Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ηλεκτρονικές και κινητές πληρωμές με χρήση σάρωσης QR Code.** **Electronic and mobile payments using QR Code scan.**
Ονοματεπώνυμο Φοιτητή	**Ανδρέας Καρμένης**
Πατρώνυμο	**Δημήτρης**
Αριθμός Μητρώου	**ΜΠΣΠ/15034**
Επιβλέπων	**Ευθύμιος Αλέπης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Δεκέμβριος, 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Επίκουρος Καθηγητής	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

ΕΥΧΑΡΙΣΤΙΕΣ & ΑΦΙΕΡΩΣΕΙΣ

Για τη διεκπεραίωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα, επίκουρο καθηγητή κ. Ευθύμιο Αλέπη, για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωσή της.

Επιπλέον, οφείλω να αφιερώσω την διπλωματική εργασία στα παιδιά μου Κυριάκο και Λευτέρη για το πολύτιμο χόνο που τους στερήσα, καθώς επίσης και στη γυναίκα μου, Γιασεμή, για την συμπαραστάση.

Πινακας περιεχομενων

1. Ευχαριστιες & αφιερωσεις.....3

2. **Περιληψη**.....6

2.1. ΕΛΛΗΝΙΚΑ.....6

2.2. ΑΓΓΛΙΚΑ.....8

3. **Εισαγωγη**.....11

4. **Ανασκοπηση πεδιου**.....16

4.1. **Axis Bank QR Payment App**.....16

4.1.1 ΣΧΕΤΙΚΑ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ.....16

4.1.2. ΤΙ ΕΙΝΑΙ Η MVISA.....16

4.1.3. ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ.....17

4.1.4. ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΗΣ MVISA.....17

4.2. **Bharat QR**.....18

4.2.1. ΤΙ ΕΙΝΑΙ ΤΟ BHARAT QR.....18

4.2.2. ΕΓΚΑΤΑΣΤΑΣΗ - ΥΛΟΠΟΙΗΣΗ ΣΥΝΑΛΛΑΓΗΣ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ……...18

4.3. **mVisa QR code payment λυσεις για τους merchants**……...19

4.3.1. ΔΥΝΑΤΟΤΗΤΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΟΝ ΕΠΑΓΓΕΛΜΑΤΙΑ.....20

4.3.2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΠΛΕΟΝΕΚΤΗΜΑΤΑ.....20

4.3.3. ΑΠΟΔΟΧΗ ΠΛΗΡΩΜΗΣ.....20

4.4. **BHIM – Making India Cashless**.....20

4.4.1. ΠΡΟΣΠΕΛΑΣΗ ΤΗΣ BHIM.....20

5. **Παρουσιαση και χρηση εφαρμογης (User manual)**.....22

5.1. PURCHASE GOODS.....22

5.2. LINK YOUR CARD.....22

5.3. SERVER CONFIGURATION.....23

5.4. PAYMENT LOGS.....23

5.5. EXIT.....23

6. **Αρχιτεκτονικη συστηματος**.....32

6.1. ΠΡΩΤΟ ΜΕΡΟΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 35

6.2. ΔΕΥΤΕΡΟ ΜΕΡΟΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 54

7. **ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ**.....63
7.1. ΣΥΜΠΕΡΑΣΜΑΤΑ.....63
7.2. ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ.....64

ΒΙΒΛΙΟΓΡΑΦΙΑ.....65

1. Περίληψη

Το QR Code Services App, είναι μια εφαρμογή Android η οποία αναλαμβάνει να διεκπαιρεώνει on-line, άμεσες συναλλαγές.

Η εφαρμογή, αποτελείται από δύο μέρη. Το ένα μέρος αφορά την android εφαρμογή η οποία εγκαθίσταται στη συσκευή του χρήστη και το δεύτερο μέρος, το λογισμικό που προσομειώνει την πλευρά της τράπεζας. Το τελευταίο, έχει αναπτυχθεί με Java και Spring boot framework. Η τράπεζα, ως είθισται, μετά από ένα αίτημα οικονομικής συναλλαγής, προβαίνει στους απαραίτητους ελέγχους και απαντά με μηνύματα - responses, τα οποία συμπεριλαμβάνουν ως περιεχόμενο την πληροφορία για μια επιτυχημένη είτε μια αποτυχημένη συναλλαγή. Στην περίπτωση της αποτυχημένης συναλλαγής, το response, περιέχει κωδικούς και μηνύματα σφαλμάτων τα οποία περιγράφουν τον λόγο της αποτυχίας.

Στις επιτυχημένες απάντησεις-responses, δεν υπάρχουν κωδικοί σφαλμάτων, αλλά υπάρχουν κωδικοί για την ένδειξη μιας πετυχημένης συναλλαγής που περιέχει περισσότερες πληροφορίες σχετικές με την συναλλαγή όπως είναι ο κωδικός ή το αναγνωριστικό της πληρωμής.

Από την άλλη, το πρώτο μέρος της εφαρμογής (Android εφαρμογή), είναι μια native εφαρμογή, και ο σκοπός ύπαρξής της είναι η απλούστευση της διαδικασίας της αγοράς ενός προϊόντος συγκριτικά με τις υφιστάμενες παραδοσιακές διαδικασίες και τους τρόπους πληρωμής που ήδη γνωρίζουμε. Ο χρήστης της εφαρμογής, απλά σαρώνει ένα QR Code το οποίο φέρει πάνω του κάποιο προϊόν, με ή χωρίς υλική υπόσταση. Για παράδειγμα, ένα τέτοιο προϊόν μπορεί να είναι ένα εμφιαλωμένο μπουκάλι νερό (υλική υπόσταση), είτε η αγορά ενός προϊόντος στο διαδίκτυο (αγορά αεροπορικού εισιτηρίου - άϋλη υπόσταση) και η εφαρμογή αποκωδικοποιεί το QR Code, εξάγει τις πληροφορίες που ενδιαφέρουν, για παράδειγμα πληροφορίες για τον επιτηδευματία, όπως τα καταστήματα Walmart αλλά και του προϊόντος, για παράδειγμα το όνομα του προϊόντος και η τιμή του.

Στη συνέχεια, επεξεργάζεται τα στοιχεία αυτά και ένας συνδυασμός από προεπιλεγμένες πληροφορίες εμφανίζονται στην οθόνη του κινητού του χρήστη. Ο ίδιος, έχει πάντα τον έλεγχο για μια συνέχεια της επικείμενης αγοράς ή ακύρωσής της. Αν επιλεγεί η συνέχεια, ο χρήστης οδηγείται στην επόμενη οθόνη και καλείται να επιλέξει μια από τις χρεωστικές/πιστωτικές κάρτες για παράδειγμα (VISA, MASTER CARD, DISCOVER) ή κάποιο IBAN λογαριασμό του, ανάλογα με ποιές κάρτες και λογαριασμούς έχει εκ των πρωτέρων καταχωρήσει. Πρέπει να τονιστεί ότι η εφαρμογή, για λόγους παρουσίασης και απλούστευσης, θεωρεί ότι ο χρήστης έχει εκ των πρωτέρων δηλώσει μια σειρά από δικές του χρεωστικές/πιστωτικές κάρτες και λογαριασμό(ους) του. Στην παρούσα υλοποίηση, οι πληροφοριες αυτές είναι δηλωμένες σε αρχείο τύπου xml, αλλά σε μια εμπορεύσιμη έκδοση της εφαρμογής, εννοείται πως θα πρέπει η εφαρμογή να επιτρέπει στον χρήστη, να προσθέτει, να αφαιρεί ή να τροποποιεί αυτά τα στοιχεία μέσα από ξεχωριστές οθόνες.

