ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία ιστότοπου Web Banking

Γεράσιμος Θ. Κοτσοβός

Εισηγητής: Ματιάτος Σπυρίδων, Καθηγητής Εφαρμογών

ΑΘΗΝΑ

ΜΑΙΟΣ 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία ιστότοπου Web Banking

Γεράσιμος Θ. Κοτσοβός

Α.Μ. 39143

Εισηγητής:

Ματιάτος Σπυρίδων, Καθηγητής Εφαρμογών

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γεράσιμος Κοτσοβός, του Θεόφιλου, με αριθμό μητρώου 39143 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, αρκετές αλλαγές, δοκιμές και συζητήσεις για το καλύτερο δυνατό αποτέλεσμα. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, κ. Σπυρίδων Ματιάτος, τον οποίο θα ήθελα να ευχαριστήσω. Επίσης θα ήθελα να ευχαριστήσω την οικογένειά, τους φίλους και τους συναδέλφους μου για την υποστήριξη και την υπομονή που υπέδειξαν καθ' όλη τη διάρκεια της εκπόνησης αυτής της εργασίας.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή έχει ως θέμα την δημιουργία μίας ιστοσελίδας Web Banking, με σκοπό την κατανόηση της δυσκολίας ανάπτυξης μιας πλήρους εφαρμογής που έχει ιδιαίτερα υψηλές προδιαγραφές ασφάλειας και λειτουργικότητας.

Οι κύριες εργασίες περιλαμβάνουν την ανάπτυξη της ιστοσελίδας (front-end), των τραπεζικών συναλλαγών (back-end), της διασύνδεσης μεταξύ των δύο (API) και μίας βάσεως δεδομένων. Ιδιαίτερη προσοχή θα δοθεί στην ταυτοποίηση (authentication) των χρηστών και στην προστασία των δεδομένων, χρησιμοποιώντας κρυπτογράφηση (encryption) σε όλα τα δεδομένα του χρήστη, στις συναλλαγές και τους κωδικούς. Επίσης θα υπάρχει φιλτράρισμα (filtering) ώστε να προστατευθεί η εφαρμογή από τυχόν κακόβουλες επιθέσεις στην βάση δεδομένων (injection).

Θα χρησιμοποιηθούν σύγχρονα εργαλεία και αρχιτεκτονικές (React.js, Redux, REST) και γενικά θα καταβληθεί προσπάθεια, ώστε η ιστοσελίδα να είναι σωστά δομημένη, εύχρηστη, λειτουργική, γρήγορη και εύκολα επεκτάσιμη.

ABSTRACT

The subject of my thesis was the development of a Web Banking website, with the purpose of understanding the complexity of the development of a full-stack application consisting of high-end specifications of security and functionality.

The main process consists of the development of the website (front-end), the banking transactions (back-end), the connection between the two (API) and a database. Special attention will be given on the authentication and the protection of the user’s data, using encryption throughout the application. Furthermore, the database will be protected from injections.

Modern tools and technologies will be used (React.js, Redux, REST) and an effort will be made in order the website to be clean, easy to use, fast and easily extendable.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Πλήρη (full-stack) web εφαρμογή.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ιστοσελίδα, τραπεζικές συναλλαγές, αρχιτεκτονική τριών επιπέδων.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

HTTP Hypertext Transfer Protocol

REST Representational State Transfer

API Application Programming Interface

JSON JavaScript Object Notation

JWT JSON Web Token

AJAX Asynchronous JavaScript and XML

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια ιστορική αναδρομή σχετικά με την εξέλιξη των τραπεζικών συναλλαγών.

1.1. Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η παρούσα πτυχιακή έχει ως θέμα την ανάπτυξη ιστοσελίδας web banking, μιας μορφής ηλεκτρονικών τραπεζικών συναλλαγών που έχει εξελιχθεί ραγδαία τα τελευταία χρόνια. Η υλοποίηση έγινε με σκοπό την κατανόηση της δυσκολίας ανάπτυξης μιας πλήρους εφαρμογής που έχει ιδιαίτερα υψηλές προδιαγραφές ασφάλειας, λειτουργικότητας καθώς και την ανάγκη της ευκολίας χρήσης από οποιοδήποτε χρήστη.

1.2. Ιστορική αναδρομή

Σε αυτό το υπό κεφάλαιο θα γίνει θα γίνει μια ιστορική αναδρομή στην εξέλιξη των τραπεζικών συναλλαγών.

1.2.1. Παραδοσιακές τραπεζικές συναλλαγές

Βασικές μορφές δανεισμού και καταθέσεων με επιτόκια άρχισαν να εμφανίζονται στις απαρχές του Δυτικού πολιτισμού ήδη από το 3.000 π.Χ. σε ναούς και βασιλικά θησαυροφυλάκια στη Μεσοποταμία. Πριν τη βασιλεία του Σαργώντος του Μέγα (2335-2280 π.Χ.) οι συναλλαγές ήταν περιορισμένες στα όρια της κάθε πόλης της Βαβυλώνας και στο ναό που βρισκόταν στο επίκεντρο της οικονομίας της καθώς οι συναλλαγές με ξένους απαγορεύονταν.[1]

Στην αρχαία Ελλάδα αναπτύχθηκαν οχυρά στα οποία φυλάσσονταν πολύτιμα αγαθά σε περιόδους πολέμου ή αστάθειας. Σε ιδιωτικές και κοινές μονάδες μέσα στην αρχαία Ελληνική κοινωνία - κυρίως σε ναούς όπως ο Παρθενώνας, ο ναός της Αρτέμιδος στην Έφεσο, το Ηραίον της Σάμου και των Δελφών και ο ναός του

Απόλλωνα- λάμβαναν χώρα οικονομικές συναλλαγές όπως καταθέσεις, αλλαγή και επικύρωση νομισμάτων καθώς και δάνεια.[2]

Στη Ρωμαϊκή Αυτοκρατορία εντοπίζεται μια απλοποίηση των τραπεζικών διαδικασιών και η αυγή των καταστημάτων-χρήματος, παρόμοια με τα υποκαταστήματα των σύγχρονων τραπεζών. Περισσότεροι πολίτες είχαν τη δυνατότητα να ανταλλάσσουν χρήματα, να παίρνουν δάνεια και να καταθέτουν. Με τη πτώση της Ρωμαϊκής αυτοκρατορίας οι οργανώσεις αυτές χάθηκαν και δεν αναπτύχθηκαν παρόμοιες διαδικασίες μέχρι και την περίοδο του Μεσαίωνα.[3]

Το 1587 ανοίγει η Banco della Piazza di Rialto στη Βενετία ως κρατική πρωτοβουλία. Σκοπός της είναι να διεκπεραιώσει τη σημαντική λειτουργία της κατοχής κεφαλαίων των εμπόρων σε ασφαλείς καταθέσεις και να επιτρέψει την πραγματοποίηση χρηματοοικονομικών συναλλαγών στη Βενετία και αλλού χωρίς τη φυσική μεταφορά κερμάτων. Η πρωτοβουλία των Βενετών, είναι μια προσπάθεια να προσφέρει ένα μέτρο ασφάλειας στις εμπορικές συναλλαγές. Αυτή η πολύ απλουστευμένη έκδοση χρεωστικής γραμμής κερδίζει αργά την αποδοχή από τα τέλη του 17ου αιώνα.[4]

Κατά τη διάρκεια του 18ου αιώνα, η Τράπεζα της Αγγλίας αναλαμβάνει σταδιακά πολλά από τα καθήκοντα που συνδέονται τώρα με μια κεντρική τράπεζα. Οργανώνει την πώληση κρατικών ομολόγων όταν χρειαζόταν κεφάλαια, λειτουργεί ως τράπεζα εκκαθάρισης για τις κυβερνητικές υπηρεσίες, διευκολύνοντας και επεξεργάζοντας τις καθημερινές συναλλαγές τους. Η Τράπεζα της Αγγλίας γίνεται επίσης τραπεζίτης σε άλλες τράπεζες του Λονδίνου, και μέσω αυτών σε μια πολύ ευρύτερη τραπεζική κοινότητα. Οι τράπεζες του Λονδίνου ενεργούν ως πράκτορες στην πρωτεύουσα για τις πολλές μικρές ιδιωτικές τράπεζες που ανοίγουν γύρω από τη χώρα στο δεύτερο μισό του 18ου αιώνα.[4]

Σήμερα, οι τράπεζες είναι παντού και προσβάσιμες από όλους. Η γκάμα των τραπεζικών συναλλαγών έχει απλουστευτεί και πολλαπλασιαστεί, και περιλαμβάνει μεταξύ άλλων: καταθέσεις, αναλήψεις, δανεισμούς, πληρωμές κάθε είδους με χρήση μετρητών, επιταγών, καρτών. Το αποτέλεσμα είναι εκατομμύρια τραπεζικές συναλλαγές να πραγματοποιούνται καθημερινά στα γκισέ και όχι μόνο των τραπεζών παγκοσμίως, από μικροποσά έως δισεκατομμύρια ευρώ.

1.2.2. Η αρχή των ηλεκτρονικών τραπεζικών συναλλαγών

Παρόλη την εξέλιξη και την προσβασιμότητα των τραπεζών από όλους, η παραδοσιακή πραγματοποίηση τραπεζικών συναλλαγών είναι μία κουραστική και πολύωρη διαδικασία για τους πελάτες αλλά και για την τράπεζα. Με την εξέλιξη όμως της τεχνολογίας και της πληροφορικής δημιουργήθηκαν τραπεζικά εργαλεία με τα οποία διευκόλυναν την διαδικασία των τραπεζικών συναλλαγών. Τα προϊόντα αυτά προωθούνται από τις τράπεζες προκειμένου να διευκολύνουν κατά κύριο λόγο τους πελάτες τους, αλλά και τις ίδιες.

Η ιδέα για την κατασκευή ενός μηχανήματος, που θα διευκόλυνε τις συναλλαγές εκτός του ωραρίου των τραπεζικών υποκαταστημάτων, ανήκε σε πολλούς τραπεζίτες σε Ανατολή και Δύση και άρχισε να διαμορφώνεται μεταπολεμικά με την ανάπτυξη των ηλεκτρονικών υπολογιστών. Στις 30 Ιουνίου 1960, ο αμερικανο-αρμένιος φωτογράφος και εφευρέτης Λούθερ Σίμτζιαν (1905-1997) κατέθεσε αίτηση ευρεσιτεχνίας για μία αυτόματη μηχανή ανάληψης μετρητών με την ονομασία Bankograph, την οποία έλαβε στις 26 Φεβρουαρίου 1963. Ενδιάμεσα, το μηχάνημα εγκαταστάθηκε πειραματικά στη Νέα Υόρκη, από τη City Bank (νυν Citibank), αλλά αποσύρθηκε ύστερα από έξι μήνες, λόγω έλλειψης ενδιαφέροντος από τους πελάτες της τράπεζας.[5]

Το 1967 στο Ένφιλντ του Βορείου Λονδίνου, έγινε η πρώτη εγκατάσταση ενός ΑΤΜ (Αυτόματη Ταμειολογιστική Μηχανή) όπως το ξέρουμε σήμερα από την Barclays, το οποίο διάβαζε αντί για κάρτες ειδικές επιταγές εμποτισμένες με άνθρακα 14, μία ουσία με ήπια ραδιενεργή ακτινοβολία. Το μηχάνημα την ανίχνευε και στη συνέχεια την ταίριαζε μ' έναν αριθμό ασφαλείας, το σημερινό μας PIN. Στην Ελλάδα τα πρώτα ΑΤΜ τοποθετήθηκαν τη δεκαετία του 1980 (Alpha Bank το 1983, Citibank το 1985) και η χρήση τους άρχισε να εξαπλώνεται την τελευταία δεκαετία του 20ου αιώνα..

Πρωταρχικός στόχος για τη δημιουργία του δικτύου ΑΤΜ ήταν και εξακολουθεί να είναι η μείωση του κόστους των τραπεζικών εργασιών. Όμως τα οφέλη ήταν περισσότερα όχι μόνο για τις τράπεζες αλλά και για τους καταναλωτές, οι οποίοι αποδεσμεύτηκαν από τα ωράρια των τραπεζών καθώς πλέον μπορούσαν να προμηθεύονται χρήματα οποιαδήποτε ώρα της ημέρας και να

πραγματοποιούν τις βασικές τους συναλλαγές ανεξάρτητα από το αν η τράπεζα ήταν ανοικτή ή όχι.

1.2.3. Σύγχρονες ηλεκτρονικές τραπεζικές συναλλαγές

Σταθμό στην ιστορία των ηλεκτρονικών τραπεζικών συναλλαγών αποτέλεσε η εισαγωγή της έννοιας του home banking από τις αμερικάνικες τράπεζες στα τέλη της δεκαετίας του ’80. Με αυτή την υπηρεσία, οι τράπεζες παρείχαν στους πελάτες τους τη δυνατότητα να πραγματοποιήσουν τις βασικές τραπεζικές τους συναλλαγές από το σπίτι μέσω ενός ηλεκτρονικού υπολογιστή. Αναπτύσσοντας λοιπόν το κατάλληλο λογισμικό, προσπάθησαν να το προωθήσουν δωρεάν ειδικά στους πιο εύπορους και απαιτητικούς. Ωστόσο, ο κύκλος ζωής του home banking ήταν σύντομος, καθώς στα μέσα της δεκαετίας του ’90 άρχισε να επικρατεί το internet banking και γενικότερα το e-banking.

Ο σημαντικότερος παράγοντας που οδήγησε στην επικράτηση του e-banking συγκριτικά με το home banking, ήταν το γεγονός ότι οι τράπεζες δεν χρειαζόταν να συντηρούν ιδιωτικά δίκτυα με υψηλό κόστος. Επιπλέον, ούτε οι πελάτες ήταν ανάγκη να εφοδιάζονται με κάποιο εξειδικευμένο λογισμικό για να έχουν πρόσβαση στα δίκτυα της τράπεζας. Το Internet, ως ανοιχτό σύστημα, ήταν μία ελκυστική πρόκληση για τις τράπεζες που διέβλεψαν σε αυτό την ευκαιρία να διευρύνουν την πελατειακή τους βάση.

Η πρόσβαση στις ηλεκτρονικές υπηρεσίες του e-banking επιτυγχάνεται αφού ο πελάτης αιτηθεί και εξασφαλίσει κωδικούς πρόσβασης, σε κάποιο υποκατάστημα της τράπεζας. Οι τραπεζικές συναλλαγές μέσω internet δεν διαφέρουν πολύ από τη χρήση των ATM. Στην οθόνη του υπολογιστή είναι δυνατόν κάποιος, να πληροφορηθεί για τα υπόλοιπα και τους τόκους των λογαριασμών του, για τις εντολές και τις πληρωμές λογαριασμών, για τις τιμές συναλλάγματος και ξένων χαρτονομισμάτων και πολλά άλλα. Αναλυτικότερα, ο πελάτης έχει τις εξής δυνατότητες και πληροφορίες για τα προϊόντα και τις συναλλαγές της τράπεζας που επιθυμεί, μέσω των ηλεκτρονικών υπηρεσιών του e-banking:

- Πληροφορίες λογαριασμών, όπου ο χρήστης μπορεί να δει και να ελέγξει τους λογαριασμούς του online και να ενημερωθεί για κάθε υποχρέωση που έχει ως προς την τράπεζα
- Πληροφορίες σχετικά με τις κάρτες του

- Πληροφορίες χαρτοφυλακίου, δηλαδή ενημερώσεις για μετοχές, ομόλογα κ.α.
- Πληροφορίες δανείων, όπου ο χρήστης που έχει λάβει οποιασδήποτε μορφής δάνειο από τράπεζα
- Πληροφορίες επιταγών
- Μεταφορές κεφαλαίων εντός τράπεζας σε λογαριασμούς του ιδίου, όπου ο χρήστης έχει τη δυνατότητα να μεταφέρει χρήματα από έναν λογαριασμό του σε κάποιον άλλον
- Μεταφορές κεφαλαίων εντός τράπεζας σε λογαριασμούς τρίτων (πληρωμή ενοικίου κ.α.)
- Μεταφορές κεφαλαίων εκτός τράπεζας-εμβάσματα: Ο χρήστης, ο οποίος επιθυμεί τη μεταφορά κεφαλαίου εκτός της τράπεζας, θα πρέπει να γνωρίζει και τον αριθμό IBAN του δικαιούχου καθώς και το SWIFT της τράπεζας που στέλνει τα χρήματα. Ειδικά αν πρόκειται για μεταφορά εκτός Ελλάδας, θα πρέπει να αναφέρεται και η χώρα αποστολής.
- Πληρωμές πιστωτικών καρτών του ιδίου
- Πληρωμή πιστωτικών καρτών τρίτου
- Πληρωμή πιστωτικών καρτών άλλης τράπεζας
- Πληρωμές δημοσίου
- Πληρωμές λογαριασμών παροχής ηλεκτροδότησης, ύδρευσης
- Πληρωμές σταθερής και κινητής τηλεφωνίας
- Πληρωμές Ασφαλιστικών: Αρκετές ασφαλιστικές εταιρίες συνάπτουν συμφωνίες με τράπεζες, δίνοντας τη δυνατότητα στους πελάτες τους να πληρώνουν τα ασφάλιστρα τους μέσω αυτών.
- Εκτέλεση πληρωμής μισθοδοσίας, υπηρεσία η οποία προσφέρεται αποκλειστικά σε επαγγελματίες πελάτες.
- Χρηματιστηριακές εντολές, με τις οποίες ο χρήστης μπορεί άμεσα να δώσει μία εντολή που αφορά αγορά ή πώληση μετοχών στην τιμή που επιθυμεί, υποδεικνύοντας το λογαριασμό χρέωσης ή πίστωσης, την τιμή

για την οποία επιθυμεί να πραγματοποιηθεί η συναλλαγή και τα τεμάχια διαπραγμάτευσης.

Επίσης, μέσω του e-banking τους οι τράπεζες παρέχουν και τη δυνατότητα στο χρήστη για ηλεκτρονικές αιτήσεις για απόκτηση των προϊόντων τους. Μετά την αίτηση ακολουθεί και η αποδοχή των όρων συναλλαγής. Μερικά είδη αιτήσεων είναι:

- Αίτηση ανοίγματος λογαριασμού
- Αίτηση χορήγησης δανείου
- Αίτηση έκδοσης πιστωτικής κάρτας
- Αίτηση χορήγησης καρνέ επιταγών

Τέλος, οι τράπεζες πέραν των υπηρεσιών που προσφέρουν στους χρήστες τους, μέσω του e-banking παρέχουν και βοηθητικά εργαλεία που διευκολύνουν τη ζωή των πελατών τους. Μερικά από αυτά είναι:

- Υπολογισμός IBAN
- Μετατροπή νομισμάτων
- Υπολογισμός δόσεων δανείων

Συνοψίζοντας, τα οφέλη από τη χρήση των ηλεκτρονικών τραπεζικών συναλλαγών είναι πολλά και πιο συγκεκριμένα τα εξής:

- Ευκολία χρήσης και διαθεσιμότητα των υπηρεσιών σε 24ωρη βάση, 7 ημέρες την εβδομάδα, 365 ημέρες το χρόνο.
- Δυνατότητα πρόσβασης στις ηλεκτρονικές υπηρεσίες της τράπεζας ανεξάρτητα από την τοποθεσία που βρίσκεται ο χρήστης και γενικά από οποιοδήποτε σημείο υπάρχει πρόσβαση στο Internet, ακόμα και στην περίπτωση που ο χρήστης βρίσκεται εν κινήσει και χρησιμοποιεί το κινητό του τηλέφωνο.
- Ταχύτητα στη διενέργεια και ολοκλήρωση των συναλλαγών σε σχέση με τους παραδοσιακούς τρόπους αλλά και σε σχέση με τα υπόλοιπα κανάλια διανομής των υπηρεσιών.

- Υψηλό επίπεδο ασφάλειας συναλλαγών, καλύτερο από οποιοδήποτε επίπεδο παρέχουν σήμερα οι παραδοσιακοί και εναλλακτικοί τρόποι διενέργειας συναλλαγών.
- Αποδοτικότερη διαχείριση των πάσης φύσεως συναλλαγών του χρήστη (πληροφοριακών, επενδυτικών, δανειακών κλπ). Οι πελάτες των τραπεζών έχουν συνολική εικόνα των λογαριασμών και των συναλλαγών τους μέσω της οθόνης του υπολογιστή τους, καθώς και πρόσβαση σε ιστορικά στοιχεία που αφορούν προηγούμενες κινήσεις και συναλλαγές.
- Πρόσβαση σε ένα ευρύ φάσμα πληροφοριών, το οποίο καλύπτει τις ποικίλες ανάγκες των τραπεζικών πελατών, τόσο των ιδιωτών όσο και των ελεύθερων επαγγελματιών και των επιχειρήσεων, ανεξαρτήτως μεγέθους και κλάδου στον οποίο δραστηριοποιούνται.
- Μείωση κόστους συναλλαγών και συνεπώς οικονομικότερη ολοκλήρωση των συναλλακτικών δραστηριοτήτων επιχειρήσεων και ιδιωτών πελατών με τα χρηματοπιστωτικά ιδρύματα.

ΚΕΦΑΛΑΙΟ 2

ΤΕΧΝΟΛΟΓΙΕΣ ΑΝΑΠΤΥΞΗΣ

Σε αυτό το κεφάλαιο αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του Web Banking.

2.1. Η γλώσσα προγραμματισμού JavaScript

Η JavaScript είναι μία scripting γλώσσα προγραμματισμού, η οποία αποτελεί μια από τις τρεις βασικές τεχνολογίες του World Wide Web (HTML, CSS και JavaScript). Χρησιμοποιείται για να προσθέσει διαδραστικές και δυναμικές δυνατότητες στις ιστοσελίδες. Τα τελευταία χρόνια, μπορεί να χρησιμοποιηθεί και ως server side γλώσσα προγραμματισμού, με την χρήση της Node.js.

2.1.1. Η αρχή

Όλα ξεκίνησαν το 1995. Ο Marc Andeersen, ιδρυτής της Netscape Communications, είχε το όραμα ότι ο ιστός (web) χρειάζεται έναν τρόπο να γίνει πιο δυναμικός. Οι κινούμενες εικόνες(animations), η αλληλεπίδραση και άλλες μορφές μικρών αυτοματισμών θα έπρεπε να αποτελούν μέρος του ιστού του μέλλοντος. Ως αποτέλεσμα, το web θα χρειαζόταν μια μικρή scripting γλώσσα που θα μπορούσε να αλληλεπιδράσει με το DOM (Document Object Model). Το DOM είναι μια διεπαφή προγραμματισμού για τα HTML documents (μια σελίδα για παράδειγμα αποτελεί ένα document). Αντιπροσωπεύει τη σελίδα σε μορφή αντικειμένου, έτσι ώστε τα προγράμματα να μπορούν εύκολα να αλλάξουν τη δομή και το περιεχόμενο του. Επίσης, και αυτή ήταν μια σημαντική στρατηγική απόφαση εκείνη την εποχή, αυτή η γλώσσα δεν θα έπρεπε να προσανατολίζεται σε προγραμματιστές με εμπειρία στο χώρο της τεχνολογίας και ανάπτυξης λογισμικού. Αντιθέτως, η γλώσσα για το διαδίκτυο θα έπρεπε να εξυπηρετεί έναν διαφορετικό τύπο κοινού: τους σχεδιαστές (designers). Εκείνη την εποχή, ο ιστός ήταν στατικός(static). Η HTML ήταν ακόμη αρκετά νέα και αρκετά απλή στην εκμάθηση για τους μη προγραμματιστές, άρα, και οι δυνατότητες που θα έκαναν το web πιο δυναμικό θα

έπρεπε να ήταν προσβάσιμες και εύκολες σε μη προγραμματιστές. Έτσι γεννήθηκε η ιδέα της Mocha (η πρώτη ονομασία της JavaScript).

Για αυτό το σκοπό ο Brendan Eich, πατέρας της JavaScript, είχε προσληφθεί από την Netscape Communications για να αναπτύξει μια γλώσσα που θα τηρούσε τις απαιτήσεις. Ωστόσο, η Netscape Communications συνομιλούσε ταυτόχρονα και με την Sun Microsystems για να κλείσει μια συμφωνία ώστε να καταστήσει την Java διαθέσιμη στα προγράμματα περιήγησης ιστού (browsers). Παρόλα αυτά, υπήρχε ένα πρόβλημα. Η Java ήταν απλά πάρα πολύ μεγάλη, πολύ σύνθετη για αυτό ρόλο, δε θα μπορούσε ποτέ να χρησιμοποιηθεί ευκολά από συγγραφείς, ερασιτέχνες, σχεδιαστές. Έτσι, η ιδέα ήταν να καταστεί η Java διαθέσιμη για έμπειρους επαγγελματίες, ενώ η Mocha θα χρησιμοποιούνταν για μικρά scripts. Με άλλα λόγια, η Mocha προοριζόταν να είναι ο scripting companion της Java, κατά τρόπο ανάλογο τη σχέσης μεταξύ C/C ++ και Visual Basic στην πλατφόρμα των Windows.

Το πρωτότυπο της Mocha ενσωματώθηκε στον Netscape Communicator τον Μάιο του 1995 και σε σύντομο χρονικό διάστημα μετονομάστηκε σε LiveScript. Το Δεκέμβριο του 1995, η Netscape Communications και η Sun έκλεισαν τη συμφωνία η LiveScript να μετονομαζόταν σε JavaScript, η οποία θα παρουσιαζόταν ως scripting γλώσσα για μικρές client-side εργασίες στον περιηγητή ιστού (browser), ενώ η Java θα προωθούνταν ως το κατάλληλο επαγγελματικό εργαλείο για την ανάπτυξη πλούσιων web εφαρμογών.

2.1.2. Η εξέλιξη της JavaScript

Από τις πρώτες ημέρες, η JavaScript έκανε σημαντική διαφορά στην εμπειρία των χρηστών, με αποτέλεσμα όλα τα ανταγωνιστικά προγράμματα περιήγησης ιστού (browsers) να μην έχουν άλλη επιλογή παρά να βρουν μια λύση, μια λειτουργική υλοποίηση της JavaScript. Επί του παρόντος, και για πολύ μεγάλο χρονικό διάστημα, τα πρότυπα του web δεν ήταν ισχυρά. Έτσι, η Microsoft υλοποίησε τη δική της έκδοση JavaScript, που ονομαζόταν JScript. Κρατώντας τη "Java" εκτός ονόματος απέφυγε πιθανά ζητήματα εμπορικών σημάτων (trademarks). Ωστόσο, η JScript ήταν διαφορετική σε κάτι περισσότερο από το όνομα. Οι μικρές διαφορές στην εφαρμογή, ιδίως σε σχέση με ορισμένες λειτουργίες του DOM, προκάλεσαν προβλήματα που γίνονται αισθητά ακόμα και σήμερα.

Η πρώτη μεγάλη αλλαγή για της JavaScript μετά τη έκδοση της είχε τη μορφή τυποποίησης ECMA. Η ECMA (European Computer Manufacturers Association, Ένωση Ευρωπαϊκών Κατασκευαστών Ηλεκτρονικών Υπολογιστών) είναι μια βιομηχανική ένωση που ιδρύθηκε το 1961 και αφορούσε αποκλειστικά την τυποποίηση των συστημάτων πληροφοριών και επικοινωνιών. Η τυποποίηση ήταν ένα σημαντικό βήμα για μια τέτοια νέα γλώσσα, και είχε ως αποτέλεσμα να ανοίξει σε ένα ευρύτερο κοινό και να δώσει σε άλλους επαγγελματίες φωνή στην εξέλιξη της γλώσσας. Για λόγους εμπορικού σήματος, η επιτροπή ECMA δεν μπόρεσε να χρησιμοποιήσει τη JavaScript ως όνομα. Οι εναλλακτικές λύσεις δεν άρεσαν σε πολλούς, οπότε μετά από συζήτηση αποφασίστηκε ότι η γλώσσα που περιγράφεται από το πρότυπο θα ονομάζεται ECMAScript. Σήμερα, η JavaScript είναι μόνο το εμπορικό όνομα για την ECMAScript.

Η ECMAScript 3 κυκλοφόρησε τον Δεκέμβριο του 1999 και οι πρώτες μεγάλες αλλαγές στη γλώσσα είδαν το φως. Αυτή η έκδοση έφερε:

- Κανονικές εκφράσεις (Regular expressions)
- Το βρόχο do-while
- Εξαιρέσεις (Exceptions)
- Περισσότερες ενσωματωμένες συναρτήσεις (built-in functions) για πίνακες (arrays) και αλφαριθμητικά (strings)
- Τους τελεστές in και instanceof
- Πολύ καλύτερη διαχείριση λαθών

Αυτή η έκδοση της ECMAScript εξαπλώθηκε ευρέως. Υποστηρίχθηκε από όλα τα μεγάλα προγράμματα περιήγησης εκείνης της εποχής και συνέχισε να υποστηρίζεται για πολλά χρόνια. Αυτό έκανε την ECMAScript 3 την βάση για πολλές βιβλιοθήκες, ακόμα και όταν εκδόθηκαν νεότερες εκδόσεις του προτύπου. Επίσης, το AJAX (Asynchronous JavaScript and XML) ήταν μια τεχνική που γεννήθηκε στα χρόνια της ECMAScript 3. Αν και δεν ήταν μέρος του προτύπου, η Microsoft υλοποίησε ορισμένες επεκτάσεις της JavaScript για τον Internet Explorer. Αυτή η λειτουργία επέτρεψε σε ένα πρόγραμμα περιήγησης να εκτελεί ένα ασύγχρονο αίτημα HTTP σε έναν server, επιτρέποντας έτσι στις σελίδες να ενημερώνονται σε πραγματικό χρόνο.

