ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	**Σύστημα Ελέγχου Πρόσβασης Βασισμένο σε Ρόλους** **Role Based Access Control System**
Ονοματεπώνυμο Φοιτητή	**Κωνσταντίνος Γεράσοβιτς**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**Π17019**
Επιβλέπων	**Ευθύμιος Αλέπης, Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2024**

Copyright ©

Διμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης
Καθηγητής
του Τμήματος Πληροφορικής

Μαρία Βίρβου
Καθηγήτρια και Πρόεδρος
του Τμήματος Πληροφορικής

Πίνακας Περιεχομένων

1 Ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στον αναπληρωτή καθηγητή κ. Ευθύμιο Αλέπη για τις συνεχείς παραινέσεις και συμβουλές του πάνω σε αυτή μου την προσπάθεια, καθώς και για την συνολική καθοδήγηση που μου παρείχε, από την έναρξη της εκπόνησης της παρούσας εργασίας, μέχρι και την ολοκλήρωσή της. Είμαι ιδιαίτερα ευγνώμων για την εξαιρετική επικοινωνία που είχαμε και τη συνεχή ανατροφοδότηση που μου παρείχε, σε όλα τα στάδια της πτυχιακής μου εργασίας, τα οποία είχαν καταλυτικό ρόλο στην επιτυχή ολοκλήρωσή της.

Με την παράλληλη ολοκλήρωση των σπουδών μου, θα ήθελα ακόμη να ευχαριστήσω θερμά τους καθηγητές του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς, για τις ουσιαστικές γνώσεις που μου μετέδωσαν και την συνολική υποστήριξή τους στα χρόνια της φοίτησής μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και ιδιαιτέρως τους γονείς μου, για την διαχρονική τους πνευματική και υλική υποστήριξη και για το υγιές περιβάλλον που πάντοτε παρείχαν.

Σεπτέμβριος 2024
Γεράσοβιτς Κωνσταντίνος

2 Παραδείγματα κώδικα

3 Κατάλογος εικόνων

4 Περίληψη

Κείμενο στα ελληνικά

Η παρούσα πτυχιακή εργασία εξετάζει την ανάπτυξη και υλοποίηση ενός συστήματος ελέγχου πρόσβασης βασισμένου σε ρόλους (RBAC), προσαρμοσμένο στις ανάγκες μιας εταιρείας ψηφιακού μάρκετινγκ. Η κεντρική ιδέα του RBAC είναι ότι οι χρήστες δεν έχουν άμεση πρόσβαση στα αντικείμενα της επιχείρησης. Αντίθετα, τα δικαιώματα πρόσβασης συνδέονται με αυστηρά καθορισμένους ρόλους και κάθε χρήστης με την σειρά του διαθέτει έναν συγκεκριμένο ρόλο. Η ιδέα αυτή απλοποιεί σε μεγάλο βαθμό τη διαχείριση της πρόσβασης σε ένα εταιρικό σύστημα, ενώ παρέχει μεγάλη ευελιξία στον καθορισμό και την επιβολή των πολιτικών προστασίας της εκάστοτε επιχείρησης. Οι χρήστες λαμβάνουν ρόλους ανάλογα με τις ευθύνες και τα προσόντα τους και μπορούν εύκολα να αποκτήσουν έναν διαφορετικό ρόλο, χωρίς να αλλάξει η υποκείμενη δομή πρόσβασης στο σύστημα. Οι ρόλοι με την σειρά τους μπορούν να τροποποιηθούν, λαμβάνοντας νέα δικαιώματα καθώς ενσωματώνονται νέες εφαρμογές και ενέργειες, και τα δικαιώματα μπορούν να ανακληθούν από τους ρόλους όταν χρειάζεται.

Η παρούσα εργασία παρέχει μια λεπτομερή ανάλυση της εξέλιξης των συστημάτων RBAC και των τρεχουσών τεχνολογιών που χρησιμοποιούνται για την υλοποίησή τους. Το front-end της εφαρμογής αναπτύχθηκε με τη χρήση των TypeScript, React, Ant Design και Axios, ενώ για την υλοποίηση του back-end χρησιμοποιήθηκαν οι Go, Gin Framework, Casbin και GORM. Επιπλέον, γίνεται αναφορά στην ανάπτυξη και υλοποίηση μιας συμπληρωματικής ψηφιακής πλατφόρμας (Portal), απευθυνόμενης τόσο στα μέλη, όσο και στους πελάτες της εν λόγω εταιρείας, με σκοπό να αναδειχθούν καλύτερα οι λειτουργίες και οι δυνατότητες της κεντρικής εφαρμογής. Τα αποτελέσματα δείχνουν ότι ένα σύστημα RBAC, λόγω της ευελιξίας και της ελευθερίας που προσφέρει στον σχεδιασμό του, είναι ικανό να τεθεί σε ισχύ σε οποιοδήποτε εταιρικό σύστημα, ανεξάρτητα από τις ιδιαιτερότητες ή την πολυπλοκότητα της δομής του. Καθίσταται συνεπώς ιδανική λύση για την ασφάλεια των δεδομένων, τόσο για μικρομεσαίες επιχειρήσεις όσο και για πολυεθνικές μεγαλύτερου βεληνεκούς.

Abstract in english

This thesis explores the development and implementation of a Role-Based Access Control (RBAC) system, tailored to a digital marketing company's needs. The central notion of RBAC is that users do not have direct access to enterprise objects. Instead, access permissions are administratively associated with roles, and users are administratively made members of appropriate roles. This idea greatly simplifies system authorization management, while providing an opportunity for great flexibility in specifying and enforcing enterprise-specific protection policies. Users can be made members of roles, as determined by their responsibilities and qualifications and can be easily reassigned from one role to another, without modifying the underlying access structure. Roles can be granted new permissions as new applications and actions are incorporated, and permissions can be revoked from roles as needed.

This work provides an in-depth analysis of the evolution of RBAC systems and the current technologies used to implement them. The front-end of the application is developed using TypeScript, React, Ant Design, and Axios, while the back-end utilizes Go, Gin Framework, Casbin, and GORM. Additionally, this thesis also details the implementation of a supplementary digital platform (Portal), aimed at both members and clients of the company, in order to better showcase the functionalities and capabilities of the main application. The results demonstrate that an RBAC system, due to the flexibility and freedom it offers in its design, is capable of being implemented in any corporate system, regardless of the specifics or complexity of its structure. Thus, it becomes an ideal solution for data security, suitable for both small and medium-sized businesses, as well as larger, multinational corporations.

5 Εισαγωγή

Ένα Σύστημα Ελέγχου Πρόσβασης βασισμένο σε Ρόλους (Role-Based Access Control System) αποτελεί κρίσιμο στοιχείο στον τομέα της προστασίας των ευαίσθητων δεδομένων και πληροφοριών μιας σύγχρονης επιχείρησης. Η ασφάλεια των δεδομένων είναι ζωτικής σημασίας όχι μόνο για μεγάλα εταιρικά συστήματα, αλλά και για εφαρμογές που χρησιμοποιούμε καθημερινά, οι οποίες διαχειρίζονται ευαίσθητες πληροφορίες [1]. Το RBAC βασίζεται σε ένα πολύ αποτελεσματικό μοντέλο ελέγχου πρόσβασης με μακροχρόνια ιστορία, η εξέλιξη του οποίου έχει επηρεάσει σε μεγάλο βαθμό τον τρόπο που πολλές εταιρείες διαχειρίζονται την πρόσβαση στα συστήματά τους τις τελευταίες δεκαετίες.

Ένα σύστημα RBAC (Role-Based Access Control) είναι υπεύθυνο για την εξουσιοδότηση ή τον περιορισμό της πρόσβασης των χρηστών σε ένα εταιρικό σύστημα, βάσει των ρόλων τους μέσα στην εταιρεία για την οποία γίνεται η αναφορά. Αυτό επιτρέπει στους χρήστες να έχουν πρόσβαση στα δεδομένα και τις εφαρμογές που χρειάζονται για να φέρουν εις πέρας τις εργασίες τους και παράλληλα μειώνει τον κίνδυνο μη εξουσιοδοτημένων προσβάσεων σε ευαίσθητες πληροφορίες ή σε ανεξουσιοδότητες εργασίες. Υπάρχουν τρία είδη ελέγχου πρόσβασης σύμφωνα με το πρότυπο RBAC, το βασικό (core), το ιεραρχικό (hierarchical) και το περιορισμένο (constrained) [2].

Βασικό RBAC: Το βασικό μοντέλο καθορίζει τα θεμελιώδη στοιχεία κάθε συστήματος ελέγχου πρόσβασης βασισμένου σε ρόλους. Περιλαμβάνει τρεις βασικούς κανόνες:

1. **Ανάθεση Ρόλου:** Ένας χρήστης έχει δικαίωμα (permission) να εκτελέσει μια λειτουργία εάν και μόνο αν του έχει του έχει ανατεθεί ένας ρόλος.
2. **Εξουσιοδότηση Ρόλου:** Κάθε ενεργός ρόλος πρέπει να είναι εξουσιοδοτημένος.
3. **Εξουσιοδότηση Δικαιωμάτων (permissions):** Ένας χρήστης μπορεί να χρησιμοποιήσει ένα δικαίωμα (permission) μόνο εάν αυτό είναι εξουσιοδοτημένο για τον ενεργό ρόλο του.

Ιεραρχικό RBAC: Το ιεραρχικό μοντέλο επεκτείνει το βασικό, προσθέτοντας ιεραρχικές σχέσεις μεταξύ των ρόλων. Αυτό επιτρέπει τη δημιουργία υπο-ρόλων και την καλύτερη οργάνωση των δικαιωμάτων πρόσβασης (permissions).

Περιορισμένο RBAC: Το περιορισμένο μοντέλο προσθέτει τον έλεγχο διαχωρισμού των καθηκόντων (Separation of Duties - SoD) στο βασικό μοντέλο. Υπάρχουν δύο είδη διαχωρισμού, ο στατικός και ο δυναμικός. Στο πλαίσιο των σχέσεων Static Separation of Duty (SSD), ένας χρήστης δεν μπορεί να έχει δύο ή περισσότερους αποκλειστικούς ρόλους, που να μην σχετίζονται. Αντιθέτως, στο μοντέλο Dynamic Separation of Duty (DSD), ένας χρήστης μπορεί να είναι μέλος αντικρουόμενων ρόλων.

6 Περιγραφή του υπό μελέτη προβλήματος

Η κύρια εφαρμογή της παρούσας εργασίας, ένα σύστημα RBAC, είναι η απάντηση σε ένα πραγματικό πρόβλημα που προέκυψε σε ένα πραγματικό εταιρικό περιβάλλον. Για να κατανοήσουμε καλύτερα το συγκεκριμένο πρόβλημα και τους λόγους για τους οποίους το RBAC προτάθηκε ως η πλέον κατάλληλη επιλογή για την επίλυση του, κρίνεται απαραίτητο να εξηγήσουμε συνοπτικά τη δομή της εταιρείας για την οποία γίνεται η αναφορά, καθώς και τις επιμέρους υπηρεσίες που προσφέρει.

Πρόκειται για μια εταιρεία Digital Marketing με αντικείμενο την online διαφήμιση και την διανομή περιεχομένου σε όλα τα μέσα κοινωνικής δικτύωσης. Απευθύνεται σε μεμονωμένα άτομα, οργανισμούς και εταιρείες που παράγουν περιεχόμενο σε διάφορες πλατφόρμες, όπως το YouTube, το Instagram, το TikTok, το Spotify, το Apple Music, το iTunes κ.ά.

Αρχικά, στην εταιρεία υπήρξε η ανάγκη αυτοματοποίησης πολλών διαδικασιών, οι οποίες μέχρι πρότινος πραγματοποιούνταν χειροκίνητα. Οι εν λόγω διαδικασίες ενσωματώθηκαν σε ένα ενιαίο σύστημα πλατφορμών, ανοιχτό σε όλους, χωρίς να ελέγχεται η πρόσβαση με οποιονδήποτε τρόπο. Γίνεται εύκολα αντιληπτό ότι η δημιουργία μιας εφαρμογής ελέγχου πρόσβασης στα εταιρικά συστήματα, κρίθηκε απαραίτητη. Σκοπός της ήταν η απαγόρευση της εισόδου σε τρίτους, δηλαδή σε όσους δεν ανήκουν στο πελατολόγιο ή στα μέλη της εταιρείας, οι οποίοι με τη σειρά τους θα έχουν πρόσβαση σε συγκεκριμένα δεδομένα και τμήματα των εφαρμογών του συστήματος, ανάλογα με τις ανάγκες τους.

Για να γίνουν πιο σαφείς οι ανάγκες των πελατών και των εργαζομένων, με σκοπό να κατανοήσουμε ποια δεδομένα του συστήματος τους αφορούν και να τους δοθεί η κατάλληλη πρόσβαση, θα πάρουμε ως παράδειγμα και θα συγκρίνουμε δύο κατηγορίες πελατών και δύο κατηγορίες υπαλλήλων. Οι πρώτοι είναι είτε φυσικά πρόσωπα, μεμονωμένες οντότητες δηλαδή, είτε εταιρείες οι οποίες διαχειρίζονται έναν μεγάλο αριθμό ατόμων. Όσον αφορά τα μέλη της εταιρείας, θα πάρουμε ως παράδειγμα τους εργαζομένους του τμήματος creators management και του λογιστηρίου της εταιρείας.

Όλα τα άτομα που προαναφέρθηκαν, αποτελούν χρήστες της εταιρικής εφαρμογής "Portal", μιας ψηφιακής πλατφόρμας που απευθύνεται τόσο στους πελάτες, όσο και στα μέλη της εταιρείας. Στην εν λόγω πλατφόρμα εμφανίζονται όλα τα οικονομικά δεδομένα και τα στοιχεία προόδου των πελατών, αναφορικά με τα προφίλ και κανάλια τους στα μέσα κοινωνικής δικτύωσης. Περιλαμβάνει επίσης reports σε μορφή παρουσιάσεων και υπολογιστικών φύλλων, τα οποία εκδίδονται κάθε μήνα και ενημερώνουν τους πελάτες για τα κέρδη και την πρόοδό τους. Το σύστημα RBAC που προτάθηκε ως η πλέον κατάλληλη επιλογή για τον έλεγχο πρόσβασης στα εταιρικά συστήματα, διαχειρίζεται και την προσβασιμότητα των χρηστών στο Portal. Για τον λόγο αυτό, θα χρησιμοποιήσουμε το Portal ως την βασική εφαρμογή στην οποία συνδέονται πελάτες και εργαζόμενοι, ώστε να αναδείξουμε καλύτερα τις λειτουργίες και τις δυνατότητες του RBAC.

Ας ξεκινήσουμε την ανάλυση των χρηστών του Portal, εξετάζοντας τις δύο κατηγορίες πελατών της εταιρείας. Η πρώτη κατηγορία αφορά ελεύθερους επαγγελματίες που διαθέτουν προφίλ στα μέσα κοινωνικής δικτύωσης, όπως το Instagram, το Facebook και το TikTok, κανάλια στο YouTube ή record labels στις μεγαλύτερες πλατφόρμες ροής μουσικής, όπως το Spotify, στο iTunes, το Deezer, το Amazon Prime και άλλα. Κατά την είσοδό τους στο Portal, βλέπουν μια αναλυτική περιγραφή των εσόδων τους από τις πλατφόρμες όπου οι ίδιοι προβάλλονται, διάφορα metrics αναφορικά με την πορεία και την εξέλιξη των καναλιών τους, καθώς και την λίστα των εγκεκριμένων performance και financial reports τους. Η δεύτερη κατηγορία πελατών περιλαμβάνει ολόκληρες εταιρείες και οργανισμούς, που απασχολούν ένα πλήθος εργαζομένων, οι οποίοι αποτελούν με τη σειρά τους χρήστες του Portal. Ανάλογα με τον ρόλο τους στην εταιρεία-πελάτη, βλέπουν συγκεκριμένα δεδομένα στην εφαρμογή, που ανταποκρίνονται ειδικά στις ανάγκες τους. Παραδείγματος χάριν, κάποιοι χρήστες μπορεί να έχουν πρόσβαση μόνο σε λογιστικές πληροφορίες και οικονομικά δεδομένα, ενώ άλλοι αποκλειστικά σε στοιχεία προόδου και performance reports.

Η δεύτερη κατηγορία χρηστών του Portal και κατ' επέκταση όλων των εταιρικών εφαρμογών, είναι τα μέλη της εταιρείας. Αρχικά θα αναλύσουμε τον ρόλο των εργαζομένων του τμήματος creators management, ή account managers, όπως ονομάζονται εσωτερικά στην εταιρεία. Έχουν συμβουλευτικό ρόλο και ο κάθε ένας από αυτούς διαχειρίζεται έναν συγκεκριμένο αριθμό πελατών. Βρίσκονται σε διαρκή επικοινωνία με τους πελάτες καθώς είναι υπεύθυνοι για την διαχείριση των καναλιών και των προφίλ τους στα κοινωνικά δίκτυα, οφείλουν να τους συμβουλεύουν και να τους κατευθύνουν, ώστε να εξελίσσουν συνεχώς το περιεχόμενο που προβάλλουν και να τους ενημερώνουν τακτικά

για την πρόοδό τους. Είναι επίσης υπεύθυνοι για τη διαχείριση των δεδομένων για την παραγωγή των performance reports των πελατών. Κάθε account manager, κατά τη σύνδεσή του στο Portal, πρέπει να βλέπει τα οικονομικά δεδομένα, τα στοιχεία προόδου, τα financial και performance reports των πελατών που ο ίδιος διαχειρίζεται. Η δημιουργία, ο έλεγχος, η επεξεργασία και η έγκριση των παρουσιάσεων προόδου των πελατών, αποτελούν κάποιες από τις βασικές αρμοδιότητες ενός account manager και μπορεί να τις φέρει εις πέρας μέσω των χρήσιμων εργαλείων που του παρέχει το Portal. Μια δεύτερη ομάδα εργαζομένων με πρόσβαση στο σύστημα των εταιρικών εφαρμογών, είναι οι υπάλληλοι του λογιστηρίου, οι οποίοι αναλαμβάνουν τη διαχείριση των οικονομικών της εταιρείας, εστιάζοντας στις ειδικές ανάγκες και απαιτήσεις που προκύπτουν από τον χαρακτήρα του digital marketing. Αυτό συμπεριλαμβάνει την παρακολούθηση των εσόδων και των εξόδων που σχετίζονται με τις δραστηριότητες της εταιρείας στον ψηφιακό τομέα, τις διαφημίσεις της στα μέσα κοινωνικής δικτύωσης, την ανάλυση των κερδών της από διαφημιστικές καμπάνιες, τη δημιουργία οικονομικών αναφορών που βοηθούν τη διοίκηση της εταιρείας στη λήψη στρατηγικών αποφάσεων κ.ά. Κατά τη σύνδεσή τους στο Portal, τα μέλη του λογιστηρίου πρέπει να βλέπουν μόνο τα οικονομικά δεδομένα και τις παρουσιάσεις των πελατών. Σε αντίθεση με τους account managers, δεν πρέπει έχουν πρόσβαση σε στοιχεία προόδου και performance reports, καθώς δεν είναι χρήσιμα για τον ρόλο τους μέσα στην εταιρεία.

Από τα παραπάνω συμπεραίνουμε ότι η ύπαρξη ενός συστήματος καθορισμού ρόλων, στο πολύπλοκο εταιρικό μας περιβάλλον, κρίνεται απαραίτητη. Ένα σύστημα ελέγχου πρόσβασης, που παραχωρεί ή αφαιρεί δικαιώματα σε χρήστες ανάλογα με το ρόλο τους, όπως το RBAC, αποτελεί την πλέον κατάλληλη λύση για την ασφάλεια των εταιρικών συστημάτων, προσαρμοσμένη στις ανάγκες του οργανισμού μας.

7 Ιστορική Εξέλιξη των Συστημάτων RBAC (Role-Based Access Control Systems)

Η εξέλιξη των συστημάτων Role-Based Access Control (RBAC) αποτελεί ένα σημαντικό κεφάλαιο στον τομέα της πληροφορικής και της ασφάλειας των υπολογιστικών συστημάτων. Η ιστορία του RBAC ξεκινά από τη δεκαετία του '70, με προσπάθειες να απλοποιηθεί η διαχείριση πρόσβασης σε μεγάλα υπολογιστικά συστήματα, μέσω ανάθεσης ρόλων και αρμοδιοτήτων στους χρήστες . Ωστόσο, οι διαδικασίες αυτές ήταν αυθαίρετες και συχνά έπρεπε να επανασχεδιάζονται για κάθε νέο σύστημα, ανάλογα με τις ιδιαιτερότητες και τις ανάγκες του.

Η πρώτη σημαντική στροφή στην εξέλιξη του RBAC προήλθε το 1992, όταν οι ερευνητές David F. Ferraiolo και D. Richard Kuhn, στα πλαίσια του 15ου Εθνικού Συνεδρίου Υπολογιστικής Ασφάλειας, παρουσίασαν ένα σημαντικό άρθρο με τίτλο "Role-Based Access Controls" [3]. Στο άρθρο αυτό ανέπτυξαν το θεμελιώδες μοντέλο που χρησιμοποιούμε και σήμερα, το οποίο αποτελεί μια διαφορετική προσέγγιση στην μέχρι τότε επικρατέστερη μέθοδο ελέγχου πρόσβασης, Discretionary Access Control (DAC).

Η έρευνά τους επιχείρησε να αποδείξει ότι η χρήση των συστημάτων DAC, δεν αποτελούσε αξιόπιστη λύση στον έλεγχο πρόσβασης για αρκετούς εμπορικούς και κυβερνητικούς οργανισμούς. Οι ίδιοι πρότειναν την χρήση ενός non-discretionary ελέγχου πρόσβασης που επικεντρώνεται στους ρόλους, Role-Based Access Control. Στο RBAC συγκεκριμένα, οι χρήστες δεν μπορούν να μεταβιβάσουν δικαιώματα πρόσβασης σε άλλους κατά το δοκούν. Αυτό αποτελεί σημαντική διαφορά από το DAC, όπου οι χρήστες έχουν τη δυνατότητα να παρέχουν ή να ανακαλούν δικαιώματα πρόσβασης σε όσους διαχειρίζονται.