Αφου επιλεγεί και η κάρτα ή ο λογαριασμός, από που θα αφαιρεθεί το χρηματικό ποσό, τότε ο χρήστης προτρέπεται να εισάγει και το pin. Το pin είναι ένα τετραψήφιο αριθμητικό δεδομένο το οποίο υπόκειται σε διάφορους ελέγχους. Σε περίπτωση διαφορετικής εισαγωγής, η εφαρμογή αναρτά σχετικά μηνύματα λάθους σε κόκκινες αποχρώσεις και εστιάζει (focuses) στο πεδίο pin έως ότου ο χρήστης δώσει το σωστό συνδυασμό. Αν και στα πλαίσια της υλοποιημένης εφαρμογής, επειδή πρόκειται για μια demo έκδοσή της, δεν θα αποτελούσε πρόβλημα να επιτρέπονταν οποιοιδήποτε συνδυασμοί, ωστόσο, για να είναι περισσότερο ρεαλιστικό, οι κωδικοί, περιορίζονται σε συγκεκριμένα pins, για κάθε κάρτα ή λογαριασμό. Στη συνέχεια, τα pin κωδικοποιούνται και ενσωματώνεται στο request. Αυτή η πληροφορία αποκρύπτεται και αργότερα αποθηκεύεται. Μπορεί μόνο να προβληθεί ως μέρος της καταγγεγραμμένης ιστορικής πληροφορίας (log) κατ' απαίτηση του χρήστη σε μια ειδική οθόνη, την server logs (εικ. 19).
Το επόμενο σταδιο, είναι οι έλεγχοι της εφαρμογής για την διαπίστωση την ύπαρξης ή όχι του ασύρματου ή mobile δικτύου. Σε περίπτωση θετικής έκβασης, δηλαδή αν μπορέσει να αποκτήσει πρόσβαση στο δίκτυο και κατόπιν απαιτούμενης έγκρισης (άδειας) του χρήστη, στέλνει ένα request στην τράπεζα (στο δεύτερο μέρος του λογισμικού της εφαρμογής) . Αν εντός δεκαπέντε δευτερολέπτων, ως χρονικό ορίο, δεν πάρει μια απάντηση τότε, η εφαρμογή απαντά με ένα time out response και αντίστοιχο μήνυμα θα προβληθεί στην οθόνη του χρήστη προκειμένου να τον ενημερώνει για το σχετικό γεγονός.
Το λογισμικό το οποίο προσομειώνει την τράπεζα, μπορεί να είναι εγκατεστημένο τοπικά ή απομακρυσμένα. Αυτό ρυθμίζεται απο μια αντίστοιχη οθόνη του χρήστη που διαθέτει η εφαρμογή.
Στη θετική έκβαση της παραπάνω απάντησης, η εφαρμογή στέλνει το request στο url το οποίο καθορίζεται από την εικ.16, 17 και το λογισμικό του δεύτερου μερους της εφαρμογής, αναλαμβάνει τους διάφορους ελέγχους εγκυρότητας των μεταφερόμενων στοιχείων και απαντά αναλόγως. Σε κάθε περίπτωση, είτε αποτυχίας είτε επιτυχίας έκβασης της συναλλαγής, μια αφιερωμένη οθόνη της εφαρμογής για αυτό το λόγο, παρουσιάζεται στον χρήστη ενημερώνοντάς τον για την κατάσταση της έκβασης. Επίσης, το κάθε αποτέλεσμα ξεχωριστά, καταγράφεται σαν log σε μια απομακρυσμένη NO-SQL βάση δεδομένων (firebase) της Google.
Τέλος, υπάρχει μια αντίστοιχη ξεχωριστή σελίδα της εφαρμογής, η οποία σε πραγματικό χρόνο, ανακτά σε μια λίστα, επεξεργάζεται και παρουσιάζει ανά μέρα ομαδοποιημένα και ταξινομημένα από το πιο πρόσφατο σε μορφή πτυσόμενης λίστας, συμπυκνωμένη πληροφορία σχετικά με συνολικά αριθμητικά πόσα, τα οποία δείχνουν των αριθμό των επιτυχημένων και αποτυχημένων συναλλαγών ανα ημέρα.

1.1 Abstract

QR Code Services App is an Android application that undertakes to execute on-line direct transactions.

The application consists of two parts. One part concerns the android application that is installed on the user's device and the second part of the software embody the side of the bank. The latter, developed with Java and Spring boot framework. The bank, as usual, after the necessary checks, responds with messages, which include information about a successful or a failed transaction as the content. In the case of a failed transaction, the response contains error codes and logs that describe the reason of the failure.

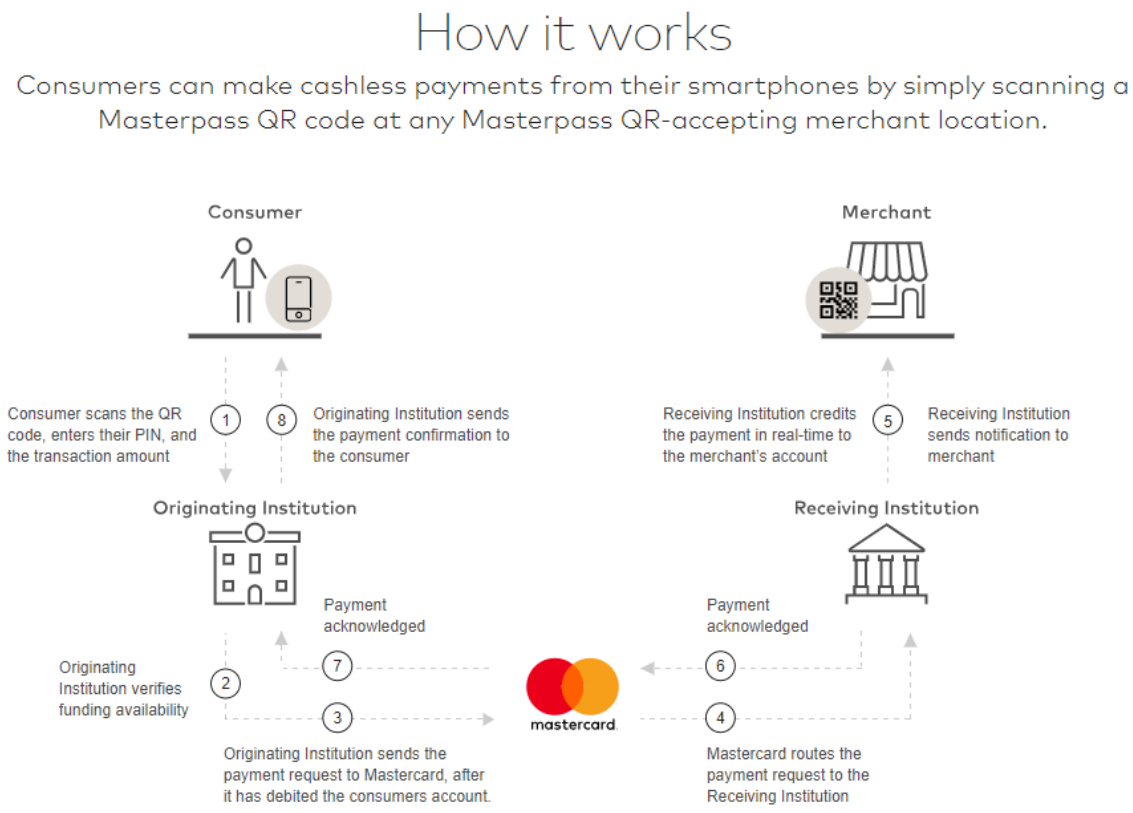
At the successful responses, there are no error codes but there are code indications of a successful transaction containing more transaction-related information such as the payment code or ID.