Τα επόμενα χρόνια δεν ήταν καλά για την ανάπτυξη της JavaScript. Μόλις ξεκίνησαν οι εργασίες για την ECMAScript 4, άρχισαν να εμφανίζονται έντονες διαφορές στην επιτροπή. Υπήρξε μια ομάδα ανθρώπων που πίστευαν ότι η JavaScript χρειάζονταν χαρακτηριστικά που θα την έκαναν μια ισχυρότερη γλώσσα για την ανάπτυξη εφαρμογών μεγάλης κλίμακας. Αυτή η ομάδα πρότεινε πολλά χαρακτηριστικά μεγάλης εμβέλειας και αλλαγών. Αρκετοί όμως θεωρούσαν ότι αυτή δεν θα ήταν η κατάλληλη εξέλιξη για τη JavaScript. Η έλλειψη συναίνεσης και η πολυπλοκότητα ορισμένων προτεινόμενων χαρακτηριστικών καθυστερούσε συνεχώς τη έκδοση της ECMAScript 4 ώσπου το 2003 σταμάτησε τελείως η ανάπτυξη της. Το 2005, ο αντίκτυπος του AJAX πυροδότησε και πάλι το ενδιαφέρον για μια νέα έκδοση της JavaScript με αποτέλεσμα να αρχίσει εκ νέου η ανάπτυξη της. Τα χρόνια περνούσαν και το σύνολο των δυνατοτήτων της συνεχώς αυξανόταν, με κάποια από τα σημαντικότερα χαρακτηρίστηκα της να είναι:

- Κλάσεις
- Διεπαφές (Interfaces)
- Namespaces
- Πακέτα (Packages)
- Πολλαπλές μεθόδους (Multi methods)
- Επαναλήπτες (Iterators)
- Παραγωγοί (Generators)
- Σωστότερο πεδίο εφαρμογής (Proper block scoping)

Παρόλα αυτά, από το 2008 και μετά, η κοινότητα επικεντρώθηκε στην ανάπτυξη της ECMAScript 3.1. Η ECMAScript 4 απορρίφθηκε. Η ECMAScript 3.1 ολοκληρώθηκε και εκδόθηκε το 2009. Η ECMAScript 4 ακόμη και αν δεν εκδόθηκε ποτέ επίσημα, αναγνωρίστηκε ως ειδική παραλλαγή της ECMAScript. Το αποτέλεσμα ήταν η απόφαση της επιτροπής να μετονομάσει την ECMAScript 3.1 σε ECMAScript 5 για να αποφευχθεί η σύγχυση. Η ECMAScript 5 έγινε μια από τις πλέον υποστηριζόμενες εκδόσεις της JavaScript και υποστηρίχθηκε εξ ολοκλήρου από τα τότε μεγάλα προγράμματα περιήγησης ιστού (browsers), όπως ο Firefox 4 (2011), ο Chrome 19 (2012), ο Safari 6 (2012), ο Opera 12.10 (2012) και ο Internet

Explorer 10 (2012). Η ECMAScript 5 ήταν μια μέτρια αναβάθμιση της ECMAScript 3 και περιλάμβανε:

- Οι μέθοδοι Getter/setter
- Νέες μεθόδους για αντικείμενα
- Νέες μεθόδους για πίνακες
- Νέες μεθόδους για ημερομηνίες
- Σύνδεση συναρτήσεων (Function binding)
- JSON
- Αμετάβλητα καθολικά αντικείμενα (Immutable global objects)

Η ECMAScript 6, που μετονομάστηκε αργότερα σε ECMAScript 2015, σχεδιάστηκε για να φέρει μεγάλες αλλαγές. Ορισμένα από τα μεγάλα χαρακτηριστικά της ECMAScript 4 εφαρμόστηκαν σε αυτή την έκδοση, ωστόσο, εφαρμόστηκαν με διαφορετική νοοτροπία. Αυτή τη φορά στην επιτροπή επικρατούσε ενότητα, με αποτέλεσμα σχεδόν κάθε αλλαγή που είχε συζητηθεί στο παρελθόν και απαιτούσε συντακτικές αλλαγές να υλοποιηθεί σε αυτήν την έκδοση. Πολλοί προμηθευτές προγραμμάτων περιήγησης ιστού άρχισαν να εργάζονταν στην υλοποίηση των χαρακτηριστικών της ECMAScript 2015, αλλά οι αλλαγές ήταν πολλές με αποτέλεσμα να χρειαστεί αρκετός χρόνος για να υλοποιηθούν. Ακόμα και σήμερα, οι browsers δεν έχουν πλήρη κάλυψη της ECMAScript 2015. Μια σύντομη περίληψη των πιο σημαντικών νέων στοιχείων είναι:

- Νέοι τρόποι δήλωσης μεταβλητών με χρήση του let και του const
- Συναρτήσεις βέλους (Arrow functions)
- Κλάσεις
- Πρότυπα αλφαριθμητικά (Template strings)
- Υποσχέσεις (Promises)
- Προεπιλεγμένοι παράμετροι (default arguments) για τις συναρτήσεις (functions)
- Το Rest/Spread συντακτικό

- Το συντακτικό ενοτήτων (Module syntax)
- Νέα collections (Set, Map)
- Αντανάκλαση (Reflection)

2.1.3. ReactJS

Ένα από τα μεγαλύτερα προβλήματα της ανάπτυξης ιστοσελίδων είναι ο χειρισμός του DOM. Ο άμεσος χειρισμός του DOM όσο αφορά τους πόρους του συστήματος είναι ακριβός. Η λήψη δεδομένων μιας JavaScript εφαρμογής και η απεικόνιση τους στο πρόγραμμα περιήγησης ιστού (browser) είναι μια δαπανηρή και αργή λειτουργία. Η ReactJS είναι η πρώτη βιβλιοθήκη που καταλήγει σε ένα προφανές συμπέρασμα: η βελτιστοποίηση του χειρισμού του DOM οδηγεί στη δημιουργία μιας γρήγορης βιβλιοθήκης και σε αυτή την περίπτωση σε μια βιβλιοθήκη που προσφέρει τη δυνατότητα ανάπτυξης ιστοσελίδων με χρήση λίγου κώδικα. Η ReactJS δημιουργήθηκε από το Facebook, και χρησιμοποιείται για τη δημιουργία σύνθετων, reactive και γρήγορων ιστοσελίδων. Ενθαρρύνει τη δημιουργία επαναχρησιμοποιήσιμων κομματιών κώδικα (components) που παρουσιάζουν δεδομένα τα οποία αλλάζουν συχνά και έχουν ως απαίτηση τις πολλές και συχνές ανανεώσεις του DOM.

Η ReactJS είναι μια βιβλιοθήκη που ακολουθεί την αρχιτεκτονική της μιας και μόνο σελίδας ανά website (Single Page Application, SPA). Αυτό επιτυγχάνεται με την δυναμική ανανέωση, και όχι επαναφόρτωση, της σελίδας που αλληλοεπιδρά με τον χρήστη, με την χρήση AJAX. Τα Single Page Application websites πλέον δε χωρίζονται σε αυτόνομες σελίδες τις οποίες ζητάμε με ένα URL από τον server, ο οποίος μας τις «σερβίρει» ανακατευθύνοντας μας σε αυτές. Αντιθέτως τα URLs έχουν πλέον τον ρόλο των διαδρομών (routes). Τα routes θα μπορούσαν να χαρακτηριστούν ως υποσελίδες της μοναδικής σελίδας του website. Όταν ζητήσουμε ένα URL, το περιεχόμενο της σελίδας ανανεώνεται δυναμικά και πλέον χωρίς επαναφόρτωση δείχνει τα περιεχόμενα τα οποία έχουμε συνδέσει με το αντίστοιχο route. Η ReactJS υλοποιεί την αρχιτεκτονική των Single Page Applications αποθηκεύοντας την κατάσταση (state) της ιστοσελίδας σε μια δικιά της εσωτερική δενδρική μορφή (εικονικός DOM), και χειρίζεται όλες τις οπτικές αλλαγές ανανεώνοντας τα περιεχόμενα της στο πρόγραμμα περιήγησης ιστού (πραγματικός DOM), μόνο όταν έχει αλλάξει η κατάσταση (state) της. Η ReactJS

πηγαίνει τα πράγματα ένα βήμα παραπέρα γιατί γνωρίζει πότε έχει ή δεν έχει αλλάξει η κατάσταση ή συγκεκριμένα στοιχεία της ιστοσελίδας, με αποτέλεσμα η ανανέωση του DOM να γίνεται αυτόματα και μόνο για τα στοιχεία που έχουν αλλάξει. Ο αλγόριθμος που χρησιμοποιεί για να κατανοεί πότε άλλαξε το δέντρο κατάστασης έχει πολυπλοκότητα O(n), χρήζοντάς την ιδανική για μεγάλες εφαρμογές Αυτό το καταφέρνει χρησιμοποιώντας ένα hash map των στοιχείων του DOM, με μοναδικά κλειδιά.

Η ReactJS χρησιμοποιεί την JSX, έναν preprocessor που προσθέτει την δυνατότητα χρήσης XML συντακτικού στην JavaScript. Δεν είναι υποχρεωτική η χρήση της JSX, αλλά προτείνεται επειδή κάνει πιο εύκολη και καθαρή την ανάπτυξη ιστοσελίδων.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello World!!
      </div>
    );
  }
}
```

Εικόνα 1: Συντακτικό ReactJS/JSX

Η ReactJS προτείνει το σπάσιμο του κώδικα σε μικρά διακριτά κομμάτια, τα οποία έχουν μια και μόνο συγκεκριμένη λειτουργία, έτσι ώστε να είναι σαφή, καθαρά και να μπορούν να επαναχρησιμοποιηθούν όσο το δυνατό περισσότερο. Αυτά τα κομμάτια κώδικα τα ονομάζει components και μπορούν να χρησιμοποιηθούν σαν HTML tags, με αποτέλεσμα ένα component να μπορεί να περιέχει μέσα του περισσότερα components.

```
class TableHeader extends React.Component {
  render() {
    return (
      <thead>
        <tr>
          <th>This is the header</th>
        </tr>
      </thead>
    );
  }
}
```

Εικόνα 2: Component για table header

```
class TableContents extends React.Component {
  render() {
    return (
      <tbody>
        <tr>
          <td>cell 1</td>
          <td>cell 2</td>
          <td>cell 3</td>
        </tr>
      </tbody>
    );
  }
}
```

Εικόνα 3: Component για table body

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <TableHeader />
        <TableContents />
      </table>
    );
  }
}
```

Εικόνα 4: Component ενός table με χρήση components για το header και το body

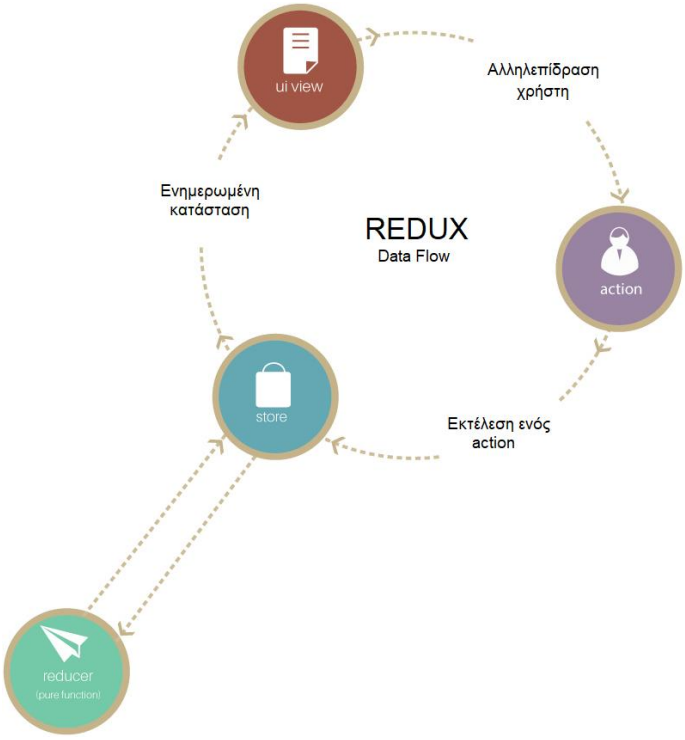
2.1.4. Redux

Καθώς οι απαιτήσεις για web εφαρμογές δημιουργημένες με JavaScript γίνονται όλο και πιο περίπλοκες, η κατάσταση της εφαρμογής (application state) που πρέπει να χειρίζεται ανά πάσα στιγμή ο κώδικας είναι μεγαλύτερη από ποτέ. Αυτή η κατάσταση μπορεί να περιλαμβάνει απαντήσεις του server και προσωρινά αποθηκευμένα δεδομένα (cached data), καθώς και τοπικά δεδομένα που δεν έχουν ακόμη αποθηκευτεί στο server. Η κατάσταση της εφαρμογής αυξάνεται επίσης λόγω της πολυπλοκότητάς της, καθώς θα πρέπει να διαχειριστούν η σελίδα που είναι ενεργή, οι επιλεγμένες καρτέλες, διάφορα animations, το σπάσιμο των δεδομένων σε σελίδες (pagination) και ούτω καθεξής. Η διαχείριση αυτής της συνεχώς μεταβαλλόμενης κατάστασης είναι δύσκολη. Σε μια κλασική MVC (Model View Controller) υλοποίηση, ένα Model μπορεί να ενημερώσει ένα άλλο Model, τότε ένα View μπορεί να ενημερώσει ένα Model, το οποίο ενημερώνει άλλο Model και αυτό και με τη σειρά του, ενδέχεται να αναβαθμίσει ένα άλλο View. Σε κάποιο

σημείο, είναι σχεδόν αδύνατο καταλάβεις και να κρατήσεις τον έλεγχο στο τι γίνεται στην εφαρμογή. Όταν ένα σύστημα είναι αδιαφανές και μη ντετερμινιστικό, είναι δύσκολο να αναπαραχθούν σφάλματα κατά τη διάρκεια του testing ή να προστεθούν νέα χαρακτηριστικά.

Αυτή η πολυπλοκότητα στη διαχείριση της κατάστασης web εφαρμογής παρουσιάζεται επειδή αναμιγνύονται δύο έννοιες που είναι πολύ δύσκολες για το ανθρώπινο μυαλό να τις καταλάβει: τη μετάλλαξη (mutation) και την ασυγχρονικότητα (asynchronicity). Αυτές οι δύο εξαιρετικές τεχνικές όταν λειτουργούν ξεχωριστά, αλλά και όταν συνδυάζονται δημιουργούν ένα χάος. Βιβλιοθήκες όπως η ReactJS επιχειρούν να λύσουν αυτό το πρόβλημα στο κομμάτι της απεικόνισης της εφαρμογής, αφαιρώντας τόσο την ασύγχρονη όσο και την άμεση χειραγώγηση του DOM. Ωστόσο, η διαχείριση της κατάστασης των δεδομένων παραμένει δύσκολη. Αυτό το πρόβλημα λύνει το Redux. Το Redux είναι μια JavaScript βιβλιοθήκη που διαχειρίζεται τη κατάσταση της εφαρμογής, καθιστώντας τις μεταλλάξεις (αλλαγές) της κατάστασης προβλέψιμες επιβάλλοντας ορισμένους περιορισμούς σχετικά με το πώς και πότε θα γίνονται. Αυτοί οι περιορισμοί αντικατοπτρίζονται στις τρεις αρχές της Redux::

- **Μια και μόνο πηγή αλήθειας** (**single source of truth).** Η κατάσταση ολόκληρης της εφαρμογής τηρείται σε ένα object δενδρικής μορφής που ονομάζεται store.
- **Η κατάσταση της εφαρμογής είναι read only.** Ο μόνος τρόπος για να αλλάξουμε την κατάσταση είναι να εκτελέσουμε μια ενέργεια (action), ένα αντικείμενο που περιγράφει τι θα αλλάξει.
- **Οι αλλαγές στην κατάσταση πραγματοποιούνται με τη χρήση των reducers.** Οι reducers είναι pure functions που παίρνουν ως παραμέτρους την τρέχουσα κατάσταση και μια ενέργεια (action), και επιστρέφουν τη νέα κατάσταση της εφαρμογής. Οι pure functions είναι συναρτήσεις που αφού ολοκληρώσουν τις λειτουργίες τους, επιστρέφουν πάντα μια νέα τροποποιημένη κατάσταση με βάση τις παραμέτρους που παίρνουν. Δεν τροποποιούν την υπάρχουσα κατάσταση. Αντίθετα, επιστρέφουν μια νέα.



Εικόνα 2.1.5: Data flow του Redux

2.2. Η πλατφόρμα .NET

Η πλατφόρμα .NET (.NET Framework) είναι ένα framework λογισμικού που αναπτύχθηκε από τη Microsoft και λειτουργεί κυρίως στα Microsoft Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη κλάσεων που ονομάζεται Framework Class Library (FCL), και παρέχει γλωσσική διαλειτουργικότητα, δηλαδή οι γλώσσες προγραμματισμού που υποστηρίζονται από το .Net Framework (C#, VB.NET και F#) μπορούν να χρησιμοποιήσουν η μία τον κώδικα της άλλης χωρίς κανένα πρόβλημα. Τα προγράμματα που γράφονται για το .NET Framework εκτελούνται σε ένα περιβάλλον λογισμικού με την ονομασία Common Language Runtime (CLR), μια εικονική μηχανή εφαρμογής που παρέχει υπηρεσίες όπως η ασφάλεια, η διαχείριση μνήμης και ο χειρισμός εξαιρέσεων (exceptions). Το FCL και το CLR αποτελούν από κοινού το .NET Framework.

Ένας από τους κύριους περιορισμούς του .NET Framework ήταν ότι έτρεχε αποκλειστικά στα Windows. Αυτό άλλαξε με την έλευση του .NET Core, το οποίο παρέχει στους προγραμματιστές βιβλιοθήκες που μπορούν να χρησιμοποιηθούν σε διάφορες πλατφόρμες, όπως Windows, Linux και Mac. Αποτελείται από το

CoreCLR, μια ολοκληρωμένη υλοποίηση του ήδη υπάρχοντος CLR, και έρχεται με ένα βελτιωμένο Just-In-Time compiler(JIT), ο οποίος ονομάζεται RyuJIT. Το .NET Core περιλαμβάνει επίσης το CoreFX, το οποίο αποτελεί μια τμηματική διακλάδωση του .NET FCL. Επίσης, το .NET Core, σε αντίθεση με το .NET Framework, είναι modular, δηλαδή επιτρέπει στους προγραμματιστές να χρησιμοποιούν μόνο τα τμήματα του framework που χρειάζονται για τα έργα τους μέσω του NuGet package manager. Τέλος, ενώ το .NET Core μοιράζεται ένα υποσύνολο των API του .NET Framework, έρχεται με το δικό του API που δεν αποτελεί μέρος του .NET Framework.

2.2.1. Η γλώσσα προγραμματισμού C#

Η C# είναι μια multi-paradigm γλώσσα προγραμματισμού βασίζεται στις αρχές του Object Oriented Programming (OOP) και του Component Oriented Programming. Αναπτύχθηκε από τη Microsoft το 2000, με τον Anders Hejlsberg ως κύριο σχεδιαστή και επικεφαλής αρχιτέκτονα. Ο Anders Hejlsberg και η ομάδα του ήθελαν να δημιουργήσουν μια νέα γλώσσα προγραμματισμού που θα μπορούσε να χρησιμοποιηθεί για την ανάπτυξη βιβλιοθηκών του .NET Framework. Από τη στιγμή της ανακοίνωσης της, η C# έχει αμφιλεγόμενα αναφερθεί ως απομίμηση της Java. Ωστόσο, με την πάροδο του χρόνου, τόσο η Java όσο και η C# παρουσίασαν ξεχωριστά χαρακτηριστικά.

Από την κυκλοφορία της C# 2.0 το Νοέμβριο του 2005, οι C# και Java εξελίχθηκαν σε ολοένα και πιο αποκλίνουσες τροχιές, καθιστώντας κάπως λιγότερο παρόμοιες. Μια από τις πρώτες μεγάλες διαφορές ήρθε με την προσθήκη των γενικών τύπων (generics) και στις δύο γλώσσες, στις οποίες ακολουθήθηκαν τελείως διαφορετικές υλοποιήσεις. Επιπλέον, από την κυκλοφορία της C# 3.0, έχουν προσθέτει αρκετά σημαντικά χαρακτηριστικά συναρτησιακού προγραμματισμού (functional programming) όπως εκφράσεις λάμδα (lambda expressions), ανώνυμοι τύποι (anonymous types), μεθόδους επεκτάσεων (extension methods), αλλά με σημαντικότερο όλων το Language Integrated Query (LINQ), το οποίο δίνει δυνατότητα εκτέλεσης queries πάνω σε δεδομένα. Αυτά τα χαρακτηριστικά επέτρεψαν τους προγραμματιστές να μειώσουν κατά πολύ τον κώδικα που έπρεπε να γραφτεί για κοινές εργασίες όπως η αναζήτηση βάσης δεδομένων, η ανάλυση ενός αρχείου XML ή η αναζήτηση σε δομές δεδομένων, όπως πίνακες ή λίστες, με αποτέλεσμα να μπορεί δοθεί περισσότερη έμφαση στην

υλοποίηση και συντήρηση της πραγματικής λογικής του προγράμματος. Επίσης, με την έλευση της C# 5.0 το 2013, δόθηκε η δυνατότητα δημιουργίας και εκτέλεσης ασύγχρονων μεθόδων.

2.2.2. Entity Framework

Το Entity Framework είναι ένα Object/Relational Mapping (ORM) framework της Microsoft, το οποίο δίνει τη δυνατότητα στους προγραμματιστές να χειρίζονται τα δεδομένα σχεσιακών βάσεων δεδομένων ως απλά αντικείμενα. Αυτό έχει ως αποτέλεσμα να μπορεί να χρησιμοποιηθεί το LINQ με το Entity Framework για τη δημιουργία queries για την ανάκτηση ή/και την παραμετροποίηση των δεδομένων, με φυσικό επακόλουθο να μη χρειάζεται να γραφτεί τίποτα σε SQL. Επίσης, οι ιδιότητες των πινάκων των σχεσιακών βάσεων, όπως οι τύποι των πεδίων, οι default τιμές των πεδίων, ή ακόμα και οι σχέσεις μεταξύ πινάκων γίνονται διαθέσιμες μέσα στο κώδικα μέσω του Entity Framework.

Το Entity Framework μπορεί να χρησιμοποιηθεί με τρεις διαφορετικούς τρόπους:

- Αν η βάση δεδομένων υπάρχει ήδη, τότε υπάρχει η δυνατότητα να παραχθούν οι πίνακες και οι σχέσεις τους αυτόματα σε κώδικα (κλάσεις, configurations κλπ.). Αυτή η μέθοδος ονομάζεται **Database First**.
- Αν υπάρχουν ήδη οι κλάσεις υλοποιημένες, μπορεί να παραχθεί αυτόματα ολόκληρη η βάση δεδομένων με όλους τους πίνακες και τις σχέσεις τους. Αυτή η μέθοδος ονομάζεται **Code First**.
- Αν δεν υπάρχει ούτε η βάση αλλά ούτε και η βάση δεδομένων, αλλά έχει αποφασιστεί να σχεδιαστεί πρώτα το σχήμα της βάσης δεδομένων και μετά να ακολουθήσει η δημιουργία της βάσεις και των κλάσεων, δίνεται η δυνατότητα να σχεδιαστεί το σχήμα της βάσης και όλες οι ιδιότητες της μέσω visual designer, και μετά να παραχθούν αυτόματα και οι κλάσεις και η βάση δεδομένων. Αυτή η μέθοδος ονομάζεται **Model First**.

2.3. Representational state transfer (REST)

Προκειμένου οι υπολογιστές να συνδεθούν μεταξύ τους μέσω ενός δικτύου και να μπορέσουν να μιλήσουν στην ίδια γλώσσα, εκατοντάδες μέθοδοι έχουν δημιουργηθεί με το πέρασμα των χρόνων. Κάθε μέθοδος ή πρωτόκολλο

υλοποιούσε αυτή τη σύνδεση με διαφορετικό τρόπο με αποτέλεσμα να μην ήταν εύκολη η επικοινωνία μεταξύ διαφορετικών συστημάτων. Για να λυθεί αυτό το πρόβλημα, το 2000 ο Roy Fielding δημιούργησε το REpresentational state transfer (REST). Το REST είναι ένα αρχιτεκτονικό στυλ για την παροχή προτύπων επικοινωνίας μεταξύ συστημάτων υπολογιστών στο διαδίκτυο, διευκολύνοντας τα συστήματα να επικοινωνούν μεταξύ τους μέσω του ίδιου μηχανισμού που χρησιμοποιεί το web, αποφεύγοντας πολλές δυσκολίες που είχαν άλλα μεγαλύτερα και πολύπλοκα πρωτόκολλα.

Η αρχή για το πώς δουλεύει το REST είναι παρόμοια με το πώς μια ιστοσελίδα και τα δεδομένα της γίνονται διαθέσιμα στο πρόγραμμα περιήγησης ιστού (browser). Η ιστοσελίδα εμφανίζεται στο browser μέσω μιας διεύθυνσης URL. Ο browser ζητά τα δεδομένα της ιστοσελίδας μέσω μίας αίτησης στο αντίστοιχο URL, λειτουργώντας ως client, και ο server του τα επιστρέφει με μία απάντηση. Άρα, τα δεδομένα βρίσκονται σε κάποια διεύθυνση την οποία ένας υπολογιστής θα χρησιμοποιούσε είτε για να ζητήσει δεδομένα είτε ακόμα και για να στείλει δεδομένα. Για αυτές τις ενέργειες χρησιμοποιούνται οι αιτήσεις HTTP GET και HTTP POST. Επίσης, το REST υποστηρίζει και άλλες αιτήσεις, όπως το PUT, DELETE, OPTONS, HEAD, TRACE, CONNECT. Αλλά οι πραγματικά πιο σημαντικές αιτήσεις είναι το GET και το POST. Το GET μπορεί να χρησιμοποιηθεί για την ανάκτηση οποιουδήποτε είδους δεδομένων παρέχει ένας server και το POST για τη δημιουργία. Με αυτές τις απλές αιτήσεις που ακόμα και ένας απλός browser χρησιμοποιεί καθημερινά, συστήματα τελείως διαφορετικά μεταξύ τους μπορούν να επικοινωνούν με αιτήσεις και απαντήσεις απλά και γρήγορα.

Στο αρχιτεκτονικό μοντέλο του REST, η υλοποίηση του client και η υλοποίηση του server μπορεί να γίνει ανεξάρτητα, χωρίς να γνωρίζει ο ένας για τον άλλο. Αυτό σημαίνει ότι ο κώδικας στην πλευρά του client μπορεί να αλλάξει ανά πάσα στιγμή χωρίς να επηρεαστεί η λειτουργία του server και ο κώδικας στην πλευρά του server μπορεί να αλλάξει χωρίς να επηρεαστεί η λειτουργία του client. Εφόσον κάθε πλευρά γνωρίζει τη μορφή των μηνυμάτων που αποστέλλονται στο άλλο, μπορούν να διατηρηθούν ως μονάδες (modular) και χωριστά. Διαχωρίζοντας τις ευθύνες των clients από τις ευθύνες των servers, βελτιώνουμε την ευελιξία της διασύνδεσης μεταξύ των συστημάτων και βελτιώνουμε την δυνατότητα κλιμάκωσης των servers.

Επίσης, τα συστήματα που ακολουθούν την αρχιτεκτονική REST δεν αποθηκεύουν πληροφορίες για τη κατάσταση προηγούμενων ανταλλαγών μνημάτων μεταξύ client και server (stateless), πράγμα που σημαίνει ότι ο server δεν χρειάζεται να γνωρίζει τίποτα σχετικά με την κατάσταση στην οποία βρίσκεται ο client και το αντίστροφο. Με αυτόν τον τρόπο, τόσο ο server όσο και ο client μπορούν να καταλάβουν οποιοδήποτε μήνυμα λαμβάνουν, ακόμη και χωρίς να βλέπουν προηγούμενα μηνύματα. Αυτός ο περιορισμός επιβάλλεται μέσω της χρήσης πόρων(resources) και όχι εντολών(commands). Επειδή τα συστήματα REST αλληλοεπιδρούν μέσω τυποποιημένων λειτουργιών, δεν βασίζονται σε υλοποίηση διεπαφών(interfaces). Αυτοί οι περιορισμοί βοηθούν να επιτευχθεί αξιοπιστία, γρήγορη απόδοση και δυνατότητα κλιμάκωσης, αφού τα κομμάτια (components) που αποτελούν τους client και πόσο μάλλον τους servers μπορούν να διαχειρίζονται, να ενημερώνονται και να επαναχρησιμοποιούνται χωρίς να επηρεάζουν το σύστημα στο σύνολό του, ακόμα και κατά τη λειτουργία του συστήματος.

Οι συνθήκες που παρέχει το REST είναι ιδανικές για την ανάπτυξη Application Programming Interfaces (APIs). Τα APIs που εφαρμόζουν την REST αρχιτεκτονική, ονομάζονται RESTful APIs. Με τη χρήση RESTful APIs διαφορετικές εφαρμογές που χρησιμοποιούνται για τον ίδιο σκοπό, π.χ. Web Banking και Mobile Banking, αντί να έχουν υλοποιημένη κάθε φορά ξεχωριστά και τη λογική του client αλλά και τη λογική του server, μπορούν να υλοποιήσουν μόνο τη λογική του client, και για τη λογική του server να χρησιμοποιηθούν τα ίδια σημεία εισαγωγής (endpoints) ενός RESTful API, τα οποία εκτελούν τις ίδιες ενέργειες για όλους και όλοι λαμβάνουν τις ίδιες απαντήσεις, γρήγορα και αξιόπιστα.

2.4. Ασφάλεια

Η σύγχρονη ανάπτυξη ιστοσελίδων έχει πολλές προκλήσεις, και μία από αυτές – ίσως η σημαντικότερη, αν και συχνά δεν δίνεται αρκετή έμφαση – είναι η ασφάλεια. Ενώ σοβαρές, πολύπλοκες και ίσως κοστοβόρες τεχνικές καταπολέμησης κακόβουλων επιθέσεων έχουν γίνει πλέον αναγκαίες, υπάρχουν και οι βασικές τεχνικές, όπως η πιστοποίηση (authentication), η εξουσιοδότηση (authorization), η κρυπτογράφηση των ευαίσθητων δεδομένων της εφαρμογής,

όπως οι κωδικοί ή τα προσωπικά δεδομένα των χρηστών, ή ακόμα και η επικύρωση (validation) των δεδομένων που εισάγει ο χρήστης, ώστε να αποφευχθούν κακόβουλες επιθέσεις προς τις βάσεις δεδομένων.

2.4.1. Πιστοποίηση (authentication)

Η πιστοποίηση (authentication) ενός χρήστη που ζητάει να αποκτήσει πρόσβαση σε μια εφαρμογή, χρησιμοποιώντας ένα user name και ένα password, υπάρχει ώστε να μπορέσει μια εφαρμογή να μάθει ποιος είναι ο τρέχων χρήστης και εάν είναι πιστοποιημένος ή όχι. Μια πλήρες πλατφόρμα (framework) πιστοποίησης μπορεί να είναι ικανή να γνωρίζει πιθανώς ορισμένα χαρακτηριστικά για τον χρήστη, όπως ένα μοναδικό αναγνωριστικό, πχ τη διεύθυνση ηλεκτρονικού ταχυδρομείου ή το ονοματεπώνυμο του. Στις περισσότερες περιπτώσεις επιλέγεται η τεχνική του session, όπου η εφαρμογή ξέρει ανά πάσα στιγμή ποιος είναι ο χρήστης, τι κάνει και τι πρέπει να του επιτραπεί να κάνει. Αντίθετα, σε εφαρμογές που είναι υλοποιημένες βάσει της REST αρχιτεκτονικής, η οποία προϋποθέτει η εφαρμογή να είναι stateless, δηλαδή να μη κρατάει κάποια κατάσταση, όπως ποιος είναι ο χρήστης που έχει πιστοποιηθεί, χρησιμοποιούνται άλλες τεχνικές, με τη πιο διαδιδομένη από αυτές να είναι η πιστοποίηση με χρήση token.

Το JSON Web Token (JWT) είναι ένα ανοικτό πρότυπο βασισμένο σε JSON για τη δημιουργία tokens πιστοποίησης, τα οποία αποτελούνται από τρία κρυπτογραφημένα μέρη:

- Την επικεφαλίδα (header), η οποία περιέχει πληροφορίες για τον τύπο του token και τον hashing αλγόριθμο που χρησιμοποιεί.
- Το φορτίο (payload), το οποίο περιέχει τους ισχυρισμούς (claims) του token, δηλαδή διάφορα δεδομένα ή metadata, που μπορούν να χρησιμοποιηθούν κατά την πιστοποίηση ή τη ταυτοποίηση των χρηστών.
- Την υπογραφή (signature), η οποία δημιουργείται με τη χρήση του hashing αλγόριθμου, της επικεφαλίδας (header) χρησιμοποιώντας ως δεδομένα την επικεφαλίδα, το φορτίο και ένα μυστικό κλειδί για ασφάλεια.