Τα επόμενα χρόνια, οι ερευνητές συνέχισαν να επεκτείνουν το RBAC, εξετάζοντας τα οφέλη του σε οικονομικό επίπεδο, καθορίζοντας ένα ενιαίο μοντέλο και προσδιορίζοντας τις μορφές διαχωρισμού των ευθυνών [4].

Συγκεκριμένα το 2000, κατά τη διάρκεια του Πέμπτου Συνεδρίου του Συλλόγου Μηχανημάτων Υπολογισμού (Fifth ACM Workshop on Role-Based Access Control) στο Βερολίνο, δημοσιεύτηκε ένα σημαντικό άρθρο με τίτλο "The NIST Model for Role-Based Access Control: Towards a Unified Standard" [5]. Οι συγγραφείς Ravi Sandhu, David Ferraiolo, και Richard Kuhn παρουσίασαν ένα ενοποιημένο μοντέλο, γνωστό ως NIST RBAC, που είχε ως στόχο την επίλυση της έλλειψης ενός κοινού προτύπου.

Το 2004, το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) αναγνώρισε και καθιέρωσε επίσημα το προαναφερόμενο μοντέλο, ως πρότυπο της βιομηχανίας (ANSI/INCITS 359-2004) [6], με αναθεώρηση το 2012 (INCITS 359-2012) [7], [8]. Η εφαρμογή του RBAC από εταιρείες όπως η IBM, Sybase, Secure Computing και Siemens από το 1994 αντιπροσωπεύει την ευρεία αποδοχή του μοντέλου.

Οικονομική Ανάλυση:

Τον Δεκέμβριο του 2010, ο μη κερδοσκοπικός οργανισμός RTI International (Research Triangle Institute) με έδρα τη Βόρεια Καρολίνα, δημοσίευσε μια έκθεση με τίτλο "Economic Analysis of Role-Based Access Control: Final Report" για το NIST. Η έκθεση αναλύει την οικονομική αξία του RBAC για τις επιχειρήσεις και για το εθνικό οικονομικό σύστημα, παρέχοντας ποσοτικά τα οικονομικά οφέλη του συστήματος για τις επιχειρήσεις που το υιοθετούν. Πιο συγκεκριμένα, αναγράφεται ότι η εφαρμογή των συστημάτων RBAC εξοικονόμησε περίπου 1,1 δισεκατομμύρια δολάρια στη βιομηχανία [9]. Σε πιο σύγχρονες έρευνες, το ποσό αυτό φαίνεται να έχει πολλαπλασιαστεί.

Σύγχρονες Προκλήσεις και Εξελίξεις:

Συνοψίζοντας, το RBAC αντιπροσωπεύει μια εξέλιξη στη διαχείριση της πρόσβασης, επιτρέποντας την καλύτερη οργάνωση και προστασία των ευαίσθητων δεδομένων σε εμπορικά, κυβερνητικά και βιομηχανικά περιβάλλοντα. Με την αύξηση των διαδικτυακών απειλών και ιδιαίτερα στην μετά COVID-19 εποχή, όπου όλο και περισσότερες ψηφιακές τεχνολογίες χρησιμοποιούνται για την επιτήρηση της ζωής των ανθρώπων [10], οι προγραμματιστές συνεχίζουν να εξελίσσουν τα συστήματα RBAC, καθώς η ανάγκη για προηγμένες λύσεις ασφαλείας και προστασίας των δικαιωμάτων απορρήτου είναι μεγαλύτερη από ποτέ. Τέλος, νέες τεχνολογίες όπως η τεχνητή νοημοσύνη, η αναγνώριση προτύπων, και η αυτοματοποίηση διαδικασιών προσθέτουν νέες δυνατότητες και προκλήσεις.

8 Τεχνολογίες που χρησιμοποιήθηκαν

Στην ενότητα αυτή θα αναλύσουμε τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος RBAC και θα αποσαφηνίσουμε κάποιες έννοιες, κρίσιμες για την κατανόηση του παρόντος εγγράφου.

8.1 Front-End

Το Front-End είναι η πρόσοψη μιας εφαρμογής η οποία είναι άμεσα προσβάσιμη από τους χρήστες και τους επιτρέπει να χρησιμοποιούν τα εργαλεία και τις υπηρεσίες που αυτή προσφέρει.

8.1.1 Typescript

Η γλώσσα προγραμματισμού TypeScript είναι αυτή που χρησιμοποιήθηκε για την υλοποίηση του frontend, του κομματιού της εφαρμογής με το οποίο αλληλεπιδρά ο χρήστης. Πρόκειται για μια αντικειμενοστραφή γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία Microsoft και αποτελεί ένα υπερσύνολο της JavaScript. Επιτρέπει στους προγραμματιστές να καθορίζουν τους τύπους των μεταβλητών και να δηλώνουν ρητά τους τύπους επιστροφής των συναρτήσεων, βελτιώνοντας την ποιότητα και τη δυνατότητα συντήρησης του κώδικα. Η TypeScript εξασφαλίζει έναν πιο αξιόπιστο και δομημένο κώδικα, με αποτέλεσμα μια ομαλή εμπειρία ανάπτυξης και τελικά, τη βελτίωση της συνολικής ποιότητας του λογισμικού.

```
/**
 * handlePermissionChange is called whenever the RBAC administrator
 * grants or denies permissions for a role.
 */
const handlePermissionChange = async (checked: boolean, permissionInfo:
↪  PermissionInfo) => {
    // add policy
    if (checked) {
        try {
            await axiosApiInstance.post('/api/permissions/', {
                newRole: selectedRole,
                newData: permissionInfo.resource,
                newPrivilege: permissionInfo.action
            })

            await getPermissionsForRole(selectedRole!)
            notification.success({message: 'Success'})
        } catch (e: any) {
            notification.error({message: e.response.data.message})
        }
    }
    // remove policy
    else {
        try {
            await axiosApiInstance.delete('/api/permissions/', {
                data: {
                    role: selectedRole,
                    data: permissionInfo.resource,
                    privilege: permissionInfo.action
                }
            })

            await getPermissionsForRole(selectedRole!)
            notification.success({message: 'Success'})
        } catch (e: any) {
            notification.error({message: e.response.data.message})
        }
    }

}
```

Listing 1: Παράδειγμα συνάρτησης γραμμένης σε Typescript

8.1.2 React

Η React είναι μία βιβλιοθήκη ανοικτού κώδικα της JavaScript, δημιουργήθηκε από την εταιρεία Facebook και χρησιμοποιείται για την ανάπτυξη διεπαφών χρήστη. Διευκολύνει τους προγραμματιστές να δημιουργούν δυναμικές web εφαρμογές μεγάλης κλίμακας, καθώς απαιτεί λιγότερο κώδικα και προσφέρει μεγάλη λειτουργικότητα, σε αντίθεση με την απλή JavaScript που γίνεται πολύπλοκη όταν πρόκειται για σύνθετες εφαρμογές.

Για να κατανοήσουμε αυτήν την λειτουργικότητα και τις ιδιαιτερότητες της React που την καθιστούν τόσο αποδοτική, πρέπει να αποσαφηνίσουμε κάποιες έννοιες όπως το JSX, το DOM, το Virtual DOM, την αρχιτεκτονική βασισμένη σε τμήματα και τα react hooks.

Το JSX (JavaScript Syntax Extension) είναι επέκταση σύνταξης της JavaScript και χρησιμοποιείται από την React για να περιγράψει πώς θα δείχνει ένα UI. Με την χρήση του JSX, γίνεται δυνατή η ένταξη HTML κώδικα σε ένα αρχείο JavaScript. Αυτό κάνει τον κώδικα πιο κατανοητό και διευκολύνει τον έλεγχο και την αποσφαλμάτωσή του.

Το DOM (Document Object Model) αποτελεί μια δομημένη αναπαράσταση των στοιχείων HTML που υπάρχουν σε μια ιστοσελίδα. Πρόκειται για μία δενδρική δομή δεδομένων αποτελούμενη από nodes, τα οποία αντιπροσωπεύουν τα στοιχεία (elements) ενός εγγράφου XML ή HTML. Κατασκευάζεται κάθε φορά που η σελίδα του φυλλομετρητή φορτώνει και επιτρέπει στα προγράμματα να διαβάζουν, να έχουν πρόσβαση και να αλλάζουν τη δομή, το στυλ και το περιεχόμενο ενός αρχείου. Το Virtual DOM (Εικονικό DOM) στην React είναι ένα αντίγραφο του πραγματικού που υπάρχει στη μνήμη και δημιουργείται κάθε φορά που υπάρχει μια αλλαγή στην κατάσταση της εφαρμογής [11]. Η κύρια διαφορά τους είναι ότι στο Virtual DOM οι αλλαγές δεν απεικονίζονται απευθείας στην οθόνη, στοιχείο που το καθιστά γρηγορότερο στην ενημέρωση. Κάθε φορά που η κατάσταση ενός στοιχείου της εφαρμογής αλλάζει, η δενδρική δομή που δημιουργείται, συγκρίνεται με την αντίστοιχη του προηγούμενου εικονικού DOM. Έτσι εντοπίζονται οι πιο πρόσφατες αλλαγές και ενημερώνονται μόνο τα απαιτούμενα στοιχεία στο πραγματικό DOM.

Το Virtual DOM βελτιώνει σε μεγάλο βαθμό την απόδοση της React. Κάθε φορά που η κατάσταση μιας συνιστώσας αλλάζει, η React ενημερώνει το Virtual DOM και όχι το πραγματικό. Στη συνέχεια, με τη διαδικασία της "σύγκρισης" (diffing), εντοπίζει τις αλλαγές και εφαρμόζει μόνο αυτές στο πραγματικό. Οι αλλαγές αυτές αποστέλλονται σε ομάδες και όχι κάθε μία ξεχωριστά, μέσω της διαδικασίας "batch updates". Κατά αυτόν τον τρόπο, μόνο τα διαφοροποιημένα στοιχεία του πραγματικού DOM γίνονται render στην σελίδα, βελτιώνοντας έτσι την συνολική απόδοση. Η παραπάνω διαδικασία ονομάζεται "Reconciliation" και εξηγεί γιατί η React, μαζί με το Virtual DOM, αποτελεί μια από τις πιο διαδεδομένες βιβλιοθήκες για την ανάπτυξη διεπαφών χρήστη και εφαρμογών παγκοσμίως.

Επιπλέον, η λογική της React ακολουθεί την αρχιτεκτονική βασισμένη σε τμήματα (component-based architecture), όπου τα διάφορα τμήματα της διεπαφής διασπώνται σε μικρότερα, επαναχρησιμοποιήσιμα components. Χωρίζονται σε δύο κατηγορίες ανάλογα με τη δομή τους, class ή functional, όμως και τα δύο έχουν την ίδια λειτουργικότητα. Τα components μπορούν να δεχτούν διάφορες παραμέτρους (props), διαχειρίζονται τη δική τους κατάσταση (state) και μπορούν να συνδέονται για την κατασκευή μεγαλύτερων, πιο σύνθετων διεπαφών.

Η React διαθέτει Hooks, τα οποία βοηθούν στην διαχείριση του state και των διάφορων χαρακτηριστικών των functional components. Είναι συναρτήσεις που επιτρέπουν την εύκολη διαχείριση των μεταβλητών του state. Τα Hooks που χρησιμοποιήθηκαν στην παρούσα εργασία, είναι το useState και το useEffect.

- Το useState χρησιμοποιείται μέσα σε ένα functional component για την δήλωση μιας μεταβλητής του state και μιας συνάρτησης, υπεύθυνης για την ανανέωσή της. Οποιαδήποτε αλλαγή στην μεταβλητή εκείνη, προκαλεί re-render ολόκληρου του component, χωρίς όμως να χαθούν οι υπόλοιπες τιμές που είναι αποθηκευμένες στο state [12].
- Το useEffect χρησιμοποιείται από τα functional components για να εκτελέσει μια λειτουργία (side effect) κάθε φορά που μια ορισμένη συνθήκη ισχύει. Δέχεται ως όρισμα έναν πίνακα μεταβλητών και εκτελεί συγκεκριμένες λειτουργίες (side effects), όταν κάποια από τις μεταβλητές αυτές αλλάξει [13].

Με τη δηλωτική της σύνταξη (declarative syntax) και την αποτελεσματικότητά της, η React έχει

γίνει δικαίως μια από τις δημοφιλέστερες επιλογές για τη δημιουργία διαδραστικών και δυναμικών web εφαρμογών.

```
const [roles, setRoles] = useState<UserEmail[]>();

const getRoles = async () => {
    //data has key and value
    //must be on .json format
    try {
        const res = await axiosApiInstance.get<Role[]>('/api/roles/')
        let data = res.data || []
        let selectOptions = data.map((r: Role) => (
            {label: r.role, value: r.role}
        ))

        setRoles(selectOptions)

        return selectOptions

    } catch (e: any) {
        notification.error({message: e.response.data.message})
        return []
    }
}


const onSaveUserRole = async (key: any, row: User & { index?: number |
↪  undefined }, newLineConfig: User & { index?: number | undefined }) => {
    try {
        await axiosApiInstance.put('/api/roles/', {
            ...row
        })
        notification.success({message: 'Success'})
    } catch (e: any) {
        notification.error({message: e.response.data.message})
        refUsersTable.current?.reload()
    }
}

// It's called everytime the values inside the array change (the array at the
↪  end, after the function)
// Now that the array is empty, useEffect is called only once, at the start
useEffect(() => {
    getRoles()
}, [])
```

Listing 2: Παράδειγμα κώδικα React

8.1.3 Ant Design

Το Ant Design είναι ένα component library με έτοιμα στοιχεία (components), που εξυπηρετεί την δημιουργία σύγχρονων διαδικτυακών διεπαφών χρήστη και εφαρμογών.

Έχει σχεδιαστεί για να παρέχει έναν δομημένο, σταθερό και κομψό σχεδιασμό. Αποτελείται από ένα σύνολο προκαθορισμένων και εύκολα παραμετροποιήσιμων στοιχείων (components), καθιστώντας το εύκολο στη χρήση. Συνοδεύεται από ένα εκτενές documentation και διατίθεται για πολλές γλώσσες προγραμματισμού, γεγονός που το καθιστά ένα από τα κορυφαία εργαλεία σχεδιασμού frontend παγκοσμίως. Τα βασικότερα χαρακτηριστικά του Ant Design περιλαμβάνουν:

Consistent Design (Σταθερός Σχεδιασμός): Στόχος του Ant Design είναι να παρέχει συνοχή και ομοιομορφία στον σχεδιασμό των διαφόρων τμημάτων της εφαρμογής. Αυτό σημαίνει ότι τα κουμπιά, οι λίστες, τα γραφήματα, οι φόρμες, τα μενού και τα υπόλοιπα components, να έχουν εναρμονισμένη εμφάνιση και συμπεριφορά σε όλη την εφαρμογή. Η συνοχή αυτή εξασφαλίζει μια καλύτερη εμπειρία πλοήγησης για τους χρήστες, καθώς οι ίδιοι μπορούν να προσδοκούν ότι τα στοιχεία θα συμπεριφέρονται και θα εμφανίζονται με παρόμοιο τρόπο.

Pre-designed Components (Προκαθορισμένα Στοιχεία): Το Ant Design παρέχει ένα εκτενές σύνολο προκαθορισμένων και εύκολα παραμετροποιήσιμων στοιχείων (components), όπως κουμπιά, πίνακες, φόρμες, γραφήματα, μενού, εικονίδια κ.ά., που μπορούν να χρησιμοποιηθούν αυτούσια για την ανάπτυξη μιας εφαρμογής. Τα στοιχεία αυτά είναι κομψά σχεδιασμένα, με σύγχρονη αισθητική και ενσωματώνονται εύκολα στο έργο του προγραμματιστή, εξοικονομώντας χρόνο και προσπάθεια στη διαδικασία ανάπτυξης κώδικα.

Responsive Design (Σχεδιασμός που ανταποκρίνεται στις απαιτήσεις του εκάστοτε συστήματος): Οι εφαρμογές που έχουν υλοποιηθεί με τη βοήθεια του Ant Design, είναι σε θέση να προσαρμόζονται σε διαφορετικές συσκευές, ανάλογα με τα χαρακτηριστικά της εκάστοτε οθόνης. Αυτό διασφαλίζει ότι όλοι οι χρήστες θα έχουν την ίδια ευχάριστη εμπειρία, ανεξάρτητα από την συσκευή και το μέγεθος της οθόνης που χρησιμοποιούν, χωρίς να απαιτείται ξεχωριστός σχεδιασμός για κάθε περίπτωση.

Συμπερασματικά, είναι σαφές ότι το Ant Design αποτελεί ένα ισχυρό εργαλείο για την ανάπτυξη διαδικτυακών εφαρμογών, με έμφαση στην σύγχρονη αισθητική και τον λειτουργικό σχεδιασμό.

```
return (
    <PageContainer>
        <Row>
            <Col span={24}>
                <ConfigProvider locale={enUSIntl}>
                    <EditableProTable<User>
                        request={async (params, sort, filter) => {
                            try {
                                const res = await
                                ↪ axiosApiInstance.get('/api/users/', {
                                    params: {
                                        keyword: params.keyword
                                    }
                                })
                                return {data: res.data, success: true, total:
                                ↪ res.data.length}
                            } catch (e: any) {
                                notification.error({message:
                                ↪ e.response.data.message})
                                return {data: [], success: false, total: 0}
                            }
                        }}
                        actionRef={refUsersTable}
                        columns={columns}
                        rowKey="id"
                        controlled={true}
                        value={dataSource}
                        onChange={(dataSource) => setDataSource(dataSource as
                        ↪ any)}
                        recordCreatorProps={false}
                        pagination={{pageSize: 8, hideOnSinglePage: true,
                        ↪ showQuickJumper: true}}
                        editable={{
                            type: 'single',
                            editableKeys: editableKeys,
                            actionRender: (row, config, defaultDoms) => {
                                return [defaultDoms.save, defaultDoms.cancel];
                            },
                            onChange: (editableKeys) =>
                            ↪ setEditableKeys(editableKeys),
                            onSave: onSaveUserRole,
                            onCancel: async (key, record, originRow, newRow) =>
                                refUsersTable.current?.reload(),
                            onlyOneLineEditorAlertMessage: 'Only one line can be
                            ↪ edited!',
                        }}
                        options={{
                            search: {placeholder: 'Please enter keyword',
                            ↪ allowClear: true},
                        }}
                        bordered
                    />
                </ConfigProvider>
            </Col>
        </Row>
    </PageContainer>
);
```

Listing 3: Παράδειγμα από components του Ant Design

8.1.4 Axios

Το Axios είναι μια βιβλιοθήκη της JavaScript που απλοποιεί τη διαδικασία των HTTP requests (αιτημάτων) τόσο σε client-side, όσο και σε server-side εφαρμογές [14].

Το Axios διευκολύνει την αποστολή αιτημάτων (requests) και τη διαχείριση των απαντήσεων (responses) στις πλατφόρμες. Επιπλέον, παρέχει ένα εύχρηστο API που καθιστά απλή την εκτέλεση κοινών λειτουργιών HTTP όπως GET, POST, PUT και DELETE [15]. Ακόμη, μπορεί εύκολα να ρυθμιστεί ώστε να λειτουργήσει με μια ευρεία γκάμα από web APIs.

Εκτός από τα παραπάνω, το Axios έχει τη δυνατότητα προσθήκης "interceptors" στην κλήση κάποιου API [16]. Με αυτόν τον τρόπο, μπορεί να εκτελέσει διάφορες εντολές και να πραγματοποιήσει αλλαγές στο request, πριν εκείνο σταλεί στο API. Ένα από τα βασικότερα χαρακτηριστικά των "interceptors" είναι ότι διευκολύνουν σε μεγάλο βαθμό την διαδικασία του error handling, σε περίπτωση που υπάρξει οποιοδήποτε σφάλμα κατά την κλήση του API [17]. Επιπλέον, το Axios διαθέτει ενσωματωμένη υποστήριξη για την ακύρωση αιτημάτων (request cancellation) [18] και την αυτόματη ανάλυση αρχείων JSON (automatic JSON parsing).

Γίνεται, επομένως, εύκολα αντιληπτό ότι το Axios είναι μια πανίσχυρη βιβλιοθήκη και δικαίως αποτελεί μια από τις κορυφαίες επιλογές για τη διαχείριση αιτημάτων σε web εφαρμογές.

```
import axiosApiInstance from "../api/axiosClient";

// GET Request
const res = await axiosApiInstance.get('/api/users/emails')
let data = res.data || []
let selectOptions = data.map((email: string) => (
    {label: email, value: email}
))

// POST Request
await axiosApiInstance.post<Role>('/api/roles/', {
                ...role
            })

// PUT Request
await axiosApiInstance.put('/api/roles/', {
                ...row
            })

// DELETE Request
await axiosApiInstance.delete('/api/firebase/' + user.id);
```

Listing 4: Παραδείγματα HTTP requests μέσω του Axios

8.2 Back-End

Το Back-End είναι το σύστημα που βρίσκεται στο πίσω μέρος μιας εφαρμογής και είναι υπεύθυνο για την ομαλή λειτουργία της. Εκεί γίνεται η διαχείριση των δεδομένων και η πραγματοποίηση όλων των ενεργειών του συστήματος.