On the other hand, the first part of the app (Android app) is a native application, and its purpose is to simplify the purchase process of a product compared to the existing traditional processes and payment methods we already know.The user of the application simply scans a QR Code on a product, whether or not it has a physical nature, for example, such a product may be a bottled water (physical material) or the purchase of a product on the Internet (the air ticket market - intangible material) and the application decodes the QR Code, extracts the information of interest, for example, information about the trader, such as Walmart and the product, for example the name of the product and its price.

It then processes these data and a combination of pre-agreed information is displayed on the mobile user screen. The user, has always under his control the continuation of the upcoming market or its cancellation. If selected to proceed, the user is taken to the next screen and has to choose one of the debit/credit cards for example (VISA, MASTER CARD, DISCOVER) or an IBAN account, depending on which cards and accounts he has previously registered. It should be emphasized that the application, for simplicity reasons of presentation, considers that the user has previously stated a series of his own debit / credit cards and account (s). In the present embodiment, this information is declared in an xml file, but in a commercial version of the application, the application should allow the user to add, remove, or modify these elements through separate and dedicated screens.

After selecting the card or account from which the money will be deducted, the user is also prompted to enter the pin. The pin is a four-digit numeric data and subjects to various checks. Only numeric digits are allowed and the pin's length, has a restriction of four digits, otherwise the application posts relative error messages in red shades and focuses in the pin field until the user gives the correct combination. Although the implemented application is a demo version and not a market one, it would not be a problem to allow any combination, however, in order to give more realism, the pins are limited to specific for each card or account. The pin is then encoded and embedded in the request. This information is hidden from the user in the forthcoming

sequence of screens and finally stored. It can only be viewed as part of the logged information upon request of the user on a special screen log.
The next step is the check which the application does to determine whether or not any wireless or a mobile network exists. In case of a positive outcome, namely if it can acquire any network, after the required authorization (license) of the user, it sends a request to the bank (the second part of the application software). If within fefteen seconds any timeout occurs, (time out is considered the maximum time which the application waits for the response from the server), then the application responds with a time out response and a relative corresponding message will be displayed on the user screen as an Android toast message to inform for the event.
The software that simulates the bank can be installed locally or remotely as well. That is a URL which decides the location and it is set by a special corresponding screen of the app (icon. 16, 17).
In the positive outcome of the above response, the application sends the request to the url and the software of the second part of the application takes over the various validity checks of the transferred data and responds accordingly.
In any case, either failure or success of the transaction outcome, a dedicated application screen for this reason, is presented to the user by informing him of the outcome status. Also, each result individually, is logged as a record in a Google's remote NO-SQL firebase.
Finally, there is a corresponding separate page of the application, which in real time, recovers historical transactions in a list, processes and presents each day grouped and sorted from the most recent event in a fold-out list, solid information about numerical quantities, which show the the totals and the number of successful and failed transactions per day.


How it works
Consumers can make cashless payments from their smartphones by simply scanning a Masterpass QR code at any Masterpass QR-accepting merchant location.
Consumer
Merchant
Consumer scans the QR code, enters their PIN, and the transaction amount
1
8
Originating Institution sends the payment confirmation to the consumer
Receiving Institution credits the payment in real-time to the merchant's account
5
Receiving Institution sends notification to merchant
Originating Institution
Receiving Institution
Payment acknowledged
Payment acknowledged
Originating Institution verifies funding availability
2
7
6
3
4
mastercard
Originating Institution sends the payment request to Mastercard, after it has debited the consumers account.
Mastercard routes the payment request to the Receiving Institution

(εικ. 1)

2. Εισαγωγή.

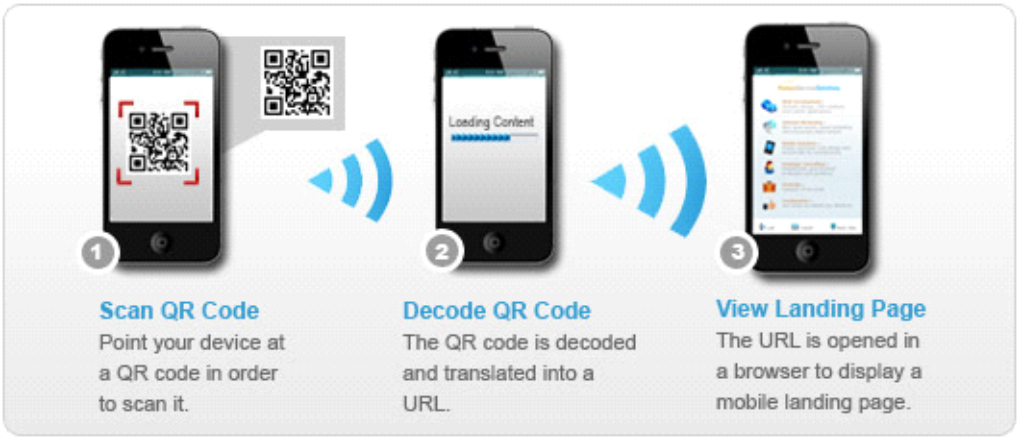
Το εγχείρημα των πληρωμών από το κινητό τηλέφωνο σαρώνοντας ένα QR Code, είναι σχετικά μια νέα καινοτομία η οποία είναι σε πειραματικό στάδιο. Πρόκειται για μια πολλά υποσχόμενη εφαρμογή η οποία απαιτεί προσαρμογή συμπεριφοράς των καταναλωτών αλλά και εμπόρων λιανικής αγοράς.

Η εφαρμογή QR Code Services, είναι σχεδιασμένη να να χρησιμοποιηθεί στις σύγχρονες φυσικές ή ηλεκτρονικές αγορές. Έτσι, αποκτάται μια υπεροχή έναντι των παραδοσιακών – γνωστών τρόπων των αγορών.

Η εφαρμογή, είναι αφιερωμένη στην περαιτέρω απλούστευση της διαδικασίας πληρωμών μέσω του smart phone, και της σάρωσης του QR Code. Αποκωδικοποιεί την πληροφορία που εμπεριέχεται σε μια QR Code σήμανση, η οποία εξάγεται αυτόματα και χωρίς λάθη, σε διαδοχικές οθόνες της εφαρμογής, χωρίς να χρειάζονται επιπρόσθετες συσκευές (ταμεία,ή pos μηχανήματα) ή/και κάποια επιπρόσθετη επεξεργασία ή χρεωστικές επιβαρύνσεις, με τα υψηλότερα standards ασφάλειας, σιγουριάς και εγγυήσεων.