Σε ένα κλασικό παράδειγμα πιστοποίησης με χρήση token, και συγκεκριμένα με τη χρήση JWT, ο server θα δημιουργούσε ένα token το οποίο έχει το claim "συνδεδεμένος ως διαχειριστής" και θα το έστελνε στον client. Ο client στη συνέχεια

θα χρησιμοποιούσε αυτό το token για να αποδείξει ότι έχει συνδεθεί ως διαχειριστής. Σε ένα RESTful API, ο χρήστης θα χρησιμοποιούσε το JWT σε κάθε αίτηση προς τον server, ώστε να αποδείξει ότι είναι πιστοποιημένος. Τα JWTs είναι σχεδιασμένα ώστε να είναι μικρά σε μέγεθος, URL-safe, να μπορούν να κρυπτογραφηθούν και είναι ιδιαίτερα χρήσιμα σε περιπτώσεις single-sign-on, δηλαδή σε περιπτώσεις όπου ο χρήστης χρησιμοποιεί το ίδιο user name και password για να αποκτήσει πρόσβαση σε πολλές εφαρμογές, όπως πχ Web Banking και Mobile Banking.

2.4.2. Κρυπτογράφηση

Όταν χρησιμοποιείτε μια απλή και συνηθισμένη σύνδεση μεταξύ δύο συστημάτων, οι χρήστες και τα δεδομένα τους εκτίθενται σε πολλούς κινδύνους που προκύπτουν από το γεγονός ότι τα δεδομένα μεταδίδονται σε απλή μορφή. Μία κακόβουλη επίθεση που μπορεί να αποκτήσει πρόσβαση στο δίκτυο επικοινωνίας μεταξύ των συστημάτων, μπορεί να υποκλέψει ή ακόμα και να παραποιήσει τα δεδομένα χωρίς να γίνει αντιληπτό. Δεν υπάρχει κανένα όριο για το τι μπορεί να κάνει ο εισβολέας, συμπεριλαμβανομένης της κλοπής της των δεδομένων, την έγχυση κακόβουλου κώδικα στα συστήματα ή την αλλαγή των δεδομένων που ανταλλάσσουν. Η καλύτερη λύση σε αυτό το πρόβλημα είναι η κρυπτογράφηση όλων των δεδομένων.

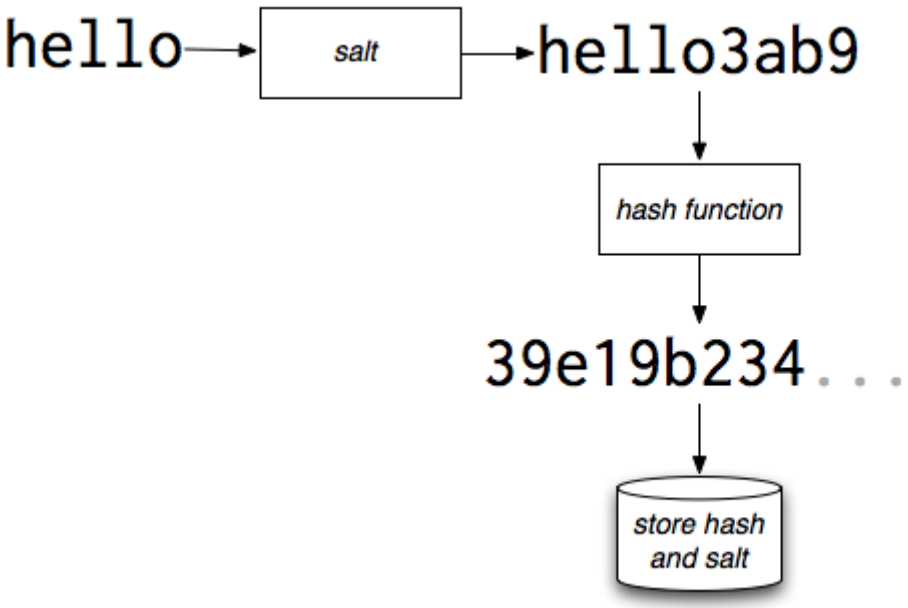
Η πιο διαδεδομένη τεχνική κρυπτογράφησης στο web είναι το HTTPS. Το HTTPS χρησιμοποιήθηκε αρχικά για τη διασφάλιση των ευαίσθητων δεδομένων που ανταλλάσσονταν στο διαδίκτυο, όπως για παράδειγμα είναι οι χρηματοπιστωτικές συναλλαγές. Πλέον, είναι συνηθισμένο να χρησιμοποιείται εξ ορισμού σε πολλές ιστοσελίδες που χρησιμοποιούμε στις καθημερινές μας ζωές, όπως η κοινωνική δικτύωση και οι μηχανές αναζήτησης. Το πρωτόκολλο HTTPS χρησιμοποιεί το πρωτόκολλο Transport Layer Security (TLS), το διάδοχο του πρωτοκόλλου Secure Sockets Layer (SSL), για την εξασφάλιση ασφαλών επικοινωνιών. Η λειτουργία του HTTPS είναι η εξής: παίρνει το ήδη υπάρχον HTTP πρωτόκολλο που χρησιμοποιείται, και προσθέτει ένα επίπεδο κρυπτογράφησης, χρησιμοποιώντας το SSL, πάνω από αυτό. Οι clients και οι servers εξακολουθούν να επικοινωνούν με ακριβώς τον ίδιο HTTP τρόπο μεταξύ τους, αλλά μέσω μιας ασφαλούς σύνδεσης SSL που κρυπτογραφεί και αποκρυπτογραφεί τα αιτήματα και τις απαντήσεις τους. Αν στηθεί και χρησιμοποιηθεί σωστά, παρέχει υψηλή

προστασία από υποκλοπές και παραποιήσεις δεδομένων. Με όλο και περισσότερους κινδύνους και επιθέσεις στο διαδίκτυο να εμφανίζονται καθημερινά, πλέον γίνεται όλο και πιο λογικό είναι να αντιμετωπίζεται όλη η κυκλοφορία των δεδομένων ως ευαίσθητη και να κρυπτογραφείται.

Παρόλα αυτά, η κρυπτογράφηση της κυκλοφορίας των δεδομένων μεταξύ client και server, δεν είναι αρκετή. Συχνά οι χρήστες δεν είναι αρκετό να προστατεύονται από κακόβουλες επιθέσεις, αλλά ακόμα και από τους ίδιους. Για παράδειγμα, η μη ασφαλή αποθήκευση κωδικών δημιουργεί κινδύνους, τόσο εσωτερικούς όσο και εξωτερικούς. Στην πρώτη περίπτωση, ένας προγραμματιστής εφαρμογών ή ένας διαχειριστής βάσεων δεδομένων ο οποίος έχει πρόσβαση και μπορεί να διαβάσει τον πίνακα που περιέχει τα user names και passwords, έχει πλέον πρόσβαση στα διαπιστευτήρια (credentials) όλων των χρηστών. Ακόμη και αν το συγκεκριμένο σενάριο δεν προκαλεί μεγάλη ανησυχία, η αποθήκευση των διαπιστευτηρίων των χρηστών χωρίς την κατάλληλη κρυπτογραφική προστασία δημιουργεί ένα κενό ασφαλείας που ουδεμία σχέση έχει με την εφαρμογή αυτή καθαυτή. Ένα μεγάλο πρόβλημα είναι το γεγονός ότι οι χρήστες ξαναχρησιμοποιούν τα ίδια διαπιστευτήρια. Πολλοί χρήστες χρησιμοποιούν τα ίδια διαπιστευτήρια που χρησιμοποιούν για το e-mail τους, το Web Banking τους, τα μέσα κοινωνικής δικτύωσης ή ακόμα και για απλά forums. Αυτό έχει ως αποτέλεσμα βάσεις δεδομένων που διατηρούν διαπιστευτήρια φαινομενικά χαμηλού κινδύνου, όπως ενός forum, να διατηρούν ταυτόχρονα διαπιστευτήρια και για τραπεζικές συναλλαγές. Αν ένας κακόβουλος υπάλληλος ή ένας εξωτερικός χάκερ κλέψει τα δεδομένα των διαπιστευτηρίων, θα μπορούσε να τα χρησιμοποιήσει για απόπειρες σύνδεσης σε τραπεζικές ιστοσελίδες μέχρι να βρεθεί το ένα άτομο που έκανε αυτό το λάθος και χρησιμοποίησε τα ίδια διαπιστευτήρια του σε μια απλή ιστοσελίδα και στο Web Banking του.

Η λύση σε αυτό το πρόβλημα είναι να μην αποθηκεύεται ποτέ αυτούσιος ο ίδιος ο κωδικός πρόσβασης, αλλά ένα hash αυτού. Ένας cryptographic hashing αλγόριθμος είναι μια μεταμόρφωση μίας διαδρομής (one-way transformation) μίας εισόδου (input) σε μία έξοδο (output), από την οποία έξοδο (output) είναι αδύνατο να ανακτηθεί η είσοδος (input). Για παράδειγμα, ένας κωδικός πρόσβασής “test123” εφαρμόζοντας τον hash αλγόριθμο BCrypt δίνει το hex αποτέλεσμα «$2a$04$..BAZ6CKB7.B6OkN/Gr6nunaMPnc6NXKeqOkSA4WnAPH

6bdDMZ8VS». Για να μπορέσει να γίνει η επικύρωση του κωδικού πρόσβασης ενός χρήστη, εφαρμόζεται ο ίδιος hash αλγόριθμος στον μη κρυπτογραφημένο κωδικό πρόσβασης και γίνεται η σύγκριση μεταξύ τους. Το πρόβλημα είναι ότι κάθε χρήστης με τον ίδιο κωδικό πρόσβασης θα έχει το ίδιο hash στη βάση δεδομένων, με αποτέλεσμα να μην λύνουμε το παραπάνω πρόβλημα όπου ένας κωδικός χρησιμοποιείται για πολλές ιστοσελίδες, και αν κλαπεί, μπορεί να δώσει πρόσβαση σε όλες. Η τεχνική για να λυθεί αυτό το πρόβλημα είναι η προσθήκη επιπλέον δεδομένων στο κωδικό πρόσβασης πριν το hashing, τα οποία ονομάζονται salt, έτσι ώστε να γίνει το αποτέλεσμα του hash αλγόριθμου κάθε φορά απρόβλεπτο, με αποτέλεσμα ο ίδιος κωδικός πρόσβασης να μην έχει το ίδιο hash, αλλά και να μην μπορεί εύκολα να αναστραφεί μηχανικά.



Εικόνα 5: Λειτουργία hash με χρήση salt

ΚΕΦΑΛΑΙΟ 3

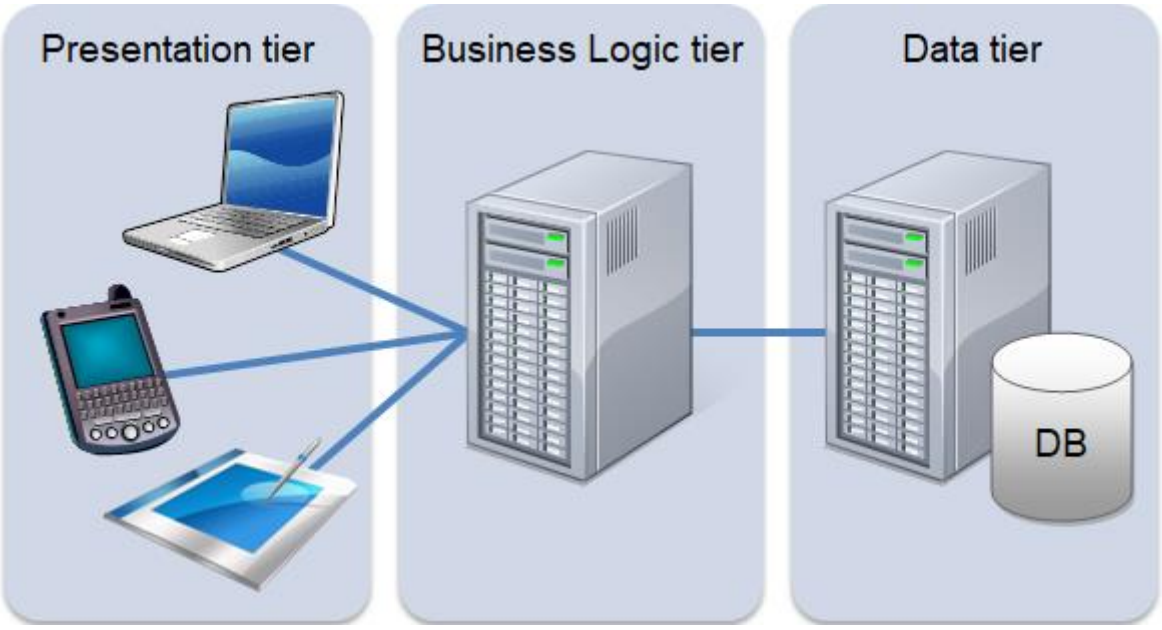
ΑΝΑΠΤΥΞΗ ΚΑΙ ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

Σε αυτό το κεφάλαιο θα περιγραφεί η δομή της εφαρμογής σε αρχιτεκτονικό, παρουσιαστικό αλλά και σε λειτουργικό επίπεδο, καθώς ο τρόπος εκτέλεσης όλων των λειτουργιών του ιστοτόπου.

3.1. Αρχιτεκτονική δομή εφαρμογής

Η εφαρμογή είναι βασισμένη στην αρχιτεκτονική τριών επιπέδων (3-Tier architecture). Το πρώτο επίπεδο (επίπεδο παρουσίασης, presentation tier) αποτελείται από την ιστοσελίδα η οποία λειτουργεί ως η διεπαφή προς τον χρήστη. Η ιστοσελίδα έχει αναπτυχθεί χρησιμοποιώντας HTM5, CSS3 και JavaScript. Πιο συγκεκριμένα, έχει γίνει χρήση του Bootstrap, το οποίο είναι ένα από τα πιο διαδεδομένα CSS frameworks, της React και του Redux. Επίσης, ως application server έχει επιλεχτεί ο Node.js. Το δεύτερο (επίπεδο επιχειρησιακής λογικής, business logic tier) αποτελείται από τον REST API server. Ο REST API server έχει αναπτυχθεί χρησιμοποιώντας τη πλατφόρμα .NET της Microsoft και πιο συγκεκριμένα, έχει γίνει χρήση της γλώσσας C#, του Entity Framework και ως server έχει χρησιμοποιηθεί ο Internet Information Services(IIS) web server. Το τρίτο και τελευταίο επίπεδο (επίπεδο δεδομένων, data tier) αποτελείται από τη βάση δεδομένων και τον κώδικα που επικοινωνεί και διαχειρίζεται αυτή. Ως βάση δεδομένων έχει χρησιμοποιηθεί ο Microsoft SQL Server Express.

Η επικοινωνία μεταξύ των επιπέδων γίνεται μόνο μεταξύ γειτονικών επιπέδων. Δηλαδή, το πρώτο επίπεδο (ιστοσελίδα) επικοινωνεί μόνο με το δεύτερο επίπεδο (REST API server), το δεύτερο επίπεδο ως ενδιάμεσο επίπεδο επικοινωνεί και με το πρώτο επίπεδο αλλά και με το τρίτο επίπεδο (κώδικας διαχείρισης βάσης δεδομένων/βάση δεδομένων) και το τρίτο επίπεδο επικοινωνεί μόνο με το δεύτερο επίπεδο.



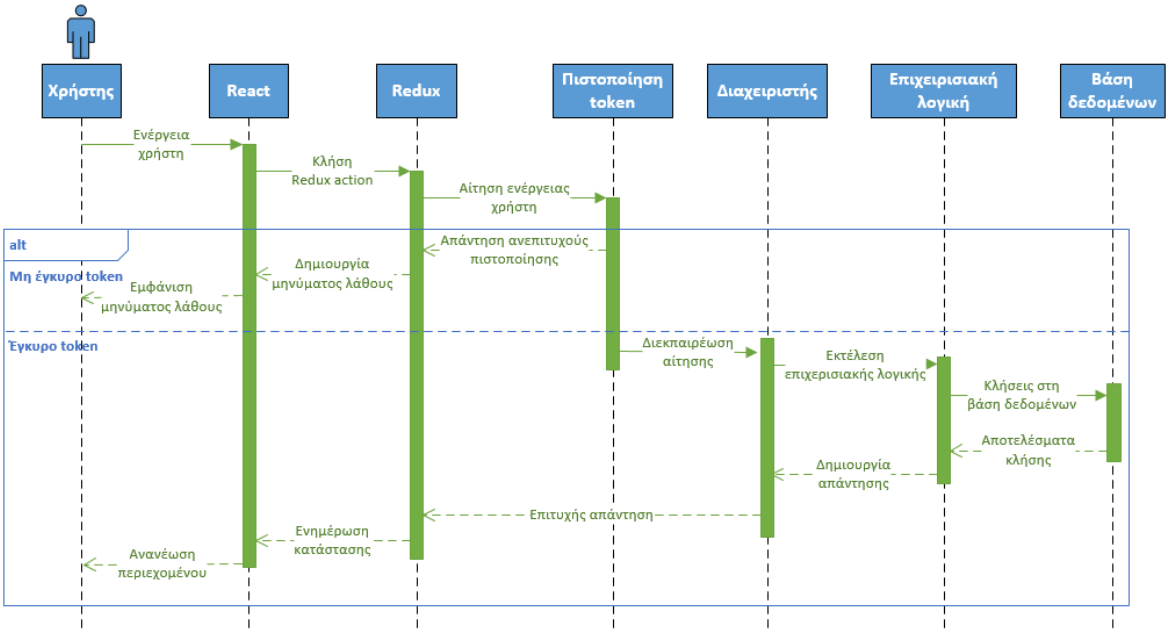
Εικόνα 6: Απεικόνιση αρχιτεκτονικής τριών επιπέδων

Η αρχιτεκτονική τριών επιπέδων επιλέχτηκε επειδή δίνει αρκετά πλεονεκτήματα και ιδιαίτερη ευελιξία κατά την υλοποίηση αλλά και κατά την επέκταση της εφαρμογής, λόγω της έλλειψης άμεσης σύνδεσης (loose coupling) μεταξύ των επιπέδων. Αναλυτικότερα, μερικά από αυτά τα πλεονεκτήματα είναι:

- Το λογισμικό που αποτελεί το κάθε επίπεδο μπορεί να γραφτεί σε τελείως διαφορετικές γλώσσες προγραμματισμού, αλλά η επικοινωνία των επιπέδων να πραγματοποιείται χωρίς προβλήματα. Στην περίπτωση αυτής της εφαρμογής, χρησιμοποιήθηκε η JavaScript στο επίπεδο παρουσίασης (ιστοσελίδα), η πλατφόρμα .net και η γλώσσα C# στο επίπεδο επιχειρησιακής λογικής (REST API server) και ο Microsoft SQL Server και η C# στο επίπεδο δεδομένων.
- Το λογισμικό που αποτελεί το κάθε επίπεδο μπορεί να αναπτυχθεί ξεχωριστά, σε διαφορετικές χρονικές περιόδους ή και παράλληλα, ακόμη και από διαφορετικούς προγραμματιστές.
- Στη περίπτωση που χρειάζεται να αλλάξει κάποια λειτουργικότητα (functionality), για παράδειγμα στο επίπεδο επιχειρησιακής λογικής (όπως ο τρόπος υπολογισμού του υπολοίπου ενός λογαριασμού), η

αλλαγή θα γίνει μόνο εκεί, και δε θα χρειαστεί να γίνει καμία απολύτως ενέργεια ή αλλαγή στα υπόλοιπα επίπεδα.

Για κάθε ενέργεια του χρήστη στο επίπεδο διεπαφής (ιστοσελίδα Web Banking), ακολουθείται η εξής ροή: Μέσω ενός React κομματιού κώδικα (component) το οποίο ανήκει σε μια από τις διαδρομές (routes) που είναι «σπασμένη» η ιστοσελίδα, ο χρήστης θα ζητήσει μια τραπεζική συναλλαγή (πχ πληρωμή). Το React component θα πραγματοποιήσει μία ή και περισσότερες κλήσεις (ανάλογα τη συναλλαγή) στο επίπεδο της επιχειρησιακής λογικής (REST API server) μέσω μιας ενέργειας (action) του reducer του Redux στο οποίο ανήκει, πάντα στέλνοντας μαζί και το token πιστοποίησης. Το πρώτο πράγμα που θα κάνει ο REST API server ανεξαρτήτως αίτησης είναι να ελέγξει την εγκυρότητα του token, δηλαδή αν είναι σωστό ή αν έχει λήξει (η μέγιστη διάρκεια ζωής των token έχουν οριστεί σε 10 λεπτά), ώστε να αποφασίσει αν θα επιτραπεί η διεκπεραίωση της αίτησης ή όχι. Στη συνέχεια, αν το token είναι έγκυρο, ανάλογα την αίτηση που έχει δεχτεί ο REST API server, εκτελείται η κατάλληλη επιχειρησιακή λογική, γίνονται οι απαραίτητες κλήσεις προς το επίπεδο δεδομένων (βάση δεδομένων) με τη χρήση του Entity Framework της πλατφόρμας .NET και αν όλα έχουν πάει καλά ετοιμάζεται η κατάλληλη HTTP απάντηση επιτυχίας, διαφορετικά ετοιμάζεται η κατάλληλη απάντηση αποτυχίας, μαζί με ένα κατάλληλο μήνυμα για το χρήστη. Αν το token δεν είναι έγκυρο, τότε ετοιμάζεται η HTTP απάντηση μη έγκυρης πιστοποίησης. Έπειτα, αποστέλλεται η απάντηση στο επίπεδο διεπαφής (ιστοσελίδα), η οποία λαμβάνεται από το action του Redux, το οποίο εν ακολουθία ανανεώνει τη κατάσταση (state) του reducer που πραγματοποίησε τη αίτηση, μέσω του component της React. Τέλος, τα περιεχόμενα του component από το οποίο ξεκίνησε όλη η διαδικασία και οποιουδήποτε άλλου συμμετέχει στη τραπεζική συναλλαγή ανανεώνονται αυτόματα λόγω της αλλαγής της κατάστασης του reducer και ο χρήστης βλέπει το αποτέλεσμα στην οθόνη του.



Εικόνα 7: Διάγραμμα ακολουθίας ενέργειας χρήστη

Αυτή η διαδικασία πραγματοποιείται ανεξαιρέτως για όλες τις τραπεζικές συναλλαγές της ιστοσελίδας, από τη μικρότερη μέχρι και τη μεγαλύτερη.

3.2. REST API Server

Όλη η λειτουργικότητα του Web Banking έχει χωριστεί σε «λειτουργικές περιοχές» οι οποίες προκύψανε από την τμηματοποίηση και ομαδοποίηση όλων των λειτουργιών σύμφωνα με τη τραπεζική οντότητα την οποία αφορούν. Οι λειτουργικές περιοχές που αναγνωρίστηκαν με αυτό τον τρόπο είναι οι εξής:

- Τράπεζες
- Πελάτης
- Λογαριασμοί
- Κάρτες
- Δάνεια
- Πληρωμές
- Μεταφορές

- Πάγιες εντολές
- Φορτίσεις
- Ιστορικό

Για κάθε μια από αυτές τις λειτουργικές περιοχές, έχει φτιαχτεί και ένας διαχειριστής (controller) ο οποίος περιέχει τα υλοποιημένα σημεία εισαγωγής (endpoints) της κάθε λειτουργικής περιοχής, που γίνονται διαθέσιμα μέσω του REST API server, πίσω από τα οποία είναι υλοποιημένη όλη η επιχειρησιακή λογική. Επίσης, υπάρχει ακόμη ένα διαθέσιμο σημείο εισαγωγής που δεν υπάγεται σε κάποια από τις παραπάνω λειτουργικές περιοχές, και αυτό είναι το σημείο εισαγωγής πιστοποίησης.

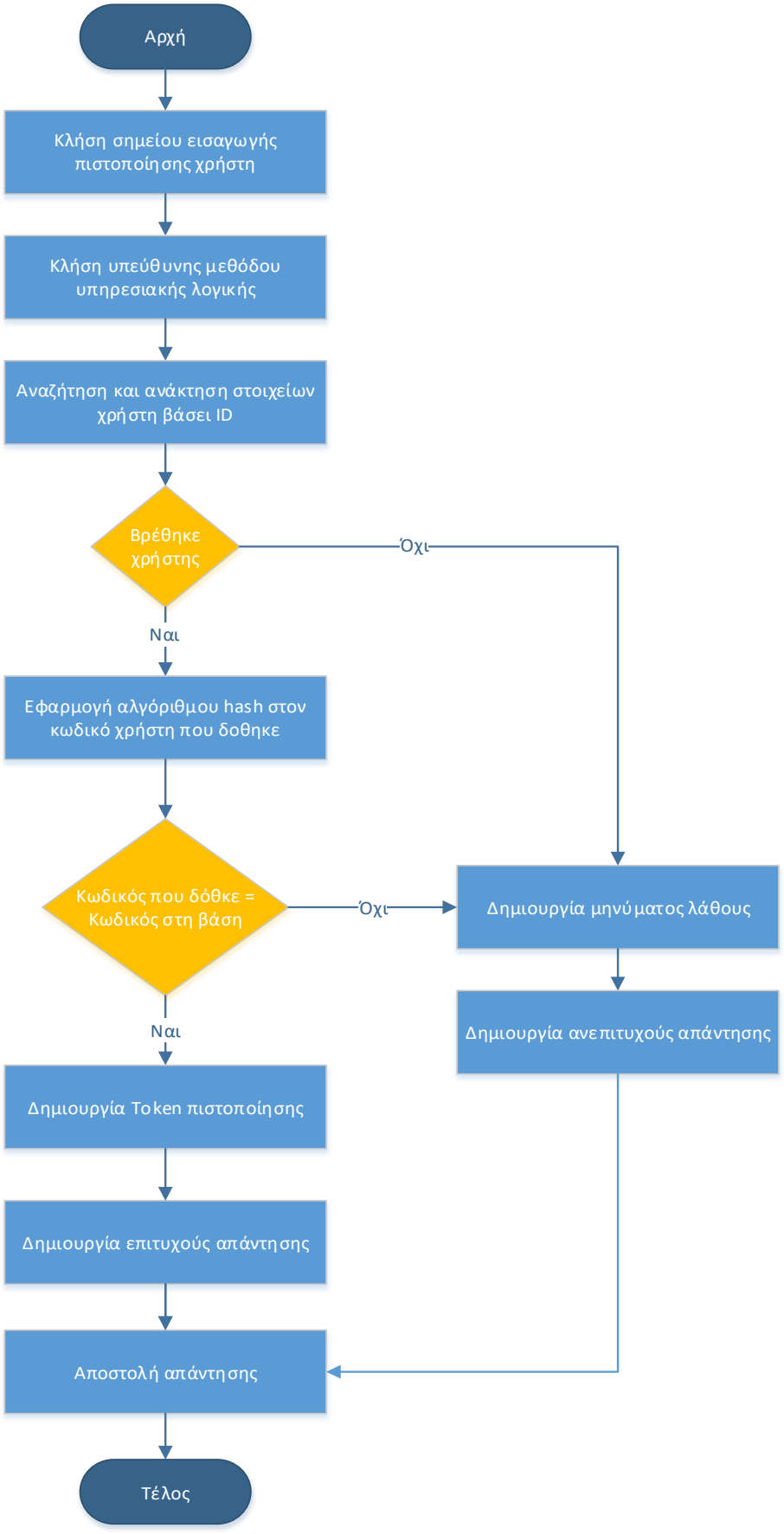
Λειτουργική περιοχή	Τύπος HTTP κλήσης	Περιγραφή κλήσης/σημείο εισαγωγής	Παράμετροι
Πιστοποίηση	POST	Πιστοποίηση χρήστη	ID και κωδικό χρήστη
Τράπεζες	GET	Ανάκτηση τραπεζών εσωτερικού	-
Πελάτης	GET	Ανάκτηση ονόματος πελάτη	Αριθμός πελάτη
Λογαριασμοί	GET	Ανάκτηση στοιχείων λογαριασμού	Αριθμός λογαριασμού
Λογαριασμοί	GET	Ανάκτηση στοιχείων όλων των λογαριασμών	Αριθμός πελάτη
Κάρτες	GET	Ανάκτηση στοιχείων όλων των καρτών	Αριθμός πελάτη
Κάρτες	GET	Ανάκτηση στοιχείων πιστωτικής κάρτας	Αριθμός πιστωτικής κάρτας
Κάρτες	GET	Ανάκτηση στοιχείων χρεωστικής κάρτας	Αριθμός χρεωστικής κάρτας
Κάρτες	GET	Ανάκτηση στοιχείων προπληρωμένης κάρτας	Αριθμός προπληρωμένης κάρτας
Κάρτες	DELETE	Αποσύνδεση προϊόντος από χρεωστική κάρτα	Αριθμός χρεωστικής κάρτας και προϊόντος
Δάνεια	GET	Ανάκτηση στοιχείων δανείου	Αριθμός δανείου
Δάνεια	GET	Ανάκτηση στοιχείων όλων των δανείων	Αριθμός πελάτη
Πληρωμές	GET	Ανάκτηση διαθέσιμων πληρωμών	-
Πληρωμές	POST	Πληρωμή προϊόντος της τράπεζας	Στοιχεία πληρωμής
Πληρωμές	POST	Πληρωμή προς τρίτους	Στοιχεία πληρωμής
Μεταφορές	POST	Μεταφορά χρημάτων	Στοιχεία μεταφοράς

Πάγιες εντολές	GET	Ανάκτηση στοιχείων όλων των πάγιων εντολών μεταφοράς	Αριθμός πελάτη
Πάγιες εντολές	POST	Δημιουργία πάγιας εντολής μεταφοράς	Στοιχεία πάγιας εντολής μεταφοράς
Πάγιες εντολές	DELETE	Διαγραφή πάγιας εντολής μεταφοράς	Αριθμός πάγιας εντολής
Πάγιες εντολές	GET	Ανάκτηση στοιχείων όλων των πάγιων εντολών πληρωμής	Αριθμός πελάτη
Πάγιες εντολές	POST	Δημιουργία πάγιας εντολής πληρωμής	Στοιχεία πάγιας εντολής πληρωμής
Πάγιες εντολές	DELETE	Διαγραφή πάγιας εντολής πληρωμής	Αριθμός πάγιας εντολής
Φορτίσεις	POST	Φόρτιση προπληρωμένης κάρτας	Στοιχεία φόρτισης προπληρωμένης κάρτας
Ιστορικό	GET	Ανάκτηση ιστορικού συναλλαγών τρέχοντος μήνα	Αριθμός προϊόντος
Ιστορικό	GET	Ανάκτηση ιστορικού συναλλαγών συγκεκριμένης περιόδου	Αριθμός προϊόντος, ημερομηνία αρχής και τέλους

Πίνακας 1: Διαθέσιμα REST API σημεία εισαγωγής

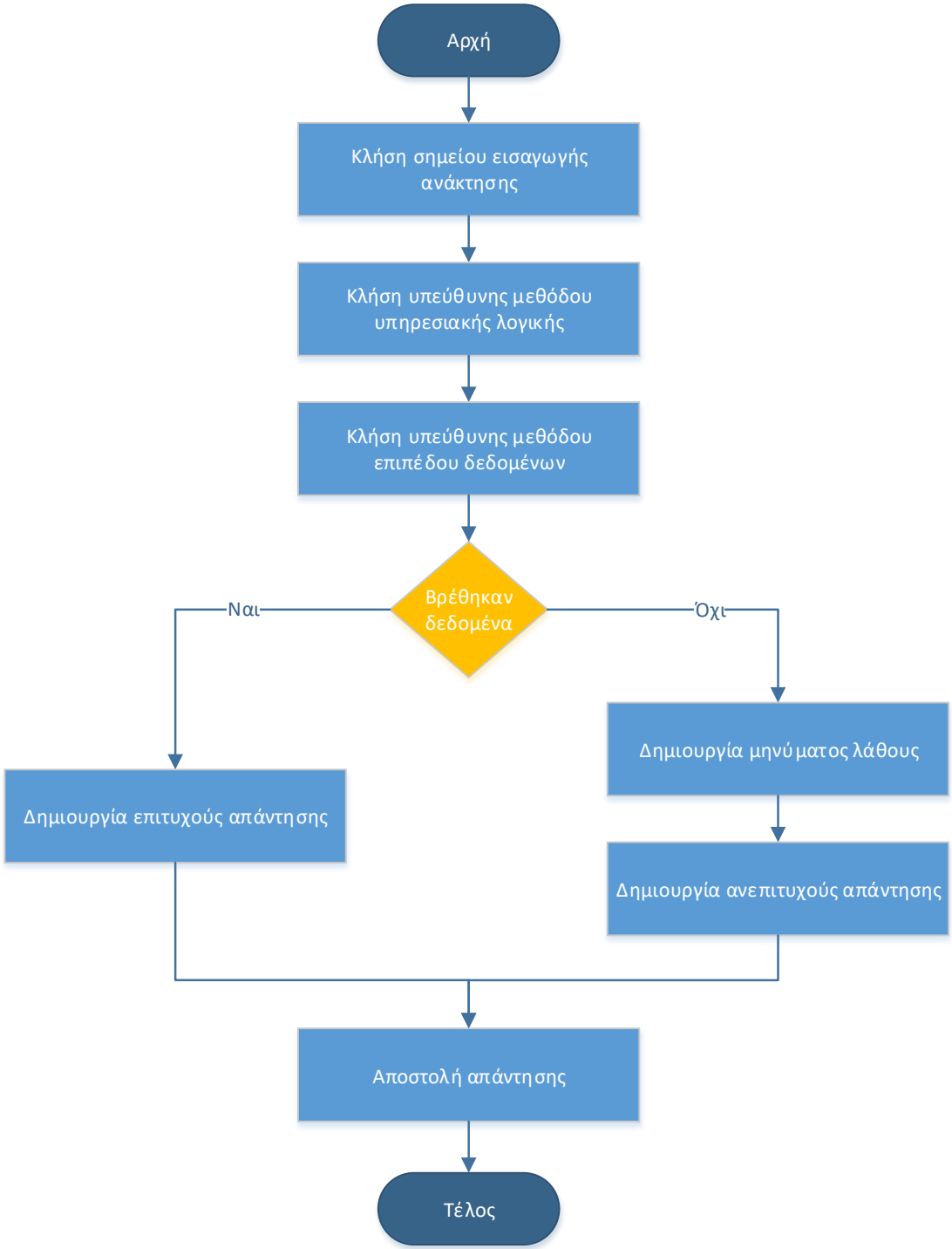
Όλες οι κλήσεις που δύναται να επιστρέψουν κάποιο μήνυμα, δέχονται επίσης ως παράμετρο και την γλώσσα (ελληνικά/αγγλικά) στην οποία θα επιστραφεί το μήνυμα, στο πλαίσιο της υποστήριξης και πλήρης λειτουργίας του Web Banking και στις δύο αυτές γλώσσες.

