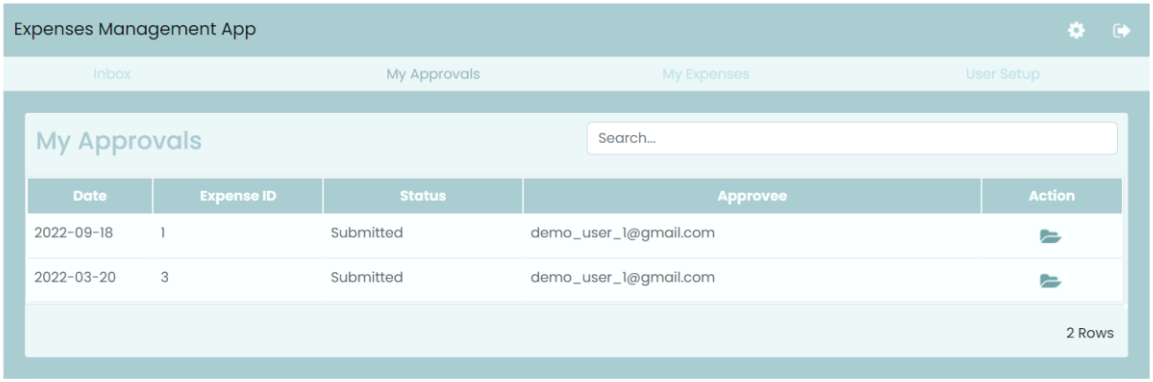
Με το Submit, πηγαίνει ενημερωτικό μήνυμα στον εγκριτή demo_user_2@gmail.com.



Εικόνα 34: Παρουσίαση Εφαρμογής Inbox

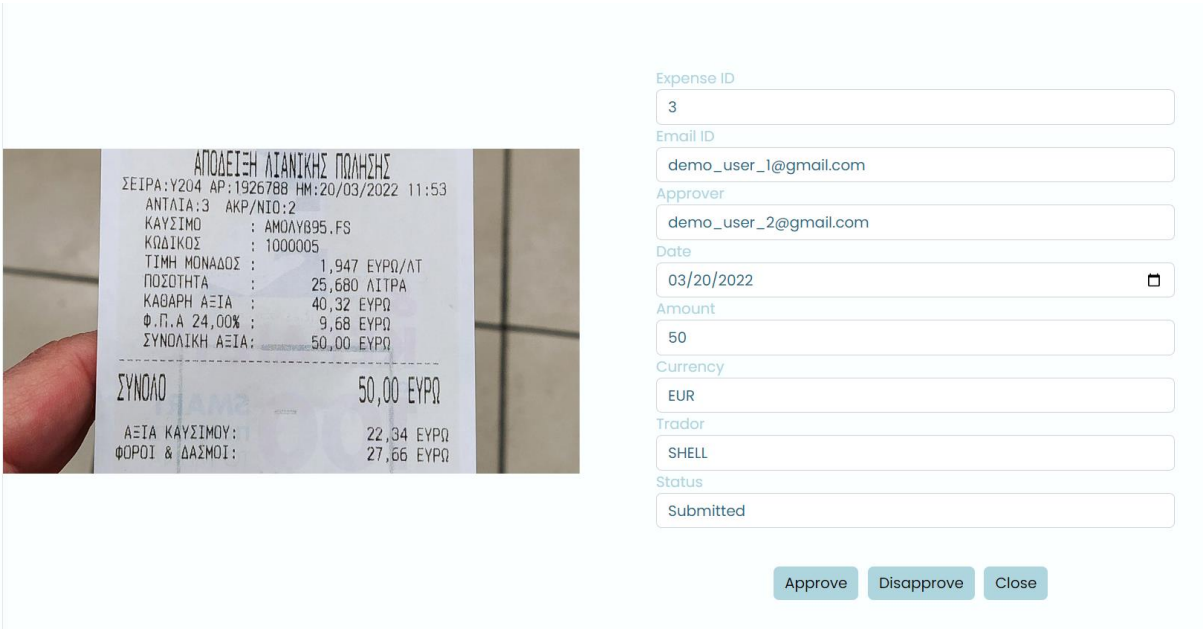
Ο χρήστης μπορεί να διαγράψει το μήνυμα επιλέγοντας το εικονίδιο .