Τα παραπάνω, πραγματοποιούνται σε τρία βήματα.

Chose QR Payment Scan QR Code Confirm Payment



(εικ. 2)

Τα QR Codes μπορούν να χρησιμοποιηθούν με διάφορους έξυπνους και αποτελεσματικούς τρόπους δελεάζοντας τους καταναλωτές να εμπλακούν στη διαδικασία.

Από τη μια πλευρά, οι σημερινοί καταναλωτές αναζητούν διαδραστικές και εξατομικευμένες εμπειρίες. Για πολλούς από αυτούς, η κινητή συσκευή τους είναι μια πρόκληση στο να επιλέγουν να συμμετάσχουν σε αυτές. Από την άλλη, αποτελούν μια ευκαιρία για τους διαφημιζόμενους να επιτύχουν υψηλότερη απόδοση επένδυσης σε κάθε διαφημιστική καμπάνια. Αυτό καθιστά την τεχνολογία QR Code ιδανική λύση για τη σύνδεση με πιθανούς νέους και πολύ ενδιαφερόμενους πελάτες.

Η σάρωση ενός QR Κώδικα, απαιτεί ελάχιστη προσπάθεια εκ μέρους του χρήστη οι οποίοι αναμένουν σαν αντάλλαγμα άμεσα αποτελέσματα, επομένως η δημιουργική χρήση του Κώδικα Ταχείας Ανταπόκρισης (Quick Response Code – QR Code), απαιτεί τη γνώση του τόπου και του τρόπου χρησιμοποίησής τους.

Οι έμποροι της λιανικής πώλησης, είναι ειδικοί στη χρήση των βιτρινών για να προσελκύσουν την προσοχή. Με την παροχή σήμανσης που φέρει τον QR Κωδικό, στους εμπόρους λιανικής πώλησης, οι πληροφορίες σχετικά με το προϊόν είναι διαθέσιμες στους χρήστες των smartphone, ακόμα και όταν τα καταστήματα είναι κλειστά.

Οι αγοραστές με κινητές συσκευές έχουν γίνει εξαιρετικά άνετοι με το "showrooming" - μια πρακτική χρήσης για προβολή προϊόντων αλλά και αγορών στο διαδίκτυο. Δεν υπάρχει κανένας καλός λόγος να μην επωφεληθούν από αυτή την τάση. Με οθόνες και σήμανση που φέρουν τα QR Codes, μπορούν να βοηθηθούν να κλείνουν μια πώληση από σχετικές πληροφορίες, όπως κριτικές πελατών και αποκλειστικές προσφορές.

Υπηρεσίες πληρωμών κατά τρόπο που προτείνει η εφαρμογή QR Code Services, μετατρέπονται σε μια ευχάριστη και άνετη εμπειρία, μειώνοντας τις ώρες αναμονής σε πολυάριθμα καταστήματα και εστιατόρια. Επίσης, πληρωμή λογαριασμών όπως αυτοί της ΔΕΗ, ΕΥΔΑΠ, του ΟΤΕ ή ηλεκτρονικών εισιτηρίων και γενικά αγορές στο διαδίκτυο ή σε αυτόματους πωλητές, μπορούν να πραγματοποιηθούν σαρώνοντας τον κωδικό QR Code που φέρουν πάνω τους, με το τηλέφωνο και να μεταφέρουν την πληρωμή από τον προσωπικό λογαριασμό τους στη συσκευή, επωφελούμενοι από την ευκολία αυτού του είδους πληρωμής.

Εύκολα γίνεται αντιληπτό πως η αναμονή σε ταμειακές ή POS μηχανημάτων ουρές ή η συμπλήρωση χρονοβόρων φορμών θέτοντας σε κίνδυνο προσωπικά ευαίσθητα δεδομένα, εξαλείφεται, αυξάνοντας έτσι την ασφάλεια και την διαφύλαξή τους μακριά από αδιάκριτα βλέμματα.

Στις παρακάτω εικόνες βλέπουμε κάποιες πληρωμές μέσω της QR Code υπηρεσίας. Στις διαδοχικές εικόνες 3,4 και 5, παρατηρούμε πως ένα νέο ζευγάρι καθώς κάνει την βόλτα του, αποσπά την προσοχή τους μια διαφημιστική καμπάνια κάποιων νέων ταινιών. Καθώς βλέπουν αποσπάσματα σε μια εξ' αυτών, η οποία διαθέτει στην άκρη κάποια QR Code σήμανση, σαρώνουν με την συσκευή τους το QR Code, αποφασίζουν είτε να κατεβαζουν

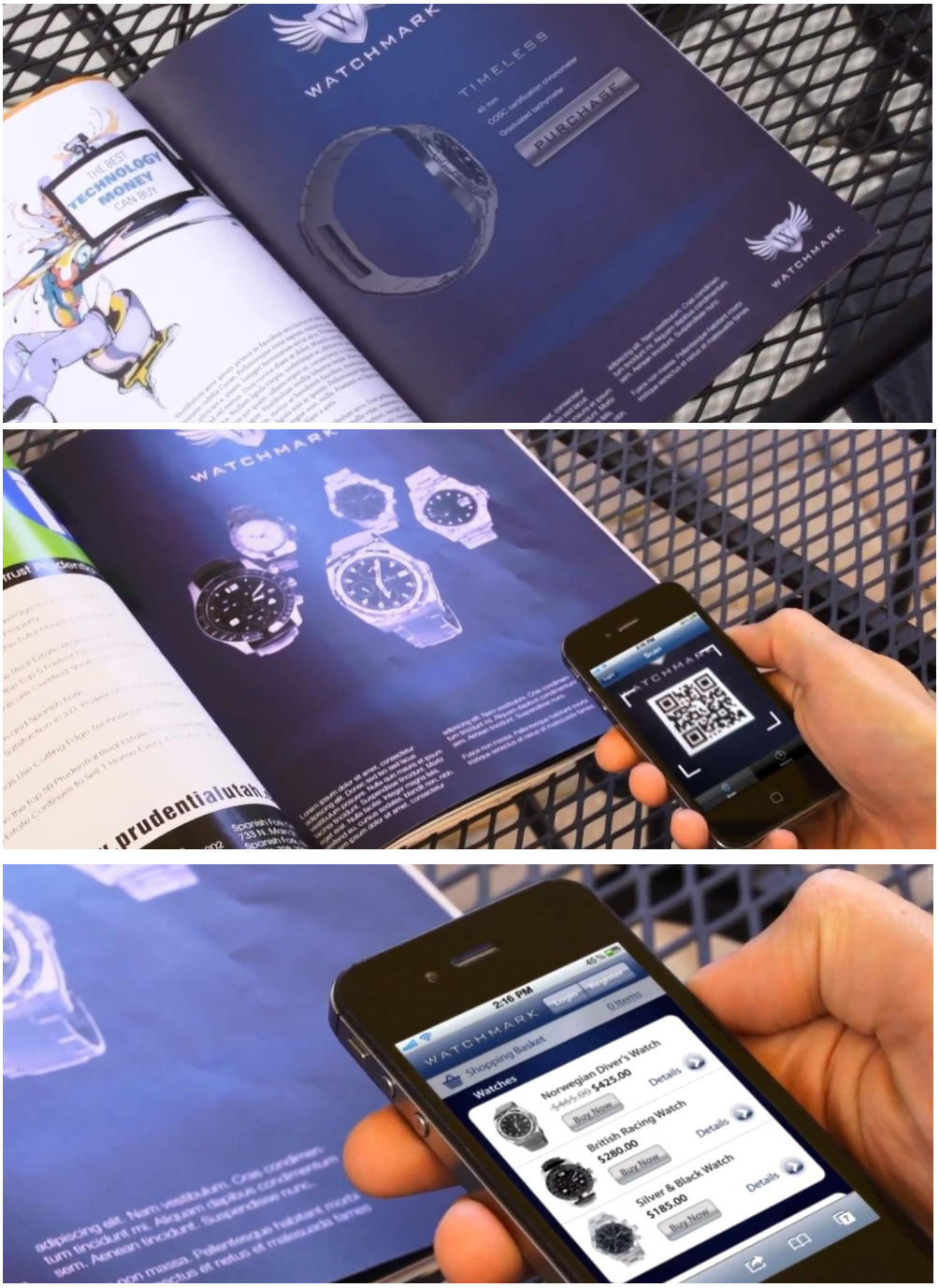
την ταινία στη συσκευή τους να την παρακολουθήσουν αργότερα είτε να αγοράζουν εκείνη τη στιγμή εισιτήρια για το σινεμά.