Η πιο σημαντική διαθέσιμη κλήση, που πρέπει να καλεστεί πρώτα και επιτυχημένα αναγκαστικά ώστε να γίνουν και τα υπόλοιπα σημεία εισαγωγής διαθέσιμα είναι αυτή της πιστοποίησης.



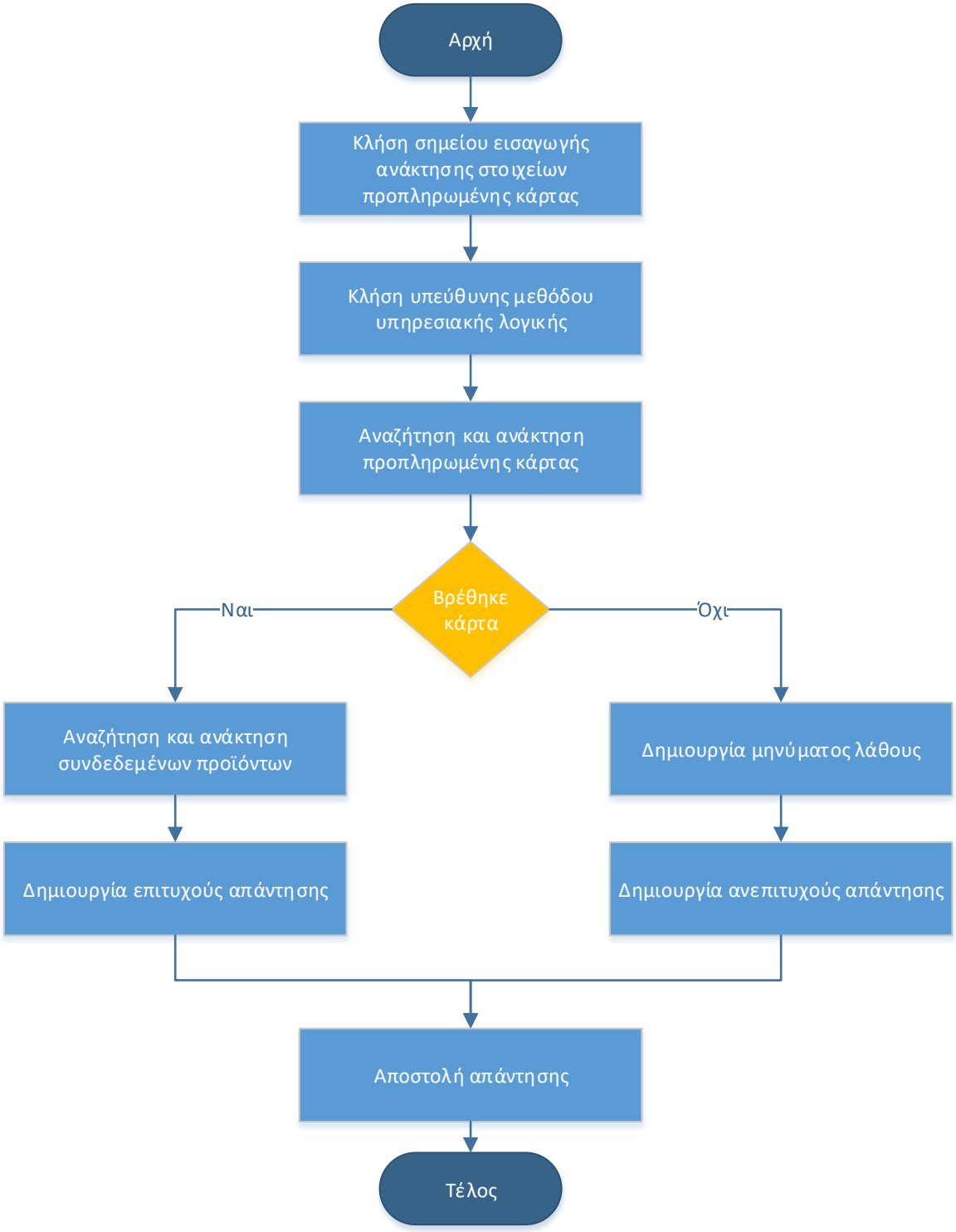
Εικόνα 8: Διάγραμμα ροής εκτέλεσης πιστοποίησης χρήστη

Οι κλήσεις οι οποίες είναι υπεύθυνες για ανάκτηση δεδομένων ακολουθούν την εξής ροή εκτέλεσης: Πραγματοποιείται η κλήση στο αντίστοιχο σημείο εισαγωγής, το οποίο καλεί μια μέθοδο στο επίπεδο επιχειρησιακής λογικής, η οποία με τη σειρά της καλεί μια άλλη μέθοδο στο επίπεδο δεδομένων, η οποία φέρνει και επιστρέφει τα στοιχεία που ζητήθηκαν. Η παραπάνω διαδικασία αποτυπώνεται στο παρακάτω διάγραμμα.



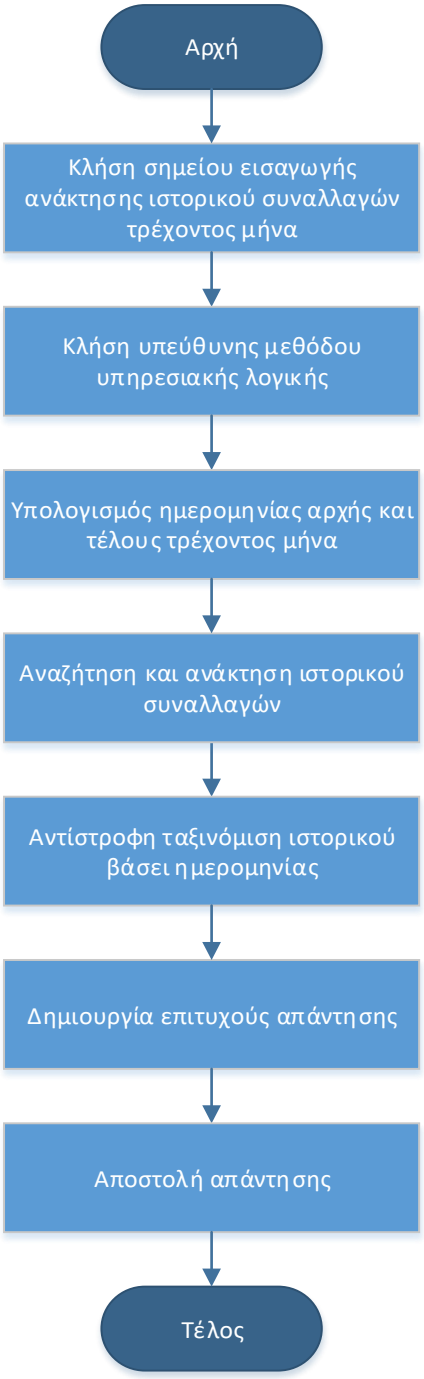
Εικόνα 9: Διάγραμμα ροής εκτέλεσης ανάκτησης δεδομένων

Εξαίρεση αποτελούν οι κλήσεις ανάκτησης στοιχείων προπληρωμένων καρτών, και οι ανακτήσεις ιστορικού συναλλαγών. Στη περίπτωση της ανάκτησης στοιχείων προπληρωμένης κάρτας χρειάζονται να γίνουν παραπάνω από μία κλήσεις στη βάση.

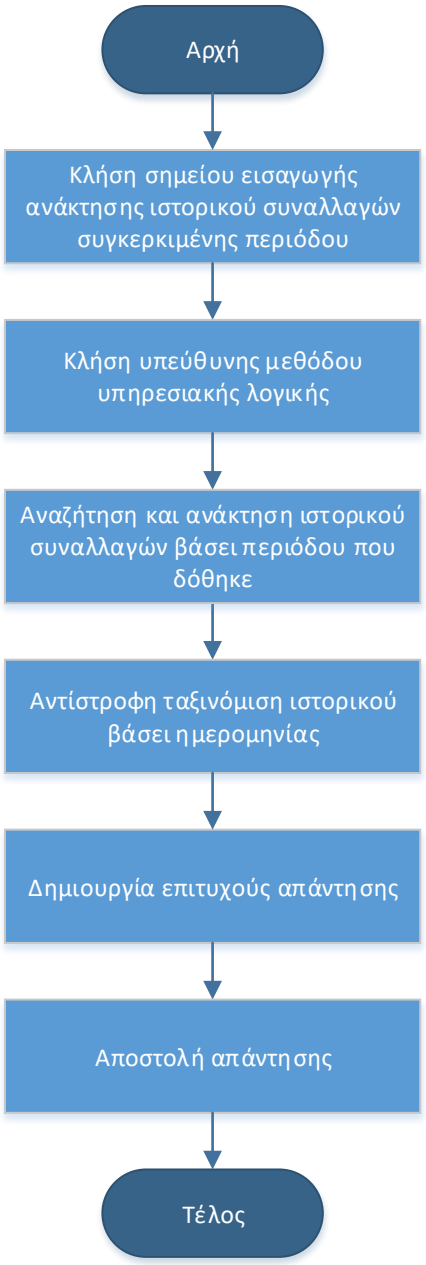


Εικόνα 10: Διάγραμμα ροής ανάκτησης στοιχείων προπληρωμένης κάρτας

Οι κλήσεις της ανάκτησης ιστορικού εκτελούν επιπλέον λογική πριν και μετά τη κλήση στη βάση. Στη περίπτωση της ανάκλησης ιστορικού συναλλαγών τρέχοντος μήνα πρέπει να υπολογιστούν οι ημερομηνίες αρχής και τέλους του μήνα, ενώ και στη περίπτωση της ανάκτησης ιστορικού συναλλαγών τρέχοντος μήνα και σε αυτή της περιόδου, το ιστορικό επιστρέφεται σε αντιστροφή σειρά, βάσει ημερομηνίας (πρώτα οι νεότερες)

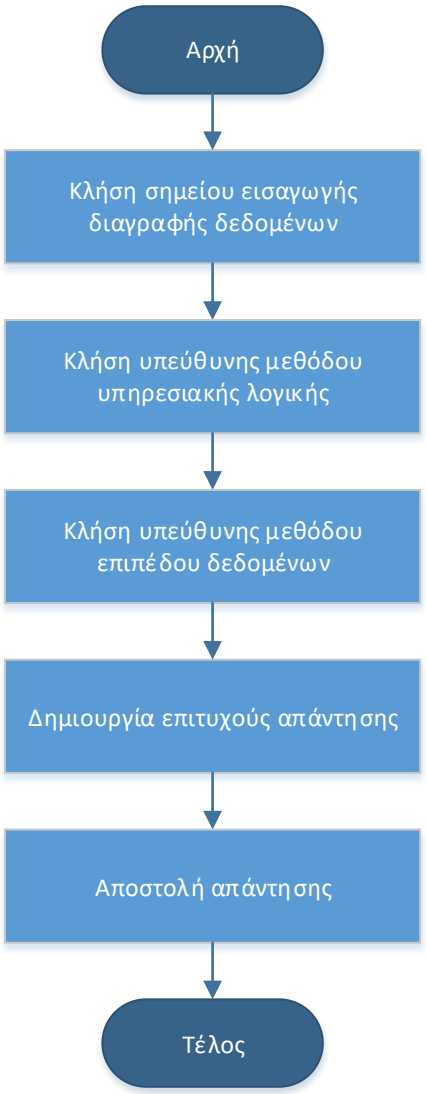


Εικόνα 11: Διάγραμμα ροής εκτέλεσης ανάκτησης ιστορικού συναλλαγών τρέχοντος μήνα



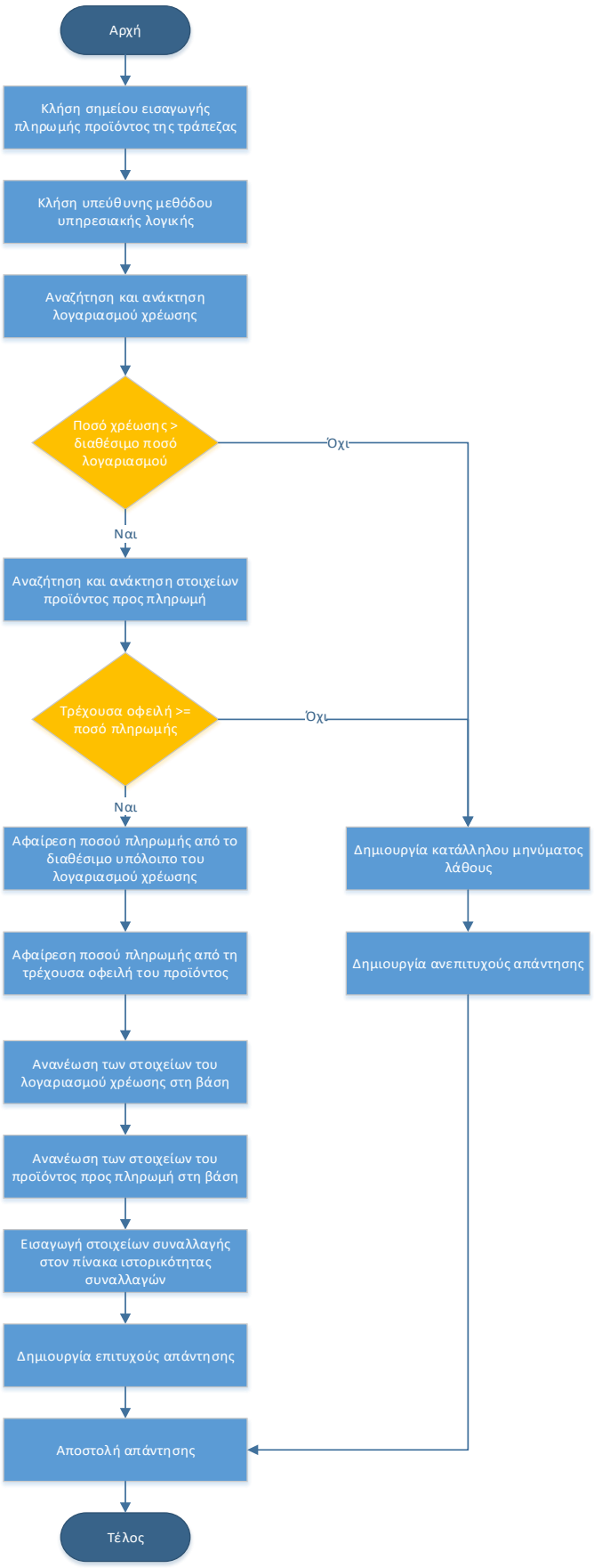
Εικόνα 12: Διάγραμμα ροής εκτέλεσης ανάκτησης ιστορικού συναλλαγών συγκεκριμένης περιόδου

Οι κλήσεις οι οποίες είναι υπεύθυνες για διαγραφή δεδομένων η ροή εκτέλεσης είναι παρόμοια με αυτή της ανάκτησης. Η μόνη διαφορά είναι ότι τα στοιχεία που αναζητούνται στη βάση δεν επιστρέφονται αλλά διαγράφονται.

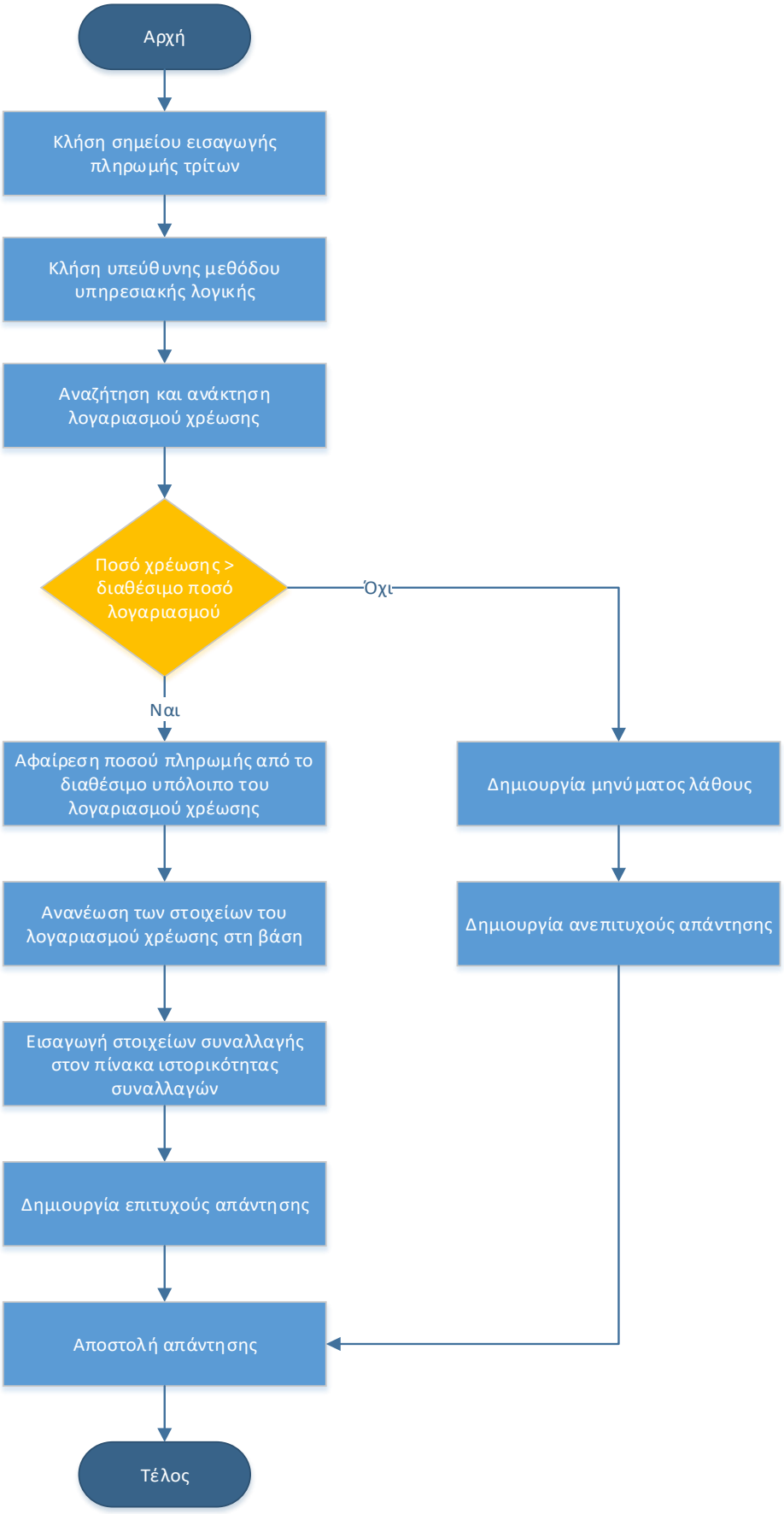


Εικόνα 13: Διάγραμμα ροής εκτέλεσης διαγραφής δεδομένων

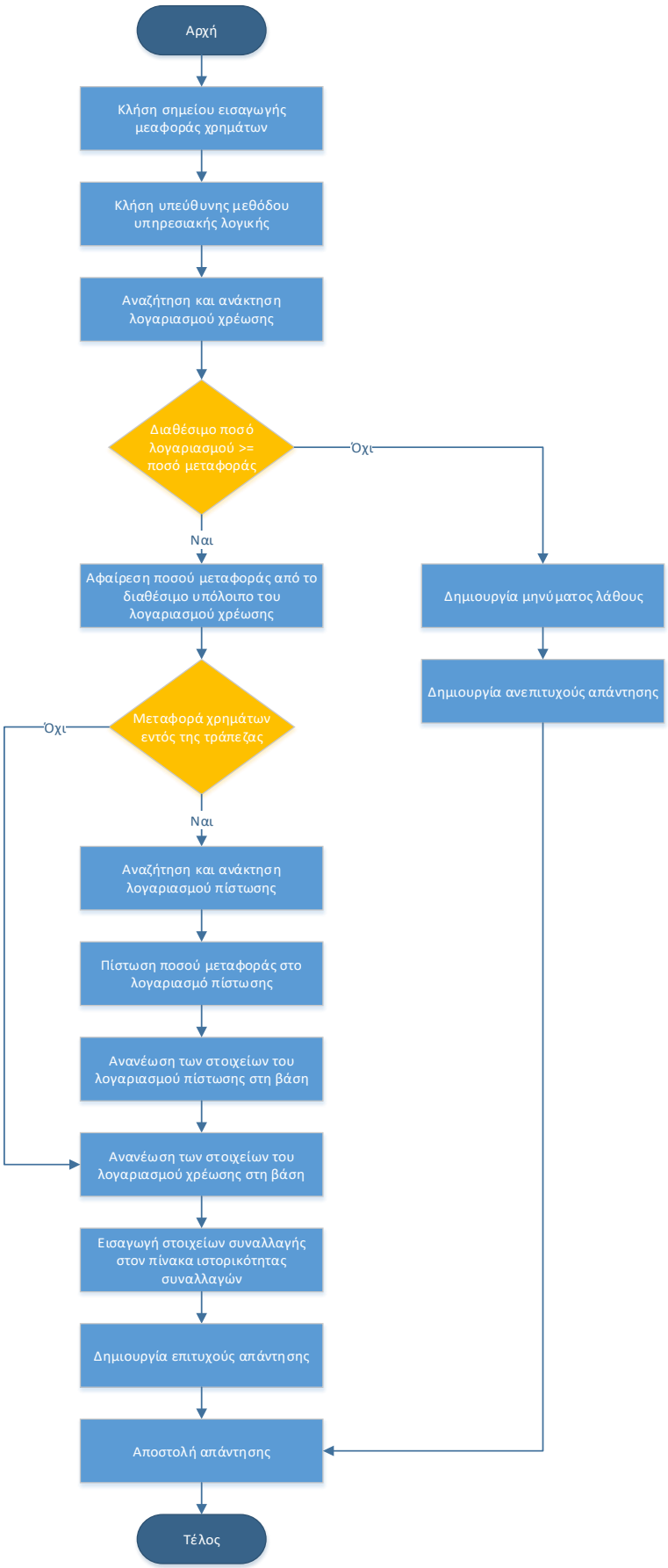
Οι υπόλοιπες κλήσεις (POST) περιέχουν πολύ περισσότερη επιχειρησιακή λογική και εκτελούν πολύ πιο σύνθετες συναλλαγές, όπως είναι η πληρωμή, η μεταφορά χρημάτων και η φόρτιση προπληρωμένων καρτών, με αποτέλεσμα οι ροές εκτέλεσης τους να είναι αρκετά διαφορετικές από τις προηγούμενες. Ακολουθούν οι ροές εκτέλεσης των υπολοίπων κλήσεων.



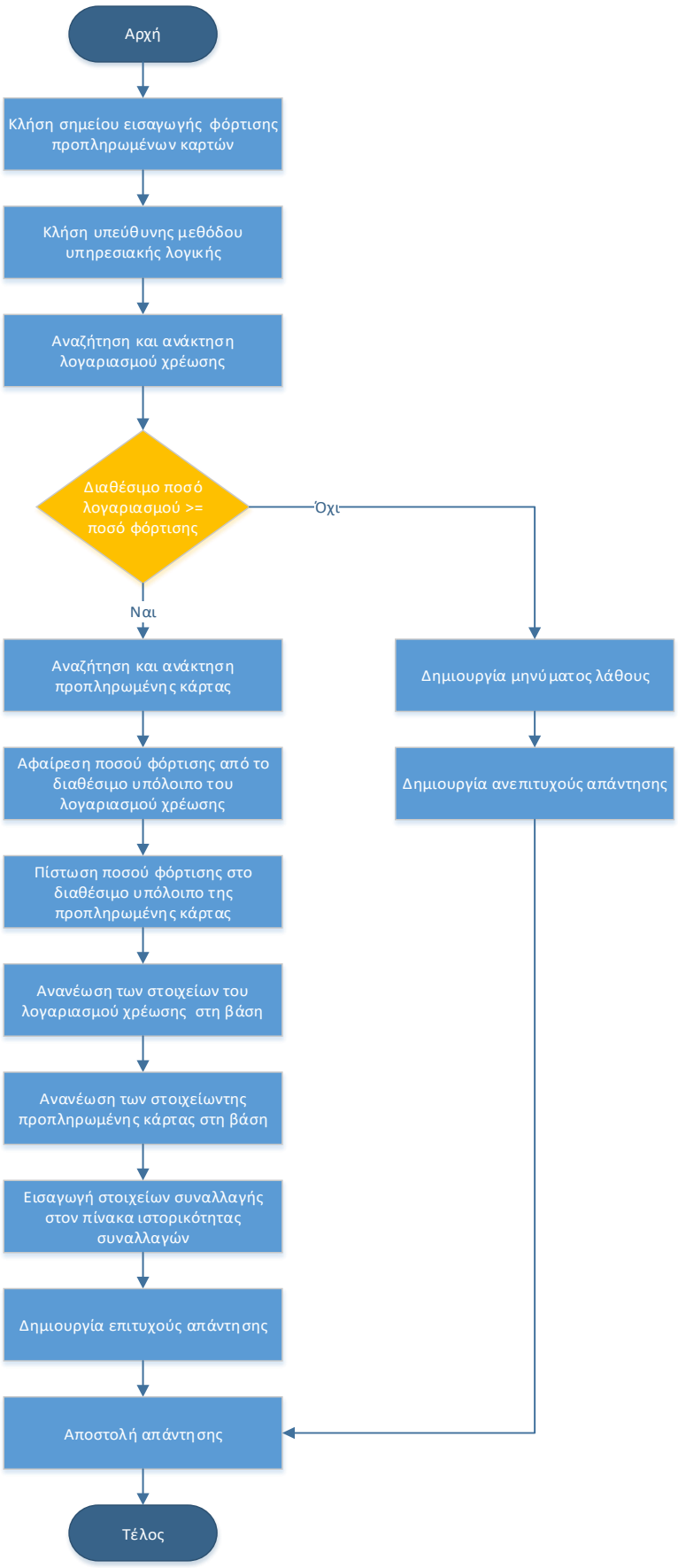
Εικόνα 14: Διάγραμμα ροής εκτέλεσης πληρωμής προϊόντος της τράπεζας



Εικόνα 15: Διάγραμμα ροής εκτέλεσης πληρωμής τρίτων



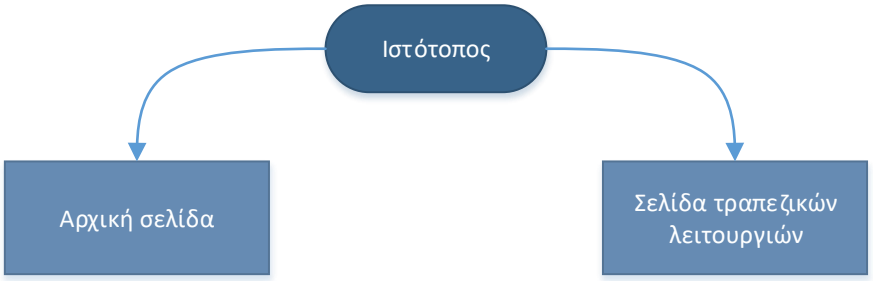
Εικόνα 16: Διάγραμμα ροής εκτέλεσης μεταφοράς χρημάτων



Εικόνα 17: Διάγραμμα ροής εκτέλεσης φόρτισης προπληρωμένης κάρτας

3.3. Δομή ιστότοπου Web Banking

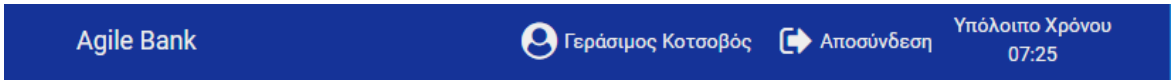
Ο ιστότοπος έχει αναπτυχθεί ως Single Page Application (SPA, εφαρμογή μίας σελίδας) χρησιμοποιώντας τη React JavaScript βιβλιοθήκη. Όλος ο ιστότοπος δηλαδή φορτώνεται σε μία και μόνο σελίδα, η οποία αλλάζει δυναμικά το περιεχόμενο της, κάνοντας διαθέσιμα διαφορετικά React κομμάτια (components) απεικόνισης. Αυτό έχει ως αποτέλεσμα ο ιστότοπος να έχει διαχωριστεί σε εσωτερικά routes (διαδρομές) και υποδιαδρομές αυτών. Κάθε URL που επισκέπτεται ο χρήστης αποτελεί και ένα ξεχωριστό route. Ο ιστότοπος αποτελείται από τρία βασικά κομμάτια: την επικεφαλίδα, τον χώρο απεικόνισης της κάθε διαδρομής, και του υποσέλιδου. Οι βασικές διαδρομές που έχει διαχωριστεί ο ιστότοπος είναι δύο: η αρχική σελίδα, και η σελίδα τραπεζικών λειτουργιών.