Ο χρήστης μπορεί να ανοίξει το έξοδο επιλέγοντας το εικονίδιο .
Το έξοδο επίσης θα εμφανίζεται στο My Approvals.



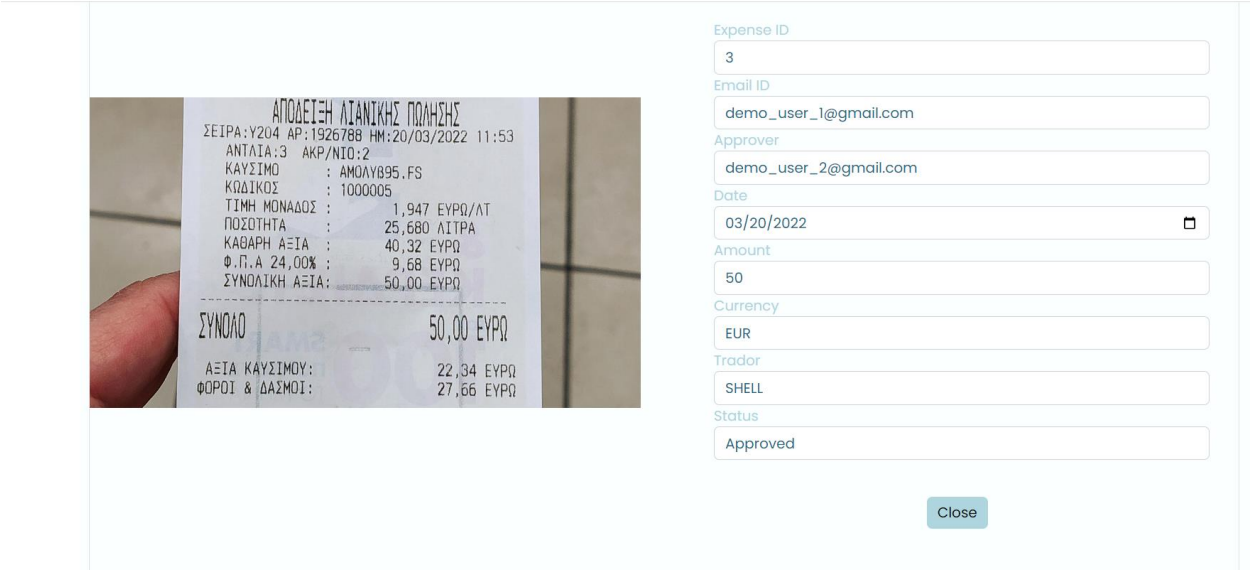
Εικόνα 35: Παρουσίαση Εφαρμογής My Approvals

Ο χρήστης μπορεί να ανοίξει το έξοδο επιλέγοντας το εικονίδιο .
Με το άνοιγμα του εξόδου εμφανίζεται η καρτέλα εξόδου.