Μια άλλη χρήση, μπορεί να παρατηρηθεί στις εικόνες 6,7 και 8 οι οποίες κάποιος πίνοτας τον καφέ του, καθώς ξεφυλλίζει κάποιο περιοδικό, η προσοχή του εστιάζεται σε κάποια ανδρικά ρολόγια. Η σελίδα αυτή διαθέτει συν τοις άλλοις και κάποιο QR Code. Τελικά, αποφασίζει να σαρώσει το QR Code και αυτόματα μεταβαίνει online για την αγορά του από μια λίστα με περισσότερες επιλογές παρόμοιων ρολογιών.

Αυτές είναι κάποιες ελάχιστες από τις πολλές δυνατότητες της εφαρμογής. Όλα τα θετικά στοιχεία και οι ιδέες των όσων προαναφέρθηκαν, ελπίζουμε πως θα εμπνεύσουν τους επαγγελματίες εμπόρους σε επόμενες καμπάνιες που θα αφορούν τέτοιου είδους πληρωμές μέσω κινητών συσκευών.


SUPERVISION BY
MIKE HILL, A.C.E.
RON HOWARD
SCAN TO
VIEW THE
TRAILER
1:53 PM
Photos
Settings
Maps
Compass
Scan
Camera
JOHN CALLEY
SCAN TO
VIEW THE
TRAILER

(εικ. 3, 4, 5)


WATCHMARK
TIMELESS
PURCHASE
THE BEST TECHNOLOGY MONEY CAN BUY
WATCHMARK
2:16 PM
WATCHMARK
Shopping Basket
Watches
Norwegian Diver's Watch
$425.00
Details
Buy Now
British Racing Watch
$280.00
Details
Buy Now
Silver & Black Watch
$185.00
Details
Buy Now

(εικ. 6, 7, 8)

3. Ανασκόπηση πεδίου.

Παρόλο που υπάρχουν πολλές εναλλακτικές όσον αφορά την αποκωδικοποίηση των QR Codes και την εξαγωγή των πληροφοριών στο app και play store, ωστόσο, ελάχιστες σε αριθμό εφαρμογές είναι ολοκλήρωμένες σε επίπεδο πληρωμών και χρησιμοποιούν QR Code σάρωση για online on the go shopping. Παρακάτω προβάλλονται ενδεικτικά, κάποιες εφαρμογές σχετικές με σαρώσεις QR Codes, οι οποίες βρίσκονται στα Android και iPhone stores.

I. Μια εφαρμογή η οποία χρησιμοποιεί QR Code σάρωση είναι η Axis Bank QR Payment App (εικ. 9) που χρησιμοποιείται ήδη από την Axis Bank της Ινδίας στη Βομβάη.

Σχετικά με την εφαρμογή.

Η Axis Bank συνεργάζεται με τη Visa για να προάγει το mVisa- καινοτόμα λύση πληρωμής, η οποία βοηθάει τους κατόχους χρεωστικών καρτών της Axis Bank Visa να πραγματοποιούν γρήγορες πληρωμές μόνο με τη σάρωση κωδικών QR διαθέσιμων σε καταστήματα λιανικού εμπορίου και τυπωμένα πάνω σε λογαριασμούς.

Τι είναι η mVisa;

Η mVisa είναι μια υπηρεσία πληρωμής (card-to-merchant payment service) από κάρτα πελάτη σε λογαριασμό εμπόρου που παρέχεται ως μέρος της εφαρμογής Axis Mobile καταχωρώντας τις Visa χρεωστικές κάρτες για χρήση των mVisa πληρωμών. Οι κάτοχοι των καρτών μπορούν να κάνουν άμεση πληρωμή λογαριασμών μέσω σάρωσης των QR Codes που βρίσκονται σε καταστήματα εμπόρων ή αποτυπωμένα πάνω σε λογαριασμούς.

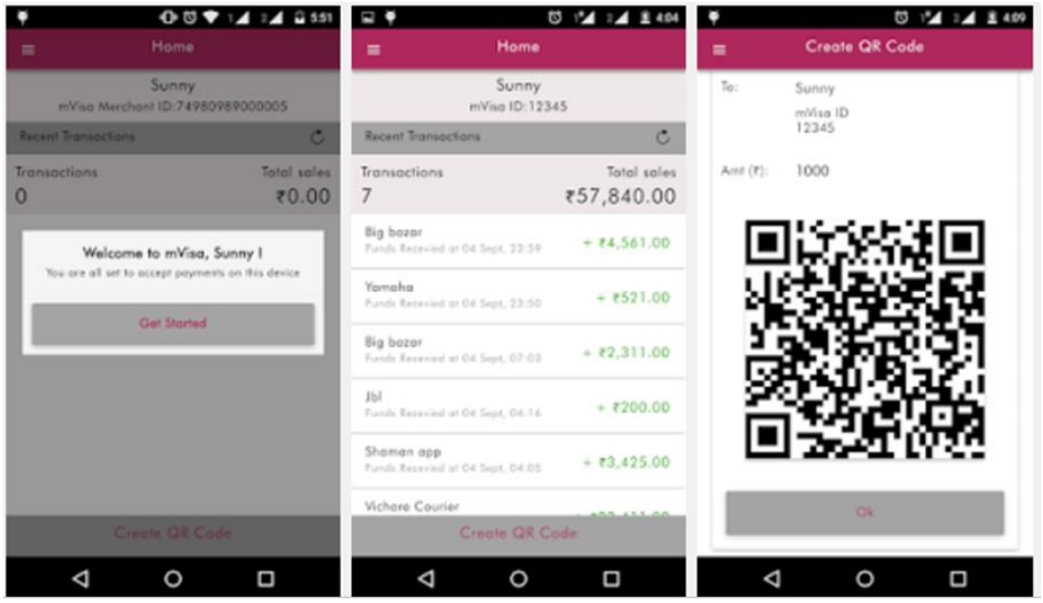
Πώς λετουργεί;

- Αρχικά απαιτείται η εγγραφή (register) του χρήστη μετά την λήψη της εφαρμογής από τα App & Play Stores.
- Δυνατότητα καταχωρήσεων χρεωστικών καρτών της Axis Bank Visa, συνδεδεμένες με τον κύριο λογαριασμό του χρήστη, για τη χρήση του mVisa.
- Δεν υπάρχει περιορισμός των αριθμών των καρτών που κάποιος χρήστης μπορεί να εισάγει στην εφαρμογή.
- Ο χρήστης κάνει click στο εικονίδιο Shop της εφαρμογής.
- Click στην επιλογή Pay με mVisa και σάρωση με QR Code.
- Επιβεβαίωση των λεπτομεριών της συναλλαγής και εισαγωγή της mPin για αυθεντικοποίηση/ταυτοποίηση.
- Επιλογή κάρτας για πληρωμή.