8.2.1 Go Programming Language

Η Go είναι μια γλώσσα προγραμματισμού διαδικαστικού (procedural) τύπου. Αναπτύχθηκε το 2007 από τους Robert Griesemer, Rob Pike και Ken Thompson στην Google, αλλά κυκλοφόρησε ως γλώσσα ανοιχτού κώδικα το 2009.

Για την καλύτερη κατανόησή της, θα πρέπει πρώτα να δοθεί ένας σύντομος ορισμός του διαδικαστικού προγραμματισμού (Procedural Programming) και της βασικής διαφοράς του από τον αντικειμενοστραφή (Object Oriented Programming).

Ο διαδικαστικός προγραμματισμός ορίζεται ως ένα προγραμματιστικό μοντέλο που προκύπτει από τον δομημένο προγραμματισμό (structured programming) και βασίζεται στην κλήση διαδικασιών (procedures). Οι διαδικασίες αυτές, γνωστές και ως ρουτίνες, υπορουτίνες ή συναρτήσεις, αποτελούνται από μια σειρά υπολογιστικών βημάτων, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο.

Στον διαδικαστικό προγραμματισμό, για την επίλυση ενός πολύπλοκου προβλήματος, ακολουθείται μια προσέγγιση από πάνω προς τα κάτω (top-down), αναλύοντας αρχικά τα πιο σύνθετα κομμάτια του προβλήματος, τα οποία διαιρούνται σταδιακά, σε μικρότερα, απλούστερα υποπροβλήματα. Η συγκεκριμένη λογική επίλυσης προβλημάτων, αποτελεί και την βασική διαφορά του διαδικαστικού από τον αντικειμενοστραφή προγραμματισμού, όπου το πρόγραμμα διαιρείται σε μικρότερα τμήματα που ονομάζονται αντικείμενα, ακολουθώντας μια προσέγγιση από τα κάτω προς τα πάνω (bottom-up) [19].

Η σύνταξη της GO έχει βασιστεί σε εκείνη της γλώσσας C, η οποία θεωρείται προπάτοράς της. Τα αρχεία ενός προγράμματος γραμμένου σε Go, ομαδοποιούνται σε πακέτα (packages), για την αποτελεσματικότερη διαχείριση των dependencies. Σημαντική επιρροή για την τμηματοποίηση των προγραμμάτων σε πακέτα (packages) και τον τρόπο δήλωσης των μεταβλητών, έχουν οι γλώσσες Pascal, Modula και Oberon [20]. Η λογική πίσω από τις διεργασίες της Go, βασίστηκε στην θεωρία του Tony Hoare για τις "Επικοινωνούσες Ακολουθιακές Διεργασίες", όπως εκείνη παρουσιάζεται στο ομώνυμο σύγγραμμά του "Communicating Sequential Processes" [21]. Η Go αποτελεί μια νεότερη εκδοχή των προαναφερθέντων, παραδοσιακών γλωσσών προγραμματισμού, αφενός μεν διατηρώντας τα καλά τους στοιχεία, αφετέρου δε, εντάσσοντας μια σύγχρονη και αποτελεσματική προσέγγιση για πολλούς τομείς.

Παρόλο που η Go διαθέτει τύπους (types), μεθόδους (methods) και επιτρέπει ένα αντικειμενοστραφές στυλ προγραμματισμού, δεν αποτελεί μια object-oriented γλώσσα, καθώς δεν υπάρχει ιεραρχία στους τύπους [22].

Για να κατανοήσουμε την έννοια της ιεραρχίας των τύπων (type hierarchy), θα πάρουμε ως παράδειγμα την Java, μια από τις πλέον διαδεδομένες αντικειμενοστραφείς γλώσσες προγραμματισμού, αναλύοντας τον τρόπο με τον οποίο ορίζει τους τύπους των μεταβλητών της. Πιο συγκεκριμένα στην Java, ο constructor της κλάσης που χρησιμοποιείται για τη δημιουργία ενός αντικειμένου, καθορίζει και τον τύπο του. Ωστόσο, στα αντικείμενα μιας κλάσης (subclass) που κληρονομεί μία δεύτερη κλάση (base class) ή χρησιμοποιεί ένα interface, ανατίθεται ο τύπος της βασικής κλάσης ή του interface, αντίστοιχα [23]. Έτσι υπάρχει μια ιεραρχία στους τύπους. Με την ίδια λογική που τοποθετούμε τα αντικείμενα του πραγματικού κόσμου σε κατηγορίες, έτσι κατηγοριοποιούμε και τα προγραμματιστικά αντικείμενα σε κλάσεις, όπου κάποιες είναι πιο γενικές από άλλες.

Η έννοια του interface στην Go είναι διαφορετική από τις υπόλοιπες γλώσσες προγραμματισμού. Στην Go συγκεκριμένα, ένα interface αποτελεί έναν προσαρμοσμένο τύπο (custom type), ο οποίος χρησιμοποιείται για τον ορισμό μίας ή περισσότερων μεθόδων του προγράμματος. Η έννοια του interface είναι αφηρημένη και έτσι δεν δίνεται η δυνατότητα δημιουργίας ενός instance του interface. Επιτρέπεται όμως στους προγραμματιστές να ορίζουν μεταβλητές τύπου interface, οι οποίες περιέχουν τις ορισμένες συναρτήσεις του και μπορεί να τους ανατεθεί μια concrete type τιμή [24]. Ως concrete type ορίζονται όλοι οι τύποι δεδομένων που δεν είναι interfaces [25]. Επομένως, ένα interface στην

Go μπορεί να λειτουργήσει ως ένα σύνολο μεθόδων ή ως ένα custom type.

Η Go υποστηρίζει την λογική του ταυτόχρονου προγραμματισμού (Concurrent Programming). Ως Concurrent Programming ορίζεται ο τρόπος σχεδίασης και συγγραφής προγραμμάτων που μπορούν να εκτελούν πολλαπλές διεργασίες ταυτόχρονα [26]. Ο συγχρονισμός στην Go επιτυγχάνεται μέσω των goroutines. Οι ρουτίνες είναι lightweight threads, τόσο ελαφριά που σε ένα πρόγραμμα μπορεί να τρέξουν χιλιάδες ταυτόχρονα, χωρίς να επηρεαστεί σημαντικά η απόδοση του κώδικα [27]. Τα goroutines ανταλλάσσουν πληροφορίες και επικοινωνούν μεταξύ τους μέσω των channels, τα οποία αποτελούν κι αυτά με τη σειρά τους built-in χαρακτηριστικά της Go [28]. Τα προαναφερθέντα χαρακτηριστικά της Go, την καθιστούν μια ιδανική επιλογή για την κατασκευή επεκτάσιμων προγραμμάτων υψηλής απόδοσης, με στόχο την επίλυση σύνθετων υπολογιστικών προβλημάτων.

Ένα από τα σημαντικότερα χαρακτηριστικά της Go είναι η ενσωματωμένη συλλογή σκουπιδιών (garbage collection), η οποία συμβάλλει στη διαχείριση της διαθέσιμης μνήμης (memory allocation) κατά την εκτέλεση προγραμμάτων. Σε αντίθεση με άλλες γλώσσες προγραμματισμού, όπως η C, στις οποίες απαιτείται η χειροκίνητη διαχείριση μνήμης από τους προγραμματιστές, στην Go γίνεται αυτόματα μέσω του Garbage Collector. Ο μηχανισμός αυτός αναλαμβάνει την απελευθέρωση της μνήμης που δεν χρειάζεται πλέον, εξαλείφοντας έτσι τον κίνδυνο για memory leaks [29].

Συμπερασματικά, η Go ως γλώσσα προγραμματισμού, συνδυάζει απλότητα στον σχεδιασμό και αποδοτικότητα στην εκτέλεση. Αυτοί αποτελούν δύο από τους κυριότερους λόγους της επιλογής της Go, ως βασικής γλώσσας ανάπτυξης του συστήματος RBAC.

```go
//
// Get all firebase users associated with given employee
//
func GetEmployeeUsers() gin.HandlerFunc {
        return func(c *gin.Context) {
                var employee models.Employee
                err := c.ShouldBindUri(&employee)
                if err != nil {
                        c.AbortWithError(http.StatusInternalServerError, err)
                        log.Println(err)
                        return
                }

                // Connect to RBAC Database (for gorm queries)
                database, err := db.ConnectToRBACGorm()
                if err != nil {
                        c.AbortWithError(http.StatusInternalServerError, err)
                        log.Println(err)
                        return
                }
                defer db.CloseDBConnectionGorm(database)

                // Initialize return data
                var associatedUsers []struct {
                        ID    string `json:"id"`
                        Email string `json:"email"`
                }

                // Get all users associated with this employee
                err = database.Debug().Table("users").
                        Joins("JOIN employees ON users.employee_id =
                        ↪  employees.id").
                        Select("users.id, users.email").
                        Where("employees.id = ?", employee.Id).
                        Find(&associatedUsers).Error
                if err != nil {
                        c.AbortWithError(http.StatusInternalServerError, err)
                        log.Println(err)
                        return
                }

                c.JSON(http.StatusOK, associatedUsers)
        }
}
```

Listing 5: Παράδειγμα συνάρτησης γραμμένης σε Go

8.2.2 Gin Framework

Το Gin είναι ένα HTTP web framework γραμμένο σε Go. Διαθέτει ένα API παρόμοιο με αυτό του Martini, αλλά με επίδοση που μπορεί να είναι μέχρι και 40 φορές ταχύτερη [30]. Για να κατανοήσουμε καλύτερα το εργαλείο Gin, θα πρέπει να δώσουμε έναν σύντομο ορισμό του Martini και να εξηγήσουμε τις βασικές του λειτουργίες.

Το Martini είναι ένα πακέτο (package) της Go και αποτελεί ισχυρό εργαλείο για την ανάπτυξη γρήγορων και ευέλικτων web εφαρμογών και υπηρεσιών. Η βασική του ιδέα είναι η επιτάχυνση της διαδικασίας ανάπτυξης, μέσω της παροχής μιας σειράς εργαλείων (handlers, routes, services) και

δυνατοτήτων που χρησιμεύουν στην οργάνωση και διαχείριση των εφαρμογών [31].

Το Martini είναι γνωστό για τον τμηματικό (modular) και μη επεμβατικό (non-intrusive), στις λειτουργίες της εφαρμογής, σχεδιασμό του. Non-instructive αφενός, διότι ο κώδικας της εφαρμογής δεν εξαρτάται άμεσα από τα αντικείμενα του framework και modular αφετέρου, καθώς αποτελείται από μικρότερα, αυτόνομα τμήματα (modules), ειδικά σχεδιασμένα να εκτελούν μια αυστηρά καθορισμένη λειτουργία το καθένα.

Το Martini και κατ' επέκταση το Gin, είναι εύκολα στη χρήση και συμβατά με πολλά άλλα πακέτα της Go. Παρέχουν μία πληθώρα από χρήσιμους handlers, routes και middlewares. Οι handlers είναι συναρτήσεις που καλούνται από το πρόγραμμα, μέσω των routes [32]. Τα routes είναι HTTP μέθοδοι που περιλαμβάνουν έναν ή περισσότερους handlers. Λειτουργούν ως Endpoints με URLs στο backend, καλούνται από το frontend και εκτελούν τους handlers που τους αντιστοιχούν [33]. Τέλος, τα middlewares είναι συναρτήσεις που λειτουργούν συμπληρωματικά των routes και εκτελούνται πριν την κλήση των αντίστοιχων handlers [34].

Στην παρούσα πτυχιακή εργασία, με την βοήθεια του Gin, υλοποιήθηκε ένα restful API με ομαδοποιημένα routes στο backend, όπου κάθε ένα από αυτά καλεί μια handler μέθοδο, υπεύθυνη για την εκτέλεση μιας ενέργειας στη βάση δεδομένων. Κάθε route, περιλαμβάνει επίσης κι ένα middleware, υπεύθυνο για το authorization, με σκοπό τον έλεγχο της άδειας του χρήστη να εκτελέσει την παραπάνω ενέργεια, ανάλογα με τον ρόλο του.

```
import (
        "github.com/gin-contrib/cors"
        "github.com/gin-gonic/contrib/static"
        "github.com/gin-gonic/gin"
)

//SetupRoutes : all the routes are defined here
func SetupRoutes(db *gorm.DB) {
    httpRouter := gin.Default()
    //CORS
    cors_conf := cors.DefaultConfig()
    cors_conf.AllowAllOrigins = true
    cors_conf.AllowCredentials = true
    cors_conf.AddAllowHeaders("authorization")
    cors_conf.AddAllowHeaders("Access-Control-Allow-Credentials")
    cors_conf.AddAllowHeaders("Access-Control-Allow-Origin")
    cors_conf.AddAllowHeaders("accept")
    httpRouter.Use(cors.New(cors_conf))
    httpRouter.MaxMultipartMemory = 1024 << 20

    // set db & firebase auth to gin context with a middleware to all incoming
    ↪  request
    httpRouter.Use(func(c *gin.Context) {
        c.Set("db", db)
        c.Set("firebaseAuth", firebaseAuth)
    })
    apiRoutes := httpRouter.Group("/api", middleware.AuthMiddleware)

    // CUSTOMERS ROUTES
    customersProtectedRoutes := apiRoutes.Group("/customers")
    {
            customersProtectedRoutes.GET("/", middleware.Authorize("rbac::data",
            ↪  "read", enforcer), handlers.GetAllCustomers())
            customersProtectedRoutes.POST("/", middleware.Authorize("rbac::data",
            ↪  "write", enforcer), handlers.AddCustomer())
            customersProtectedRoutes.DELETE("/:id",
            ↪  middleware.Authorize("rbac::data", "write", enforcer),
            ↪  handlers.DeleteCustomer(enforcer))
            customersProtectedRoutes.PUT("/", middleware.Authorize("rbac::data",
            ↪  "write", enforcer), handlers.UpdateCustomer())

            associations := customersProtectedRoutes.Group("/associations")
            {
                    associations.GET("/:id", middleware.Authorize("rbac::data",
                    ↪  "read", enforcer), handlers.GetCustomerUsers(enforcer))
                    associations.PUT("/", middleware.Authorize("rbac::data",
                    ↪  "read", enforcer),
                    ↪  handlers.ToggleCustomerUserAccess(enforcer))
                    associations.POST("/", middleware.Authorize("rbac::data",
                    ↪  "write", enforcer),
                    ↪  handlers.AddCustomerUserAssociation(enforcer))
                    associations.DELETE("/", middleware.Authorize("rbac::data",
                    ↪  "write", enforcer),
                    ↪  handlers.DeleteCustomerUserAssociation(enforcer))
            }
    }
}
```

Listing 6: Παραδείγματα από routes του back-end, με τη χρήση του Gin

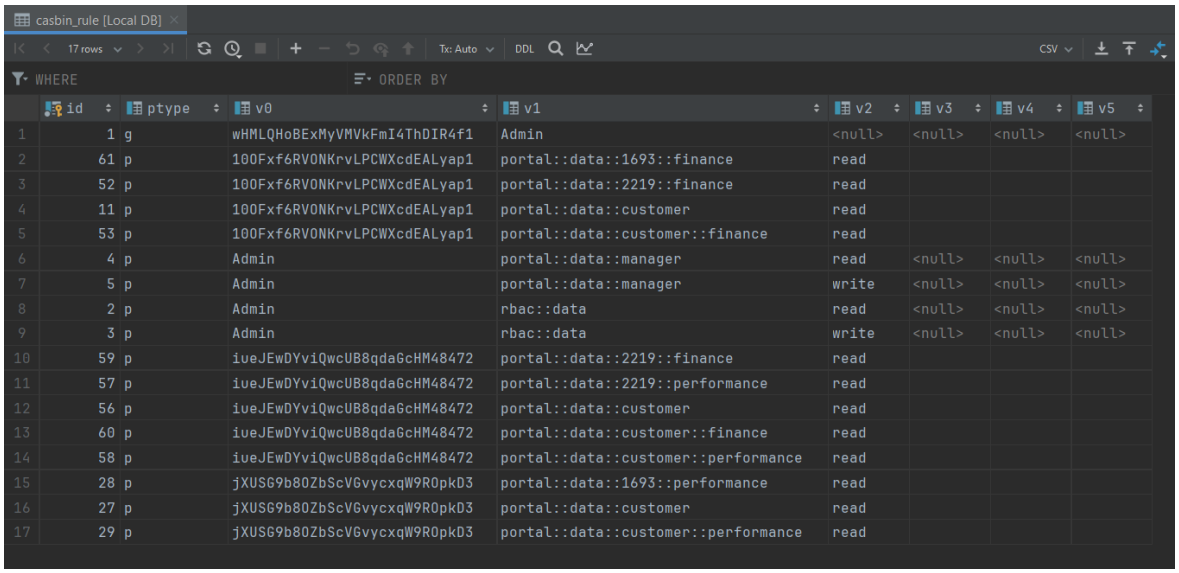
8.2.3 Casbin

Το Casbin είναι μια ισχυρή και αποτελεσματική open-source βιβλιοθήκη που χρησιμοποιείται για το authorization πληροφοριακών συστημάτων. Υποστηρίζει διάφορα μοντέλα ελέγχου πρόσβασης για την εξουσιοδότηση των χρηστών σε διάφορα επίπεδα.

Η λογική του Casbin είναι αρκετά απλή. Ορίζουμε ένα σύνολο κανόνων, με τους οποίους καθορίζεται ποιος χρήστης ή οντότητα, έχει πρόσβαση σε ποιο αντικείμενο και ποιες ενέργειες επιτρέπεται να εκτελέσει. Αυτό το μοντέλο subject, object, action, ονομάζεται "κλασική ροή" και αποτελεί την πιο συνηθισμένη περίπτωση χρήσης του Casbin [35].

Στην παρούσα πτυχιακή εργασία, οι κανόνες πρόσβασης (policies) ορίζονται στον πίνακα "`casbin_rule`" της βάσης δεδομένων, όπου περιέχονται όλα τα υποκείμενα (subjects), αντικείμενα (objects) και επιθυμητές ενέργειες (actions) του συστήματος. Στο αρχείο "`rbac_model.conf`", ορίζεται η μορφή των κλήσεων (requests), των κανόνων πρόσβασης (policies) και των ρόλων (roles), το αποτέλεσμα (effect) ενός έγκυρου policy και οι συνθήκες εξουσιοδότησης (matchers) του συστήματος. Το Casbin παρέχει έναν Enforcer για την επικύρωση των εισερχόμενων requests, βασισμένο στον πίνακα "`casbin_rule`" και στο αρχείο "`rbac_model.conf`".

Γίνεται εύκολα αντιληπτό λοιπόν, ότι το Casbin αποτελεί βασικό χαρακτηριστικό της παρούσας πτυχιακής εργασίας. Μέσω αυτού, ορίστηκαν οι κανόνες που καθορίζουν ποιος χρήστης έχει πρόσβαση, σε ποια τμήματα των εταιρικών εφαρμογών, ανάλογα με τον ρόλο που διαθέτει. Επιπλέον, με τη χρήση της συνάρτησης Authorize(), ελέγχεται αν ο χρήστης είναι εξουσιοδοτημένος να εκτελέσει κάποια ενέργεια, πάνω σε ένα αντικείμενο του συστήματος.



casbin_rule [Local DB]

17 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	id	ptype	v0	v1	v2	v3	v4	v5
1	1	g	wHMLQHoBExMyVMVkFmI4ThDIR4f1	Admin	<null>	<null>	<null>	<null>
2	61	p	10OFxf6RVONKrvLPCWXcdEALyap1	portal::data::1693::finance	read			
3	52	p	10OFxf6RVONKrvLPCWXcdEALyap1	portal::data::2219::finance	read			
4	11	p	10OFxf6RVONKrvLPCWXcdEALyap1	portal::data::customer	read			
5	53	p	10OFxf6RVONKrvLPCWXcdEALyap1	portal::data::customer::finance	read			
6	4	p	Admin	portal::data::manager	read	<null>	<null>	<null>
7	5	p	Admin	portal::data::manager	write	<null>	<null>	<null>
8	2	p	Admin	rbac::data	read	<null>	<null>	<null>
9	3	p	Admin	rbac::data	write	<null>	<null>	<null>
10	59	p	iueJEwDYviQwcUB8qdaGcHM48472	portal::data::2219::finance	read			
11	57	p	iueJEwDYviQwcUB8qdaGcHM48472	portal::data::2219::performance	read			
12	56	p	iueJEwDYviQwcUB8qdaGcHM48472	portal::data::customer	read			
13	60	p	iueJEwDYviQwcUB8qdaGcHM48472	portal::data::customer::finance	read			
14	58	p	iueJEwDYviQwcUB8qdaGcHM48472	portal::data::customer::performance	read			
15	28	p	jXUSG9b8OZbScVGvycxqW9ROpkD3	portal::data::1693::performance	read			
16	27	p	jXUSG9b8OZbScVGvycxqW9ROpkD3	portal::data::customer	read			
17	29	p	jXUSG9b8OZbScVGvycxqW9ROpkD3	portal::data::customer::performance	read			

Figure 1: Πίνακας `casbin_rule` στην βάση δεδομένων

```
// Remove financial policy from user, if exists
if enforcer.HasPolicy(user.Id, fmt.Sprintf("portal::data::%d::finance",
↪  customer.Id), "read") {
        _, err = enforcer.RemovePolicy(user.Id,
        ↪  fmt.Sprintf("portal::data::%d::finance", customer.Id), "read")
        if err != nil {
                c.AbortWithError(http.StatusInternalServerError, err)
                log.Println(err)
                return
        }
}
// Remove performance policy from user, if exists
if enforcer.HasPolicy(user.Id, fmt.Sprintf("portal::data::%d::performance",
↪  customer.Id), "read") {
        _, err = enforcer.RemovePolicy(user.Id,
        ↪  fmt.Sprintf("portal::data::%d::performance", customer.Id), "read")
        if err != nil {
                c.AbortWithError(http.StatusInternalServerError, err)
                log.Println(err)
                return
        }
}
```

Listing 7: Παράδειγματα queries στην βάση, με τη χρήση του Casbin Enforcer

8.2.4 GORM (Go Object Relational Mapping)

GORM ονομάζεται η βιβλιοθήκη ORM (Object Relational Mapping), γραμμένη για την γλώσσα προγραμματισμού Go [36]. Η ORM είναι μια τεχνική που χρησιμοποιείται για τη δημιουργία μιας "γέφυρας" μεταξύ αντικειμενοστραφών προγραμμάτων και σχεσιακών βάσεων δεδομένων.