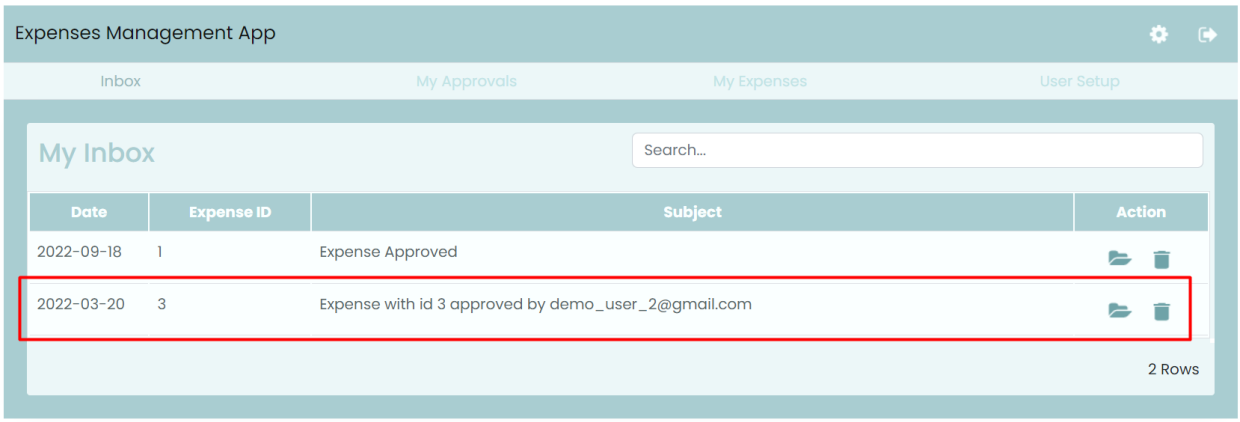


Εικόνα 36: Παρουσίαση Εφαρμογής Expense Approver

Ο εγκριτής μπορεί είτε να κάνει Approve είτε Disapprove το έξοδο. Σε κάθε περίπτωση το έξοδο θα πάρει το αντίστοιχο status (Approved/Disapproved) και θα σταλεί ενημερωτικό μήνυμα στον υπάλληλο.

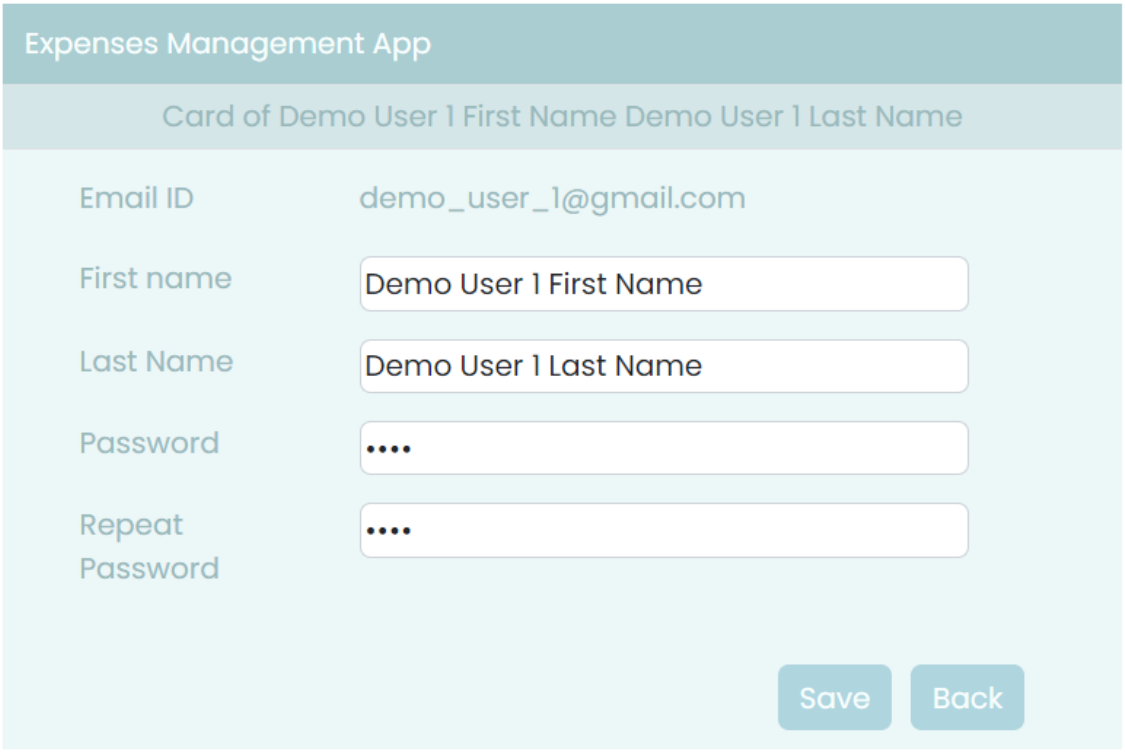


Εικόνα 37: Approved Expense



Εικόνα 38: Εγκεκριμένο έξοδο μήνυμα

Ο χρήστης μπορεί να αλλάξει τα στοιχεία του επιλέγοντας το εικονίδιο  . Τα στοιχεία που μπορεί να αλλάξει είναι το First Name, το Last Name και τον κωδικό του.