- Click στο check box το οποίο βρίσκεται δίπλα στο 'I Agree' για αποδοχή των κανονισμών και των συνθηκών που δείχνει ο σύνδεσμος.
- Click στο confirm.
- Η επιβεβαίωση συναλλαγής θα προβληθεί στην οθόνη του χρήστη.

Πλεονεκτήματα της mVisa.

- Ασφαλείς συναλλαγές καθώς οι πελάτες δεν θα πρέπει να παραδώσουν την κάρτα στον έμπορο.
- Εύκολες και άμεσες πληρωμές
- Χρησιμοποιήση της κάρτας Visa / Mastercard / Rupay Debit, για πραγματοποιήση πληρωμών σε τοποθεσίες εμπόρων σε ολόκληρη τη χώρα που εμφανίζουν έναν QR Code.
- Με την Axis Bank QR Payment, δεν χρειάζεται να μεταφερθούν και τα χρήματα παραμένουν σε ηλεκτρονικά πορτοφόλια για να γίνονται πληρωμές χρησιμοποιώντας κώδικες QR. Τα χρήματα παραμένουν στον Λογαριασμό Ταμιευτηρίου / Πιστωτικής Κάρτας και χρησιμοποιούνται μόνο κατά τη στιγμή της αγοράς.
- Οι αγορές που πραγματοποιούνται με την εφαρμογή πληρωμής Axis QR επικυρώνονται από έναν κωδικό που είναι γνωστός μόνο στον χρήστη. Επιπλέον, ο κωδικός πρόσβασης μπορεί να οριστεί χρησιμοποιώντας τον Αριθμό Χρεωστικής Κάρτας και το PIN του ATM που είναι επίσης γνωστοί μόνο στον χρήστη. Σε περίπτωση που το τηλέφωνό χαθεί, απλά θα χρειαστεί να γίνει μια κλήση στην υπηρεσία εξυπηρέτησης πελατών για ενημερώση του συμβάντος και ο χρήστης θα ενημερώνεται για όλες τις κάρτες που σχετίζονται με το αναγνωριστικό πελάτη.
- Στις επιτυχείς συναλλαγές πληρωμής, τόσο ο χρήστης όσο και ο έμπορος λαβάνουν μια άμεση ειδοποίηση που θα ενημερώνει για την πληρωμή, με το ίδιο περιεχόμενο.



(εικ. 9)

II. Μια επόμενη εφαρμογή είναι αυτή της Bharat QR (εικ. 10).

Η Axis Bank συνεργάζεται με τις Visa, MasterCard και Rupay για το λανσάρισμα της Bharat QR - μια μοναδική λύση πληρωμής, η οποία βοηθάει τους κατόχους χρεωστικών καρτών Axis Bank Visa, MasterCard και Rupay να πραγματοποιούν γρήγορες πληρωμές μόνο με μία σάρωση ενός QR Code, διαθέσιμου σε σημεία πώλησης και διάφορους λογαριασμούς, για να πραγματοποιούνται πληρωμές σε τοποθεσίες εμπόρων, σε ολόκληρη τη χώρα που εμφανίζουν έναν QR Code.

Τι είναι το Bharat QR;

Το Bharat QR είναι μια υπηρεσία πληρωμής μέσω πιστωτικής κάρτας (card-to-merchant payment service) που παρέχεται μέσω της εφαρμογής Πληρωμής QR της Axis Bank, μετά από προεγγραφή των χρεωστικών κάρτων τις Visa, MasterCard και Rupay για πληρωμή. Οι κάτοχοι καρτών μπορούν να κάνουν άμεση πληρωμή λογαριασμών στους εμπόρους. Λεπτομέρειες για τον έμπορο περιλαμβάνονται σε QR codes τα οποία είναι διαθέσιμα σε τοποθεσίες εμπόρων σε ολόκληρη τη χώρα που εμφανίζουν έναν QR Code.

Εγκατάσταση, αρχικοποίηση και υλοποίηση συναλλαγής με την εφαρμογή.

- Αρχικά απαιτείται η εγγραφή (register) του χρήστη μετά την λήψη της εφαρμογής από τα App & Play Stores.
- Εγχώρηση χρεωστικής κάρτας, ημερομηνίας λήξης αυτής, καθώς και του ATM Pin στην σχετική οθόνη.

- Λήψη κωδικού μιας χρήσης (One Time Password) μέσω του δηλωμένου αριθμού τηλεφώνου.
- Εισαγωγή του OTP στην οθώνη.
- Εισαγωγή εξαψήφιου κωδικού πρόσβασης.
- Επιλογή Scan & Pay.
- Σάρωση του QR Code στην πλευρά του Merchant.
- Εισαγωγή ποσού
- Click στο Pay.

(εικ. 10)

III. Axis Bank mVisa QR code payment λύσεις για τους Axis Bank merchants.

Η κρατική τράπεζα Bharat QR Merchant Application προσφέρει έναν απλό και αξιόπιστο τρόπο για να λήψη πληρωμών από τον έμπορο.

Δυνατότητες της εφαρμογής που παρέχονται στον επαγγελματία:

Παρέχεται εμπειρία γρήγορης πληρωμής και χωρίς προβλήματα στον πελάτη.

Λήψη πληρωμών με την εμφάνιση στατικού ή δυναμικού QR κώδικα.

Οι έμποροι, πρέπει να συνδεθούν στην εφαρμογή με έγκυρα διαπιστευτήρια (User ID και Password).

Δυνατότητα χρήσης της επιλογής "Δημιουργία Στατικού ή Δυναμικού QR Code" για τη δημιουργία QR κώδικα.

Ο έμπορος λαμβάνει επιβεβαίωση πληρωμής ως ειδοποίηση στο Bharat QR Merchant App.

Χαρακτηριστικά και Πλεονεκτήματα

Η Axis Bank mVisa λύση πληρωμής με βάση QR code για τους εμπόρους της Axis Bank, προσφέρει έναν εύκολο και ασφαλή τρόπο για να κάνει λήψη ροών πληρωμών από πελάτες. Πρόκειται για μια μοναδική λύση όπου η εφαρμογή γίνεται σημείο πώλησης (Point of Sale - POS) χωρίς να χρειάζεται η παραδοσιακή Point of sale συσκευή.

Δύο τύποι QR κωδικών που μπορούν να δημιουργηθούν από την εφαρμογή είναι:

- Ένας μοναδικός κωδικός QR για κάθε συναλλαγή εισάγωντας το ποσό.
- Ένας γενικός κωδικός QR, ο οποίος μπορεί στη συνέχεια να σαρωθεί από τους πελάτες για να πραγματοποιηθεί η πληρωμή.