Οι αντικειμενοστραφείς γλώσσες προγραμματισμού (object-oriented programming languages) αλληλεπιδρούν με τις βάσεις δεδομένων, εκτελώντας CRUD εντολές όπως δημιουργία (CREATE), ανάγνωση (READ), ενημέρωση (UPDATE), και διαγραφή (DELETE) δεδομένων. Οι συγκεκριμένες εντολές, εκτελούν queries γραμμένα σε γλώσσα SQL.

Η GORM και κατ' επέκταση η ORM, αποδίδει τον κώδικα SQL που απαιτείται για τη διαχείριση των δεδομένων της βάσης, σε εντολές Go. Ο προγραμματιστής διαχειρίζεται τις οντότητες της βάσης δεδομένων, μέσω αντικειμένων της Go. Οι τύποι των αντικειμένων είναι ειδικά διαμορφωμένες δομές (structures), γραμμένες σε Go και υπάρχει μια ένα-προς-ένα αντιστοιχία μεταξύ των μοντέλων αυτών και των πινάκων της βάσης. Με αυτήν την λογική, η διαχείριση των δεδομένων της βάσης γίνεται με κώδικα γραμμένο σε Go, χωρίς να απαιτείται η συγγραφή εντολών SQL.

Η GORM υποστηρίζει διάφορους τύπους συσχετίσεων (Associations) μεταξύ των μοντέλων, που αντιστοιχούν στις σχέσεις μεταξύ των πινάκων της βάσης. Κάποια χαρακτηριστικά παραδείγματα είναι τα Has One, Has Many, Belongs To, Many To Many, Polymorphism και Single-table inheritance. Η GORM προσφέρει επίσης διάφορα βοηθητικά hooks, τα οποία εκτελούνται συμπληρωματικά με τις CRUD εντολές, καθώς και δύο πολύ χρήσιμες λειτουργίες που διασφαλίζουν την ακεραιότητα των δεδομένων, τις συναλλαγές (transactions) και τα save points της βάσης.

Οι συναλλαγές (Transactions) της GORM, εξασφαλίσουν τη συνέπεια και την ακεραιότητα των δεδομένων. Μια συναλλαγή είναι μια ακολουθία λειτουργιών που εκτελούνται ως ένα μοναδικό σύνολο εργασιών. Τα χαρακτηριστικά ACID (Atomicity, Consistency, Isolation, Durability) μιας συναλλαγής, εξασφαλίζουν ότι είτε όλες οι λειτουργίες της συναλλαγής θα εκτελεστούν με επιτυχία, είτε καμία από αυτές. Κατ' αυτόν τον τρόπο, εγγυάται η συνεκτικότητα της βάσης δεδομένων, ακόμη και στην περίπτωση σφάλματος κατά την εκτέλεση της συναλλαγής. Οι συναλλαγές είναι ιδιαίτερα σημαντικές σε περιπτώσεις όπου πολλές ταυτόχρονες λειτουργίες μπορεί να έχουν πρόσβαση στα ίδια δεδομένα, όπως σε μια web εφαρμογή ή σε μια πλατφόρμα e-commerce. Σε τέτοιες περιπτώσεις, οι συναλλαγές παρέχουν έναν τρόπο διαχείρισης της ταυτόχρονης πρόσβασης στη βάση δεδομένων και αποτρέπουν τις συγκρούσεις (conflicts) που μπορεί να οδηγήσουν σε αλλοίωση των δεδομένων. Συνολικά, οι συναλλαγές είναι ένα ισχυρό εργαλείο για τη διατήρηση της ακεραιότητας των δεδομένων και είναι ένα σημαντικό στοιχείο κάθε ανθεκτικής εφαρμογής.

Τα σημεία αποθήκευσης της βάσης δεδομένων (Save Points) στην GORM, είναι ένας μηχανισμός που επιτρέπει την δημιουργία ενδιάμεσων σημείων ελέγχου εντός μιας συναλλαγής. Μπορούν να χρησιμοποιηθούν για να αναιρέσουν ένα μέρος της συναλλαγής αυτής, αντί να αναιρεθεί ολόκληρη. Τα σημεία αποθήκευσης μπορούν να είναι χρήσιμα σε περιπτώσεις όπου εκτελείται ένα πολύπλοκο transaction, με πολλαπλά στάδια και πρέπει να αναιρεθεί ένα εξ' αυτών, χωρίς να χρειαστεί να τρέξει ολόκληρη η συναλλαγή από την αρχή. Για παράδειγμα, εάν μια λειτουργία της βάσης δεδομένων αποτύχει στη μέση ενός transaction, η χρήση σημείων αποθήκευσης θα επιτρέψει στους προγραμματιστές να αναιρέσουν τις αλλαγές από το πιο πρόσφατο save point και να επαναληφθεί μόνο η αποτυχημένη λειτουργία, αντί ολόκληρης της συναλλαγής. Τα σημεία αποθήκευσης δημιουργούνται και διαγράφονται μέσω SQL εντολών, εντός ενός transaction και υποστηρίζονται από τις περισσότερες σύγχρονες σχεσιακές βάσεις δεδομένων. Συνολικά, η χρήση των σημείων αποθήκευσης παρέχει στους προγραμματιστές μεγαλύτερη ευελιξία και έλεγχο επί των συναλλαγών, κάνοντας ευκολότερη τη διαχείριση των σφαλμάτων και τη διατήρηση της ακεραιότητας των δεδομένων.

Μία ακόμα βασική λειτουργία της GORM που αξίζει να αναφερθεί, είναι το auto migration των πινάκων της σχεσιακής βάσης δεδομένων, με βάση τα αντίστοιχα μοντέλα γραμμένα σε GO. Το auto migration αποτελεί ένα αποτελεσματικό εργαλείο για τη διαχείριση του σχήματος (schema) της βάσης. Με τη χρήση αυτής της λειτουργίας, η GORM αναλαμβάνει την αυτόματη δημιουργία και ενημέρωση των πινάκων, ώστε να αντικατοπτρίζουν τις αλλαγές στα μοντέλα δεδομένων (Models) της εφαρμογής. Αυτό εξασφαλίζει ότι οι αλλαγές στη δομή της βάσης δεδομένων πραγματοποιούνται αυτόματα, χωρίς να υπάρξει ανάγκη χειροκίνητων επεμβάσεων από τους προγραμματιστές, διατηρώντας έτσι τη συνέπεια και την ακεραιότητα των δεδομένων [37].

Τέλος, το eager loading αποτελεί μία από τις κυριότερες λειτουργίες της GORM. Στο πλαίσιο μιας

σχεσιακής βάσης δεδομένων, υπάρχουν συσχετίσεις μεταξύ ορισμένων πινάκων. Κατά την ανάγνωση των δεδομένων ενός πίνακα, μπορεί να χρειαστεί να φορτωθούν επιπλέον δεδομένα από διάφορους άλλους πίνακες. Μέσω της εντολής Preload, η GORM προβλέπει, κατά την εκτέλεση ενός query, την φόρτωση των σχετικών δεδομένων εκ των προτέρων.

```
package models

type User struct {
        Id                 string `json:"id" db:"id" uri:"id" gorm:"primaryKey"`
        Email              string `json:"email" db:"email"`
        CreationTimestamp  int    `json:"creation_timestamp"
        ↪  db:"creation_timestamp"`
        LastLoginTimestamp int    `json:"last_login_timestamp"
        ↪  db:"last_login_timestamp"`
        // Association
        // User can be associated with one employee
        EmployeeID *int `json:"employee_id"` //* means it can be null
        // One (firebase) user can be associated with more than one customer
        Customers []*Customer `gorm:"many2many:customer_user;"`
}
```

Listing 8: Μοντέλο GORM για τον πίνακα users της βάσης

```
// Connect to RBAC Database (for gorm queries)
database, err := db.ConnectToRBACGorm()
if err != nil {
    c.AbortWithError(http.StatusInternalServerError, err)
    log.Println(err)
    return
}
defer db.CloseDBConnectionGorm(database)

var users []struct {
    Id                  string `json:"id" db:"id"`
    Email               string `json:"email" db:"email"`
    HasPerformanceAccess bool   `json:"has_performance_access"`
    HasFinancialAccess   bool   `json:"has_financial_access"`
}

err = database.Debug().
    Table("users").
    Joins("JOIN customer_user ON customer_user.user_id = users.id").
    Joins("JOIN customers ON customers.id = customer_user.customer_id").
    Where("customers.id = ?", customer.Id).
    Select("users.id, users.email, 0 AS has_performance_access, 0 AS
    ↪  has_financial_access").
    Scan(&users).Error
if err != nil {
    c.AbortWithError(http.StatusInternalServerError, err)
    log.Println(err)
    return
}
```

Listing 9: Παράδειγμα GORM query

8.3 Βάσεις Δεδομένων

Ως βάση δεδομένων ορίζουμε μία οργανωμένη δομή μέσα στην οποία συλλέγονται δεδομένα και κατηγοριοποιούνται κατάλληλα, με σκοπό την καλύτερη αξιοποίηση τους. Κάθε βάση δεδομένων δίνει την δυνατότητα διαχείρισης και ελέγχου μεγάλου όγκου δεδομένων.

8.3.1 MariaDB

Η MariaDB είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων, ανοικτού κώδικα. Αποτελεί έναν από τους πιο δημοφιλείς διακομιστές (server) βάσεων δεδομένων στον κόσμο, με σημαντικούς χρήστες όπως η Wikipedia, το WordPress.com και η Google. Ο Server της MariaDB κυκλοφορεί υπό την άδεια ανοικτού κώδικα GPLv2 και είναι εγγυημένο ότι θα παραμείνει open source [38].

Η ιστορία της MariaDB ξεκινά όταν ο προκάτοχός της, η MySQL, αγοράστηκε από την εταιρεία Oracle το 2009. Ο ιδρυτής της MySQL, Michael Widenius, διακρίνοντας μια γενικότερη ανησυχία σχετικά με την εποπτεία της Oracle, αποφάσισε να ξεκινήσει ένα νέο, ξεχωριστό έργο, την MariaDB. Η MySQL είχε πάρει το όνομά της από την πρώτη κόρη του Widenius, την My, ενώ η MariaDB πήρε το όνομά της από την δεύτερη κόρη του, την Maria [39].

Ο server της MariaDB μπορεί να χρησιμοποιηθεί για αναλύσεις, ως ενσωματωμένος διακομιστής (embedded server), καθώς είναι εύκολα επεκτάσιμος (scalable). Ένας Embedded Application Server επεκτείνει τις δυνατότητες ενός τυπικού web server, επιτρέποντας στον διακομιστή να χειριστεί πολλά περισσότερα, πέραν των απλών HTTP requests και responses, καθιστώντας εύκολη και γρήγορη την ανάπτυξη μιας λογικής λήψης αποφάσεων σε πραγματικό χρόνο και μιας επιχειρηματικής λογικής [40]. Αυτό το χαρακτηριστικό καθιστά τον διακομιστή της MariaDB, ως έναν Embedded Application Server, ιδανικό για την ανάπτυξη web-based εφαρμογών.

Είναι σημαντικό να αναφερθεί επίσης ότι η MariaDB μπορεί να χρησιμοποιηθεί για τη διαχείριση των δεδομένων συναλλαγών υψηλής διαθεσιμότητας. Οι clusters υψηλής διαθεσιμότητας (high availability clusters) είναι μια ομάδα από servers που λειτουργούν ως ένα ενιαίο σύστημα. Στην περίπτωση που ένας διακομιστής δεν ανταποκρίνεται, ένας άλλος server του cluster αναλαμβάνει αυτόματα την εκτέλεση των ενεργών συναλλαγών, εξασφαλίζοντας ότι η υπηρεσία ή η εφαρμογή που υποστηρίζεται, παραμένει λειτουργική [41]. Ο διακομιστής της MariaDB αποτελεί μια ιδανική επιλογή για τέτοιου είδους λειτουργίες ανάκτησης (recovery procedures) σε έναν cluster [42], εξασφαλίζοντας την ομαλή λειτουργία του.

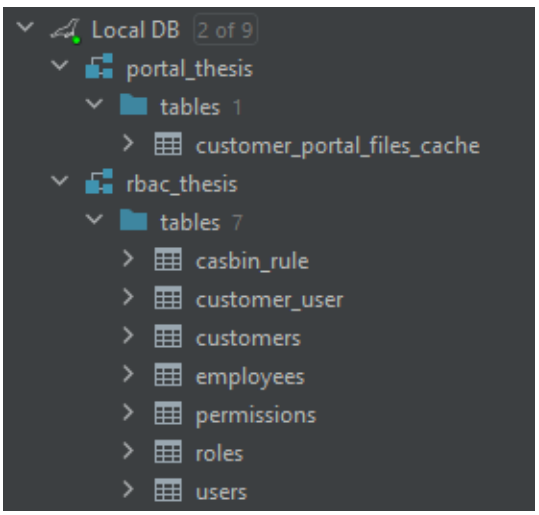
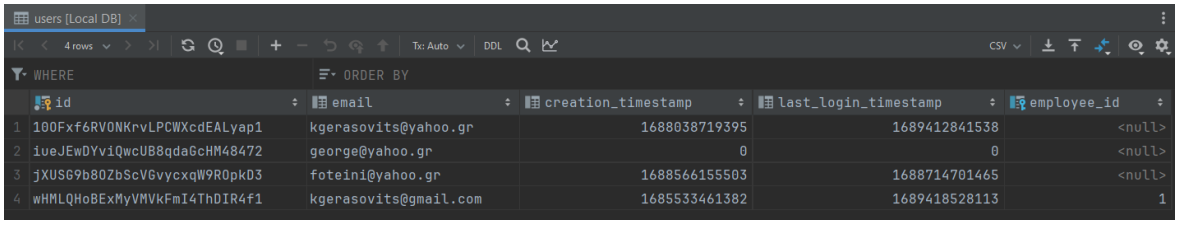


Figure 2: Πίνακες της βάσης δεδομένων



users [Local DB]

4 rows | Tx: Auto | DDL | CSV

WHERE ORDER BY

	id	email	creation_timestamp	last_login_timestamp	employee_id
1	10OFxf6RVONKrvLPCWXcdEALyap1	kgerasovits@yahoo.gr	1688038719395	1689412841538	<null>
2	iueJEwDYviQwcUB8qdaGcHM48472	george@yahoo.gr	0	0	<null>
3	jXUSG9b8OZbScVGvycxqW9ROpkD3	foteini@yahoo.gr	1688566155503	1688714701465	<null>
4	wHMLQHoBExMyVMVkFmI4ThDIR4f1	kgerasovits@gmail.com	1685533461382	1689418528113	1

Figure 3: Πίνακας users στην βάση δεδομένων

8.3.2 Firebase Authentication

Η Firebase είναι μια πλατφόρμα ανάπτυξης εφαρμογών από την Google, η οποία προσφέρει ευέλικτες λύσεις για την δημιουργία cloud-based εφαρμογών. Ανάμεσα στις υπηρεσίες που παρέχει, βρίσκεται και το Firebase Authentication, το οποίο δίνει τη δυνατότητα αυθεντικοποίησης των χρηστών ενός συστήματος [43]. Με την εν λόγω υπηρεσία, οι προγραμματιστές μπορούν εύκολα να προσθέσουν στις εφαρμογές τους μια πληθώρα διαφορετικών τρόπων σύνδεσης των χρηστών, όπως sign-in με email και κωδικό πρόσβασης, σύνδεση μέσω Google, Facebook, GitHub, Microsoft, Yahoo και πολλών άλλων αντίστοιχων υπηρεσιών. Το Firebase Authentication, παρέχει επιπλέον λειτουργίες όπως η πολυπαραγοντική αυθεντικοποίηση (multi-factor authentication), διάφορες μεθόδους αποκλεισμού χρηστών (blocking functions), το audit logging για την καταγραφή και τον καλύτερο έλεγχο όλων των δραστηριοτήτων, υποστηρίζει πρωτόκολλα όπως το SAML και το generic OpenID Connect κ.ά.

Το Firebase Authentication χρησιμοποιήθηκε ως ένα εργαλείο αυθεντικοποίησης και εξουσιοδότησης των χρηστών των διαδικτυακών εφαρμογών RBAC και Portal που υλοποιήθηκαν στην παρούσα εργασία. Τα στοιχεία των πελατών και των μελών της εταιρείας με δικαίωμα πρόσβασης σε μία από τις παραπάνω πλατφόρμες, είναι αποθηκευμένα στην Firebase. Οι εν λόγω χρήστες συνδέονται στις εφαρμογές μέσω email και password ή μέσω Google με τον εταιρικό τους Google λογαριασμό. Η επιλογή του Firebase Authentication ως μηχανισμού ελέγχου ταυτοποίησης, προσφέρει την επιθυμητή ισορροπία μεταξύ ασφάλειας, ευκολίας χρήσης και ευελιξίας για τους χρήστες του συστήματος.

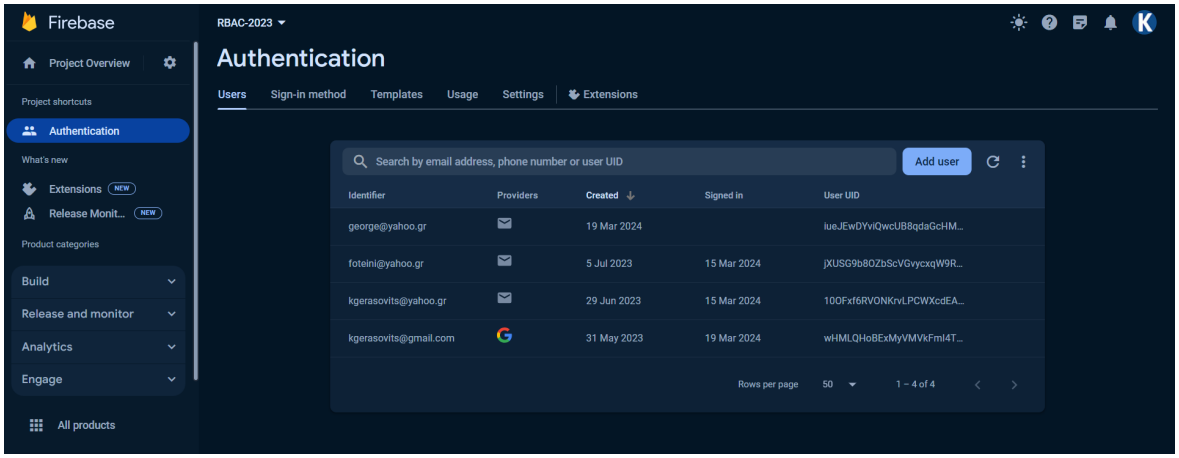


Figure 4: Χρήστες της εφαρμογής, αποθηκευμένοι στο Authentication της Firebase

8.3.3 Google Drive

Το Google Drive είναι μια υπηρεσία αποθήκευσης αρχείων στο cloud που παρέχεται από την εταιρεία Google. Με το Google Drive, οι χρήστες μπορούν να αποθηκεύουν τα αρχεία τους σε έναν ασφαλή διαδικτυακό χώρο και να έχουν πρόσβαση σε αυτά από οποιαδήποτε συσκευή. Το Google Drive επιτρέπει την κοινή χρήση αρχείων και φακέλων με άλλους χρήστες και δίνει τη δυνατότητα για την παράλληλη επεξεργασία τους. Το Google Drive αποτελεί έναν εύχρηστο τρόπο οργάνωσης και διαχείρισης αρχείων για ατομική και επαγγελματική χρήση.

Κάθε μήνα, η εταιρεία εκδίδει μηνιαία (monthly), τριμηνιαία (quarter), εξαμηνιαία (half-year) ή ετήσια (yearly) financial και performance reports, σε μορφή παρουσιάσεων και υπολογιστικών φύλλων, για τον εκάστοτε πελάτη της. Κατά αυτόν τον τρόπο, οι πελάτες ενημερώνονται για την πρόοδο των καναλιών και των προφίλ τους στα κοινωνικά δίκτυα, καθώς και για τα έσοδά τους από τις πλατφόρμες όπου οι ίδιοι προβάλλονται.

Τα εν λόγω αρχεία, αποθηκεύονται σε έναν ασφαλή χώρο στο Google Drive, ο οποίος εσωτερικά στην εταιρεία ονομάζεται “Customer’s Vault”. Μέσω του RBAC, κάθε πελάτης έχει αποκλειστική πρόσβαση στη δική του θυρίδα στο Vault, η οποία είναι ασφαλισμένη και αυστηρά προσωπική.