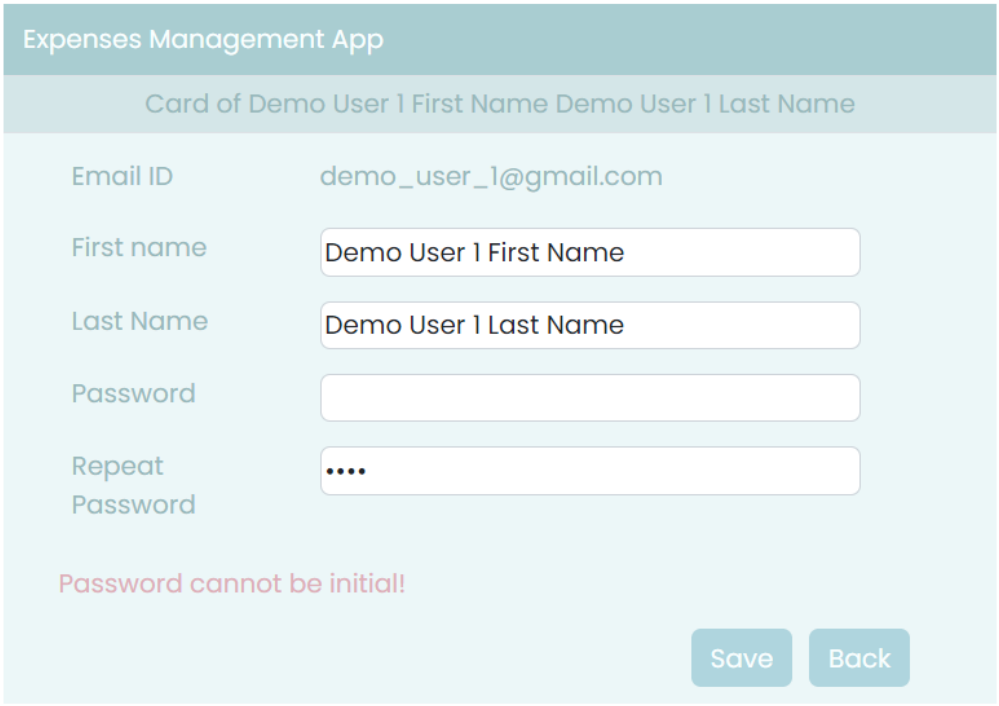


Εικόνα 39: Παρουσίαση Εφαρμογής Settings

Κατά την αποθήκευση το σύστημα κάνει του εξής ελέγχους:

- Το First Name δεν πρέπει να είναι κενό. Αν είναι κενό βγαίνει μήνυμα: “First Name cannot be initial!”.
- Το Last Name δεν πρέπει να είναι κενό. Αν είναι κενό βγαίνει μήνυμα: “Last Name cannot be initial”.
- Το Password δεν πρέπει να είναι κενό. Αν είναι κενό βγαίνει μήνυμα: “Password cannot be initial!”.
- Το Repeat Password δεν πρέπει να είναι κενό. Αν είναι κενό βγαίνει μήνυμα: “Repeat Password cannot be initial!”.
- Το Password πρέπει να είναι ίδιο με το Repeat Password. Αν δεν είναι ίδιο βγαίνει μήνυμα: “ Password and Repeat Password do not match!”.

Το μήνυμα εμφανίζεται στο παρακάτω σημείο της οθόνης:



Εικόνα 40: Παρουσίαση Εφαρμογής My Settings

Ο χρήστης μπορεί να βγει από την εφαρμογή επιλέγοντας το εικονίδιο  .

6 Σύνοψη – Συμπεράσματα

Στην σημερινή εποχή, λόγω της πανδημίας και του νέου μοντέλου εργασίας από το σπίτι, οι μικρομεσαίες επιχειρήσεις έχουν να αντιμετωπίσουν μια ακόμα πρόκληση: την αυτοματοποίηση ακόμα και των πιο απλών διαδικασιών, χωρίς να απαιτείται ανθρώπινη παρουσία.