Οι mVisa συναλλαγές αρχικοποιούνται από τις ενεργές εφαρμογές του εκδότη (active issuer apps), συνεπώς δεν υπάρχει καμιά προσπάθεια αντιμετώπισης χειρισμών των καρτών / μετρητών. (Καμία ενολητική διαχείρηση).

Ο έμπορος θα δει μειωμένα επιχειρησιακά ζητήματα, καθώς δεν υπάρχει υλικό για να το φροντίσει.

Ειδοποίηση άμεσης πληρωμής στον Πελάτη και τον Έμπορο.

Ο έμπορος δεν χρειάζεται να διευθετήσει τις συναλλαγές mVisa.

Αποδοχή πληρωμής:

Ο πελάτης επιλέγει mVisa για τις ενεργές τραπεζικές εφαρμογές

Ο πελάτης σαρώνει τον QR κωδικό στην οθόνη: αυτοκόλλητο / App / TV / Bill

Ο πελάτης επιβεβαιώνει την πληρωμή

Ο Πελάτης και Έμπορος λαμβάνουν ειδοποίηση άμεσης πληρωμής στις επιτυχημένες συναλλαγές

IV. BHIM – Making India Cashless

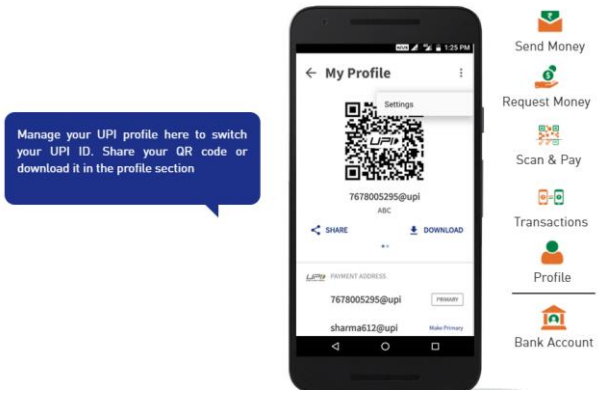
Η Bharat Interface for Money (BHIM), είναι μια εφαρμογή που επιτρέπει την πραγματοποίηση απλών, εύκολων και γρήγορων συναλλαγών χρησιμοποιώντας το Unified Payments Interface (UPI). Ο χρήστης της εφαρμογής, μπορεί εύκολα να κάνει άμεσα τραπεζικές πληρωμές προς οποιονδήποτε χρήστη του UPI χρησιμοποιώντας το αναγνωριστικό ID τους ή σαρώνοντας το δικό τους QR Code με την BHIM εφαρμογή. Επίσης, ο χρήστης, μπορεί εύκολα να αιτηθεί για χρηματικά ποσά διαμέσω της εφαρμογής από το UPI ID (εικ. 11, 12).

Προσπέλαση της BHIM.

- Αρχικά γίνεται λήψη της εφαρμογής από τα App & Play Stores.
- Επιλογή γλώσσας προτίμησης.
- Επιλογή SIM με τον αριθμό τηλεφώνου ο οποίος έχει εγχωρηθεί στην τράπεζα κατά την εγγραφή του χρήστη.
- Πρόσβαση στην εφαρμογή αφού επιλεγεί ένα τετραψήφιο συνθηματικό.
- Επιλογή και σύνδεση με τον επιθυμητό λογαριασμό τράπεζας.
- Διαμόρφωση του προσωπικού UPI Pin παρέχοντας τα τελευταία έξι ψηφία καθώς και την ημερομηνία λήξης της χρεωστικής κάρτας.
- Σε αυτό το σημείο, ο λογαριασμός χρήστη έχει εγχωρηθεί και είναι έτοιμος για χρήση. Από αυτό το σημείο και στην συνέχεια, μπορεί να επιχειρηθεί αποστολή ή/και λήψη χρήματων χωρίς παρουσία μετρητών.


Make payments quicker by scanning the QR of the person you want to send money to. Share your QR to let others make payments to you as well!
Scan & Pay
Scanning QR Code
Send Money
Request Money
Scan & Pay
Transactions
Profile
Bank Account

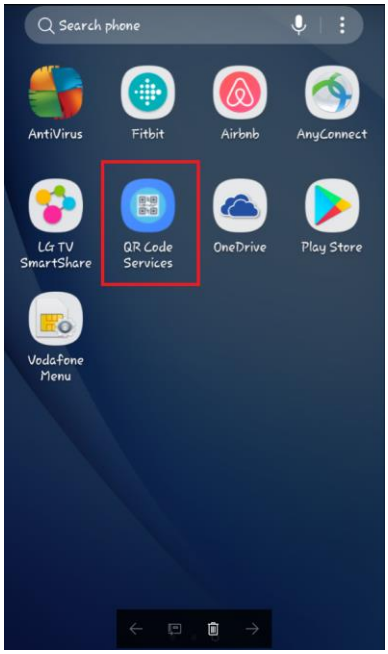
(εικ. 11)


Manage your UPI profile here to switch your UPI ID. Share your QR code or download it in the profile section
My Profile
Settings
7678005295@upi
ABC
SHARE
DOWNLOAD
PAYMENT ADDRESS
7678005295@upi
PRIMARY
sharma612@upi
Make Primary
Send Money
Request Money
Scan & Pay
Transactions
Profile
Bank Account

(εικ. 12)

4. Παρουσίαση και χρήση εφαρμογής (User manual).

Η εφαρμογή, ενεργοποιείται με το πάτημα του σκιαγραφόμενου παρακάτω εικονιδίου (εικ. 13).



(εικ. 13)

Η πρώτη οθόνη της εφαρμογής που συναντά ο χρήστης της, φαίνεται στην (εικ. 14).

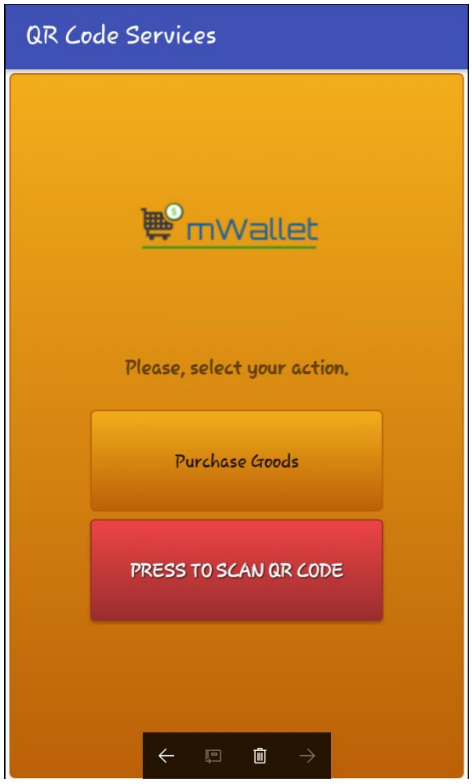
Στην εικόνα αυτή, υπάρχουν δύο πλήκτρα, το πρώτο αφορά τις επιλογές της εφαρμογής. Αν πιέσουμε το πλήκτρο αυτό, θα αναρτηθεί ένα μενού επιλογών όπως φαίνεται στην (εικ. 15).