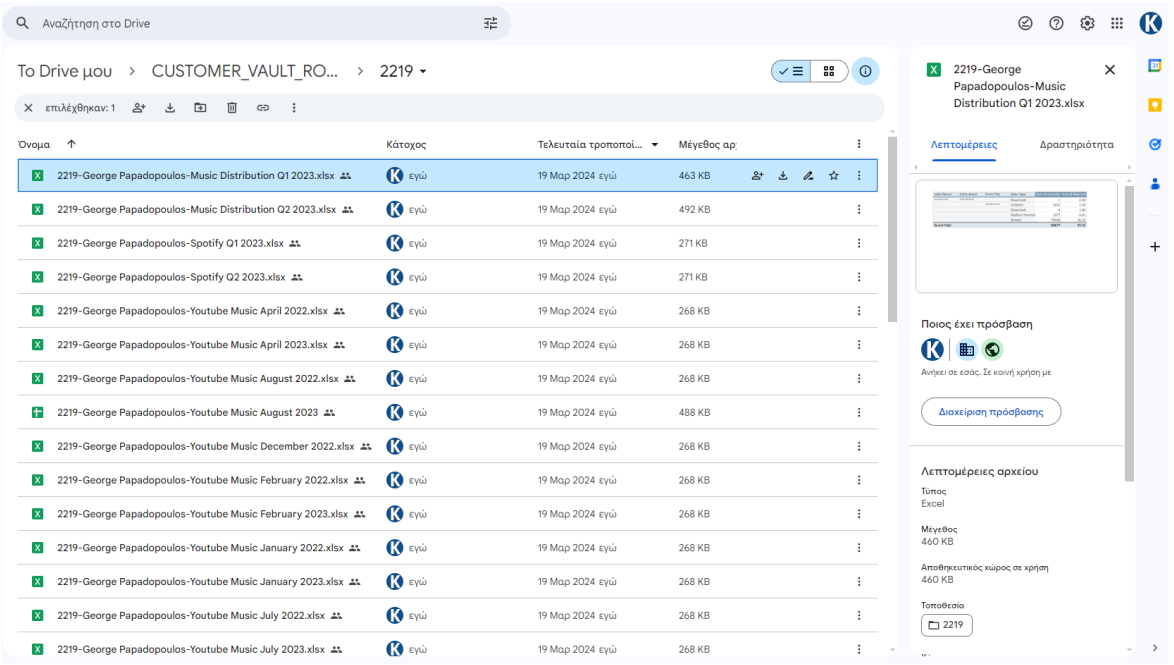


Figure 5: Παράδειγμα Vault ενός πελάτη στο Google Drive

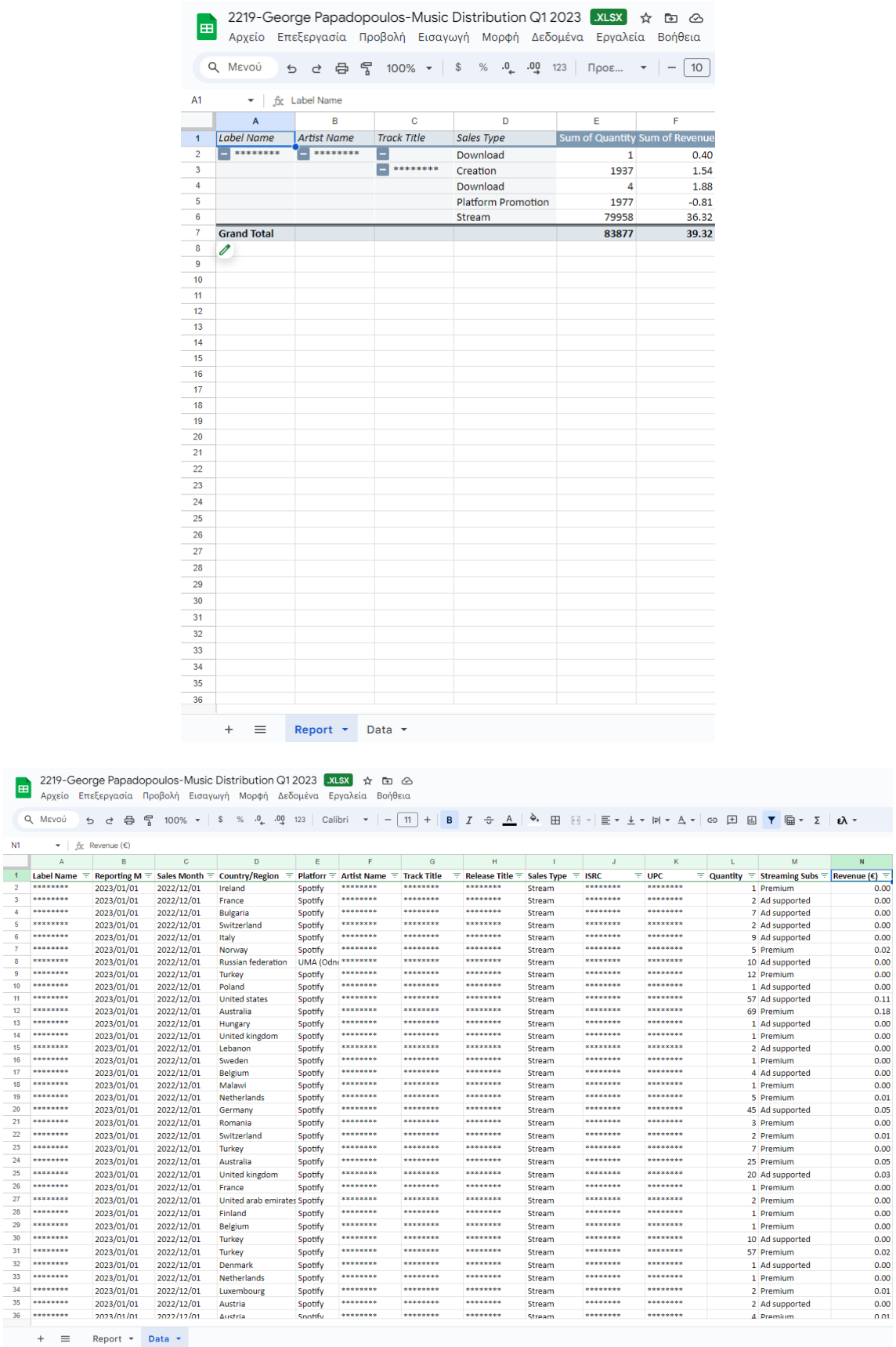


Figure 6: Παράδειγμα financial report ενός πελάτη

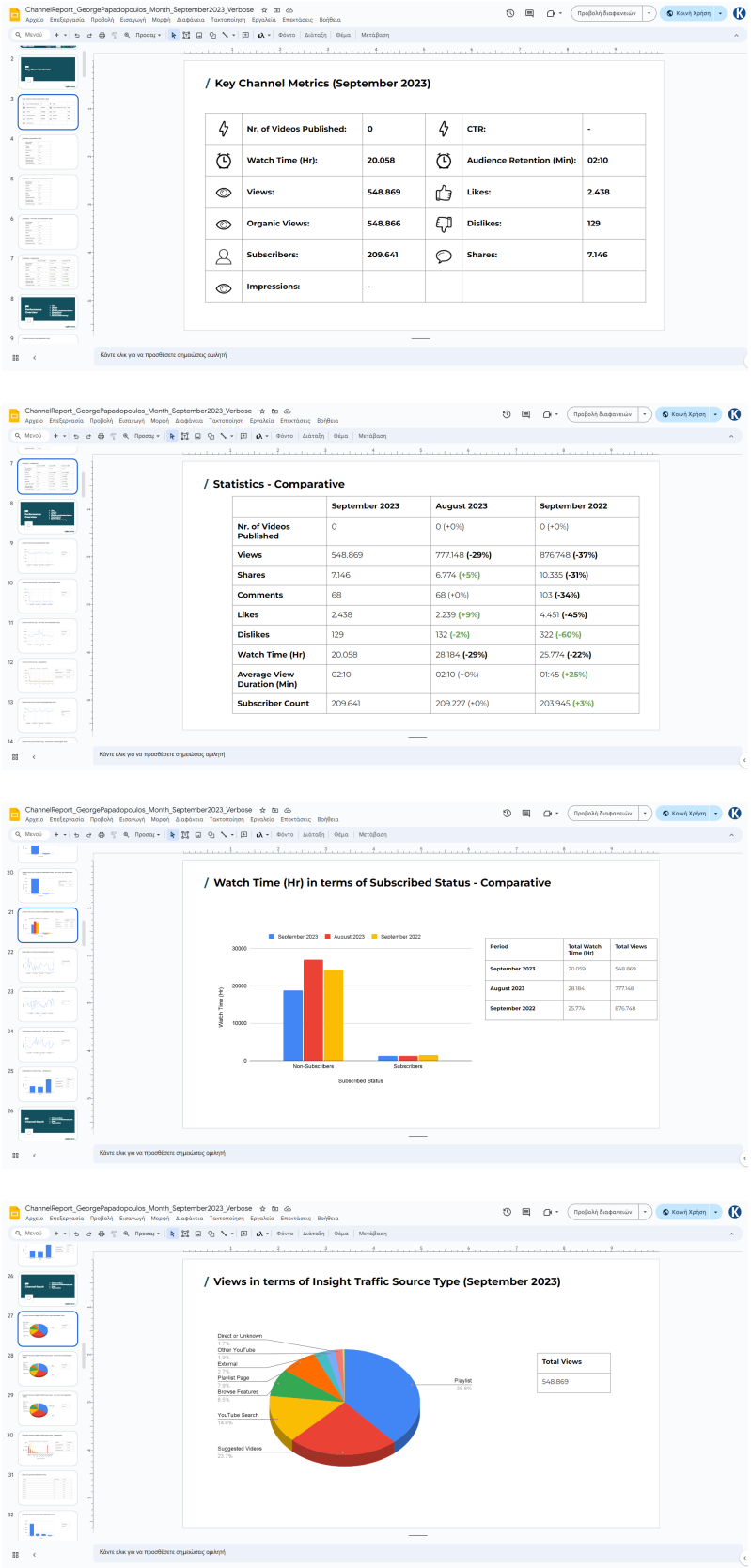


Figure 7: Παράδειγμα performance report ενός πελάτη

9 Χρήστες της εφαρμογής RBAC

Στην ενότητα αυτή θα δοθεί μια ενδελεχής περιγραφή των χρηστών που διαθέτουν πρόσβαση στην εφαρμογή RBAC, αναλύοντας τον ρόλο και τις λειτουργείες τους εντός του συστήματος.

9.1 Διαχειριστής (Administrator) του RBAC

Ο ρόλος του RBAC administrator είναι κρίσιμος για την ομαλή λειτουργία όλων των συστημάτων της εταιρείας. Ο διαχειριστής έχει αποκλειστική πρόσβαση στην εφαρμογή του RBAC και είναι υπεύθυνος για την διαχείριση των δικαιωμάτων πρόσβασης και την ανάθεση ρόλων στους χρήστες του συστήματος.

Ο administrator του RBAC δημιουργεί και επεξεργάζεται τους διάφορους ρόλους των χρηστών. Οι ρόλοι αντιστοιχούν στις θέσεις εργασίας των μελών της εταιρείας, διαμορφώνοντας διαφορετικά επίπεδα πρόσβασης στα εταιρικά συστήματα. Ο διαχειριστής είναι υπεύθυνος και για τον καθορισμό των δικαιωμάτων πρόσβασης (permissions) που συνδέονται με τον εκάστοτε ρόλο. Καθορίζει συνεπώς ποιες ενέργειες επιτρέπονται ή απαγορεύονται για κάθε ρόλο και σε ποια τμήματα της εφαρμογής έχει πρόσβαση.

Επιπλέον, ο διαχειριστής είναι υπεύθυνος για τη δημιουργία και επεξεργασία των προφίλ των εργαζομένων και των πελατών. Κάθε ένα προφίλ συνδέεται με ένα ή περισσότερα emails, μέσω των οποίων οι χρήστες έχουν πρόσβαση στα συστήματα. Τα μέλη της εταιρείας διαχειρίζονται συγκεκριμένους πελάτες και έχουν πρόσβαση στα δεδομένα όσων τους αφορούν. Με την ίδια λογική, ένας πελάτης που συνδέεται στο σύστημα, βλέπει μόνο τα προσωπικά του οικονομικά δεδομένα και στοιχεία προόδου. Υπάρχει όμως και η δυνατότητα σύνδεσης ενός email με πολλαπλά προφίλ πελατών. Με αυτόν τον τρόπο, ο administrator επιτρέπει σε έναν λογαριασμό να έχει πρόσβαση στα δεδομένα πολλών διαφορετικών πελατών.

Μια ακόμα βασική αρμοδιότητα του διαχειριστή του RBAC αποτελεί η εισαγωγή ενός καινούργιου πελάτη ή ενός νέου μέλους της εταιρείας στο σύστημα. Στην πρώτη περίπτωση, ο administrator δημιουργεί το προφίλ του πελάτη και το συνδέει με ένα ή περισσότερα emails, δίνοντάς τους πρόσβαση στα δεδομένα και τα οικονομικά του στοιχεία. Στην περίπτωση εισαγωγής νέου μέλους, ο διαχειριστής δημιουργεί το αντίστοιχο προφίλ και το συνδέει με το εταιρικό email του υπαλλήλου, μέσω του οποίου θα έχει πρόσβαση στο σύστημα και στα δεδομένα των πελατών που διαχειρίζεται.

Η διαδικασία αφαίρεσης ενός πελάτη ή ενός μέλους της εταιρείας, γίνεται πολύ εύκολα με την διαγραφή του προφίλ του από την αντίστοιχη λίστα στην εφαρμογή του RBAC. Κατά αυτόν τον τρόπο, τα email των χρηστών που δεν είναι συνδεδεμένα με κάποιο προφίλ, διαγράφονται αυτόματα από το Authentication της Firebase και οι χρήστες που αφαιρούνται, χάνουν το δικαίωμα πρόσβασης σε οποιαδήποτε εφαρμογή του συστήματος.

Συνοψίζοντας, ο διαχειριστής του RBAC είναι υπεύθυνος για τη διαχείριση των χρηστών, των ρόλων και των δικαιωμάτων πρόσβασής τους στα εταιρικά συστήματα, καθώς και για την ασφάλεια και την ομαλή λειτουργία των εφαρμογών της εταιρείας γενικότερα.

10 Υλοποίηση της εφαρμογής RBAC

Στην ενότητα αυτή θα αναλύσουμε τα τμήματα από τα οποία αποτελείται η εφαρμογή RBAC, καθώς και όλες τις λειτουργίες που μπορούν να εκτελέσουν οι χρήστες στις επιμέρους σελίδες της.

10.1 Login Screen

Η εφαρμογή RBAC (Role Based Access Control) που υλοποιήθηκε στην παρούσα εργασία, περιλαμβάνει μια σελίδα σύνδεσης χρήστη (Login Screen), η οποία αποτελεί την πρώτη διαδικασία πιστοποίησης (authentication) και εξουσιοδότησης (authorization) των χρηστών που επιθυμούν πρόσβαση στο σύστημα. Η εφαρμογή απευθύνεται μόνο στον διαχειριστή του RBAC, ο οποίος έχει αποκλειστική πρόσβαση μέσω του εταιρικού του λογαριασμού Google.

Η σελίδα σύνδεσης αποτελείται από ένα πλαίσιο στο οποίο φαίνονται με ευδιάκριτο τρόπο, το λογότυπο της εταιρείας, ο τίτλος της εφαρμογής και ένα κουμπί “Sign in with Google”, το οποίο προτρέπει τον χρήστη να συνδεθεί στην εφαρμογή, χρησιμοποιώντας το εταιρικό του gmail. Η χρήση του λογαριασμού Google ως μηχανισμού ελέγχου ταυτοποίησης εξασφαλίζει μια γρήγορη, ασφαλή και ευέλικτη διαδικασία εισόδου στην εφαρμογή, ενώ ταυτόχρονα διασφαλίζει την προστασία των δεδομένων και την εμπιστευτικότητα των πληροφοριών του χρήστη.

Κατά την είσοδο του διαχειριστή στο σύστημα, ο λογαριασμός του ελέγχεται από την υπηρεσία της Google, μέσω firebase authentication, ενώ στη συνέχεια γίνεται εσωτερικός έλεγχος των δικαιωμάτων πρόσβασης στην εφαρμογή, βάσει του ρόλου του στην εταιρεία. Αυτός ο έλεγχος εξασφαλίζει ότι ο κάθε χρήστης έχει πρόσβαση μόνο στις πλατφόρμες, τις λειτουργίες και τα δεδομένα που του επιτρέπονται.

Η ενσωμάτωση σελίδων σύνδεσης (Login pages) στις εφαρμογές της εταιρείας, αποτελεί σημαντικό βήμα προς τη δημιουργία ενός ασφαλούς και ελεγχόμενου συστήματος πρόσβασης σε εταιρικά δεδομένα και λειτουργίες. Η επιλογή του firebase authentication ως μηχανισμού ελέγχου ταυτοποίησης, προσφέρει την επιθυμητή ισορροπία μεταξύ ασφάλειας, ευκολίας χρήσης και ευελιξίας για τους χρήστες.

Figure 8: Login Screen της εφαρμογής RBAC

10.2 Homepage

Η πρώτη σελίδα ονομάζεται ”Home” και λειτουργεί ως το κεντρικό μενού πλοήγησης της εφαρμογής. Περιέχει πέντε εικονίδια/κουμπιά με τα οποία ο διαχειριστής του RBAC μπορεί να μεταβεί στις υπόλοιπες σελίδες της εφαρμογής (Users, Employees, Customers, Roles και Permissions).

Η εφαρμογή του συστήματος διαχείρισης RBAC, διαθέτει επιπλέον ένα Navigation Bar στο πάνω μέρος της σελίδας, διευκολύνοντας την πλοήγηση του χρήστη μέσα στην εφαρμογή, σε οποιαδήποτε σελίδα κι αν βρίσκεται.

Στο δεξιό μέρος του Navigation Bar, βρίσκεται το εικονίδιο του συνδεδεμένου χρήστη (User Profile Icon). Κάνοντας hover πάνω από το εικονίδιο, εμφανίζεται ένα μικρό παράθυρο με τα στοιχεία του χρήστη, όνομα και email, καθώς και το κουμπί Logout για αποσύνδεση από την εφαρμογή.

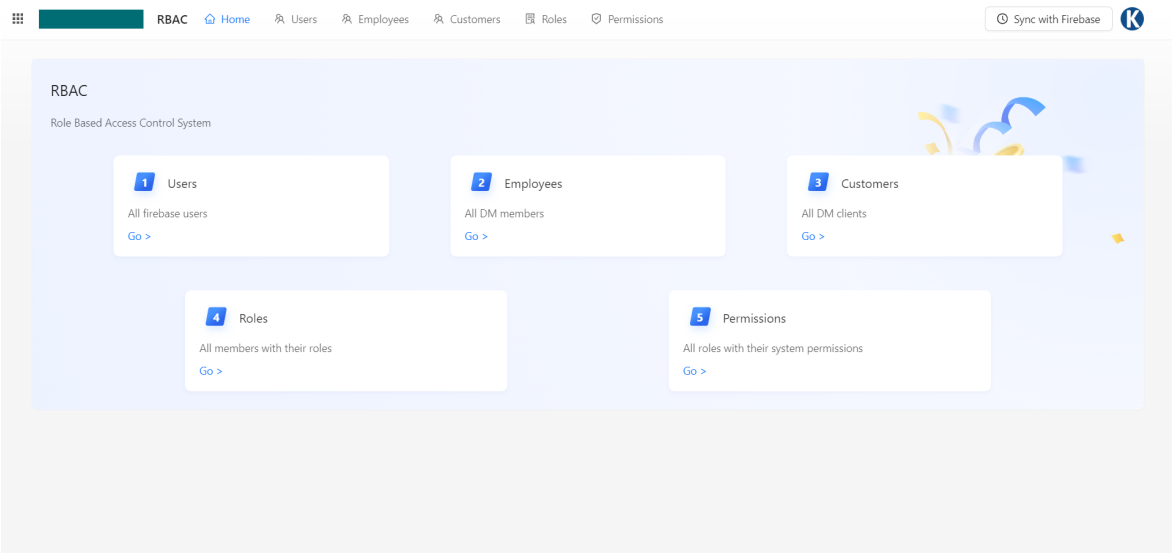


Figure 9: Homepage της εφαρμογής RBAC

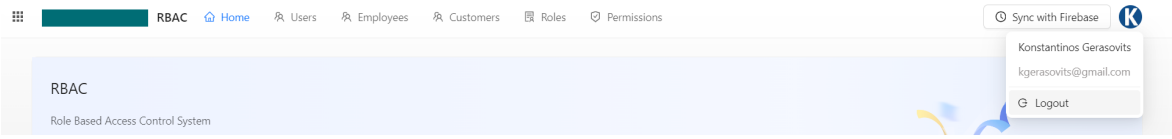


Figure 10: Navigation Bar της εφαρμογής RBAC

10.3 Users

Η δεύτερη σελίδα της εφαρμογής έχει τίτλο ”Users” και περιλαμβάνει έναν πίνακα με όλους τους χρήστες του συστήματος. Οι χρήστες αυτοί είναι αποθηκευμένοι στο Authentication της Firebase, μέσω sign-in με email και password. Με την προσθήκη κάθε νέου χρήστη στην Firebase, το email του συνδέεται με ένα μοναδικό αναγνωριστικό ”User UID”.

Κάθε γραμμή του πίνακα αποτελεί το προφίλ ενός χρήστη. Οι πληροφορίες που περιέχει είναι το email και το μοναδικό Firebase Id του.

Στον διαχειριστή του RBAC δίνεται η δυνατότητα ένταξης και διαγραφής κάποιου χρήστη από το σύστημα της εταιρείας. Πατώντας το κουμπί ”Add new user”, εμφανίζεται μια φόρμα όπου ο administrator μπορεί να συμπληρώσει τα στοιχεία του καινούργιου χρήστη, email και κωδικό πρόσβασης, ώστε να προστεθεί αυτόματα στην Firebase και στην MariaDB. Αντίστοιχα, με το κουμπί ”Delete”, που βρίσκεται στο δεξιό μέρος κάθε γραμμής του πίνακα, μπορεί εύκολα να διαγράψει έναν επιλεγμένο χρήστη, αφαιρώντας του έτσι κάθε δικαίωμα πρόσβασης στις εφαρμογές της εταιρείας.