Εικόνα 18: Βασικές διαδρομές ιστοτόπου Web Banking

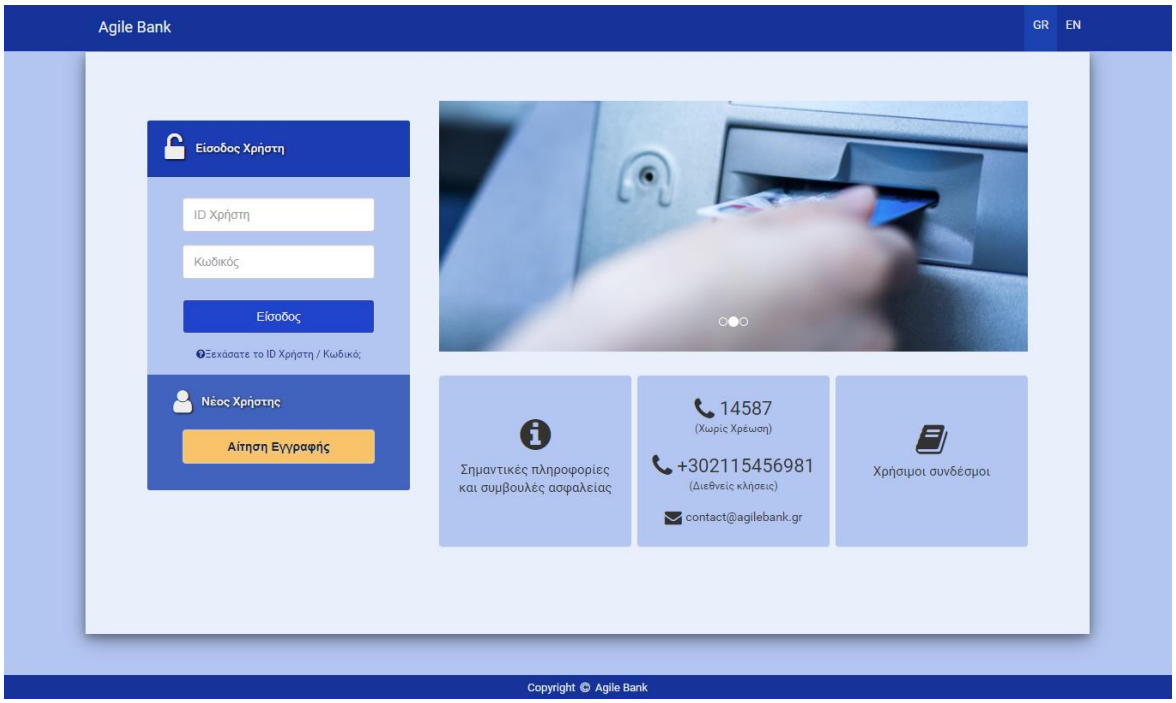
Η επικεφαλίδα αλλάζει τα περιεχόμενα της συναρτήσει της ενεργής διαδρομής. Στη περίπτωση που η αρχική σελίδα είναι ενεργή τότε περιέχει το όνομα της τράπεζας και επιλογή γλώσσας, ενώ όταν η σελίδα τραπεζικών λειτουργιών είναι ενεργή, τη θέση της επιλογής της γλώσσας παίρνει το ονοματεπώνυμο του χρήστη, το κουμπί αποσύνδεσης καθώς και έναν μετρητής ο οποίος για λόγους ασφαλείας όταν τελειώσει, όταν λήξει δηλαδή το token πιστοποίησης, αποσυνδέει αυτόματα τον χρήστη.

Εικόνα 19: Επικεφαλίδα αρχικής σελίδας



Εικόνα 20: Επικεφαλίδα σελίδας τραπεζικών λειτουργιών

Η πρώτη βασική διαδρομή, αυτή της αρχικής σελίδας, αποτελείται από τη φόρμα σύνδεσης του χρήστη, ένα κεντρικό πλαίσιο το οποίο αρχικά απεικονίζει φωτογραφίες σύγχρονων μεθόδων τραπεζικών συναλλαγών, το οποίο όμως αλλάζει το περιεχόμενο του ανάλογα με τις επιλογές του χρήστη (υπενθύμιση κωδικού, αίτηση εγγραφής, χρήσιμες πληροφορίες κ.α.) καθώς και τρία μικρότερα πλαίσια που περιέχουν χρήσιμες πληροφορίες. επικεφαλίδα αλλάζει τα περιεχόμενα της συναρτήσει της ενεργής διαδρομής.

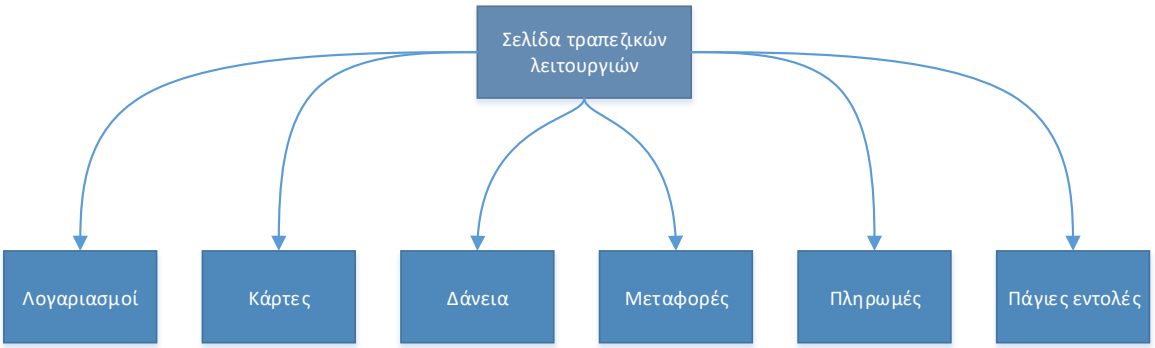


Εικόνα 21: Αρχική σελίδα του ιστοτόπου Web Banking

Η δεύτερη βασική διαδρομή, αυτή της σελίδας τραπεζικών λειτουργιών είναι προσβάσιμη μόνο από τους πιστοποιουμένους χρήστες και περιέχει όλες τις διαθέσιμες τραπεζικές λειτουργίες και πληροφορίες και διαθέτει ο ιστότοπος Web Banking. Επειδή αυτή η διαδρομή περιέχει πολύ πληροφορία και πολλές λειτουργίες, έχει διαχωριστεί σε υποδιαδρομές:

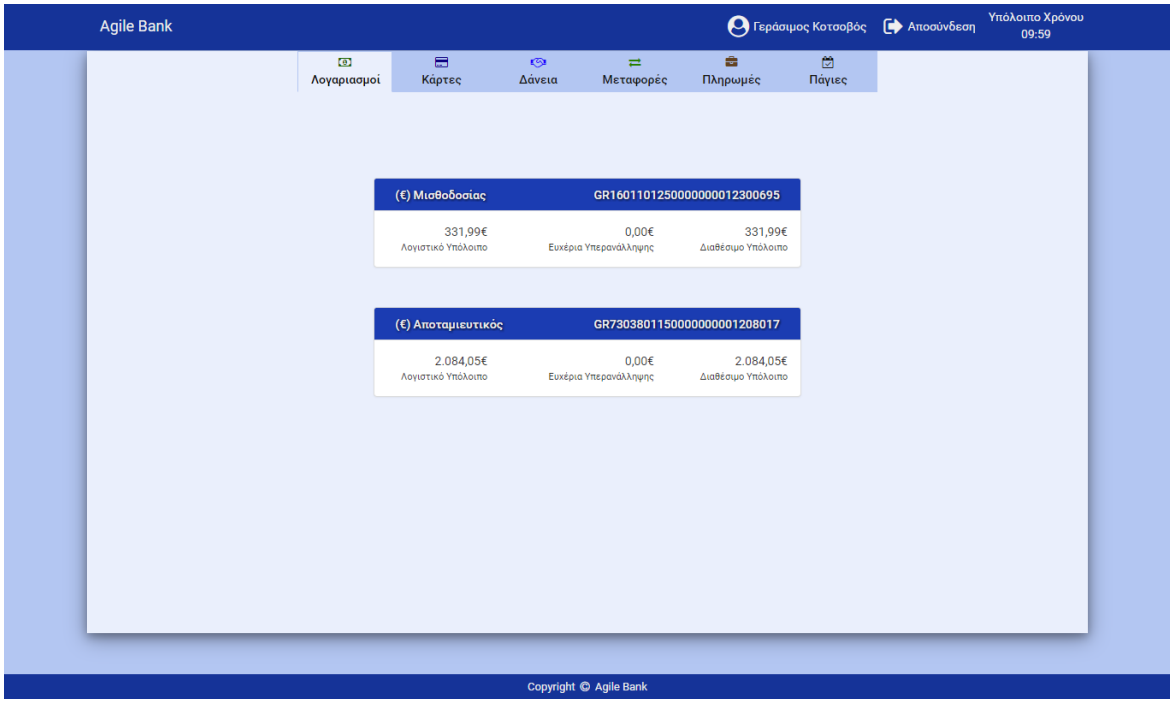
- Διαδρομή λογαριασμών
- Διαδρομή καρτών

- Διαδρομή δανείων
- Διαδρομή μεταφορών
- Διαδρομή πληρωμών
- Διαδρομή πάγιων εντολών



Εικόνα 22: Δομή σελίδας τραπεζικών λειτουργιών

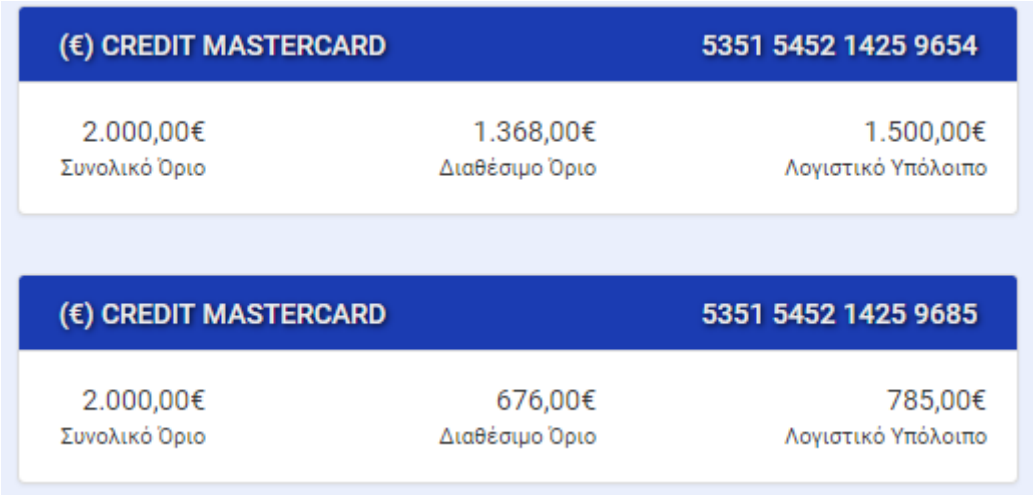
Κάθε υποδιαδρομή είναι υλοποιημένη σε μορφή καρτέλας, όπου ο χρήστης μπορεί να πλοηγηθεί και να εκτελέσει τις ενέργειες που θέλει.



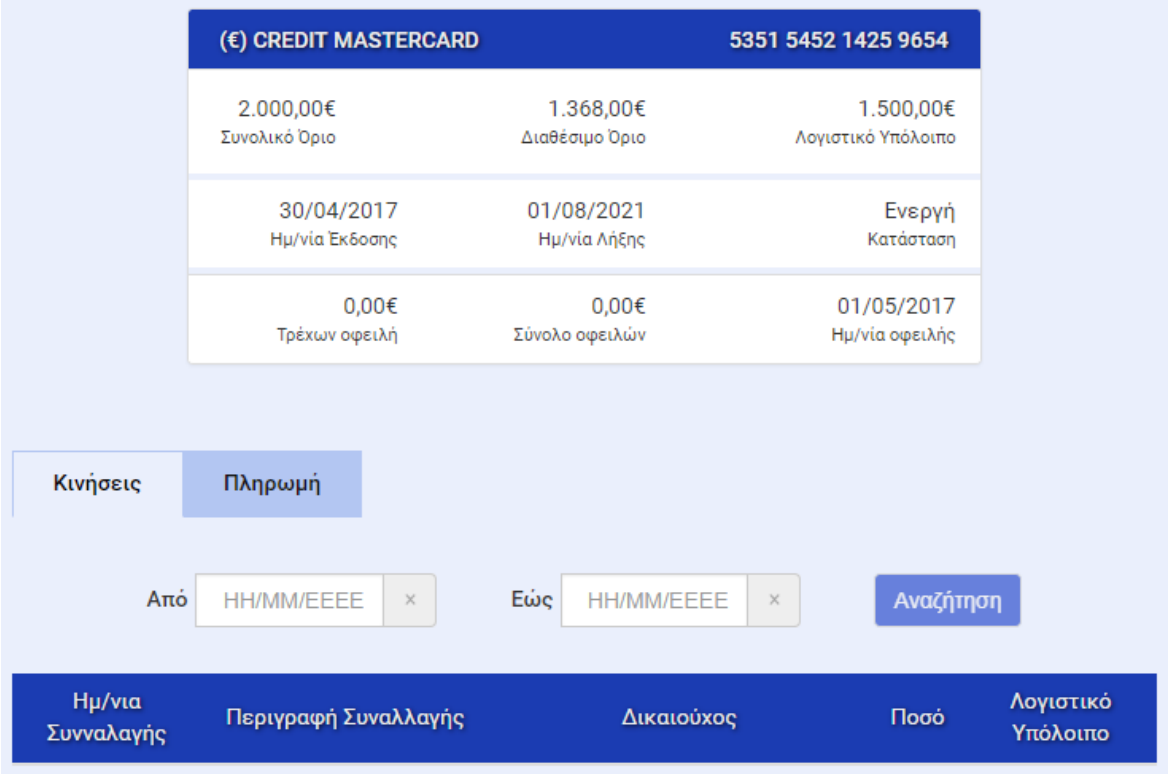
Εικόνα 23: Σελίδα τραπεζικών λειτουργιών

Οι διαδρομές των λογαριασμών, των καρτών και των δάνειων παρουσιάζουν στοιχεία των προαναφερθέντων προϊόντων με τον ίδιο ακριβώς τρόπο: στην αρχή

παρουσιάζουν όλα τα προϊόντα του χρήστη, και αν αυτός πατήσει πάνω σε ένα, τότε θα εμφανιστούν περισσότερες πληροφορίες, καθώς και οι διαθέσιμες ενέργειες για αυτό.



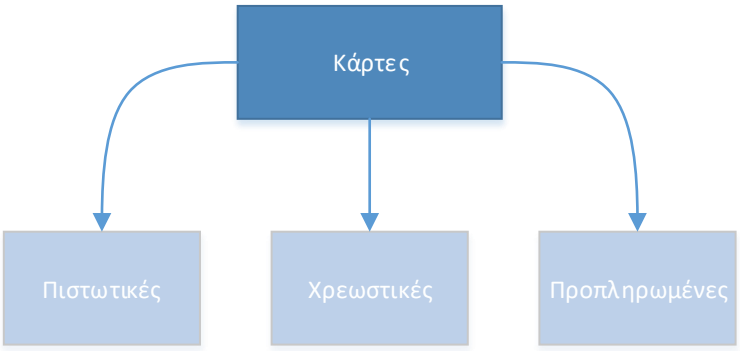
Εικόνα 24: Απεικόνιση βασικών στοιχείων όλων των προϊόντων



Εικόνα 25: Λεπτομερής απεικόνιση στοιχείων προϊόντος και διαθέσιμες ενέργειες

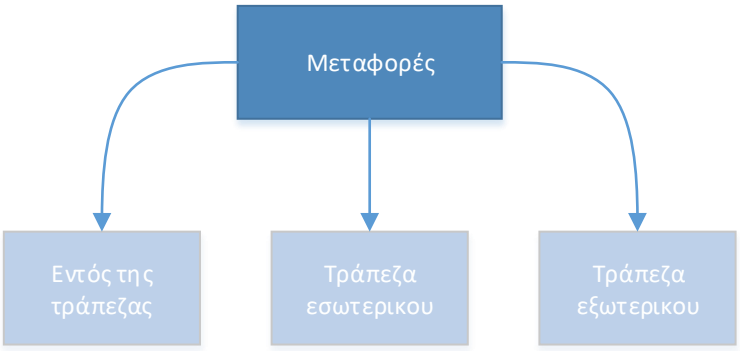
Οι διαδρομές των καρτών, των μεταφορών, των πληρωμών και των πάγιων εντολών είναι πιο σύνθετες. Κάθε μια από αυτές περιέχει δικές της υποδιαδρομές,

είτε για λόγους απεικόνισης είτε για λόγους λειτουργιών. Στη περίπτωση της διαδρομής των καρτών χρειάστηκε να δημιουργηθούν τρεις υποδιαδρομές και για λόγους απεικόνισης αλλά και για λόγους διαφορετικών διαθέσιμων λειτουργιών, μία για κάθε τύπο κάρτας: υποδιαδρομή χρεωστικών, πιστωτικών και προπληρωμένων καρτών.



Εικόνα 26: Δομή διαδρομής καρτών

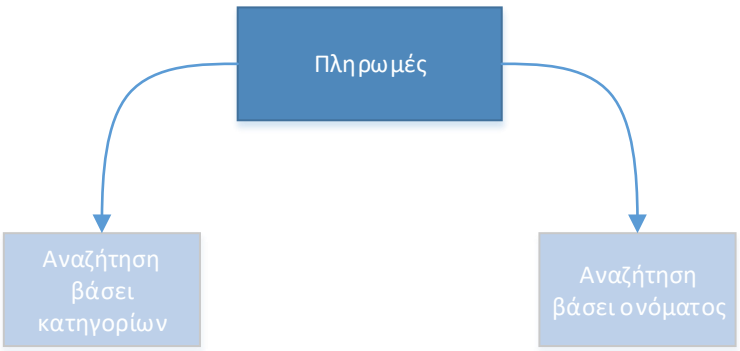
Η διαδρομή μεταφορών έχει διαχωριστεί και αυτή σε τρεις υποδιαδρομές για λόγους διαφορετικής λειτουργικότητας βάσει του τύπου της μεταφοράς χρημάτων: μεταφορές εντός της τράπεζας, μεταφορές σε τράπεζα εσωτερικού και μεταφορές σε τράπεζα εξωτερικού. Σε κάθε μια από αυτές τις περιπτώσεις ο χρήστης καλείτε να συμπληρώσει διαφορετικά στοιχεία, τα οποία θα αναλυθούν στο επόμενο υποκεφάλαιο (3.4).



Εικόνα 27: Δομή διαδρομής μεταφορών

Η διαδρομή πληρωμών έχει διαχωριστεί σε δύο υποδιαδρομές για λόγους διαφορετικής λειτουργικότητας βάσει του τρόπου αναζήτησης της πληρωμής: αναζήτηση βάσει κατηγοριών και αναζήτηση βάσει ονόματος. Σε κάθε μια από

αυτές τις περιπτώσεις ο χρήστης καλείτε να συμπληρώσει ή να επιλέξει διαφορετικά στοιχεία, τα οποία θα αναλυθούν στο επόμενο υποκεφάλαιο (3.4).



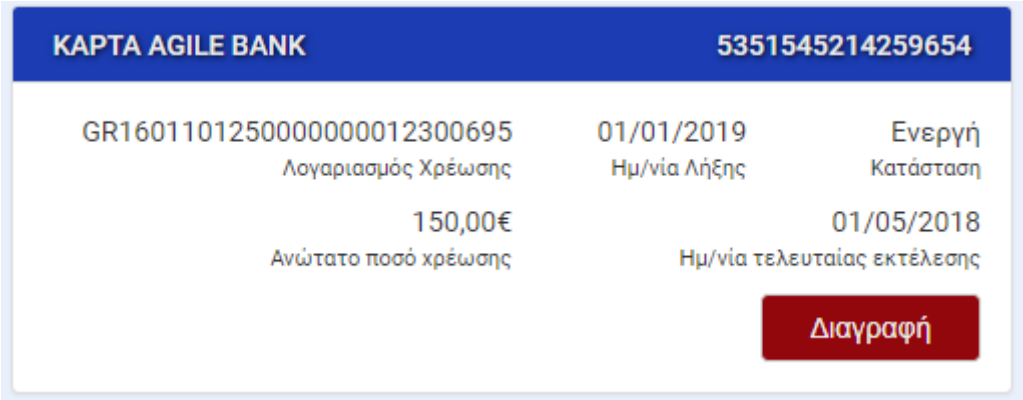
Εικόνα 28: Δομή διαδρομής πληρωμών

Η διαδρομή πάγιων εντολών έχει διαχωριστεί σε δύο υποδιαδρομές και για λόγους απεικόνισης αλλά και για λόγους διαφορετικών διαθέσιμων λειτουργιών, μία για κάθε τύπο πάγιας εντολής: πάγια εντολή μεταφοράς και πάγια εντολή πληρωμής.

Εικόνα 29: Δομή διαδρομής πάγιων εντολών

Εκτός από τις διαφορές στις διαθέσιμες λειτουργίες οι οποίες θα αναλυθούν στο επόμενο υποκεφάλαιο (3.4), υπάρχουν διαφορές και στο τρόπο απεικόνισης, λόγω των διαφορετικών στοιχείων που έχει ο κάθε διαφορετικός τύπος πάγιας εντολής.

Εικόνα 30: Απεικόνιση στοιχείων πάγιας εντολής μεταφοράς



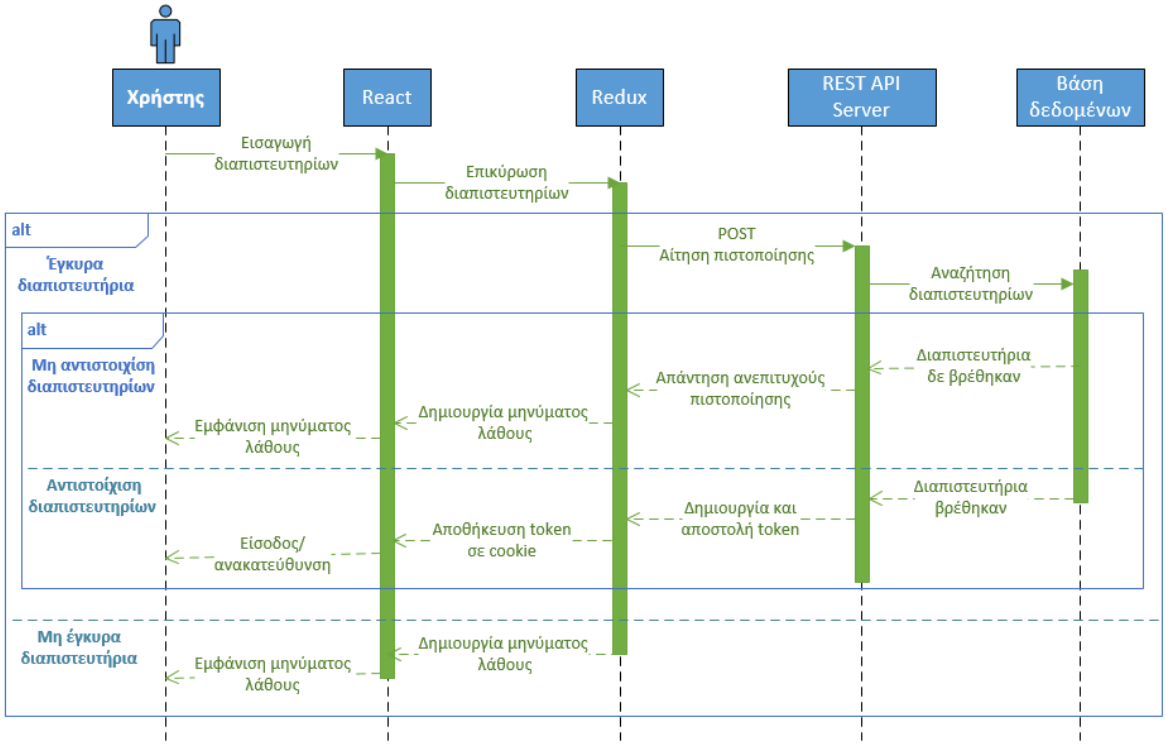
Εικόνα 31: Απεικόνιση στοιχείων πάγιας εντολής πληρωμής

3.4. Λειτουργίες ιστότοπου Web Banking

Οι λειτουργίες του ιστοτόπου χωρίζονται σε διαφορετικές κατηγορίες και διαφέρουν αρκετά μεταξύ τους και ως προς τη λειτουργικότητα άλλα και ως προς τη διαθεσιμότητα προς το χρήστη. Για παράδειγμα οι διαθέσιμες λειτουργίες προς το χρήστη είναι άλλες σύμφωνα με το αν έχει πιστοποιηθεί ή όχι. Ή υπάρχουν περισσότεροι τους ενός τρόποι να πραγματοποιήσει ο χρήστης πληρωμή ενός δανείου του. Μια ακόμη διαφορετική κατηγορία είναι οι λειτουργίες που εκτελούνται χωρίς ο χρήστης να έχει κάποια γνώση για αυτές. Σε αυτή τη περίπτωση αυτές οι λειτουργίες έχουν ως σκοπό την απροβλημάτιστη λειτουργικότητα του ιστότοπου του Web Banking. Παρόλα αυτά, **κάθε λειτουργία** ακολουθεί τη ροή που περιγράφηκε στην ***Εικόνα 7:*** *Διάγραμμα ακολουθίας ενέργειας χρήστη*.

3.4.1. Λειτουργίες μη πιστοποιημένου χρήστη

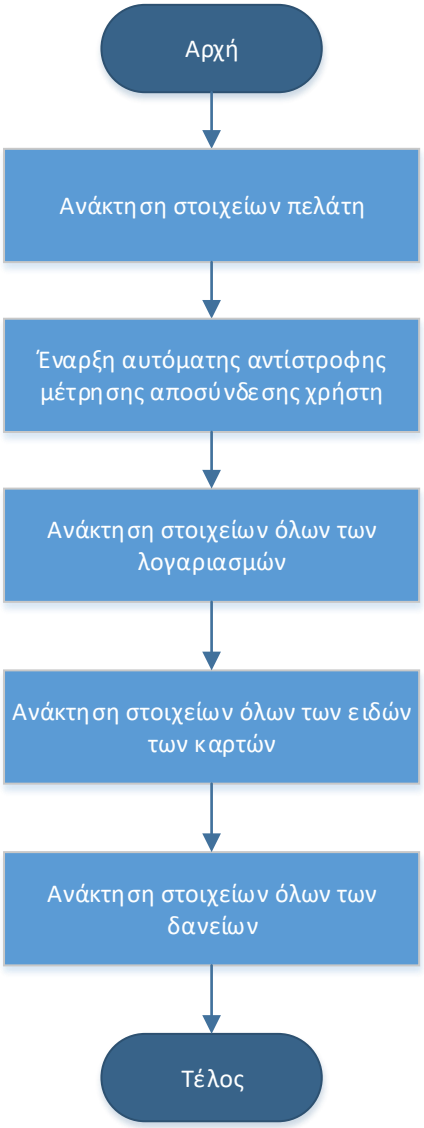
Οι λειτουργίες του ιστοτόπου που είναι διαθέσιμες προς το χρήστη στη περίπτωση που δεν έχει πιστοποιηθεί από το σύστημα είναι λίγες, για την ακρίβεια μόνο μια: αυτή της πιστοποίησης του χρήστη, η οποία είναι διαθέσιμη στην αρχική σελίδα του ιστοτόπου. Ο χρήστης καλείται να συμπληρώσει τα διαπιστευτήρια του (ID Χρήστη και Κωδικό) και να πατήσει το κουμπί εισόδου. Όταν πατηθεί το κουμπί εισόδου πραγματοποιείται μια επικύρωση των διαπιστευτήρια. Στη περίπτωση που επικυρωθούν, εκτελείται η υπεύθυνη ενέργεια του Redux η οποία στέλνει τα στοιχεία του χρήστη με μια HTTP POST αίτηση στο σημείο εισαγωγής του REST API server που είναι υπεύθυνο για την πιστοποίηση του χρήστη. Αν η απάντηση του server είναι θετική τότε στέλνεται μαζί και ένα token πιστοποίησης το οποίο αποθηκεύεται σαν cookie στο πρόγραμμα περιήγησης ιστού, ώστε να μπορεί να χρησιμοποιηθεί για την πιστοποίηση του χρήστη σε οποιαδήποτε άλλη ενέργεια πραγματοποιήσει. Στη συνέχεια ο χρήστης ανακατευθύνεται στην σελίδα με τις λειτουργίες του web banking. Στην περίπτωση που η απάντηση του server είναι αρνητική ή δεν περάσει η επικύρωση των πεδίων, τότε εμφανίζεται αντίστοιχο μήνυμα λάθους.



Εικόνα 32: Διάγραμμα ακολουθίας ενέργειας πιστοποίησης χρήστη

3.4.2. Λειτουργίες παρασκηνίου

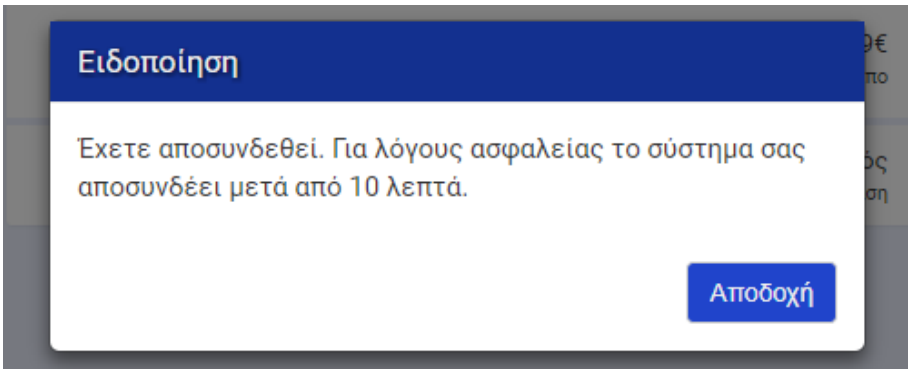
Οι λειτουργίες παρασκηνίου εκτελούνται εν ερήμην του χρήστη και έχουν ως μοναδικό σκοπό την απροβλημάτιστη λειτουργία του ιστότοπου Web Banking. Οι πιο συνηθισμένες είναι αυτές που φέρνουν ή ανανεώνουν τα στοιχεία των προϊόντων του χρήστη καθώς αυτός περιηγείται στον ιστότοπο. Αναλυτικότερα, κάθε φορά που ο χρήστης εισέλθει στη σελίδα τραπεζικών λειτουργειών τότε εκτελούνται μία σειρά από ενέργειες πριν ακόμη φορτώσει η ίδια η σελίδα των τραπεζικών λειτουργιών: ανάκτηση όλων των βασικών στοιχείων του χρήστη, και έναρξη αντίστροφης μέτρησης αποσύνδεσης του για λόγους ασφαλείας.