Μεγάλος μέρος των συστημάτων διαχείρισης επιχειρησιακών πόρων που υπάρχουν στην αγορά, και απευθύνονται σε μικρομεσαίες επιχειρήσεις, επικεντρώνονται στην λογιστική και εμπορική διαχείριση και διαδικασίες. Αυτό έχεις ως αποτέλεσμα αυτόνομες διαδικασίες, μικρότερης εμβέλειας, να μην μπορούν να καλυφθούν, μια εκ των οποίων είναι η έγκριση εξόδων στα πλαίσια παροχών προς τους εργαζομένους.

Με αυτό σαν αφετηρία, μελετήθηκαν και αναλύθηκαν οι λειτουργίες που θα μπορούσε να έχει μια τέτοια εφαρμογή με γνώμονα τον περιορισμό κόστους και πολυπλοκότητας. Κατόπιν μελέτης διαφορετικών προϊόντων που υπάρχουν στην αγορά, καταλήξαμε σε κάποιες βασικές λειτουργίες της οποίες τις προσεγγίσαμε αρχικά σχεδιαστικά μέσα από την δημιουργία ροών περιγραφής χρήσης και διαγραμμάτων. Με βάση τις λειτουργίες αυτές, σχεδιάσαμε την διεπαφή χρήστη, το οπτικό κομμάτι

δηλαδή των οθονών. Για την εύκολη και ευχάριστη χρήση της εφαρμογής, προσπαθήσαμε να απλοποιήσουμε όσο περισσότερο γίνεται την πληροφορία στις οθόνες και την συμμετοχή του χρήστη για την ολοκλήρωση της διαδικασίας.

Βασιζόμενοι στον λειτουργικό και οπτικό σχεδιασμό, καταλήξαμε στον τεχνικό σχεδιασμό, τον οποίο χρησιμοποιήσαμε και ως βάση για την τελική υλοποίηση της εφαρμογής. Σε αυτό το στάδιο, σκοπός ήταν να φτιάξουμε μια εφαρμογή που θα μπορούσε τεχνικά να εξελιχθεί εύκολα στο μέλλον ανάλογα με τις ανάγκες. Λόγω της τεχνολογίας που χρησιμοποιήθηκε, θα μπορούσαμε στο μέλλον να εξελίξουμε την εφαρμογή ώστε να μπορεί να χρησιμοποιηθεί και από άλλες συσκευές, ακόμα και από τρίτα προγράμματα, χωρίς να χρειάζεται η τροποποίηση του λειτουργικού υπόβαθρου.

Στο τελευταίο κεφάλαιο της εφαρμογής, παρουσιάστηκε η χρήστη της εφαρμογής αυτής μελετώντας κάποια βασικά σενάρια. Το συγκεκριμένο κεφάλαιο σκοπός ήταν πέρα από την τελική παρουσίαση του αποτελέσματος, να αποτελέσει και ένα εγχειρίδιο χρήσης της εφαρμογής.

Μέσω της εκπόνησης της συγκεκριμένης μεταπτυχιακής διατριβής, μας δόθηκε η δυνατότητα να προσεγγίσουμε μια επιχειρησιακή διαδικασία από πολλές οπτικές γωνίες. Αρχικά μελετήθηκε και αξιολογήθηκε η κατάσταση στην αγορά, προκειμένου να καταλήξουμε στην ιδέα κατανοώντας τις ανάγκες που υπάρχουν αυτή την στιγμή στην αγορά. Στην συνέχεια προσεγγίσαμε την ιδέα αυτή λειτουργικά και σχεδιαστικά, προκειμένου να καταλήξουμε στον τεχνικό σχεδιασμό. Στα πλαίσια του λειτουργικού και τεχνικού σχεδιασμού, έγινε έρευνα προκειμένου να κατανοήσουμε την τεχνολογία που θα χρησιμοποιηθεί για την υλοποίηση. Η προεργασία αυτή απλοποίησε αρκετά το στάδιο υλοποίησης, κατά το οποίο αξιοποιήθηκαν αρκετά εργαλεία και στα όποια μας έγιναν γνωστές νέες τεχνολογίες. Τέλος, προσεγγίσαμε την εφαρμογή και σε επίπεδο χρήσης, τόσο για να ελεγχθεί η σωστή λειτουργία και η εύκολη χρήση, αλλά και προκειμένου να δημιουργήσουμε ένα εγχειρίδιο χρήσης για μελλοντικούς χρήστες.