Στο δεξιό μέρος του Navigation Bar της σελίδας, βρίσκεται το κουμπί “Sync with Firebase”. Κατά την εγγραφή ενός νέου χρήστη στο σύστημα, τα στοιχεία του αποθηκεύονται αυτόματα στο Authentication της Firebase, όπου του αποδίδεται και το μοναδικό UID του. Για λόγους ταχύτητας και απόδοσης του συστήματος, τα δεδομένα για τον πίνακα των χρηστών στην σελίδα Users, προέρχονται από τον server της MariaDB και όχι απευθείας από την Firebase. Το “Sync with Firebase” εξασφαλίζει την αρμονία στα δεδομένα των δύο βάσεων, ενημερώνοντας σε δευτερόλεπτα τον πίνακα των χρηστών της MariaDB από τον αντίστοιχο στο Authentication της Firebase.

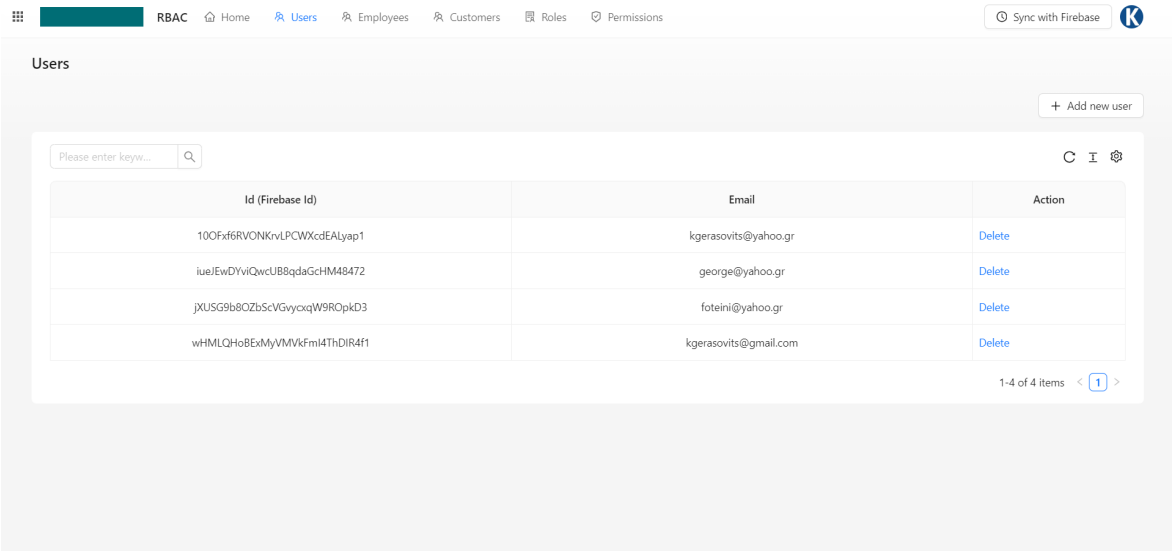


Figure 11: Σελίδα ”Users” της εφαρμογής RBAC

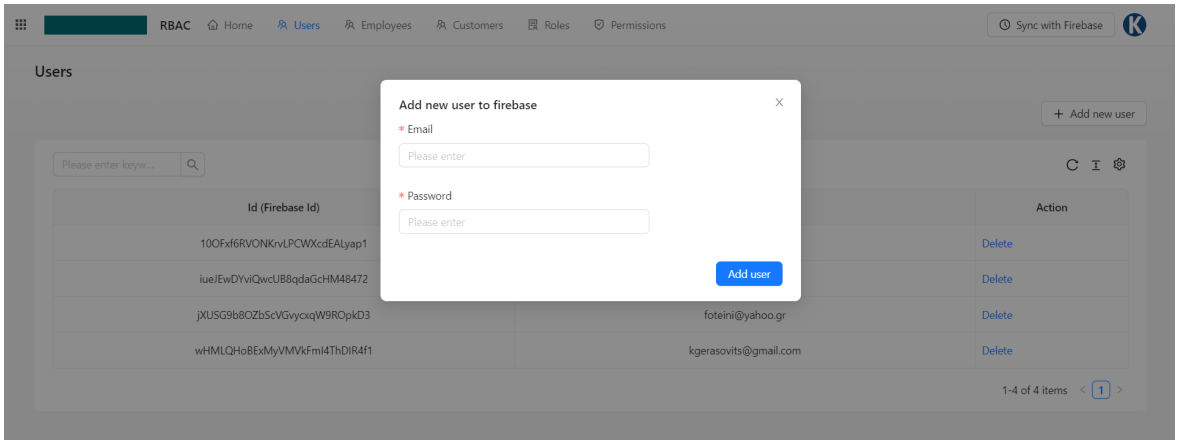


Figure 12: Προσθήκη νέου χρήστη

10.4 Employees

Η τρίτη σελίδα της εφαρμογής έχει τίτλο ”Employees” και περιλαμβάνει έναν πίνακα με όλους τους υπαλλήλους της εταιρείας. Εκείνοι έχουν πρόσβαση στις εφαρμογές του συστήματος, μέσω των εταιρικών τους emails. Βασική αρμοδιότητα του διαχειριστή του RBAC είναι να αντιστοιχίσει κάθε μέλος της εταιρείας με το email του, το οποίο είναι αποθηκευμένο στην Firebase μέσω Google authentication.

Ο πίνακας των υπαλλήλων περιλαμβάνει το πλήρες όνομα του εργαζομένου και ένα μοναδικό auto-increment id, το οποίο αποδίδεται αυτόματα στο κάθε μέλος από την MariaDB, κατά την εγγραφή του στο σύστημα. Κάθε γραμμή του πίνακα αποτελεί το προφίλ ενός εργαζομένου. Ο διαχειριστής του RBAC μπορεί να δει το συνδεδεμένο με τον εκάστοτε υπάλληλο email, πατώντας το κουμπί “+” στο αριστερό μέρος του προφίλ του.

Δύο από τις βασικές αρμοδιότητες του administration, που σχετίζονται με την σελίδα “Employees”, είναι η προσθήκη και η αφαίρεση ενός εργαζομένου από το σύστημα της εταιρείας, κατά την πρόσληψη ή την αποχώρησή του, αντίστοιχα. Στην πρώτη περίπτωση, ο διαχειριστής οφείλει να δημιουργήσει το προφίλ του νέου μέλους, συμπληρώνοντας σωστά τα στοιχεία του, προσθέτοντας με αυτόν τον τρόπο μια επιπλέον γραμμή στον πίνακα των εργαζομένων. Τέλος, είναι απαραίτητο να αντιστοιχίσει τον καινούργιο υπάλληλο με το εταιρικό του email, μέσω του οποίου θα έχει πρόσβαση στο σύστημα. Στην περίπτωση αποχώρησης ενός μέλους της εταιρείας, ο administrator διαγραφεί το προφίλ του από τον πίνακα των εργαζομένων, αφαιρώντας του έτσι κάθε δικαίωμα πρόσβασης στις εφαρμογές της εταιρείας.

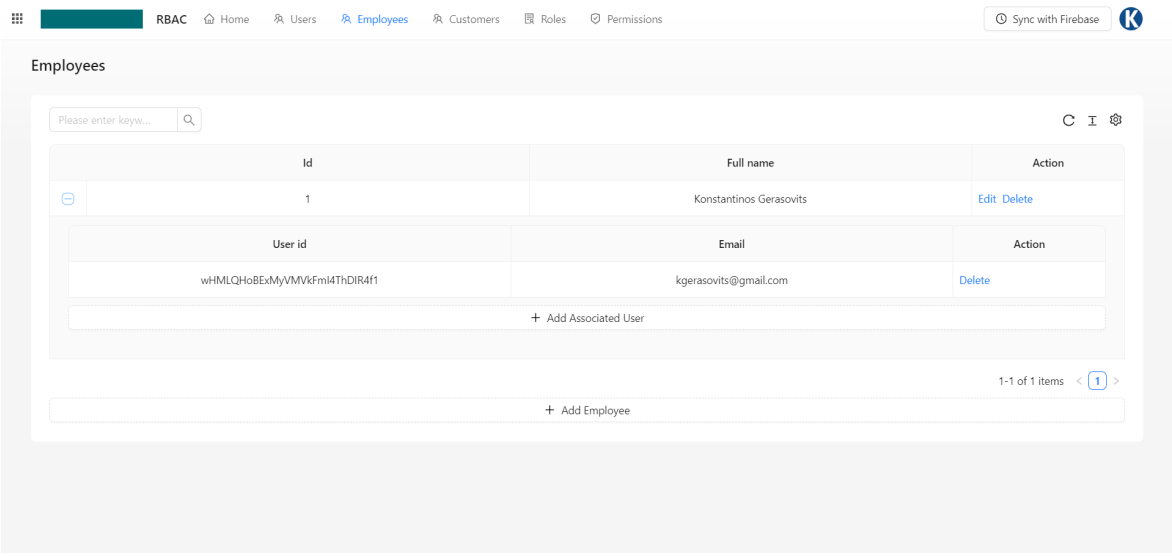


Figure 13: Σελίδα "Employees" της εφαρμογής RBAC

10.5 Customers

Η τέταρτη σελίδα της εφαρμογής έχει τίτλο "Customers" και περιλαμβάνει έναν πίνακα με όλους τους πελάτες της εταιρείας. Βασική αρμοδιότητα του διαχειριστή του RBAC είναι να αντιστοιχίσει κάθε πελάτη με τα emails των χρηστών που θα έχουν πρόσβαση στα δεδομένα του.

Κάθε γραμμή του πίνακα αποτελεί το προφίλ ενός πελάτη. Οι πληροφορίες που περιέχει είναι το όνομα και ένα μοναδικό, τετραψήφιο αναγνωριστικό id (Podio Id). Στον administrator δίνεται η δυνατότητα προσθήκης, επεξεργασίας και διαγραφής ενός προφίλ από τον πίνακα. Με την αφαίρεση ενός πελάτη, όλοι οι συνδεδεμένοι με αυτόν χρήστες, χάνουν το δικαίωμα πρόσβασης στα οικονομικά δεδομένα και τα στοιχεία προόδου του.

Ο διαχειριστής του RBAC μπορεί πολύ εύκολα να δει ποιοι χρήστες έχουν δικαίωμα πρόσβασης στα δεδομένα του εκάστοτε πελάτη. Πατώντας το κουμπί "+" στο αριστερό μέρος κάθε γραμμής του πίνακα, εμφανίζεται μια λίστα με όλα τα συνδεδεμένα με τον πελάτη emails. Οι χρήστες αυτοί βλέπουν την πλήρη εικόνα των οικονομικών στοιχείων και της απόδοσης του συγκεκριμένου πελάτη, κατά την είσοδό τους στο σύστημα (εφαρμογή Customer Portal).

Ο διαχειριστής έχει την δυνατότητα εισαγωγής ή αφαίρεσης ενός email από την παραπάνω λίστα, προσθέτοντας ή αφαιρώντας έτσι από τον αντίστοιχο χρήστη το δικαίωμα πρόσβασης στα δεδομένα του πελάτη. Τέλος, ο administrator καθορίζει αν ένα email και κατ' επέκταση ο χρήστης, θα έχει πρόσβαση μόνο στα οικονομικά δεδομένα ενός πελάτη, βλέποντας αποκλειστικά τα financial reports, μόνο στα στοιχεία προόδου του, βλέποντας αποκλειστικά τα performance reports ή και στα δύο.

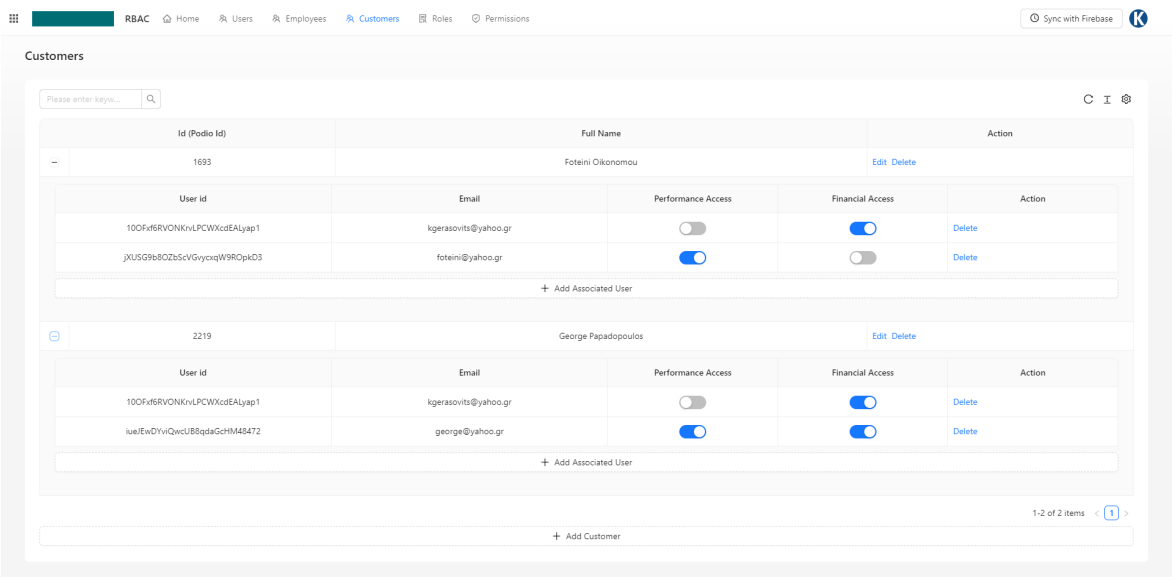


Figure 14: Σελίδα "Customers" της εφαρμογής RBAC

10.6 Roles

Η πέμπτη σελίδα της εφαρμογής έχει τίτλο "Roles". Εδώ, ο διαχειριστής του RBAC αποδίδει τους ρόλους στα μέλη της εταιρείας. Η εν λόγω σελίδα περιλαμβάνει έναν πίνακα με τα στοιχεία κάθε υπαλλήλου. Συγκεκριμένα, περιέχει στήλες με το μοναδικό αναγνωριστικό id που του έχει αποδοθεί από την firebase (User id), το πλήρες όνομά του, το εταιρικό του email, καθώς και τον ρόλο του μέσα στην εταιρεία.

Η ανάθεση ρόλου σε κάθε εργαζόμενο, αποτελεί βασική αρμοδιότητα του administrator του RBAC. Ο ίδιος έχει τη δυνατότητα προσθήκης και αλλαγής του ρόλου ενός υπαλλήλου, μέσω του κουμπιού "Edit" που βρίσκεται στην τελευταία στήλη του πίνακα.

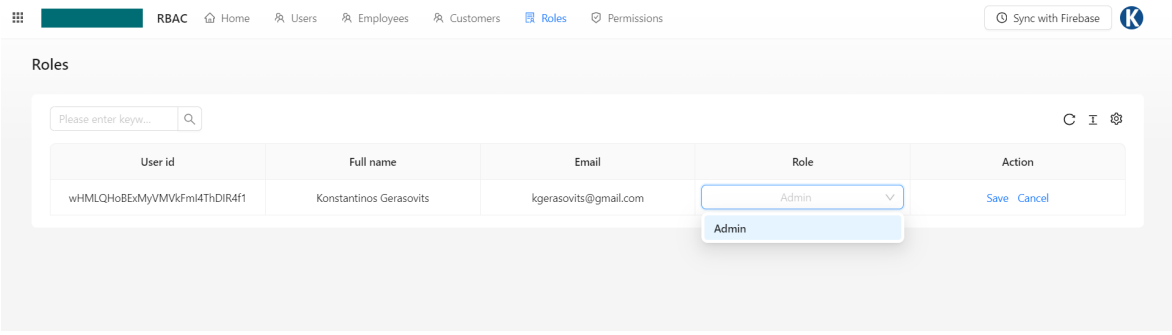


Figure 15: Σελίδα "Roles" της εφαρμογής RBAC

10.7 Permissions

Κάθε ρόλος μέσα στην εταιρεία και κατ' επέκταση κάθε χρήστης που του έχει αποδοθεί ο ρόλος αυτός, διαθέτει συγκεκριμένα δικαιώματα πρόσβασης (permissions). Ανάλογα με το αν ένα permission είναι εγκεκριμένο, το σύστημα επιτρέπει ή απαγορεύει την πρόσβαση στα τμήματα της εφαρμογής που απαιτούν το εν λόγω permission. Έτσι, οι χρήστες μέσω των ρόλων τους, έχουν πρόσβαση μόνο στις επιτρεπόμενες σελίδες των εφαρμογών, χρησιμοποιώντας μόνο εκείνα τα εργαλεία που τους είναι απαραίτητα. Το παραπάνω στοιχείο κρίνεται αναγκαίο για την διαφύλαξη των ευαίσθητων δεδομένων, από όσους δεν διαθέτουν δικαίωμα πρόσβασης στις συγκεκριμένες πληροφορίες.

Η έκτη και τελευταία σελίδα της εφαρμογής ονομάζεται "Permissions" και περιλαμβάνει τη λίστα

των ρόλων, που αντιστοιχούν στις θέσεις εργασίας των εργαζομένων. Οι ρόλοι βρίσκονται εντός ενός μπλε πλαισίου στο αριστερό μέρος της σελίδας. Ο administrator του RBAC έχει την δυνατότητα δημιουργίας ενός νέου και διαγραφής ενός υπάρχοντος ρόλου. Απαραίτητη προϋπόθεση για την διαγραφή ενός ρόλου, είναι να μην υπάρχει κανένας συσχετισμένος εργαζόμενος με τον εν λόγω ρόλο.

Πατώντας πάνω σε έναν ρόλο, εμφανίζεται ολόκληρη η λίστα των permissions του συστήματος, με εγκεκριμένα μόνο όσα κρίνονται απαραίτητα για τον συγκεκριμένο ρόλο. Για παράδειγμα, για τον λογιστή μιας εταιρείας, το δικαίωμα πρόσβασης στα οικονομικά της στοιχεία είναι εγκεκριμένο. Υπεύθυνος για τον καθορισμό των δικαιωμάτων πρόσβασης κάθε ρόλου είναι ο διαχειριστής του RBAC. Η έγκριση και η αφαίρεση ορισμένων permissions από τον εκάστοτε ρόλο, αποτελούν και αυτά με την σειρά τους, δύο από τις βασικές του αρμοδιότητες. Τέλος, ο administrator έχει την δυνατότητα δημιουργίας ενός νέου τύπου δικαιώματος πρόσβασης, εφόσον υπάρξει αυτή η ανάγκη από τα συνεχώς αναπτυσσόμενα συστήματα της εταιρείας, καθώς και την δυνατότητα αφαίρεσης ενός υπάρχοντος permission, στην σπάνια περίπτωση που τα τμήματα των εφαρμογών που το χρειάζονται, πάψουν να χρησιμοποιούνται.

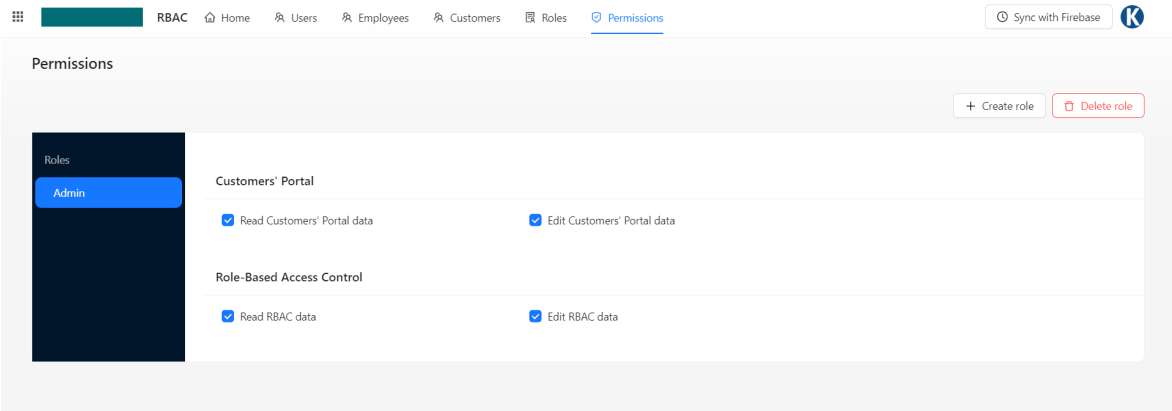


Figure 16: Σελίδα "Permissions" της εφαρμογής RBAC

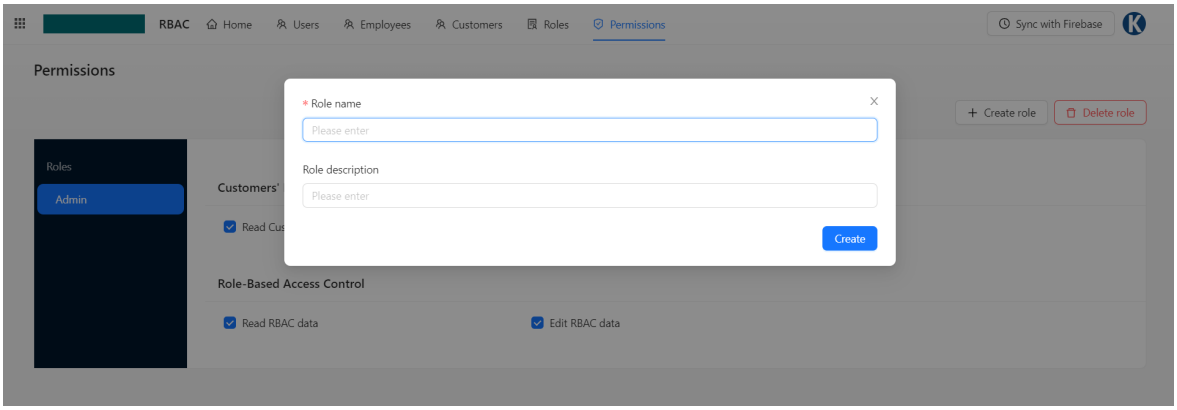


Figure 17: Προσθήκη νέου ρόλου

11 Ψηφιακή Πλατφόρμα Πελατών / Portal

11.1 Εισαγωγή

Το Portal είναι μια ψηφιακή πλατφόρμα που λειτουργεί συμπληρωματικά με την βασική εφαρμογή της παρούσας εργασίας, ώστε να αναδείξει τις λειτουργίες και τις δυνατότητες του συστήματος RBAC. Απευθύνεται τόσο στους πελάτες, όσο και στα μέλη της εταιρείας. Χάριν ευκολίας, όταν αναφερόμαστε στην πλατφόρμα από τη σκοπιά ενός πελάτη, θα χρησιμοποιούμε τον όρο "Customer Portal". Αντιστοίχως, για εσωτερική χρήση θα χρησιμοποιούμε τον όρο "Internal Portal".