Εικόνα 33: Λειτουργίες παρασκηνίου κατά την εισαγωγή του χρήστη

Επίσης, κάθε φορά που ο χρήστης επισκέπτεται μια τις διαθέσιμες καρτέλες που έχει διαχωρισθεί η σελίδα των τραπεζικών λειτουργιών τότε ανακτώνται και ανανεώνονται τα δεδομένα της εκάστοτε καρτέλας. Για παράδειγμα, κάθε φορά που ο χρήστης επισκέπτεται τη καρτέλα των λογαριασμών, γίνεται η κατάλληλη διαδικασία για να ανακτηθούν τα στοιχεία όλων των λογαριασμών του, έτσι ώστε ο χρήστης να βλέπει πάντα την τελευταία εικόνα τους.

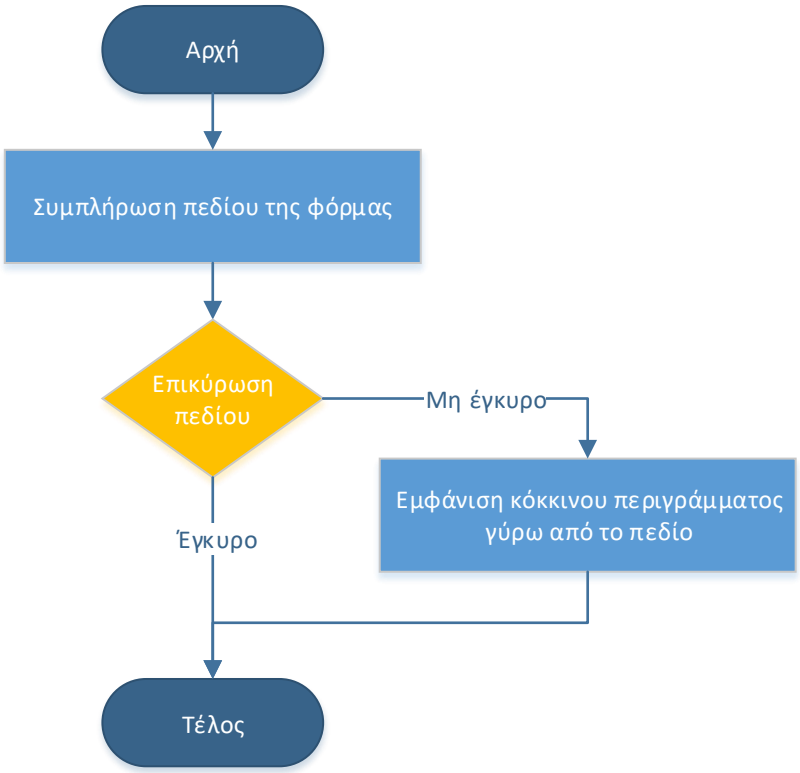
Η τελευταία ενέργεια που εκτελείται στο παρασκήνιο αυτόματα, είναι αυτή της αυτόματης αποσύνδεσης του χρήστη μετά τη πάροδο 10 λεπτών, δηλαδή όταν λήξει το token πιστοποίησης. Όταν περάσουν αυτά τα 10 λεπτά, τότε εμφανίζεται το παρακάτω μήνυμα ειδοποίησης, και μετά ο χρήστης αποσυνδέεται.



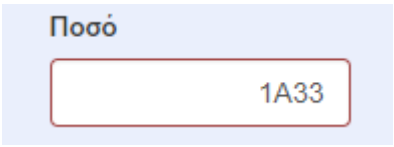
Εικόνα 34: Ειδοποίηση αυτόματης αποσύνδεσης

3.4.3. Βασικές και επαναχρησιμοποιούμενες λειτουργίες

Οι περισσότερες διαθέσιμες λειτουργίες προς το χρήστη συνήθως περιλαμβάνουν τη συμπλήρωση κάποιων πεδίων μιας φόρμας, πριν αυτή μπορεί να εκτελεστεί. Κάθε φορά που ο χρήστης συμπληρώνει ένα τέτοιο πεδίο, είναι αναγκαίο να γίνεται μια επικύρωση της τιμής του πεδίου, έτσι ώστε κατά την εκτέλεση της λειτουργίας που θέλει ο χρήστης να μη δημιουργηθούν προβλήματα. Στη περίπτωση που η τιμή του πεδίο δεν είναι έγκυρη, τότε εμφανίζεται ένα κόκκινο περίγραμμα γύρω από το πεδίο.

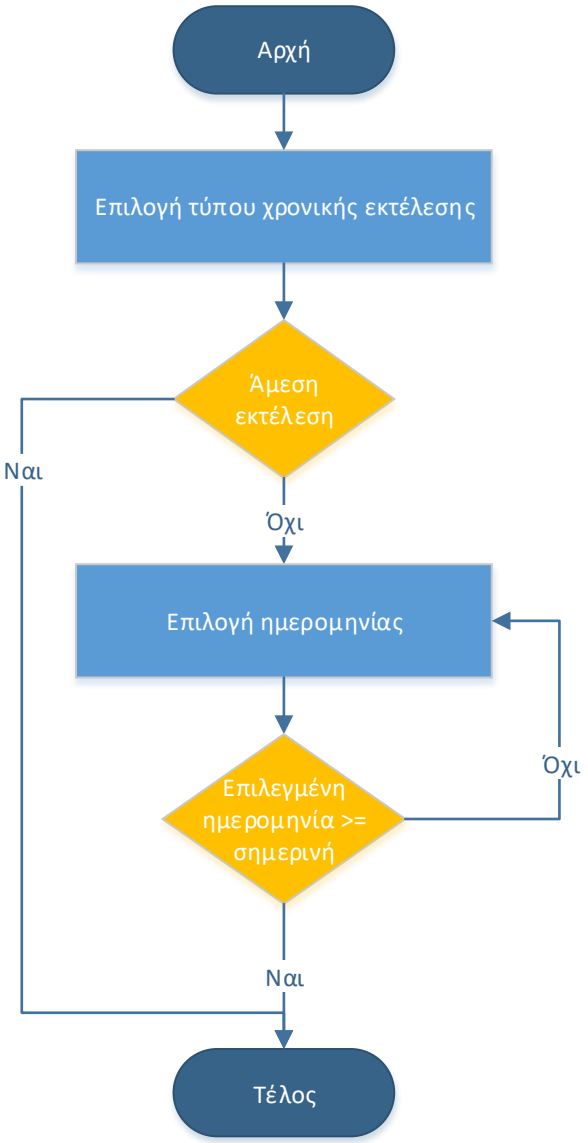


Εικόνα 35: Επικύρωση πεδίου εισαγωγής τιμής



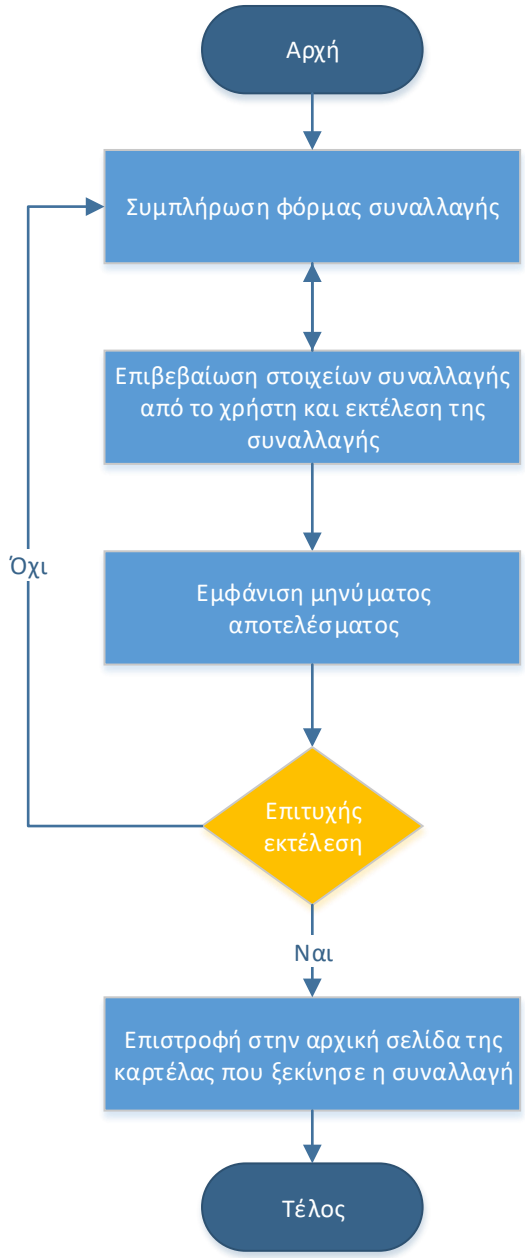
Εικόνα 36: Παράδειγμα μη επικυρωμένου πεδίου

Επίσης, όταν υπάρχει διαθέσιμη επιλογή ημερομηνίας εκτέλεσης μιας ενέργειας, για παράδειγμα η ημερομηνία εκτέλεσης μιας μεταφοράς χρημάτων, τότε υπάρχουν δύο τρόποι χρονικής εκτέλεσης: η άμεση εκτέλεση και η επιλογή της ημερομηνίας εκτέλεσης από το χρήστη. Στη περίπτωση που ο χρήστης επιλέξει ημερομηνία, τότε αυτή πρέπει να είναι έγκυρη.



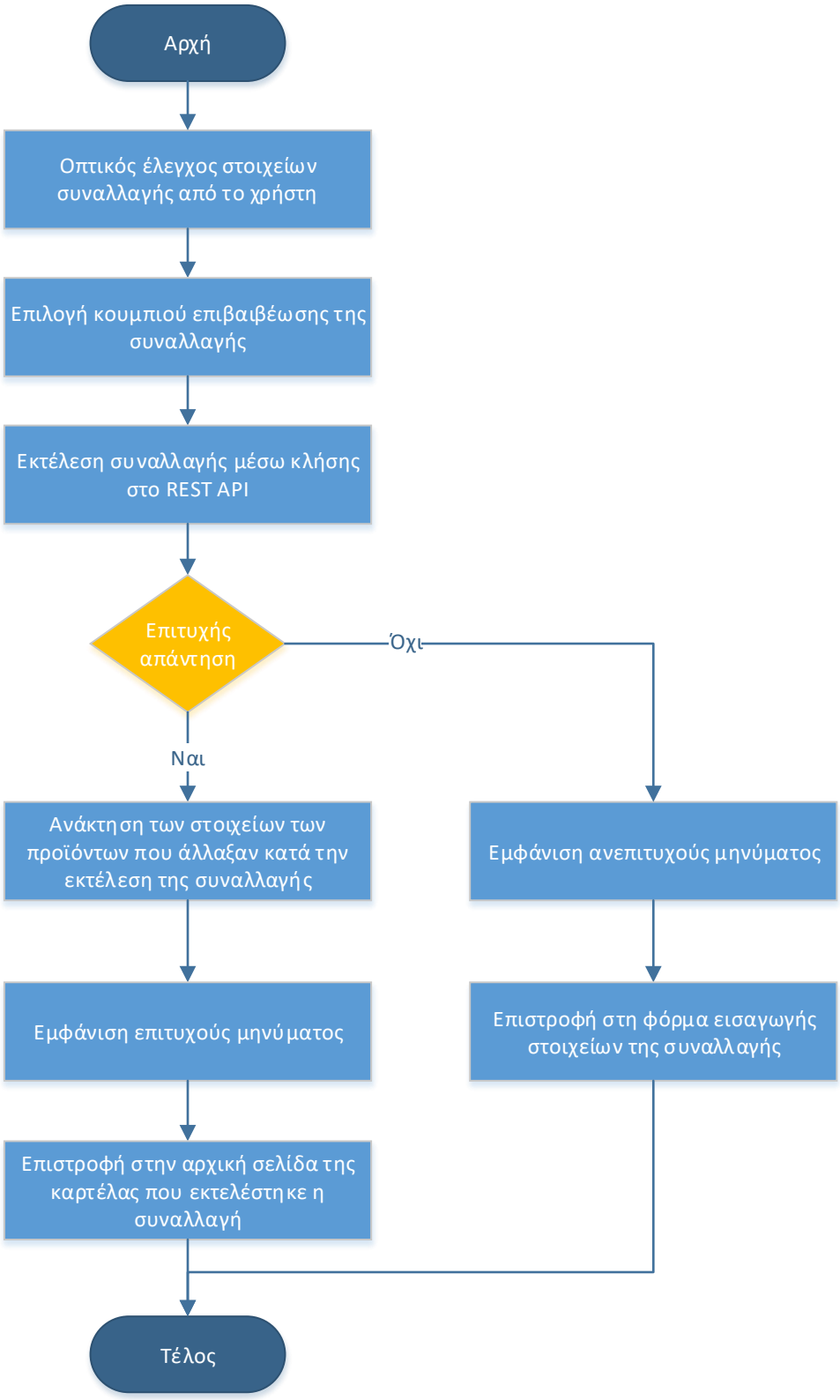
Εικόνα 37: Επιλογή χρονικής εκτέλεσης ενέργειας

Κάθε συναλλαγή που θέλει να εκτελέσει ο χρήστης ακολουθεί την ίδια ροή εκτέλεσης: συμπλήρωση των στοιχείων της συναλλαγής, επιβεβαίωση των στοιχείων που δόθηκαν από το χρήστη, εκτέλεση της συναλλαγής, εμφάνιση αποτελέσματος και επιστροφή στην σελίδα που ξεκίνησε η συναλλαγή. Στη περίπτωση που ο χρήστης δεν είναι ευχαριστημένος με τα στοιχείων που συμπλήρωσε, μπορεί να επιστρέφει στη φόρμα συναλλαγής και να αλλάξει ότι θέλει, πριν επιβεβαιώσει τελικά τα στοιχεία, και έτσι συνεχίσει η παραπάνω ροή.



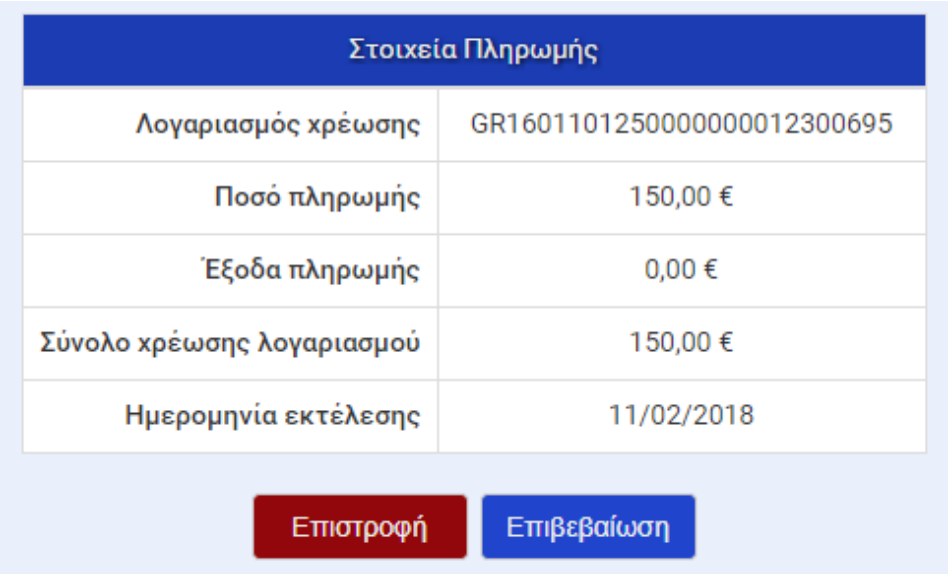
Εικόνα 38: Ροή εκτέλεσης συναλλαγής

Το βήμα όπου τα στοιχεία της συναλλαγής αξιολογούνται από το χρήστη και τα βήματα όπου εκτελείται η συναλλαγή αναλύονται περεταίρω από το παρακάτω διάγραμμα.

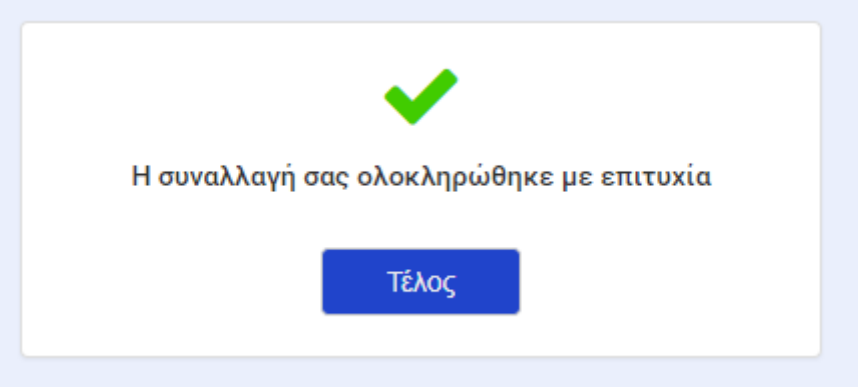


Εικόνα 39: Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας

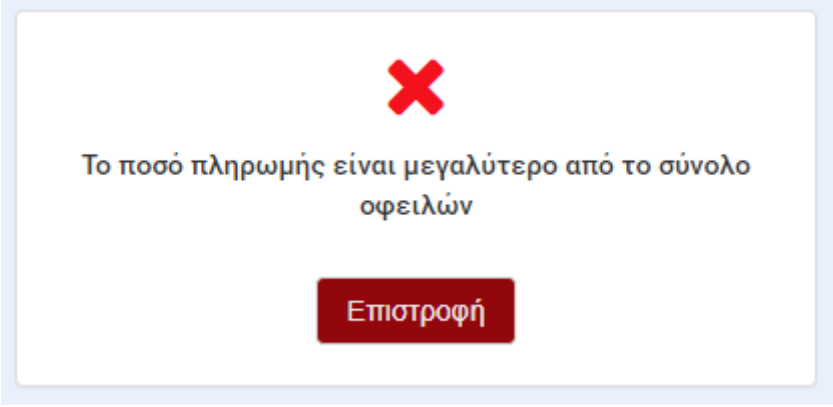
Ακολουθούν μερικά παραδείγματα απεικόνισης των παραπάνω βημάτων.



Εικόνα 40: Παράδειγμα πίνακα επιβεβαίωσης στοιχείων συναλλαγής

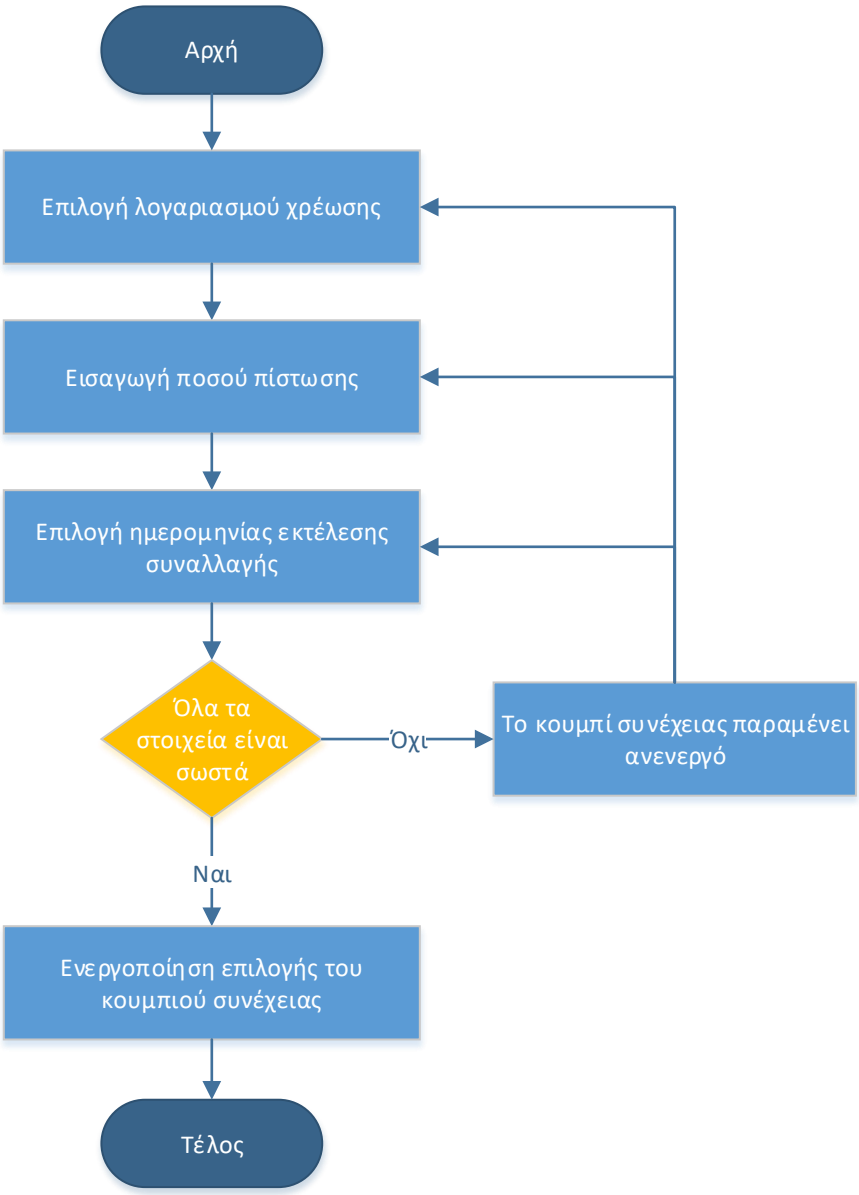


Εικόνα 41: Παράδειγμα μηνύματος επιτυχούς εκτέλεσης συναλλαγής

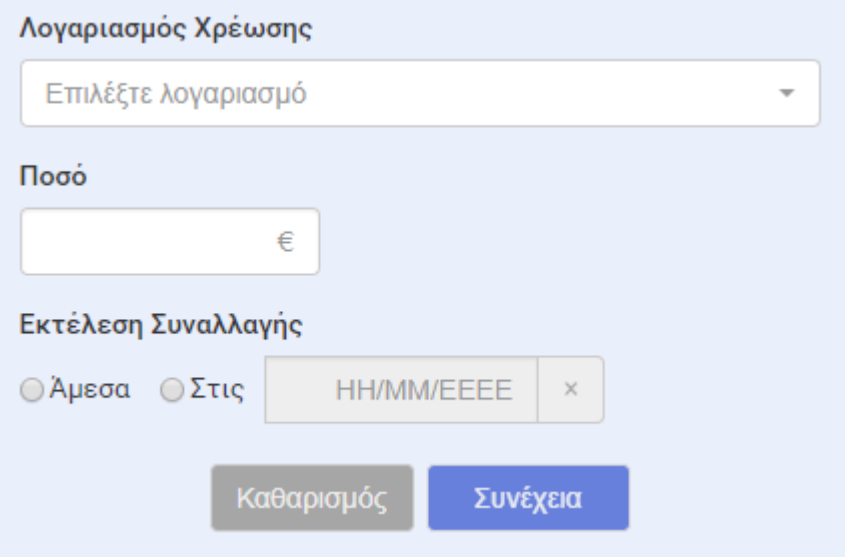


Εικόνα 42: Παράδειγμα μηνύματος ανεπιτυχούς εκτέλεσης συναλλαγής

Μια φόρμα συναλλαγής η οποία έχει επαναχρησιμοποιηθεί αρκετά είναι αυτή όπου ένα ποσό πρέπει να πιστωθεί σε κάποιο προϊόν της τράπεζας, είτε αυτό είναι λόγο πληρωμής, είτε λόγω φόρτισης (θα αναφερθούν ξεχωριστά στα επόμενα κεφάλαια). Σε αυτή τη συναλλαγή ο χρήστης καλείται να επιλέξει ένα λογαριασμό χρέωσης από οποιοδήποτε προϊόν του, να εισάγει το ποσό πίστωσης, να επιλέξει την ημερομηνία εκτέλεσης και να συνεχίσει την εκτέλεση σύμφωνα με τα βήματα που έχουν περιγραφεί προηγουμένως.

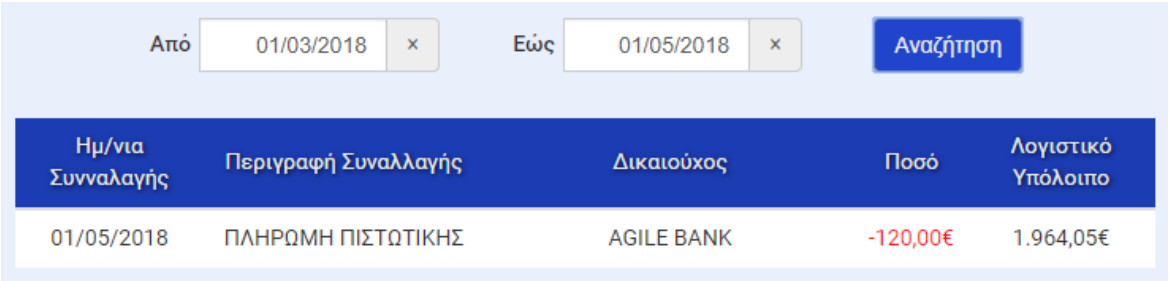


Εικόνα 43: Ροή συμπλήρωσης βασικής φόρμα συναλλαγής πίστωσης προϊόντος της τράπεζας

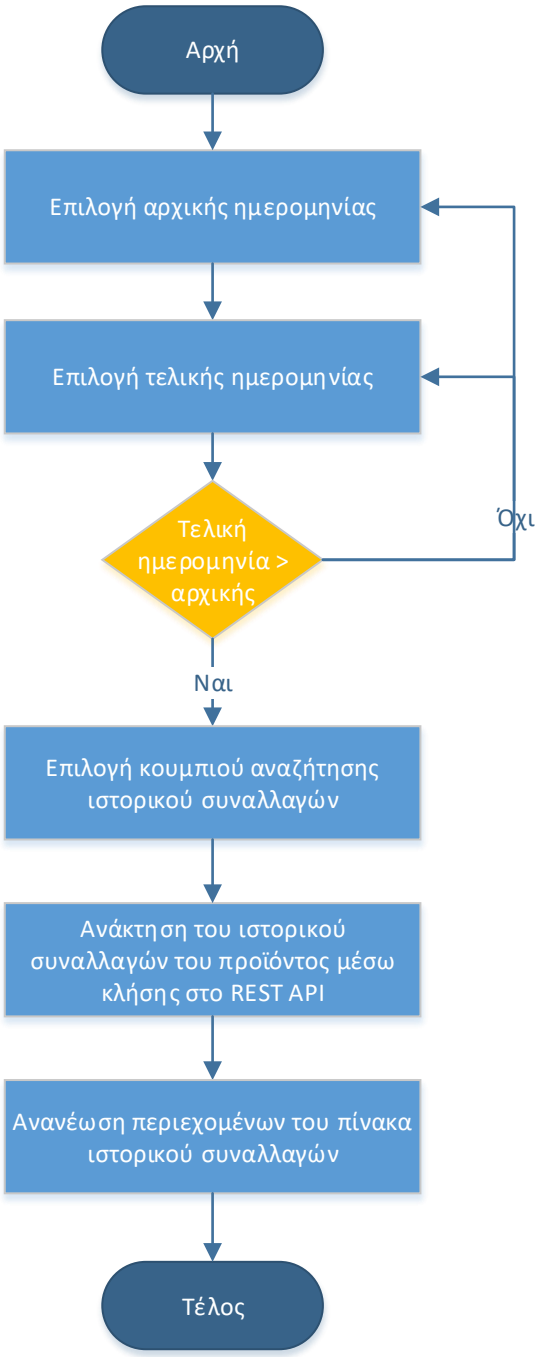


Εικόνα 44: Βασική φόρμα συναλλαγής πίστωσης προϊόντος της τράπεζας

Η τελευταία επαναχρησιμοποιήσιμη λειτουργία είναι αυτή της ανάκτησης του ιστορικού συναλλαγών ενός προϊόντος βάση περιόδου, στην οποία ο χρήστης καλείται να επιλέξει την αρχική ημερομηνία και την τελική ημερομηνία περιόδου. Αυτή η λειτουργία είναι διαθέσιμη σε όλα τα προϊόντα τα οποία μπορούν να πιστώσουν χρήματα σε κάποιο άλλο λογαριασμό ή προϊόν (θα αναφερθούν αναλυτικά στα επόμενα κεφάλαια).



Εικόνα 45: Ιστορικό κινήσεων μετά από επιλογή εύρους ημερομηνιών



Εικόνα 46: Ροή επιλογής εύρους ημερομηνιών ιστορικού συναλλαγών

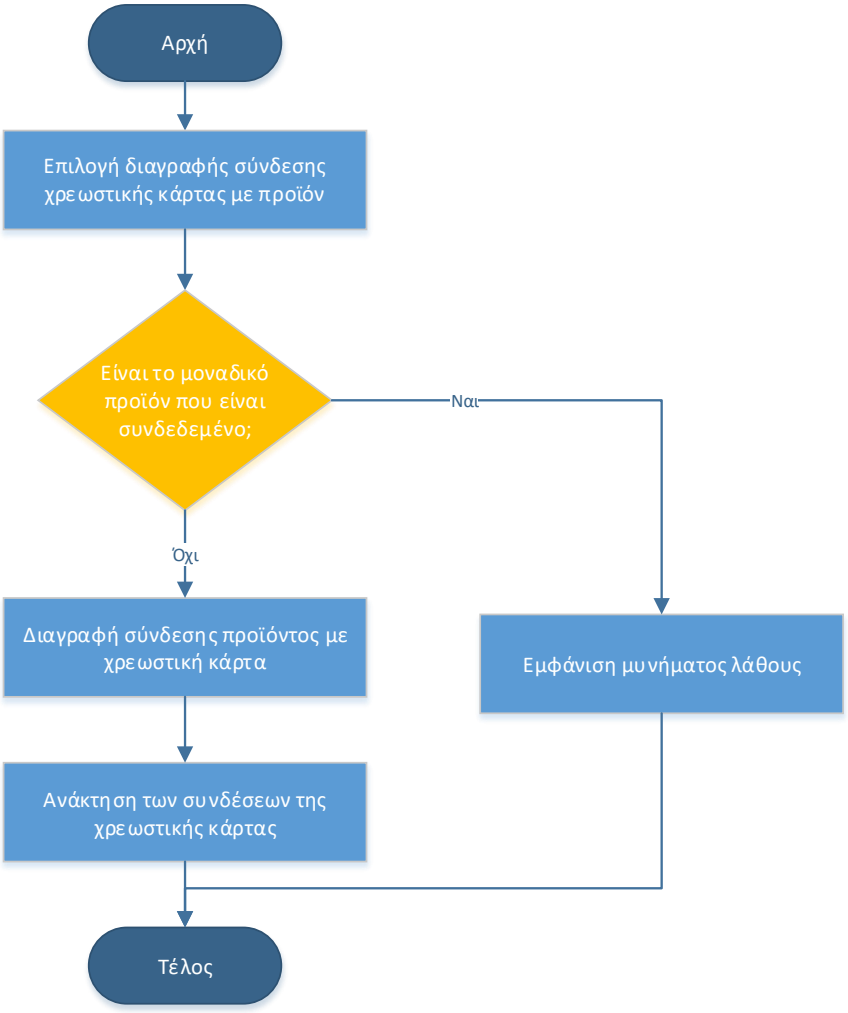
3.4.4. Λειτουργίες λογαριασμών

Η μοναδική διαθέσιμη λειτουργία που έχει ένας λογαριασμός είναι αυτή της αναζήτησης ιστορικού συναλλαγών που περιγράφηκε στο «*Ροή επιλογής εύρους ημερομηνιών ιστορικού συναλλαγών*», στο προηγούμενο κεφάλαιο.