7 Πηγές – Βιβλιογραφία

7.1 Στάδια Υλοποίησης

7.1.1 Joe Johnston (23.04.2019). How to build a web app: A beginner's guide (2022). Budibasae.

7.2 Έρευνα Αγοράς & Παρόμοια Προϊόντα

7.2.1 Julie Bawden-Davis (05.01.2022). 7 Best Business Tracker Apps for 2022. American Express.

7.2.2 Elizabeth Gravier (05.08.2022). Here are the 5 best expense tracker apps of 2022. CNBC.

7.2.3 Rajat Sharma (19.09.2022). Best Expense Tracker Apps to Download. The Balance Money.

7.2.4 Steve McCaskill, Brian Turner, Rob Clymo (14.07.2022). Best expense tracker app of 2022. Techradar.

7.2.5 Hillary Crawford (15.08.2022). Best Business Expense Trackers for Small Businesses. Nerdwallet.

7.2.6 Kathy Haan (01.08.2022). Best Expense Tracker Apps. Investopedia.

7.2.7 (01.10.2022). Best Expense Management Software. G2.

7.2.8 Sean Peek (03.08.2022). 10 Top Expense Trackers. Business News Daily.

7.2.9 Expensify Pricing, Features, Reviews and Alternatives. Getapp.

7.2.10 Expensify. Saasworthy.

7.2.11 Expensify. G2.

7.2.12 SAP Concur Pricing, Features, Reviews and Alternatives. Getapp.

7.2.13 SAP Concur. Saasworthy.

7.2.14 SAP Concur. G2.

7.2.15 Zoho Expense Pricing, Features, Reviews and Alternatives. Getapp.

7.2.16 Zoho Expense. Saasworthy.

7.2.17 Zoho Expense. G2.

7.2.18 Rydoo. Saasworthy.

7.2.19 Rydoo. G2.

7.2.20 Everlance Pricing, Features, Reviews and Alternatives. Getapp.

7.2.21 Everlance. Saasworthy.

7.3 Ροές & Διαγράμματα

7.3.1 Kupe Kupersmith, Paul Mulvey, Kate McGoey (26.03.2016). How to Create Use Case Description for Your Business Analysis Report. Dummies.

7.4 Τεχνικός Σχεδιασμός

7.4.1 What are microservices. Microservices.

7.4.2 Spring Boot Introduction. Tutorialspoint.

7.4.3 Web Service. Wikipedia.

7.4.4 (08.05.2022). What is a REST API. Redhat.

7.4.5 Model-view-controller. Wikipedia.

7.4.6 PostgreSQL: The World's Most Advanced Open Source Relational Database. Postgresql.

7.4.7 PostgreSQL. Wikipedia.

7.4.8 SQL. Wikipedia.

7.4.9 pgAdmin - PostgreSQL Tools. PgAdmin.

7.4.10 Aveek Das (10.06.2021). An overview of PGAdmin – PostgreSQL Management Tool. SQLShack.

7.4.11 IntelliJ IDEA. Wikipedia.

7.4.12 React - A JavaScript library for building user interfaces. Reactjs.

7.4.13 Visual Studio Code. Wikipedia.

7.4.14 React (JavaScript library). Wikipedia.

7.4.15 Postman Software. Wikipedia.

8 Παραρτήματα

8.1 Παράρτημα Εικόνων

8.2 Παράρτημα Πινάκων

8.3 Παράρτημα Διαγραμμάτων

8.4 Παράρτημα Κώδικα

8.5 Παράρτημα Μηνυμάτων API

8.5.1 Requests

8.5.2 Responses