Κάθε μήνα η εταιρεία ενημερώνει τους πελάτες της για την πρόοδό τους στα μέσα κοινωνικής δικτύωσης, στέλνοντάς τους performance reports, σε μορφή παρουσιάσεων. Οι παρουσιάσεις αυτές περιέχουν αναλυτικά γραφήματα, τα οποία περιγράφουν την πορεία των καναλιών και των διάφορων προφίλ των πελατών στο χρόνο. Τα γραφήματα περιέχουν χρήσιμες μετρικές όπως views, likes, comments, shares κ.ά. Τα reports μπορεί να είναι μηνιαία (monthly), τριμηνιαία (quarter), εξαμηνιαία (half-year) ή ετήσια (yearly) και αφορούν την εξέλιξη ενός καναλιού για το αντίστοιχο χρονικό διάστημα.

Υπάρχει και μία δεύτερη κατηγορία αναφορών, οικονομικού χαρακτήρα. Ονομάζονται financial reports και αφορούν μόνο τα έσοδα των πελατών από τις πλατφόρμες κοινωνικής δικτύωσης. Έχουν την μορφή υπολογιστικών φύλλων, αποστέλλονται κάθε τέλος του μήνα και περιέχουν αναλυτικά τις πηγές και τα ποσά των εσόδων των πελατών.

Μετά τη δημιουργία τους, τα reports των πελατών αποθηκεύονται σε έναν ασφαλή χώρο στο Google Drive. Κάθε πελάτης έχει τη δική του, προσωπική "θυρίδα", στην οποία έχει αποκλειστική πρόσβαση. Με την βοήθεια του RBAC, τα περιεχόμενα του φακέλου του εκάστοτε πελάτη είναι ορατά μόνο στον ίδιο, μέσω του email του. Το Portal λειτουργεί ως ένας ψηφιακός χώρος παρουσίασης όλων των reports που βρίσκονται στο Google Drive.

Το Internal Portal αποτελεί την βασική εφαρμογή διαχείρισης των πελατών από τους account managers της εταιρείας. Εκεί γίνεται ο έλεγχος και η επεξεργασία των δεδομένων των παραγόμενων reports, με δυνατότητα αποθήκευσης, έγκρισης και προβολής κάθε αρχείου στο Google Drive. Μόνο τα εγκεκριμένα financial και performance reports είναι ορατά στους πελάτες.

Το Customer Portal αποτελεί την μοναδική πλατφόρμα του συστήματος της εταιρείας, όπου έχουν πρόσβαση οι πελάτες. Εκεί βλέπουν τη λίστα με τα εγκεκριμένα performance και financial reports, με δυνατότητα προβολής και αποθήκευσης κάθε αρχείου τοπικά. Έτσι ενημερώνονται για τα έσοδα και την πρόοδό τους κάθε μήνα.

11.2 Χρήστες της εφαρμογής Portal

Στην ενότητα αυτή θα δοθεί μια ενδελεχής περιγραφή των χρηστών που διαθέτουν πρόσβαση στην ψηφιακή πλατφόρμα Portal, αναλύοντας τον ρόλο, τις λειτουργείες και τις δυαντότητές τους εντός της εφαρμογής.

11.2.1 Μέλη της εταιρείας

Τα μέλη της εταιρείας αποτελούν τους κύριους χρήστες των εταιρικών εφαρμογών, καθώς τις χρησιμοποιούν σε καθημερινή βάση για την διεκπεραίωση των εργασιών και των αιτημάτων που τους έχουν ανατεθεί. Κάθε εργαζόμενος διαθέτει ένα εταιρικό Gmail, μέσω του οποίου έχει πρόσβαση στα εταιρικά συστήματα. Κάθε μέλος έχει δικαίωμα πρόσβασης σε συγκεκριμένες πλατφόρμες, τμήματα εφαρμογών, δεδομένα και λειτουργίες, ανάλογα με τον ρόλο του μέσα στην εταιρεία.

Για την καλύτερη κατανόηση των χρηστών του Internal Portal, θα πάρουμε ως παράδειγμα τους εργαζομένους δύο τμημάτων της εταιρείας και συγκεκριμένα του creators management department και του λογιστηρίου.

Οι εργαζόμενοι του τμήματος creators management, γνωστοί και ως account managers, έχουν συμβουλευτικό ρόλο και διαχειρίζονται έναν συγκεκριμένο αριθμό πελατών. Βρίσκονται σε διαρκή επικοινωνία με τους πελάτες καθώς είναι υπεύθυνοι για την διαχείριση των καναλιών και των προφίλ τους στα κοινωνικά δίκτυα, οφείλουν να τους συμβουλεύουν και να τους κατευθύνουν, ώστε να εξελίσσουν συνεχώς το περιεχόμενο που προβάλλουν και να τους ενημερώνουν τακτικά για την πρόοδό τους. Τέλος, είναι υπεύθυνοι για τη διαχείριση των δεδομένων για την παραγωγή των performance reports των πελατών.

Κάθε account manager, κατά τη σύνδεσή του στο Internal Portal, βλέπει τα οικονομικά δεδομένα, τα στοιχεία προόδου, τα financial και performance reports των πελατών που ο ίδιος διαχειρίζεται. Η δημιουργία, ο έλεγχος, η επεξεργασία και η έγκριση των παρουσιάσεων προόδου των πελατών, αποτελούν κάποιες από τις βασικές αρμοδιότητες ενός account manager και μπορεί να τις φέρει εις πέρας μέσω των χρήσιμων εργαλείων που του παρέχει το Internal Portal.

Οι υπάλληλοι του λογιστηρίου αναλαμβάνουν να διαχειριστούν τα οικονομικά της εταιρείας, εστιάζοντας στις ειδικές ανάγκες και απαιτήσεις που προκύπτουν από τον χαρακτήρα του digital marketing. Αυτό συμπεριλαμβάνει την παρακολούθηση των εσόδων και των εξόδων που σχετίζονται με τις δραστηριότητες της εταιρείας στον ψηφιακό τομέα, τις διαφημίσεις της στα μέσα κοινωνικής δικτύωσης, την ανάλυση των κερδών της από διαφημιστικές καμπάνιες που η ίδια αναλαμβάνει, καθώς και την προετοιμασία και υποβολή φορολογικών δηλώσεων. Επιπλέον, αναλαμβάνουν τη δημιουργία και ανάλυση οικονομικών αναφορών, που βοηθούν τη διοίκηση της εταιρείας στη λήψη στρατηγικών αποφάσεων.

Κατά τη σύνδεσή τους στο Internal Portal, τα μέλη του λογιστηρίου βλέπουν μόνο τα οικονομικά δεδομένα και τις παρουσιάσεις των πελατών. Σε αντίθεση με τους account managers, δεν έχουν πρόσβαση σε στοιχεία προόδου και performance reports, καθώς δεν είναι χρήσιμα για τον ρόλο τους.

Συμπερασματικά, έχει καταστεί σαφές ότι με τη βοήθεια του RBAC, κάθε μέλος της εταιρείας έχει πρόσβαση σε συγκεκριμένα τμήματα, λειτουργίες και δεδομένα της ψηφιακής πλατφόρμας Internal Portal, ανάλογα με το ρόλο του. Αυτό είναι ένα απαραίτητο στοιχείο για τη διαφύλαξη των ευαίσθητων πληροφοριών και εταιρικών δεδομένων.

11.2.2 Πελάτες

Το Customer Portal αποτελεί την μοναδική πλατφόρμα της εταιρείας που έχουν πρόσβαση οι πελάτες. Εκεί βλέπουν τη λίστα με τα εγκεκριμένα performance και financial reports, με δυνατότητα προβολής και αποθήκευσης κάθε αρχείου τοπικά. Έτσι ενημερώνονται για τα έσοδα και την πρόοδό τους κάθε μήνα.

Οι πελάτες μπορεί να είναι είτε φυσικά πρόσωπα, μεμονωμένες οντότητες δηλαδή, είτε εταιρείες οι οποίες διαχειρίζονται έναν μεγάλο αριθμό ατόμων.

Οι μεν είναι ελεύθεροι επαγγελματίες που διαθέτουν προφίλ στα μέσα κοινωνικής δικτύωσης, όπως το Instagram, το Facebook και το TikTok, κανάλια στο YouTube ή record labels στις μεγαλύτερες πλατφόρμες ροής μουσικής, όπως το Spotify, στο iTunes, το Deezer, το Amazon Prime και άλλα.

Κάθε ένας από τους παραπάνω πελάτες, αποτελεί και ξεχωριστό χρήστη, με δικαίωμα πρόσβασης στο Customer Portal.

Οι δε είναι ολόκληρες εταιρείες και οργανισμοί, που απασχολούν ένα πλήθος εργαζομένων, οι οποίοι αποτελούν με τη σειρά τους χρήστες του Customer Portal. Ανάλογα με τον ρόλο τους στην εταιρεία-πελάτη, βλέπουν συγκεκριμένα δεδομένα στην εφαρμογή, που ανταποκρίνονται ειδικά στις ανάγκες τους. Παραδείγματος χάριν, κάποιοι χρήστες μπορεί να έχουν πρόσβαση σε λογιστικές πληροφορίες και οικονομικά δεδομένα, ενώ άλλοι μόνο σε στοιχεία προόδου και performance reports.

11.3 Υλοποίηση της εφαρμογής Portal

Στην ενότητα αυτή θα αναλύσουμε τα τμήματα από τα οποία αποτελείται η εφαρμογή Portal, καθώς και όλες τις λειτουργίες που μπορούν να εκτελέσουν οι χρήστες στις επιμέρους σελίδες της.

11.3.1 Login Screen

Η ψηφιακή πλατφόρμα Portal που υλοποιήθηκε στην παρούσα εργασία, περιλαμβάνει μια σελίδα σύνδεσης χρήστη (Login page), η οποία αποτελεί την πρώτη διαδικασία πιστοποίησης (authentication) και εξουσιοδότησης (authorization) των χρηστών που επιθυμούν πρόσβαση στο σύστημα. Η εφαρμογή απευθύνεται τόσο στους πελάτες, όσο και στα μέλη της εταιρείας. Οι μεν συνδέονται με το email και τον κωδικό πρόσβασης που τους έχει αποδώσει ο διαχειριστής του RBAC και οι δε μέσω του εταιρικού τους λογαριασμού Google.

Στην σελίδα σύνδεσης φαίνεται με ευδιάκριτο τρόπο το λογότυπο της εταιρείας και η ίδια αποτελείται από δύο τμήματα. Το πρώτο τμήμα αφορά τους πελάτες και περιλαμβάνει δύο πεδία συμπλήρωσης των στοιχείων τους, email και κωδικού πρόσβασης, ένα κουμπί "Login" για την πραγματοποίηση της σύνδεσής τους στην πλατφόρμα, καθώς και ένα κουμπί "Forgot Password?". Το εν λόγω κουμπί δίνει τη δυνατότητα σε έναν πελάτη να ανανεώσει τον κωδικό πρόσβασής του σε περίπτωση που ο ίδιος τον ξεχάσει. Η διαδικασία αλλαγής password είναι πολύ απλή. Ο χρήστης πατώντας το κουμπί "Forgot Password?", συμπληρώνει το email του σε μία εμφανιζόμενη φόρμα και του αποστέλλεται αυτόματο μήνυμα αλλαγής κωδικού από την firebase. Το μόνο που πρέπει να κάνει, είναι να ανοίξει τον σύνδεσμο που στάλθηκε στο email του και να συμπληρώσει τον νέο κωδικό πρόσβασης.

Το δεύτερο τμήμα της σελίδας Login, αφορά τα μέλη της εταιρείας και περιλαμβάνει ένα κουμπί "Sign in as manager". Το εν λόγω κουμπί επιτρέπει στους εργαζομένους να συνδεθούν στην πλατφόρμα, χρησιμοποιώντας το εταιρικό τους Gmail. Η χρήση του λογαριασμού Google ως μηχανισμού ελέγχου ταυτοποίησης εξασφαλίζει μια γρήγορη, ασφαλή και ευέλικτη διαδικασία εισόδου στην εφαρμογή, ενώ ταυτόχρονα διασφαλίζει την προστασία των δεδομένων και την εμπιστευτικότητα των πληροφοριών του χρήστη.

Κατά την είσοδο ενός υπαλλήλου στο σύστημα, ο λογαριασμός του ελέγχεται αρχικά από την υπηρεσία της Google, μέσω firebase authentication, ενώ στη συνέχεια γίνεται εσωτερικός έλεγχος των δικαιωμάτων πρόσβασης στην εφαρμογή, βάσει του ρόλου του στην εταιρεία. Αντιστοίχως, με τη σύνδεση ενός πελάτη στην πλατφόρμα, πραγματοποιείται έλεγχος των στοιχείων του μέσω του firebase authentication και ακολουθεί εσωτερικός έλεγχος των permissions του, ώστε να του εμφανίζονται μόνο τα οικονομικά δεδομένα και τα στοιχεία προόδου των πελατών που δικαιούται να βλέπει. Οι παραπάνω έλεγχοι εξασφαλίζουν ότι ο κάθε χρήστης έχει πρόσβαση μόνο στις πλατφόρμες, τις λειτουργίες και τα δεδομένα που του επιτρέπονται.

Η ενσωμάτωση σελίδων σύνδεσης (Login pages) στις εφαρμογές της εταιρείας, αποτελεί σημαντικό βήμα προς τη δημιουργία ενός ασφαλούς και ελεγχόμενου συστήματος πρόσβασης σε εταιρικά δεδομένα και λειτουργίες. Η επιλογή του firebase authentication ως μηχανισμού ελέγχου ταυτοποίησης, προσφέρει την επιθυμητή ισορροπία μεταξύ ασφάλειας, ευκολίας χρήσης και ευελιξίας για τους χρήστες.

Figure 18: Login Screen της εφαρμογής Portal

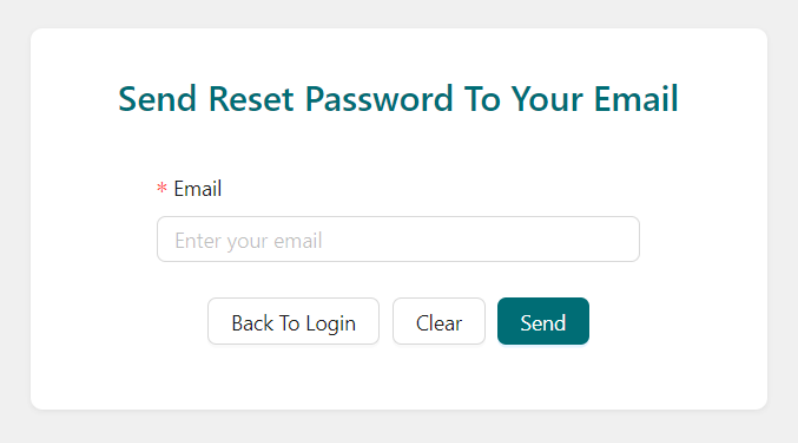


Figure 19: Φόρμα αποστολής email για ανανέωση του κωδικού πρόσβασης

11.3.2 Navigation Bar

Η ψηφιακή πλατφόρμα Portal, διαθέτει ένα Navigation Bar στο πάνω μέρος της σελίδας, το οποίο αποτελεί το κεντρικό μενού πλοήγησης του χρήστη μέσα στην εφαρμογή. Στην περίπτωση που ο συνδεδεμένος χρήστης είναι πελάτης, το μοναδικό tab του Navigation Bar είναι το "Reports", όπου ο ίδιος μπορεί να δει όλα τα εγκεκριμένα performance και financial reports που τον αφορούν, με δυνατότητα προβολής και αποθήκευσης κάθε αρχείου τοπικά στον υπολογιστή του. Στην περίπτωση σύνδεσης ενός εργαζομένου, το Navigation Bar περιέχει το tab "File Approval", όπου εμφανίζεται μια λίστα με το σύνολο των παραγόμενων reports των πελατών που ο ίδιος διαχειρίζεται, με δυνατότητα αποθήκευσης, έγκρισης και προβολής κάθε αρχείου στο Google Drive.

Στο δεξιό μέρος του Navigation Bar, βρίσκεται το εικονίδιο του συνδεδεμένου χρήστη (User Profile Icon). Όσον αφορά το Customers Portal, ο πελάτης κάνοντας hover πάνω από το εικονίδιό του, βλέπει ένα μικρό παράθυρο με το email του, ένα κουμπί "Logout" για αποσύνδεση από την εφαρμογή, καθώς και ένα κουμπί "Change Password". Το εν λόγω κουμπί, δίνει την δυνατότητα στον πελάτη να αλλάξει τον κωδικό του. Η παραπάνω διαδικασία είναι πολύ απλή. Ο χρήστης πατώντας το κουμπί "Change Password", μεταβαίνει σε μία φόρμα της firebase, όπου συμπληρώνει τον νέο κωδικό πρόσβασης στο αντίστοιχο πεδίο "New password" και στη συνέχεια τον αποθηκεύει με το πάτημα ενός κουμπιού "Save". Στην περίπτωση του Internal Portal, ο συνδεδεμένος στην πλατφόρμα χρήστης, είναι μέλος της εταιρείας. Κάνοντας hover πάνω από το εικονίδιό του, εμφανίζεται ένα μικρό παράθυρο με τα στοιχεία του, όνομα και εταιρικό email, καθώς και ένα κουμπί "Logout" για αποσύνδεση από την εφαρμογή.

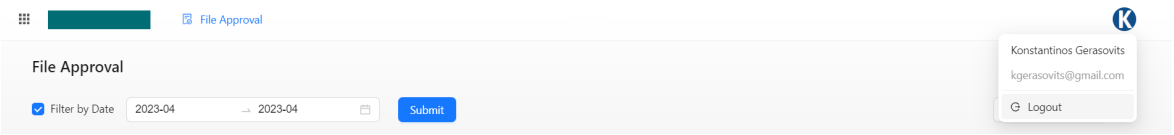


Figure 20: Navigation Bar από την σκοπιά ενός υπαλλήλου

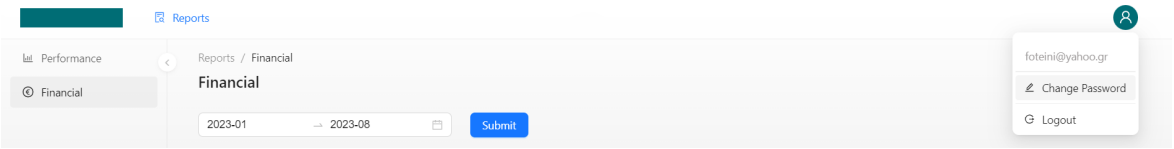


Figure 21: Navigation Bar από την σκοπιά ενός πελάτη

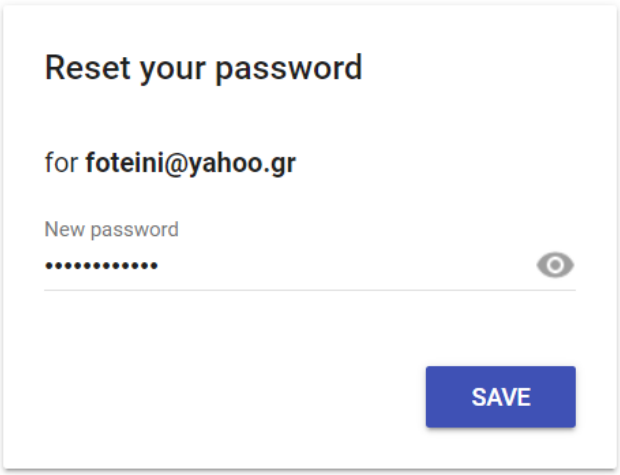


Figure 22: Φόρμα ανανέωσης κωδικού πρόσβασης του Customer Portal

11.3.3 Reports

Η σελίδα "Reports" αποτελεί το βασικό interface του Customer Portal και απευθύνεται αποκλειστικά στους πελάτες της εταιρείας. Μέσω της συγκεκριμένης σελίδας, οι ίδιοι έχουν πρόσβαση στα εγκεκριμένα financial και performance reports τους, με δυνατότητα προβολής των παρουσιάσεων στο Google Drive, καθώς και αποθήκευσης κάθε αρχείου τοπικά στον υπολογιστή τους.

Στο αριστερό μέρος της σελίδας, βρίσκεται ένα μενού με δύο επιλογές, "Performance" και "Financial", μέσω των οποίων ο χρήστης επιλέγει ποιο από τα δύο είδη report θα δει. Ανάλογα με το αν ο πελάτης έχει δικαίωμα πρόσβασης μόνο σε οικονομικά στοιχεία, μόνο σε στοιχεία προόδου ή και στα δύο, βλέπει και τις αντίστοιχες επιλογές στο μενού πλοήγησης.

Και στα δυο τμήματα που προαναφέρθηκαν, περιλαμβάνεται ένας πίνακας με όλες τις εγκεκριμένες παρουσιάσεις που έχει εκδώσει η εταιρεία προς ενημέρωση του εκάστοτε πελάτη, ταξινομημένες με βάση την ημερομηνία έκδοσής τους, σε φθίνουσα σειρά. Ο χρήστης, πατώντας το όνομα, το εικονίδιο του Google Drive ή τον τύπο του αρχείου, μπορεί να μεταβεί στην παρουσίαση, η οποία βρίσκεται αποθηκευμένη σε έναν ασφαλή χώρο στο cloud, όπου ο ίδιος έχει αποκλειστική πρόσβαση.