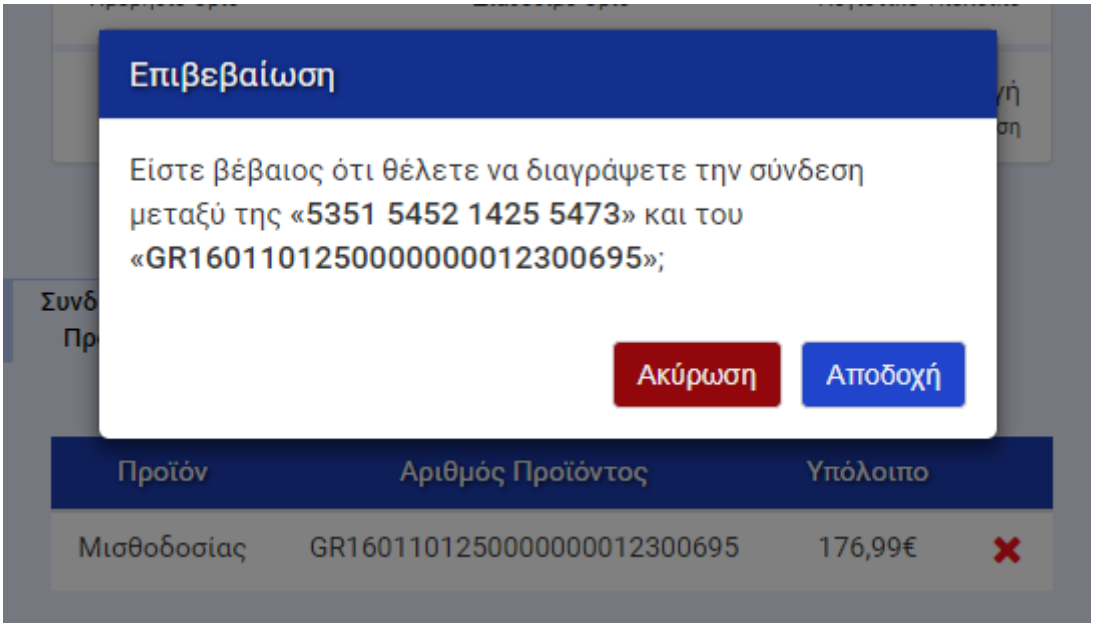
3.4.5. Λειτουργίες καρτών

Οι λειτουργίες που είναι διαθέσιμες για τα κάρτες χωρίζονται σε δύο κατηγορίες, τις κοινές λειτουργίες, δηλαδή τις λειτουργίες που έχουν όλες οι κάρτες ανεξαρτήτως τύπου, και τις λειτουργίες που είναι μοναδικές σε κάθε τύπο κάρτας. Η λειτουργία που είναι διαθέσιμη σε όλους τους τύπους των καρτών είναι η λειτουργία αναζήτησης ιστορικού συναλλαγών που περιγράφηκε στο «*Ροή επιλογής εύρους ημερομηνιών ιστορικού συναλλαγών*».

Οι χρεωστικές κάρτες, εκτός της προαναφερθείσας λειτουργίας, έχουν διαθέσιμη τη λειτουργία διαγραφής υπάρχουσας σύνδεσης με προϊόν, αρκεί αυτή η σύνδεση να μην είναι η μοναδική διαθέσιμη. Αναλυτικότερα, κάθε χρεωστική κάρτα έχει συνδεδεμένο τουλάχιστον ένα προϊόν που μπορεί να το χρησιμοποιήσει. Ο χρήστης έχει την επιλογή να διαγράψει αυτή τη σύνδεση, μόνο όταν αυτή η σύνδεση δεν είναι η τελευταία που έχει απομείνει.



Εικόνα 47: Αποσύνδεση προϊόντος από χρεωστική κάρτα



Εικόνα 48: Παράδειγμα μηνύματος επιβεβαίωσης αποσύνδεσης προϊόντος

Οι πιστωτικές κάρτες έχουν διαθέσιμη τη λειτουργία της πληρωμής. Η διαδικασία της πληρωμής πιστωτικής κάρτας όταν επιλέγεται να γίνει από τη διαδρομή (route) των πιστωτικών καρτών, υλοποιείται με το τρόπο που περιγράφηκε στη «*Ροή συμπλήρωσης βασικής φόρμα συναλλαγής πίστωσης προϊόντος της τράπεζας*» και ολοκληρώνεται με τη διαδικασία που περιγράφηκε στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*». Οι επικυρώσεις που χρειάζονται να γίνουν είναι: να είναι σωστός ο λογαριασμός χρέωσης, το ποσό πληρωμής να είναι μικρότερο ή ίσο του διαθέσιμου υπολοίπου του λογαριασμού και του συνόλου της οφειλής, και να είναι σωστή η ημερομηνία εκτέλεσης της συναλλαγής.

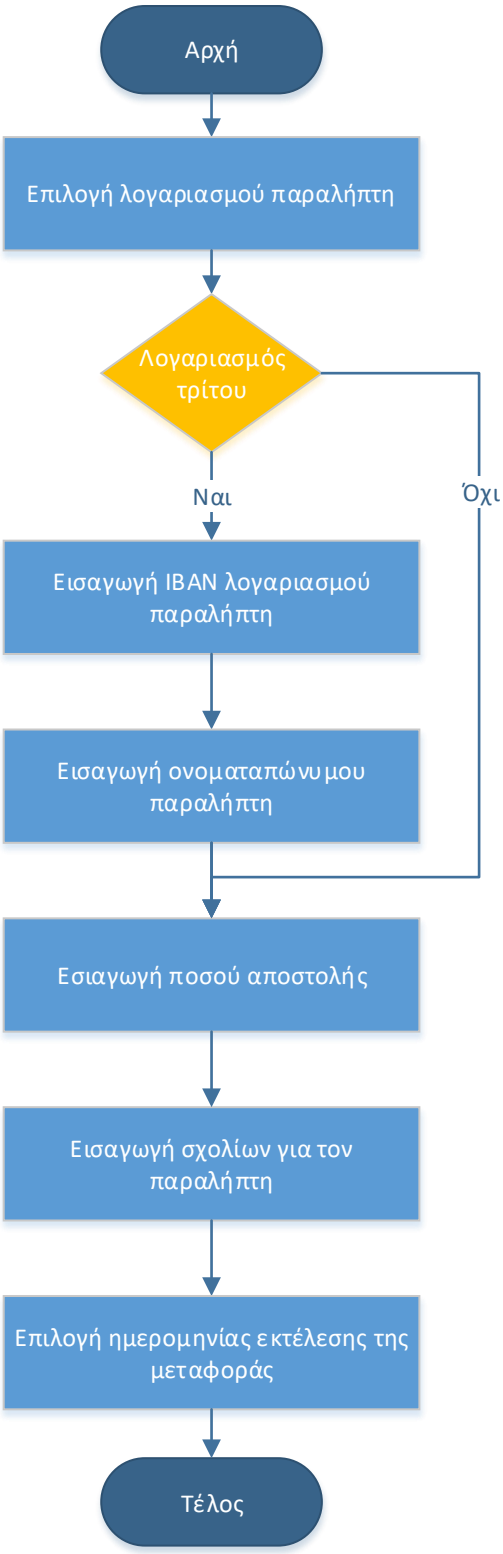
Οι πιστωτικές κάρτες έχουν διαθέσιμη τη λειτουργία της φόρτισης. Η διαδικασία της φόρτισης προπληρωμένης υλοποιείται με το τρόπο που περιγράφηκε στη «*Ροή συμπλήρωσης βασικής φόρμα συναλλαγής πίστωσης προϊόντος της τράπεζας*» και ολοκληρώνεται με τη διαδικασία που περιγράφηκε στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*». Οι επικυρώσεις που χρειάζονται να γίνουν είναι: να είναι σωστός ο λογαριασμός χρέωσης, το ποσό φόρτισης να είναι μικρότερο ή ίσο του διαθέσιμου υπολοίπου του λογαριασμού, και να είναι σωστή η ημερομηνία εκτέλεσης της συναλλαγής.

3.4.6. Λειτουργίες δανείων

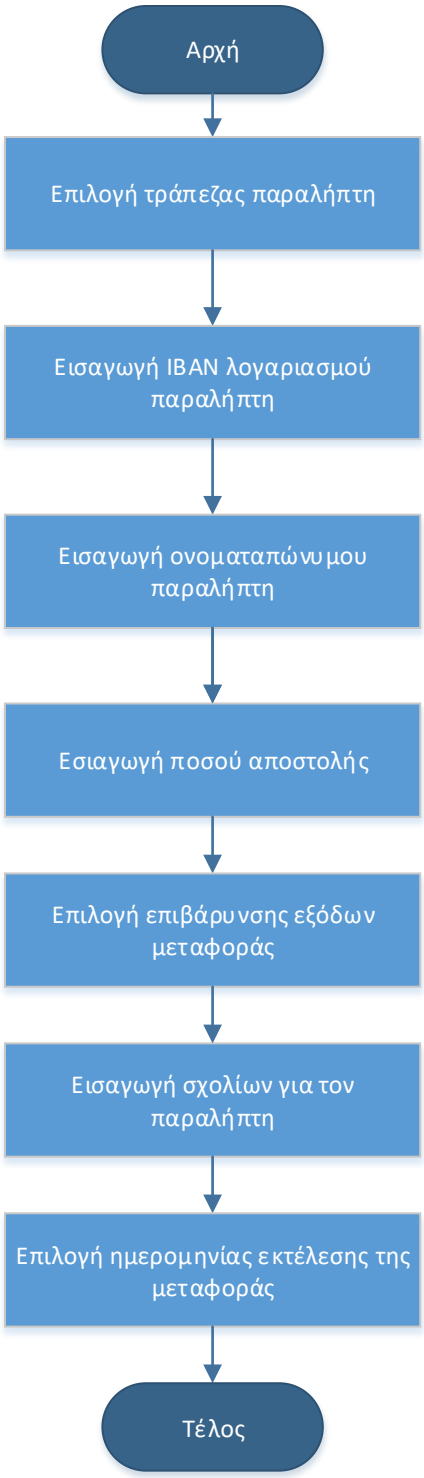
Τα δάνεια έχουν διαθέσιμες τις λειτουργίες της αναζήτησης ιστορικού συναλλαγών που περιγράφηκε στο «*Ροή επιλογής εύρους ημερομηνιών ιστορικού συναλλαγών*», και τη λειτουργία της πληρωμής. Η διαδικασία της πληρωμής πιστωτικής κάρτας όταν επιλέγεται να γίνει από τη διαδρομή (route) των δανείων, υλοποιείται με το τρόπο που περιγράφηκε στη «*Ροή συμπλήρωσης βασικής φόρμα συναλλαγής πίστωσης προϊόντος της τράπεζας*» και ολοκληρώνεται με τη διαδικασία που περιγράφηκε στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*». Οι επικυρώσεις που χρειάζονται να γίνουν είναι: να είναι σωστός ο λογαριασμός χρέωσης, το ποσό πληρωμής να είναι μικρότερο ή ίσο του διαθέσιμου υπολοίπου του λογαριασμού και του συνόλου της οφειλής, και να είναι σωστή η ημερομηνία εκτέλεσης της συναλλαγής.

3.4.7. Λειτουργίες μεταφορών

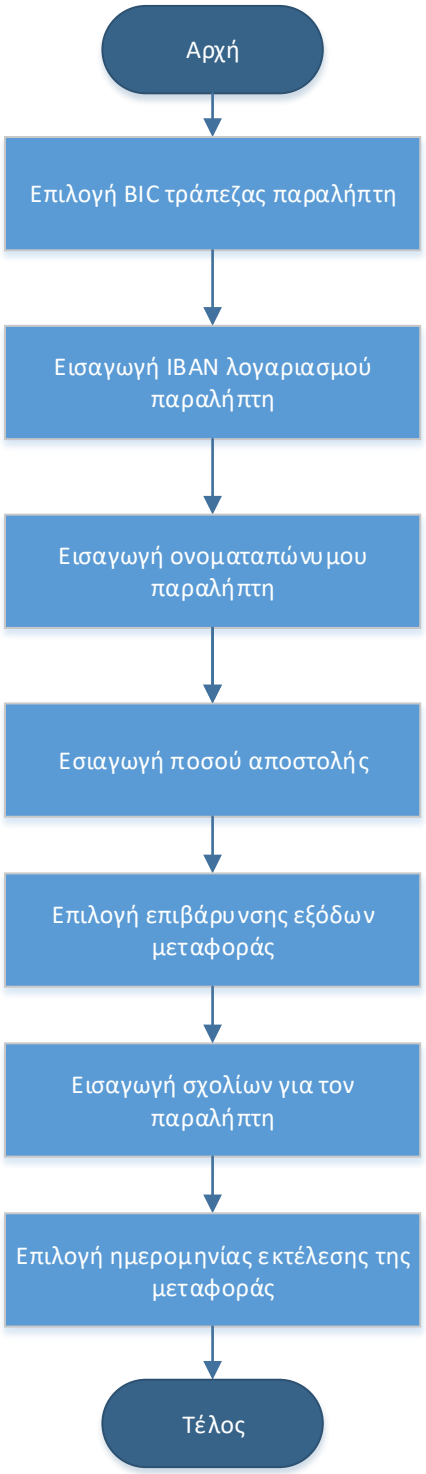
Η διαθέσιμη λειτουργία στη διαδρομή (route) των μεταφορών είναι μία: αυτή της μεταφοράς. Παρόλα αυτά η λειτουργία μεταφοράς έχει διαχωριστεί σε τρεις διαφορετικές υπολειτουργίες, βάσει του τύπου της μεταφοράς χρημάτων: μεταφορές εντός της τράπεζας, μεταφορές σε τράπεζα εσωτερικού και μεταφορές σε τράπεζα εξωτερικού. Η διαδικασία μεταφοράς χρημάτων αρχικά είναι κοινή για κάθε τύπο μεταφοράς, και ξεκινάει με την επιλογή του λογαριασμού χρέωσης. Τα απόμενα βήματα διαφέρουν από λίγο έως πολύ ανάμεσα στις διαφορετικές κατηγορίες μεταφοράς χρημάτων. Η ροή για κάθε διαδικασία αποτυπώνεται στα διαγράμματα ροής που ακολουθούν.



Εικόνα 49: Ροή συμπλήρωσης φόρμας μεταφοράς χρημάτων εντός της τράπεζας



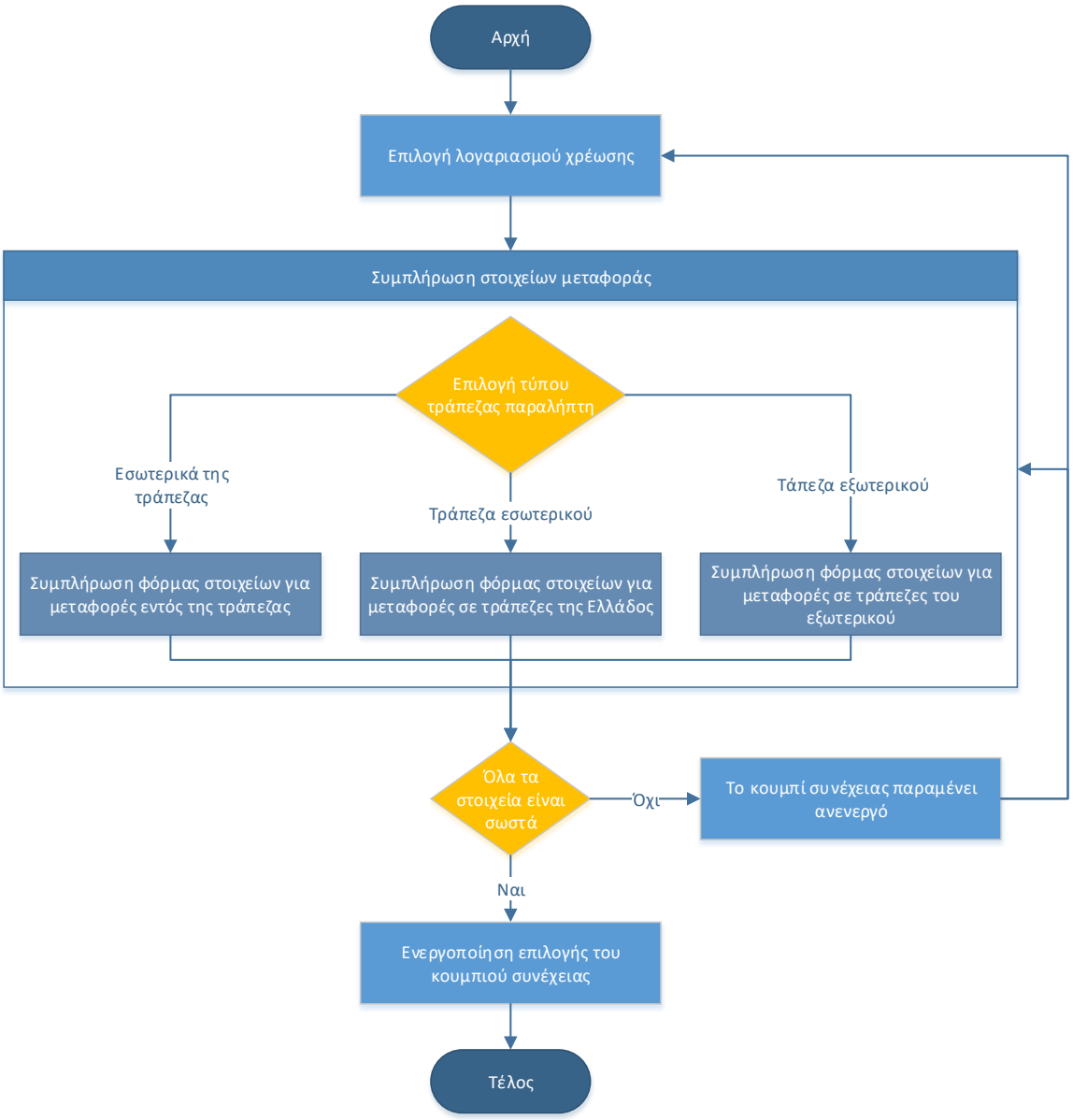
Εικόνα 50: Ροή συμπλήρωσης φόρμας μεταφοράς χτημάτων σε τράπεζα εσωτερικού



Εικόνα 51: Ροή συμπλήρωσης φόρμας μεταφοράς χρημάτων σε τράπεζα του εξωτερικού

Στη συνέχεια, και αφού συμπληρωθούν τα εκάστοτε πεδία κάθε τύπου μεταφοράς χρημάτων, πραγματοποιείται επικύρωση των στοιχείων, και σε περίπτωση που τα στοιχεία είναι σωστά, ο χρήστης μπορεί να συνεχίσει την

εκτέλεση της συναλλαγής, με το τρόπο που περιγράφεται στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*».

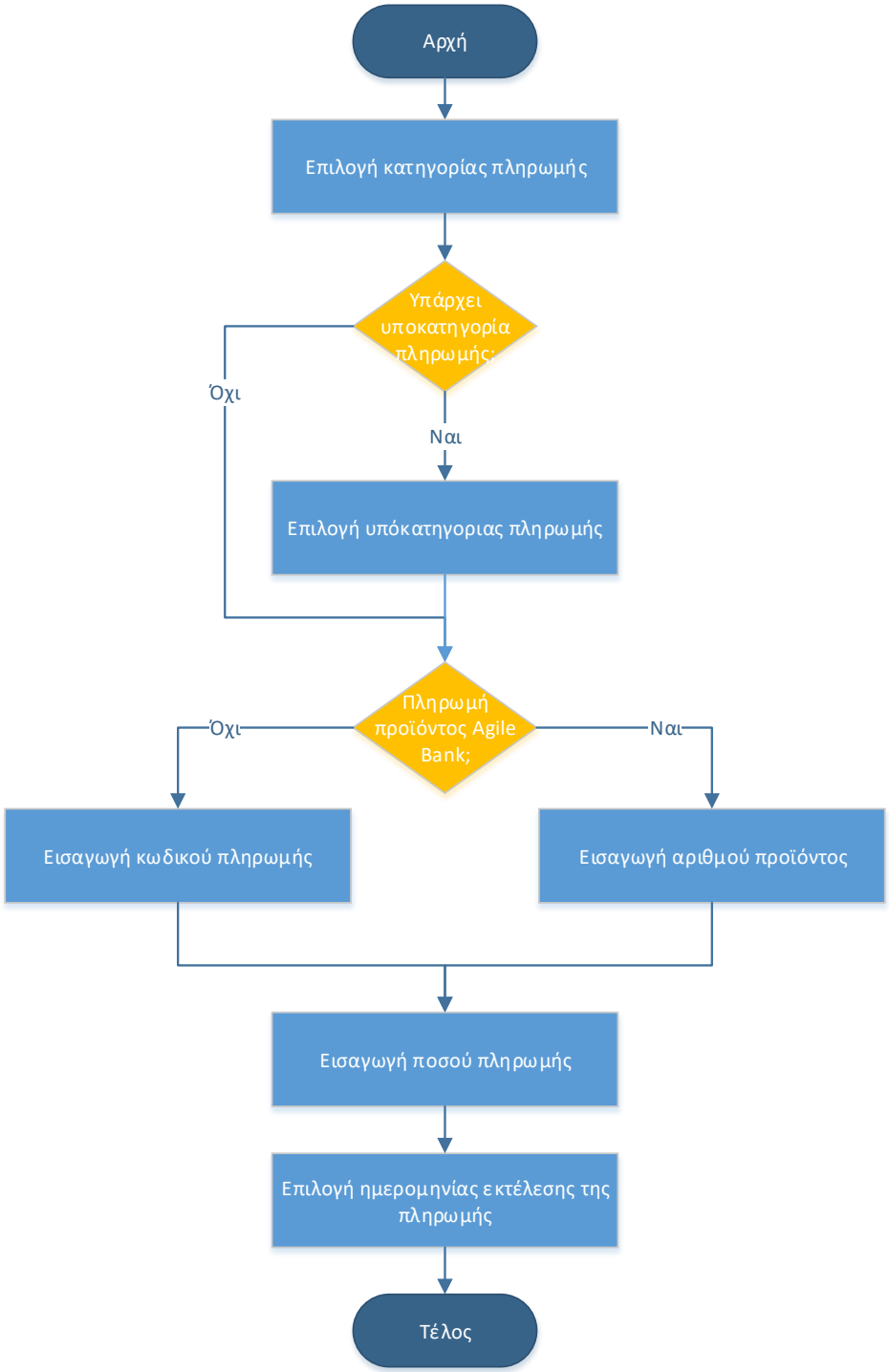


Εικόνα 52: Πλήρη ροή διαδικασίας συμπλήρωσης φόρμας μεταφοράς χρημάτων

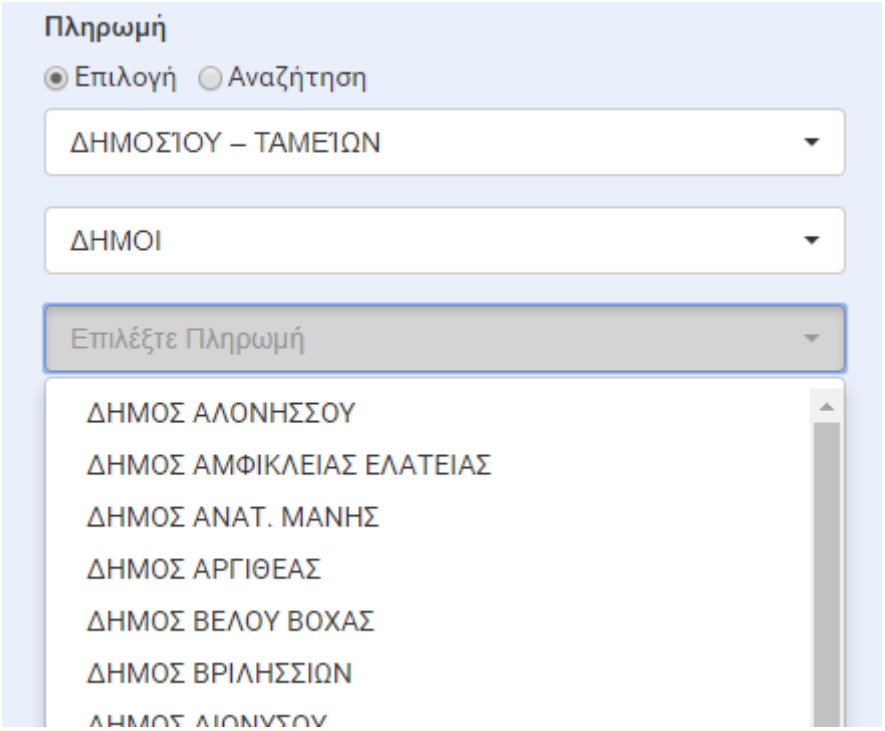
3.4.8. Λειτουργίες πληρωμών

Η διαθέσιμη λειτουργία στη διαδρομή (route) των πληρωμών είναι μία: αυτή της πληρωμής. Παρόλα αυτά η λειτουργία της πληρωμής έχει διαχωριστεί σε δύο διαφορετικές υπολειτουργίες, βάσει του τρόπου αναζήτησης της πληρωμής: αναζήτηση βάσει κατηγοριών και αναζήτηση βάσει ονόματος. Η διαδικασία πληρωμής αρχικά είναι κοινή και ξεκινάει με την επιλογή του λογαριασμού

χρέωσης, ενώ στη συνέχεια διαφοροποιείται. Στη περίπτωση της αναζήτησης βάσει κατηγοριών, ο χρήστης καλείται να επιλέξει ανάμεσα σε όλες τις διαθέσιμες κατηγορίες, και υποκατηγορίες (αν υπάρχουν) μέσω αναπτυσσόμενων λιστών, μέχρι να φτάσει στο σημείο να επιλέξει τη πληρωμή που θέλει να πραγματοποιήσει. Αυτή η διαδικασία ακολουθείται όταν ο χρήστης δεν είναι σίγουρος για την ονομασία της πληρωμής.

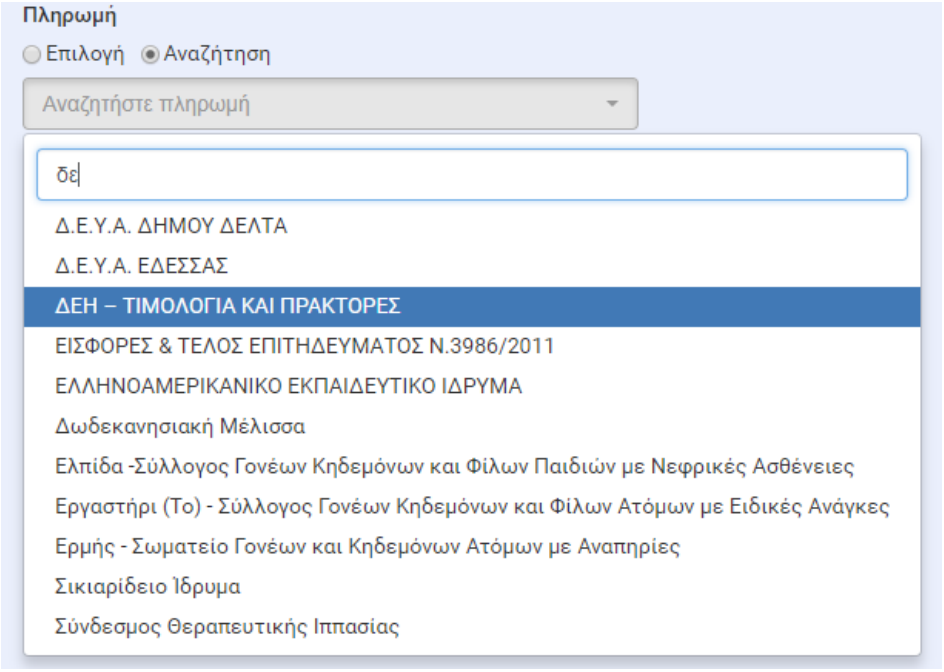


Εικόνα 53: Αναζήτηση πληρωμής βάσει κατηγοριών

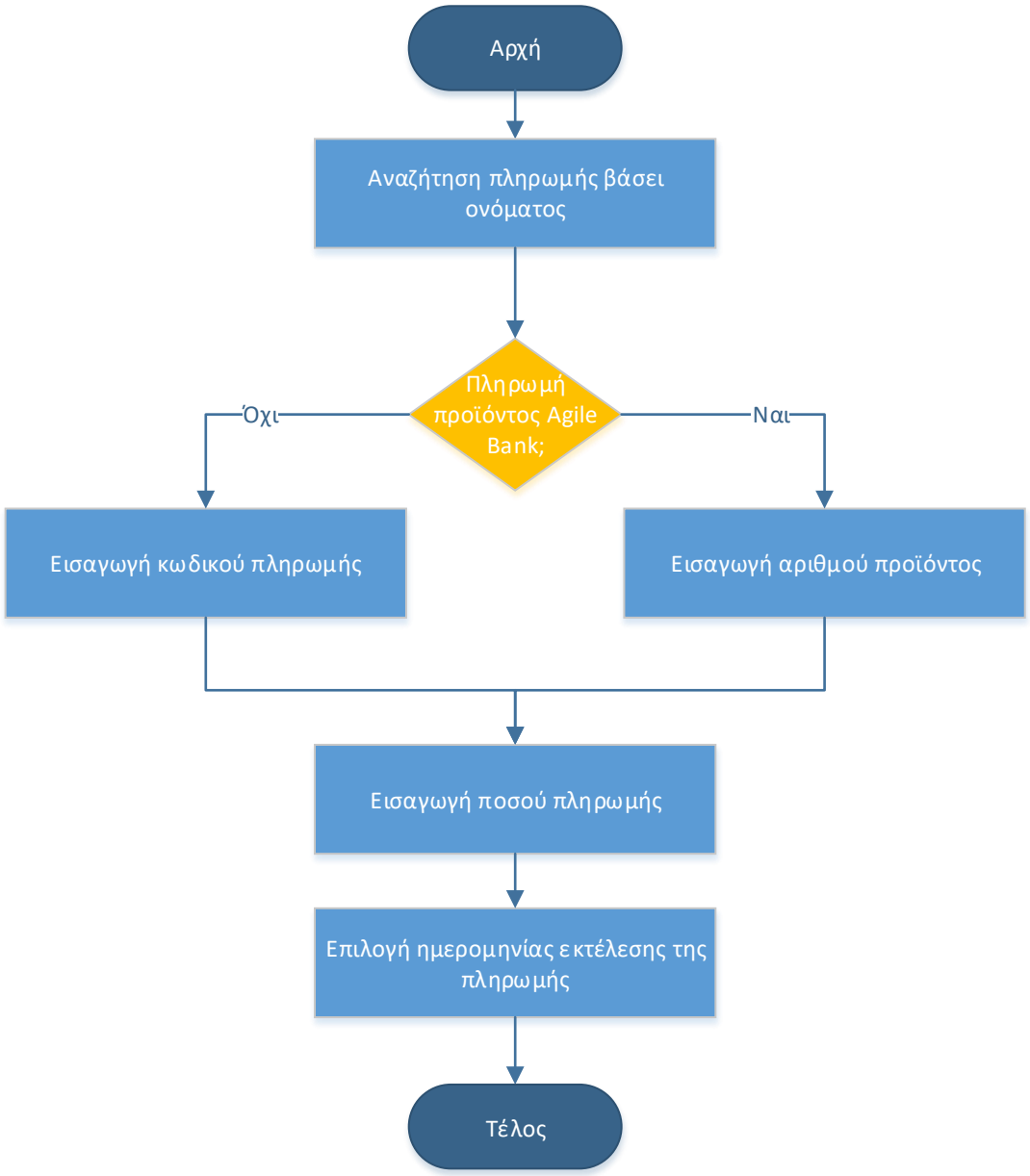


Εικόνα 54: Παράδειγμα αναζήτησης πληρωμής βάσει κατηγοριών

Αντίθετα, στη περίπτωση που ο χρήστης γνωρίζει ακριβώς, ή έστω περίπου το όνομα της πληρωμής που θέλει να πραγματοποιήσει, χρησιμοποιεί τη λειτουργία της αναζητείς πληρωμής βάσει ονόματος. Σε αυτή τη λειτουργία, ο χρήστης μπορεί να πληκτρολογήσει το όνομα της πληρωμής, ή έστω μέρος αυτού, για να ψάξει ανάμεσα σε όλες τις διαθέσιμες πληρωμές.