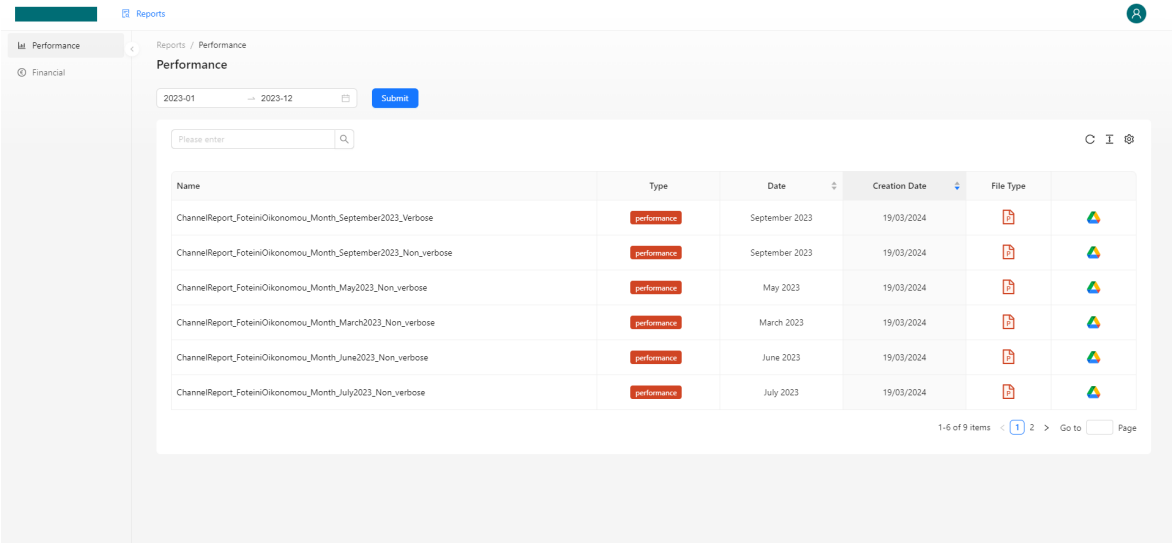


Figure 23: Λίστα των performance reports στην σελίδα "Reports" του Customer Portal

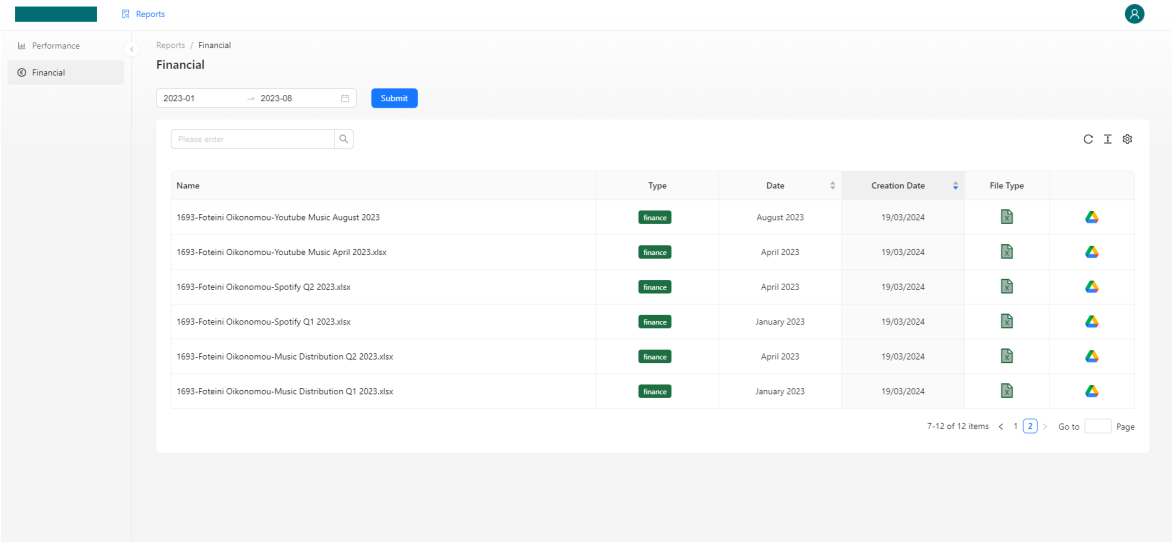


Figure 24: Λίστα των financial reports στην σελίδα "Reports" του Customer Portal

11.3.4 File Approval

Η σελίδα "File Approval" αποτελεί το βασικό interface του Internal Portal και απευθύνεται αποκλειστικά στα μέλη της εταιρείας που διαχειρίζονται έναν συγκεκριμένο αριθμό πελατών. Μέσω της συγκεκριμένης σελίδας, οι account managers έχουν πρόσβαση στο σύνολο των financial και performance reports των πελατών για τους οποίους είναι υπεύθυνοι, με δυνατότητα προβολής, επεξεργασίας και έγκρισης των παρουσιάσεων στο Google Drive, καθώς και αποθήκευσης κάθε αρχείου τοπικά στον υπολογιστή τους.

Η επεξεργασία ενός αρχείου, αφορά τόσο τα metadata του, όπως το όνομα, τον τύπο, τον μήνα και την χρονιά στην οποία αναφέρεται, καθώς και τα οικονομικά δεδομένα και τα γραφήματα στο εσωτερικό των παρουσιάσεων και των υπολογιστικών φύλλων. Η επεξεργασία και εν συνεχεία η έγκριση των reports αποτελούν δύο κύριες αρμοδιότητες ενός account manager, καθώς μόνο οι εγκεκριμένες παρουσιάσεις είναι ορατές στους πελάτες.

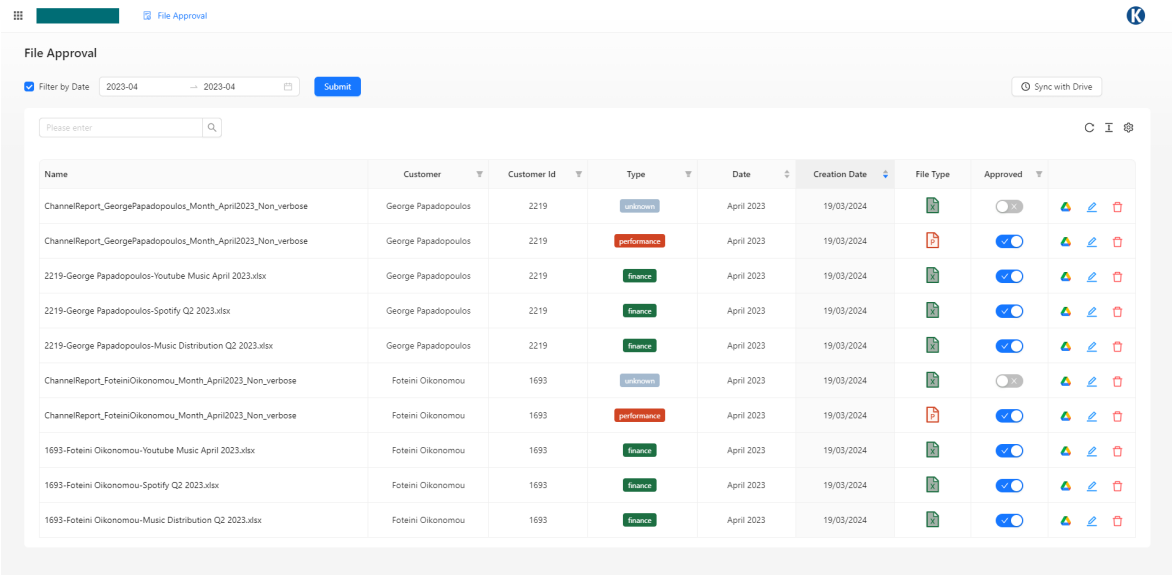


Figure 25: Σελίδα "File Approval" του Internal Portal

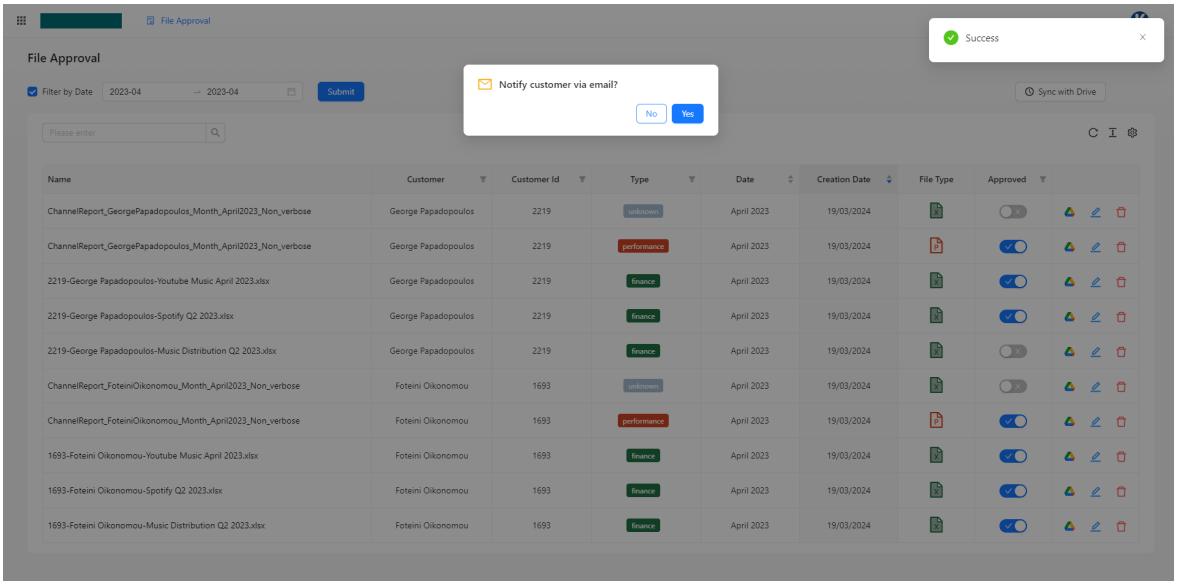


Figure 26: Έγκριση ενός αρχείου

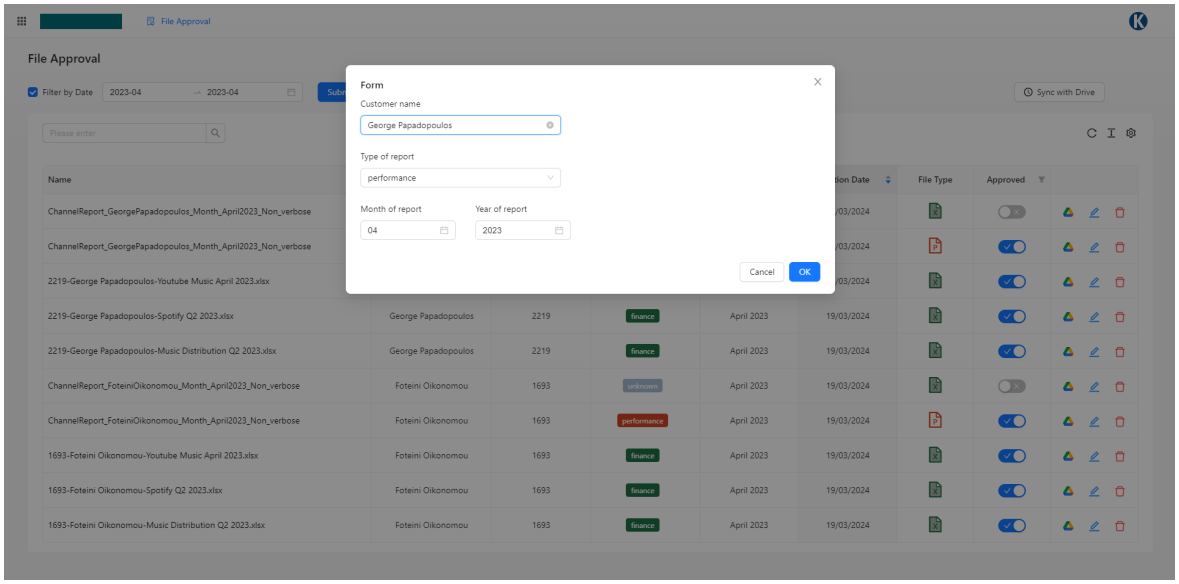


Figure 27: Επεξεργασία των metadata ενός αρχείου

12 Συμπεράσματα

Το σύστημα ελέγχου πρόσβασης με βάση τους ρόλους (RBAC) που υλοποιήθηκε στην παρούσα εργασία, προέκυψε από την ανάγκη για ασφάλεια των εταιρικών συστημάτων ενός πραγματικού εργασιακού περιβάλλοντος και όπως αποδείχθηκε, αποτέλεσε την πλέον κατάλληλη επιλογή για τον έλεγχο πρόσβασης στις εφαρμογές της εν λόγω εταιρείας digital marketing.

Ένα από τα βασικότερα στοιχεία που χαρακτηρίζουν το σύστημα RBAC, είναι η ευελιξία. Όπως γνωρίζουμε, ο διαχειριστής του μπορεί εύκολα να επεξεργαστεί τους ρόλους των εργαζομένων, να προσθέσει ή να αφαιρέσει δικαιώματα πρόσβασης (permissions), καθώς και να εντάξει νέα άτομα στο σύστημα, αποδίδοντάς τους δικαιώματα πρόσβασης ανάλογα με τη θέση τους στην εταιρεία. Συνεπώς, όταν υπάρξει ανάγκη για εσωτερική αλλαγή στην δομή ενός οργανισμού, το σύστημα RBAC είναι ικανό να ακολουθήσει, σε πραγματικό χρόνο, οποιαδήποτε αλλαγή και να την αποτυπώσει στο σύστημα, με ευκολία στο χειρισμό και γρήγορη ανταπόκριση στην εκάστοτε συνθήκη.

Επιπλέον, το RBAC προσφέρει μεγάλη ελευθερία στον καθορισμό των ρόλων και των δικαιωμάτων πρόσβασης. Ανάλογα με τους χρήστες του εκάστοτε εταιρικού συστήματος που καλείται να ασφαλίσει, ο διαχειριστής του RBAC καθορίζει τους διαφορετικούς ρόλους των χρηστών και τα permissions που τους αντιστοιχούν, δίνοντάς τους την κατάλληλη πρόσβαση στα επιτρεπόμενα τμήματα, τις λειτουργίες και τα δεδομένα που τους αφορούν.

Από τα παραπάνω, συμπεραίνουμε ότι το RBAC είναι ένα σύστημα που μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον. Η ευελιξία και η ελευθερία που προσφέρει, το καθιστούν ικανό να τεθεί σε ισχύ σε οποιοδήποτε εταιρικό σύστημα, ανεξάρτητα από τις ιδιαιτερότητες ή την πολυπλοκότητα στην δομή του. Ακόμα και μια εταιρεία με μεταβαλλόμενη δομή και μέγεθος, μπορεί να εξυπηρετήσει τις ανάγκες της με τη βοήθεια του RBAC, καθώς είναι ικανό να ακολουθήσει οποιαδήποτε αλλαγή, αλλάζοντας κι αυτό με σειρά του τους ρόλους, την ιεραρχία και τα δικαιώματα των χρηστών μέσα στο σύστημα. Συνεπώς, το RBAC απευθύνεται τόσο σε μικρομεσαίες επιχειρήσεις, όσο και σε μεγάλες πολυεθνικές, οι οποίες μπορούν να επωφεληθούν το ίδιο από τη χρήση ενός συστήματος ελέγχου πρόσβασης, βασισμένο στους ρόλους των μελών τους.

13 Βιβλιογραφικές Πηγές

[1] A. Papageorgiou, M. Strigkos, E. Politou, E. Alepis, A. Solanas, and C. Patsakis, "Security and privacy analysis of mobile health applications: The alarming state of practice," *IEEE Access*, vol. 6, pp. 9390–9403, 2018. DOI: 10.1109/ACCESS.2018.2799522.

[2] StrongDM, *The Definitive Guide to Role-Based Access Control (RBAC)*. [Online]. Available: https://www.strongdm.com/rbac.

[3] D. Ferraiolo and D. Kuhn, "Role-Based Access Controls. 15th National Computer Security Conf," 1992.

[4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, Revised October 26, 1995.

[5] R. Sandhu, D. Ferraiolo, R. Kuhn, *et al.*, "The NIST Model for Role-Based Access Control: Towards A Unified Standard," in *ACM workshop on Role-based access control*, vol. 10, 2000.

[6] A. INCITS, "INCITS 359-2004," *Role Based Access Control*, 2004.

[7] INCITS, "INCITS 359-2012," *Role Based Access Control*, 2012.

[8] I. T. L. Computer Security Division, *Role Based Access Control: CSRC*, Jun. 2020. [Online]. Available: https://csrc.nist.gov/projects/role-based-access-control.

[9] A. C. O'Connor and R. J. Loomis, "2010 Economic Analysis of Role-Based Access Control," *NIST, Gaithersburg, MD*, vol. 20899, 2010.

[10] E. Politou, E. Alepis, M. Virvou, and C. Patsakis, *Privacy and data protection challenges in the distributed era*. Springer, Oct. 2022.

[11] GfG, *ReactJS Virtual DOM*, Nov. 2023. [Online]. Available: https://www.geeksforgeeks.org/reactjs-virtual-dom/.

[12] React, *React useState*. [Online]. Available: https://react.dev/reference/react/useState.

[13] React, *React useEffect*. [Online]. Available: https://react.dev/reference/react/useEffect.

[14] Axios, *Getting started — Axios Docs*. [Online]. Available: https://axios-http.com/docs/intro.

[15] Axios, *POST Requests — Axios Docs*. [Online]. Available: https://axios-http.com/docs/post_example.

[16] Axios, *Interceptors — Axios Docs*. [Online]. Available: https://axios-http.com/docs/interceptors.

[17] Axios, *Handling Errors — Axios Docs*. [Online]. Available: https://axios-http.com/docs/handling_errors.

[18] Axios, *Cancellation — Axios Docs*. [Online]. Available: https://axios-http.com/docs/cancellation.

[19] Κ. Θραμπουλίδης, *Διαδικαστικός Προγραμματισμός - C (Τόμος Α)*, 2η έκδοση. Εκδόσεις ΤΖΙΟΛΑ, 2002.

[20] Go, *Frequently Asked Questions (FAQ) — What are Go's ancestors?* [Online]. Available: https://go.dev/doc/faq#ancestors.

[21] C. A. R. Hoare *et al.*, *Communicating Sequential Processes*. Prentice-hall Englewood Cliffs, 1985, vol. 178.

[22] Go, *Frequently Asked Questions (FAQ) — Is Go an object-oriented language?* [Online]. Available: https://go.dev/doc/faq#Is_Go_an_object-oriented_language.

[23] *Type Hierarchy*. [Online]. Available: https://www.eecs.qmul.ac.uk/~mmh/AMCM048/inheritance/hierarchy.html.

[24] GfG, *Interfaces in Golang*. [Online]. Available: https://www.geeksforgeeks.org/interfaces-in-golang/amp/.

[25] Go, *A Tour of Go — Interface values*. [Online]. Available: https://go.dev/tour/methods/11.

[26] *Concurrent vs Parallel Programming: What's the Difference?* [Online]. Available: https://www.linkedin.com/advice/0/whats-difference-between-concurrent-parallel-programming.

[27] Go, *A Tour of Go — Goroutines*. [Online]. Available: https://go.dev/tour/concurrency/1.

[28] Go, *A Tour of Go — Channels*. [Online]. Available: https://go.dev/tour/concurrency/2.

[29] Go, *A Guide to the Go Garbage Collector - The Go Programming Language*. [Online]. Available: https://tip.golang.org/doc/gc-guide.
[30] Gin, *Documentation — Gin Web Framework*. [Online]. Available: https://gin-gonic.com/docs/.
[31] Go, *Martini Package - Go Packages*. [Online]. Available: https://pkg.go.dev/gopkg.in/martini.v0#section-readme.
[32] Go, *Handlers — Martini Package*. [Online]. Available: https://pkg.go.dev/gopkg.in/martini.v0#readme-handlers.
[33] Go, *Routing — Martini Package*. [Online]. Available: https://pkg.go.dev/gopkg.in/martini.v0#readme-routing.
[34] Go, *Middleware Handlers — Martini Package*. [Online]. Available: https://pkg.go.dev/gopkg.in/martini.v0#readme-middleware-handlers.
[35] Casbin, *Overview — Casbin*. [Online]. Available: https://casbin.org/docs/overview/.
[36] Go, *GORM Package - Go Packages*. [Online]. Available: https://pkg.go.dev/gorm.io/gorm#section-readme.
[37] GORM, *GORM - The fantastic ORM library for Golang, aims to be developer friendly*. [Online]. Available: https://gorm.io/.
[38] O. S. Initiative, *GNU General Public License version 2*, Jun. 2023. [Online]. Available: https://opensource.org/license/gpl-2-0.
[39] MariaDB, *MariaDB in brief - MariaDB.org*. [Online]. Available: https://mariadb.org/en/.
[40] R. T. Logic, *What is an Embedded Application Server?* [Online]. Available: https://realtimelogic.com/articles/What-is-an-Embedded-Application-Server.
[41] Cisco, *What Is High Availability? - Cisco*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/hybrid-work/what-is-high-availability.html.
[42] IBM, *Transactional high availability - IBM Documentation*. [Online]. Available: https://www.ibm.com/docs/en/was-nd/9.0.5?topic=server-transactional-high-availability.
[43] Firebase, *Firebase Authentication*. [Online]. Available: https://firebase.google.com/docs/auth.