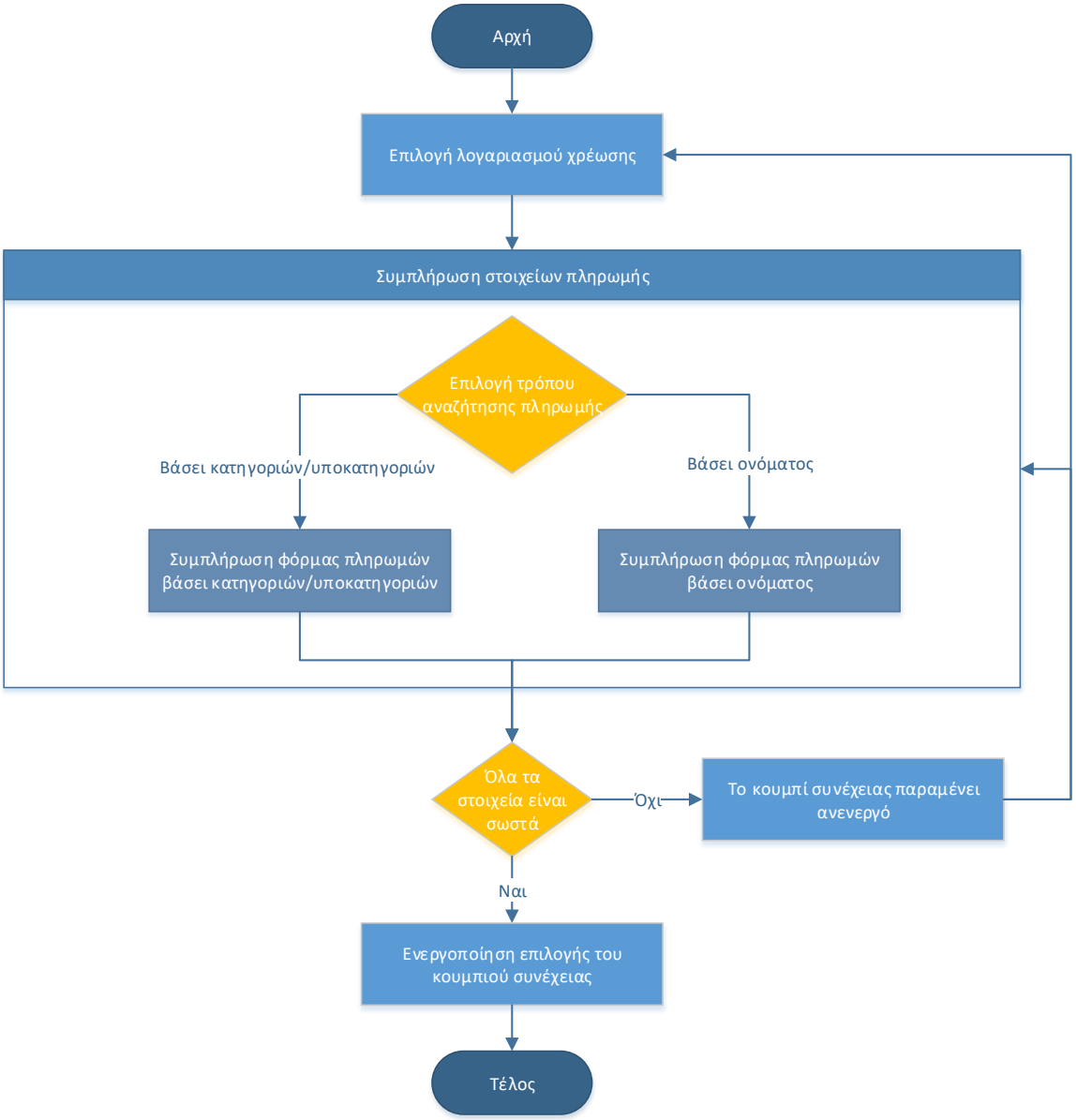


Εικόνα 55: Παράδειγμα αναζήτησης πληρωμής βάσει ονόματος



Εικόνα 56: Αναζήτηση πληρωμής βάσει ονόματος

Στη συνέχεια, και αφού συμπληρωθούν όλα τα πεδία της φόρμας της πληρωμής, πραγματοποιείται επικύρωση όλων των στοιχείων, και σε περίπτωση που τα στοιχεία είναι σωστά, ο χρήστης μπορεί να συνεχίσει την εκτέλεση της συναλλαγής, με το τρόπο που περιγράφεται στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*».

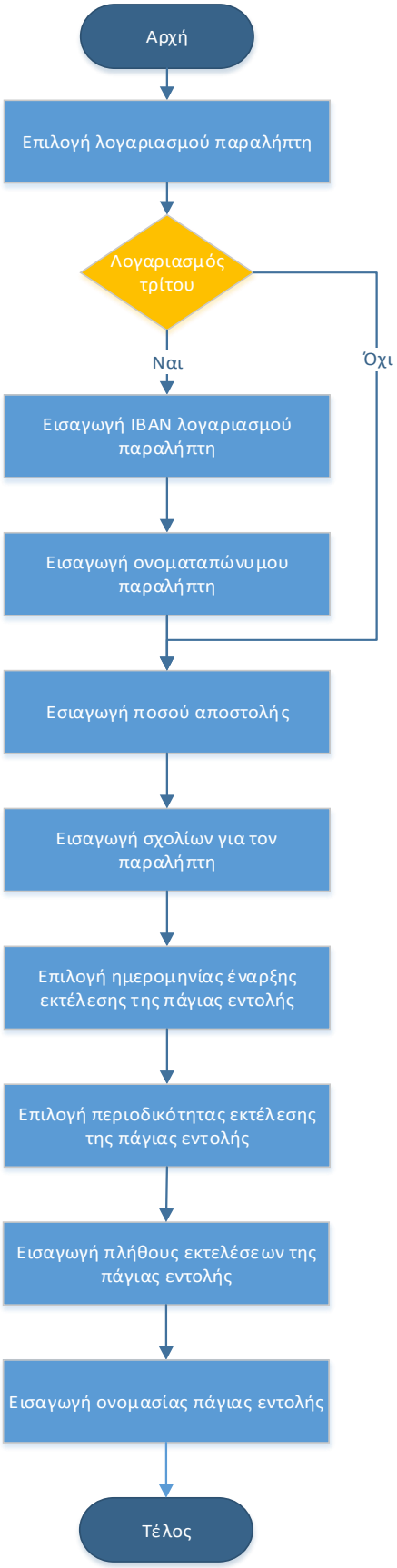


Εικόνα 57: Πλήρη ροή διαδικασίας συμπλήρωσης φόρμας πληρωμής

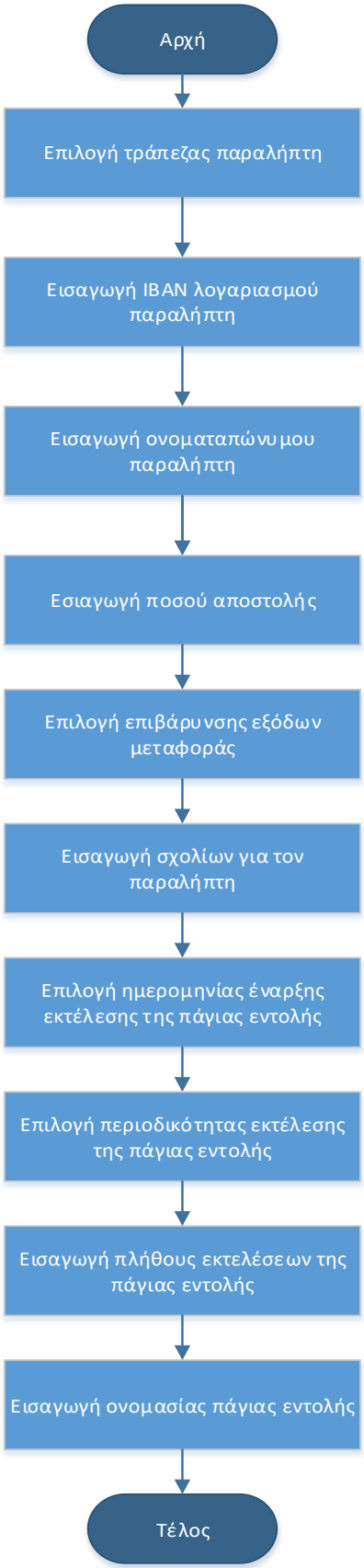
3.4.9. Λειτουργίες πάγιων εντολών

Οι λειτουργίες που είναι διαθέσιμες στη διαδρομή (route) των πάγιων εντολών είναι δύο: η δημιουργία και η διαγραφή πάγιας εντολής. Η δημιουργία της πάγιας εντολή διαχωρίζεται σε δύο υποκατηγορίες: δημιουργία πάγιας εντολής μεταφοράς χτημάτων, και πάγια εντολή πληρωμής. Επίσης, οι πάγιες εντολές μεταφορές διαχωρίζονται και αυτές με τη σειρά τους σε τρεις υποκατηγορίες, βάσει του τύπου μεταφοράς χρημάτων: πάγιες εντολές μεταφοράς εντός της τράπεζας, πάγιες εντολές μεταφοράς σε τράπεζα εσωτερικού και πάγιες εντολές μεταφοράς σε τράπεζα εξωτερικού. Η διαδικασία δημιουργίας πάγιας εντολής μεταφοράς

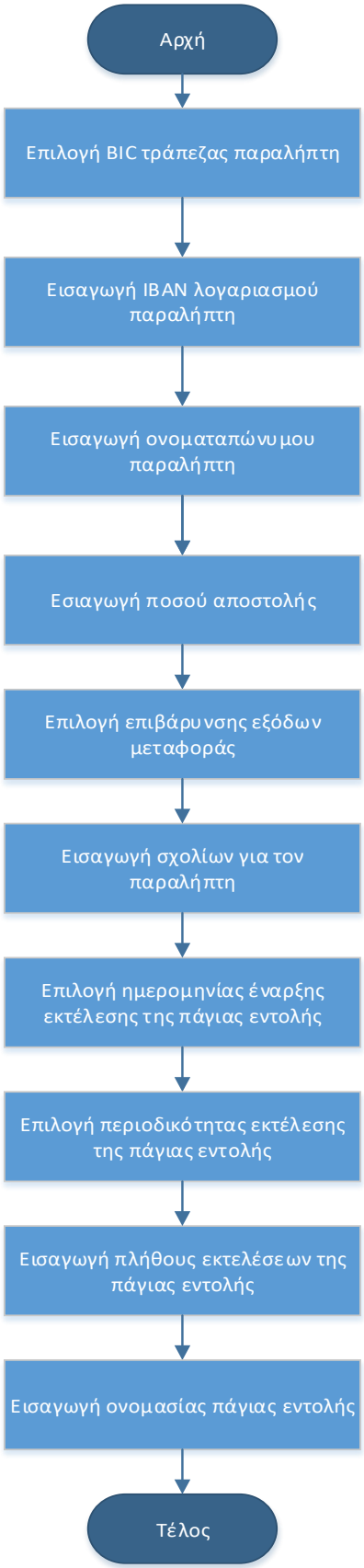
χρημάτων αρχικά είναι κοινή για κάθε τύπο μεταφοράς, και ξεκινάει με την επιλογή του λογαριασμού χρέωσης, ενώ τα απόμενα βήματα διαφοροποιούνται σύμφωνα με τον τύπο της μεταφοράς. Η ροή για κάθε διαδικασία δημιουργίας πάγιας εντολής μεταφοράς χρημάτων αποτυπώνεται στα διαγράμματα ροής που ακολουθούν.



Εικόνα 58: Ροή συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής μεταφοράς χρημάτων εντός της τράπεζας

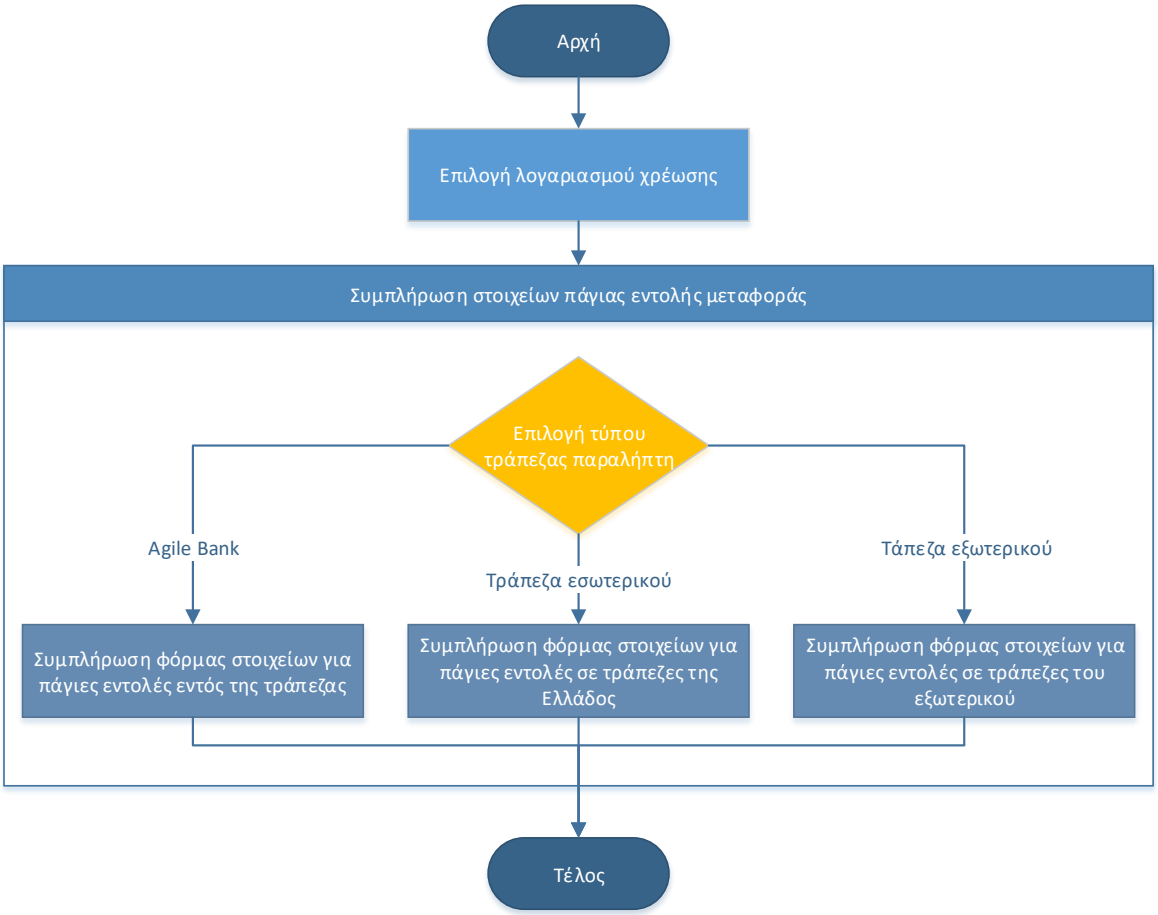


Εικόνα 59: Ροή συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής μεταφοράς χρημάτων σε τράπεζα εσωτερικού



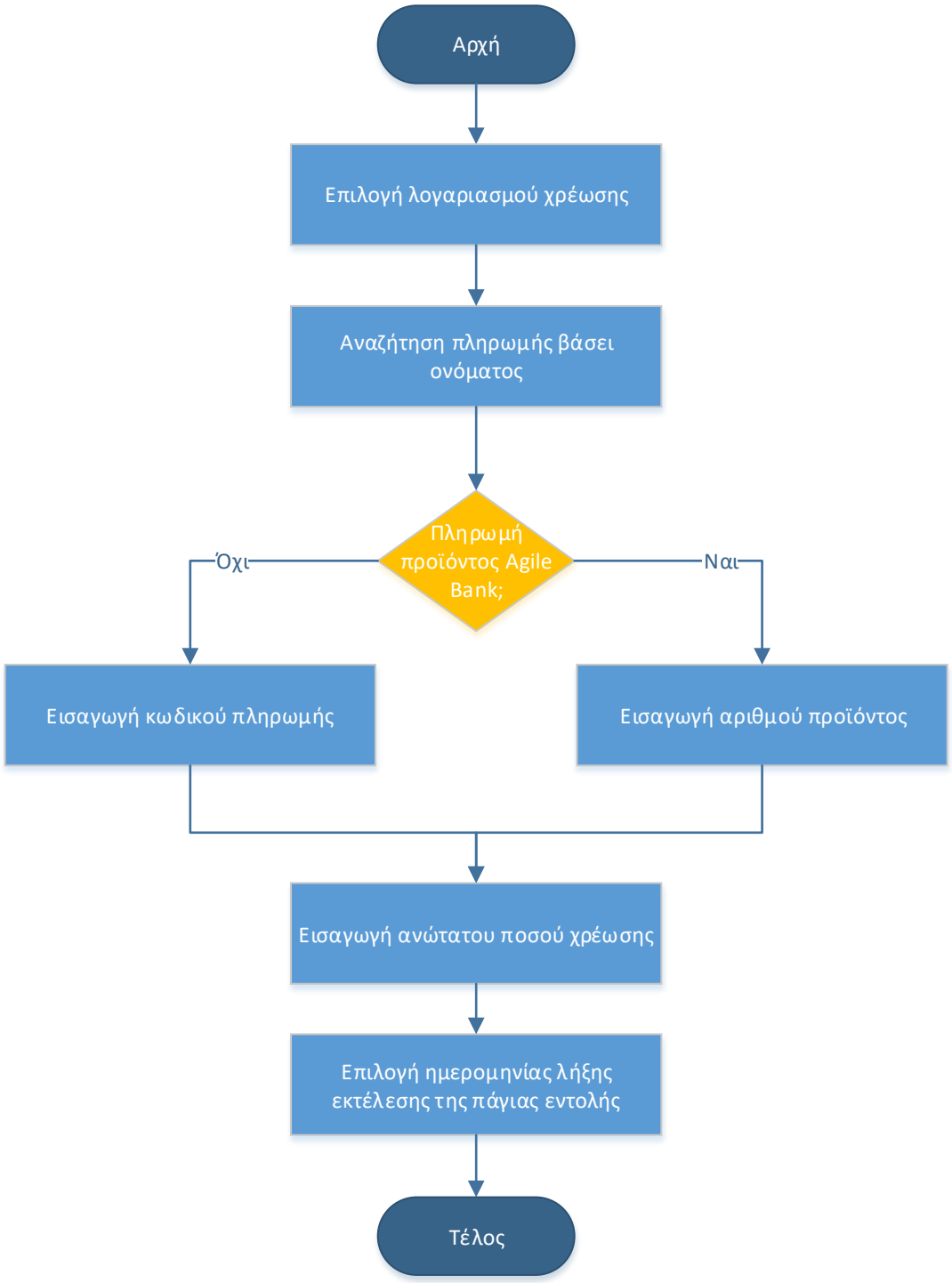
Εικόνα 60: Ροή συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής μεταφοράς χρημάτων σε τράπεζα εξωτερικού

Συνοψίζοντας, τα βήματα συμπλήρωσης της κάθε διαφορετικές φόρμας δημιουργίας πάγιας εντολής μεταφορών αποτυπώνονται στο παρακάτω διάγραμμα ροής.



Εικόνα 61: Συνολική εικόνα ροής συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής μεταφοράς χρημάτων

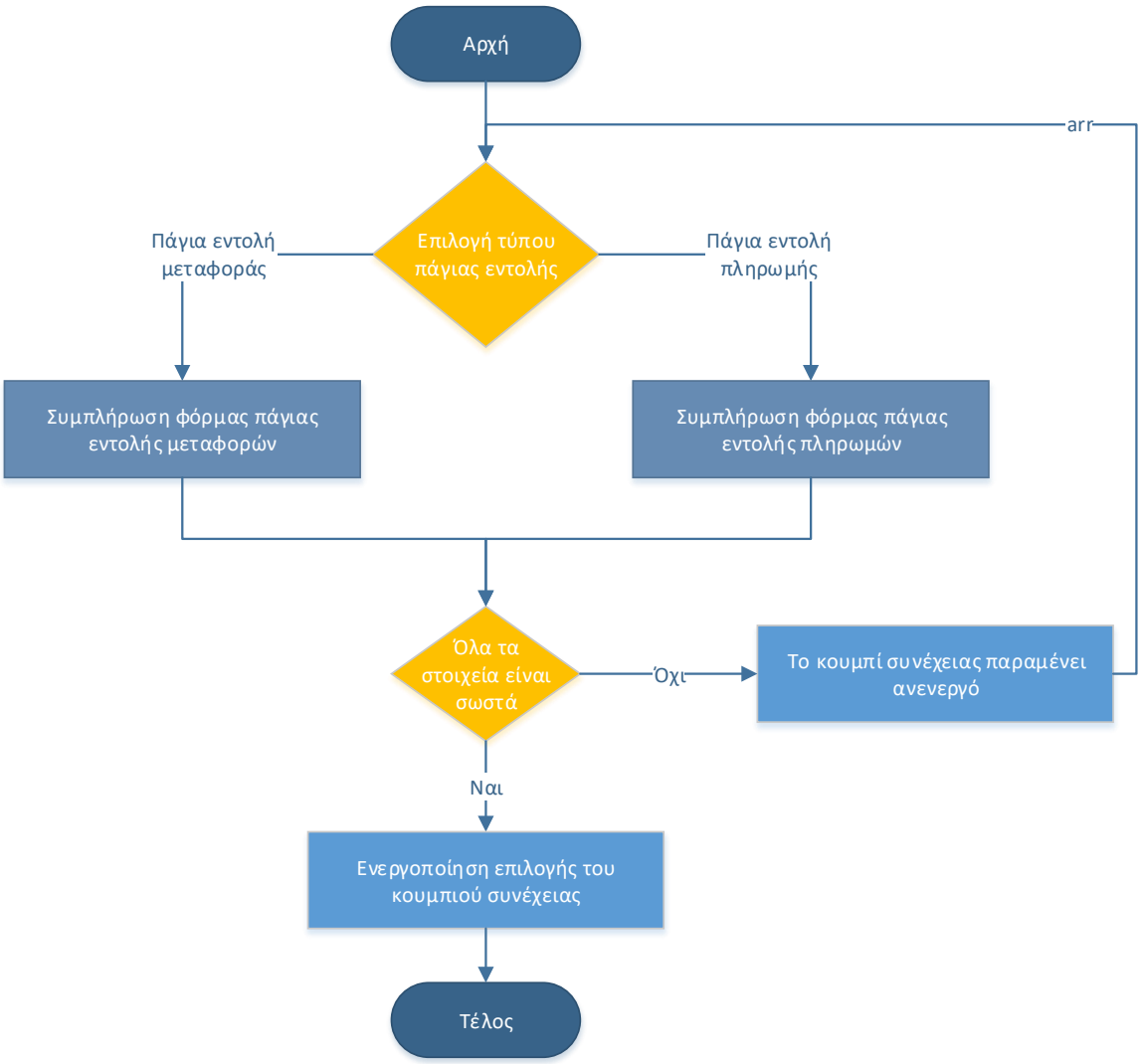
Στη περίπτωση της δημιουργίας πάγιας εντολής πληρωμής τα βήματα είναι αρκετά πιο απλά και δε χωρίζονται σε υποκατηγορίες. Αντίθετα ο χρήστης καλείται να διαλέξει το λογαριασμό χρέωσης, να αναζητήσει τη πληρωμή που θέλει βάσει του ονόματος της, να εισάγει το μοναδικό αριθμό πληρωμής, το μέγιστο ποσό χρέωσης και την ημερομηνία λήξης της πάγιας εντολής. Η διαδικασία αυτή αποτυπώνεται στο παρακάτω διάγραμμα ροής.



Εικόνα 62: Διαδικασία συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής πληρωμής

Στη συνέχεια, και αφού συμπληρωθούν όλα τα πεδία της εκάστοτε πάγιας εντολής που έχει επιλέξει να δημιουργήσει ο χρήστης, πραγματοποιείται επικύρωση των στοιχείων, και σε περίπτωση που τα στοιχεία είναι σωστά, ο χρήστης μπορεί

να συνεχίσει την εκτέλεση της συναλλαγής, με το τρόπο που περιγράφεται στη «*Ροή εκτέλεσης συναλλαγής μετά τη συμπλήρωση της φόρμας*».



Εικόνα 63: Πλήρη ροή διαδικασίας συμπλήρωσης φόρμας δημιουργίας πάγιας εντολής

ΚΕΦΑΛΑΙΟ 4

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο θα γίνει μια σύνοψη της εμπειρίας που αποκομίσθηκε από την εκπόνησή της πτυχιακής εργασίας, θα εξαχθούν συμπεράσματα και θα περιγραφούν πιθανές προοπτικές περαιτέρω ανάπτυξης.

4.1. Σύνοψη

Η υλοποίηση όλων των στόχων που είχαν τεθεί πριν την υλοποίηση του ιστοτόπου Web Banking και οι οποίο αναφέρονται στη περίληψη της παρούσας πτυχιακής εργασίας, δηλαδή ολόκληρη η εφαρμογή να είναι σωστά δομημένη, εύχρηστη, λειτουργική, γρήγορη και εύκολα επεκτάσιμη αποδείχτηκε ιδιαίτερα δύσκολή και απαιτητική στην υλοποίηση της. Η χρησιμοποίηση της αρχιτεκτονικής τριών επιπέδων υπήρξε σημείο κλειδί, διότι έδωσε την ελευθερία χρησιμοποίησης διαφορετικών τεχνολογιών σε κάθε επίπεδο, με αποτέλεσμα την επιλογή των βέλτιστων εργαλείων για την υλοποίηση του κάθε επιπέδου. Η React ως βιβλιοθήκη εφαρμογής μίας σελίδας (Single Page Application) σε επίπεδο παρουσίασης έκανε τον ιστότοπο εξαιρετικά γρήγορο, ενώ σε επίπεδο κώδικα βοήθησε πάρα πολύ στη σωστή δόμηση και επαναχρησιμοποίηση πολλών κομματιών κώδικα. Η υλοποίηση της επιχειρησιακής λογικής στη μορφή ενός REST API και της εφαρμογής των κανόνων της συγκεκριμένης αρχιτεκτονικής είχε ως αποτέλεσμα οι λειτουργίες να εκτελούνται γρήγορα και ασύγχρονα, δίνοντας στο χρήστη τη βέλτιστη εμπειρία χρήσης του Web Banking. Τέλος, η ανάπτυξη της εφαρμογής με χρήση της αρχιτεκτονικής τριών επίπεδών, έδωσε τη δυνατότητα στην εφαρμογή να είναι εξαιρετικά επεκτάσιμη και σε δυνατότητες, αλλά και σε ευκολία επέκτασης.

4.2. Προοπτικές

Οι δυσκολίες που υπερνικήθηκαν και οι ώρες που δαπανήθηκαν, κατέληξαν σε ένα λειτουργικό μεν τελικό αποτέλεσμα, το οποίο όμως σίγουρα επιδέχεται

βελτιώσεις και χρήζει περαιτέρω ανάπτυξης. Μια αρκετά καλή λειτουργία που θα μπορούσε να προστεθεί είναι αυτή της αποθήκευσης πληρωμών και μεταφορών, έτσι ώστε όταν ο χρήστης επιλέξει να πραγματοποιήσει μια πληρωμή ή μεταφορά, να έχει μια λίστα από αποθηκευμένες συναλλαγές που έχει πραγματοποιήσει στο παρελθόν και να μη χρειάζεται να πληκτρολογεί όλα τα στοιχεία ξανά και ξανά.

Επίσης, μπορεί να εξελιχθεί η αυτόματη λειτουργία αποσύνδεσης. Στην παρούσα υλοποίηση ο χρήστης αποσυνδέεται σε 10 λεπτά λόγω της μικρής χρονικής ζωής των token πιστοποίησης που λαμβάνει κατά την είσοδο του, ανεξάρτητα αν πραγματοποιεί κάποια ενέργεια ή όχι σε αυτό το διάστημα. Θα μπορούσε να βελτιωθεί η λειτουργία, ώστε κάθε φορά που ο χρήστης θα πραγματοποιεί κάποια κίνηση (ακόμα και αλλαγή καρτέλας) να λαμβάνει ένα νέο token πιστοποίησης που θα έχει πάλι 10 λεπτά ζωής, έτσι ώστε για όσο χρονικό διάστημα ο χρήστης είναι ενεργός, να μην αποσυνδέεται.

Τέλος, η πιο ενδιαφέρουσα αλλά και ταυτόχρονα μεγαλύτερη και πιο χρονοβόρα ανάπτυξη, θα ήταν να χτιστεί και μια εφαρμογή για mobile banking, χρησιμοποιώντας το ήδη υπάρχον REST API. Η μοναδική δυσκολία του εγχειρήματος θα ήταν η υλοποίηση της client εφαρμογής, δηλαδή μια επιπρόσθετη υλοποίηση του ήδη υπάρχοντος επιπέδου παρουσίας. Η αρχιτεκτονική που έχει χρησιμοποιηθεί κατά την ανάπτυξη του Web Banking ευκολύνει αυτό το εγχείρημα, αφού τα υπόλοιπα δύο επίπεδα, δηλαδή αυτό της επιχειρησιακής λογικής (REST API) και του επιπέδου δεδομένων (βάση δεδομένων και τον κώδικα που επικοινωνεί και τη διαχειρίζεται) είναι ήδη υλοποιημένα και χτισμένα έτσι ώστε να είναι όσο το δυνατόν stateless και επεκτάσιμα, με αποτέλεσμα οι επιπρόσθετες αλλαγές που θα χρειάζονταν να είναι ελάχιστες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Wikipedia, “History of Banking”, http://en.wikipedia.org/wiki/History_of_banking

[2] Gilbart, James William, “The history and principles of banking”

[3] Heathcote, Edwin, “Bank Builders”

[4] HistoryWorld, History of Banking, http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ac19

[5] Σαν σήμερα, “Η ιστορία του ATM” https://www.sansimera.gr/articles/1252

[6] Auth0, “A Brief History of JavaScript”, https://auth0.com/blog/a-brief-history-of-javascript/

[7] The New Stack, “JavaScript’s History and How it Led To ReactJS”, https: /thenewstack.io/javascripts-history-and-how-it-led-to-reactjs

[8] Redux, http://redux.js.org/

[9] Entity Framework Tutorial, http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[10] Martin Flower, “The Basics of Web Application Security”, https://martinfowler.com/articles/web-security-basics.html