Σαν επικεφαλίδα του μενού, εμφανίζεται ένα μήνυμα προτροπής του χρήστη ο οποίος καλείται να επιλέξει μια από τις προεπιλογές. Οι δε επιλογές, έχουν ως εξής:

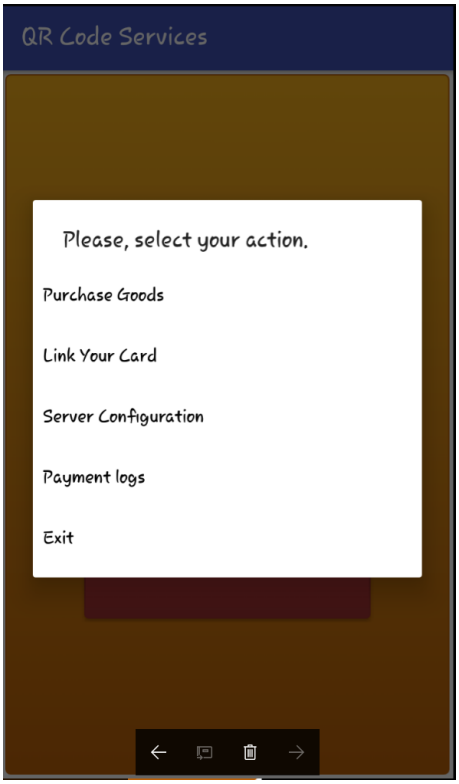
1. Purchase Goods. Ο σκοπός της συγκεκριμένης επιλογής, είναι να δίνει την δυνατότητα στον χρήστη της εφαρμογής, να προβεί σε κάποια αγορά προϊόντος. Επιλέγοντάς την, θα οδηγούσε τον χρήστη στην ενεργοποίηση της κάμερας της συσκευής η οποία με τη σειρά της, θα σάρωνε αυτόματα το QR Code του προϊόντος και θα αποκωδικοποιούσε τις πληροφορίες της. Ωστόσο, η λειτουργία της, δεν υλοποιήθηκε. Αντί αυτού, και για τη λειτουργία αυτή, υπάρχει το δεύτερο πλήκτρο της οθόνης της (εικ. 14).
2. Link Your Card. Η επιλογή της, είχε σκοπό να δώσει την δυνατότητα στον χρήστη να μπορέσει να μεταφερθεί σε μια ειδική οθόνη της φόρμας η οποία θα κατέγραφε και θα συνέδεε τις διαθέσιμες χρεωστικές, πιστωτικές και IBAN λογαριασμο/ύς του με τον προσωπικό τραπεζικό λογαριασμό του. Επίσης, στην οθόνη αυτή, θα μπορούσε επίσης, να τροποποιεί τα παραπάνω στοιχεία. Η δυνατότητα αυτή, επίσης δεν προχώρησε σε υλοποίηση, για λόγους απλούστευσης της διαδικασίας. Σε μια όμως, επαγγελματική έκδοση

της εφαρμογής, εννοείται πως κρίνεται απαραίτητη η πλήρη ανάπτυξή της. Στην υλοποιημένη παρούσα εφαρμογή, τα στοιχεία αυτά είναι διαθέσιμα και λειτουργήσιμα μέσω xml αρχείων.

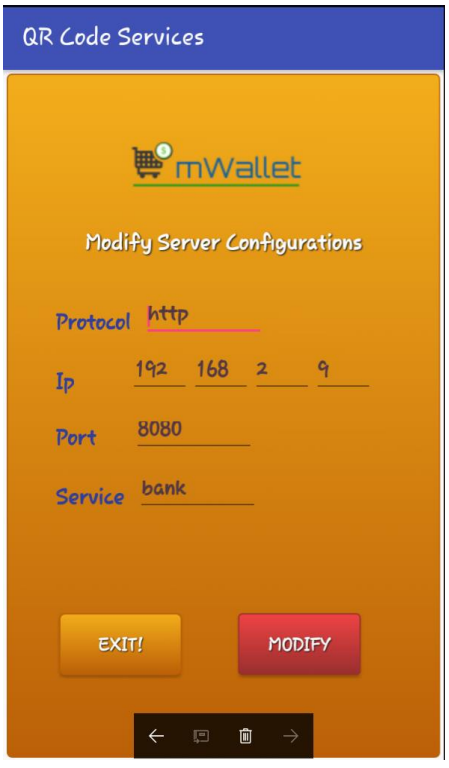
3. Server Configuration. Η επιλογή της, έχει σκοπό την διαμόρφωση των παραμέτρων του Server (εικ. 18, 19). Παράμετροι όπως το Host (IP), η υπηρεσία (Service) και η θύρα (port), δημιουργούν μαζί, την διεύθυνση (url) η οποία δείχνει την τοποθεσία (location) που βρίσκεται τοποθετημένο το λογισμικό το οποίο προσομειώνει την τράπεζα (εικ.34). Το Server Side το οποίο αποτελεί το δεύτερο μέρος του λογισμικού της εφαρμογής, μπορεί να είναι τοποθετημένο οπουδήποτε στον ιστό, είτε τοπικά, είτε απομακρυσμένα. Αυτή η δυνατότητα ακριβώς, καθορίζεται με την Server Configuration επιλογή σε αυτό το σημείο. Μετά την εγγραφή μιας διεύθυνσης (url) ή την ενημέρωσή της, οι επιλογές αυτές μπορούν να αποθηκεύονται πατώντας το πλήκτρο ‘MODIFY’. Τέλος, υπάρχει η δυνατότητα επιστροφής στην αρχική οθόνη της εφαρμογής, πατώντας το πλήκτρο ‘EXIT’.
4. Payment logs. Είναι μια επιλογή η οποία αν προτιμηθεί, οδηγεί σε μια άλλη οθόνη η οποία σε πραγματικό χρόνο κάνει ανάκτηση μιας λίστας με ιστορικά δεδομένα αγορών που έχουν πραγματοποιηθεί κατά καιρούς από μια βάση on cloud (firebase) της Google. Η λίστα, είναι χρονολογικά ταξινομημένη από την πιο πρόσφατη στην πιο παλαιά αγορά/ές που έχουν πραγματοποιηθεί. Επίσης, κάθε λίστα, ομαδοποιείται ανά ημέρα και κάθε εσωτερική εγγραφή της, ομαδοποιείται με αγορές που αφορούν την εν λόγω ημέρα που επίσης ταξινομείται χρονικά. Τέλος, κάθε επικεφαλίδα της λίστας, εκτός της ημερομηνίας, φέρει πάνω της επιπλέον πληροφορία η οποία δείχνει τον συνολικό αριθμό των συναλλαγών που έχουν λάβει χώρα την συγκεκριμένη μέρα αλλά, και τον αριθμό των επιτυχημένων και αποτυχημένων συναλλαγών, αφήνοντας έτσι τον χρήστη, ενήμερο για την κατάσταση με την οποία πρόκειται να έρθει αντιμέτωπος, αν αποφασίσει να δει τα στοιχεία της λίστας για περισσότερες λεπτομέρειες (εικ. 20, 21).
5. Exit. Η επιλογή αυτή δίνει την δυνατότητα στον χρήστη, μετά απο την απαραίτητη επιβεβαίωσή του (εικ. 22), να εγκαταλήψει ή όχι την εφαρμογή.



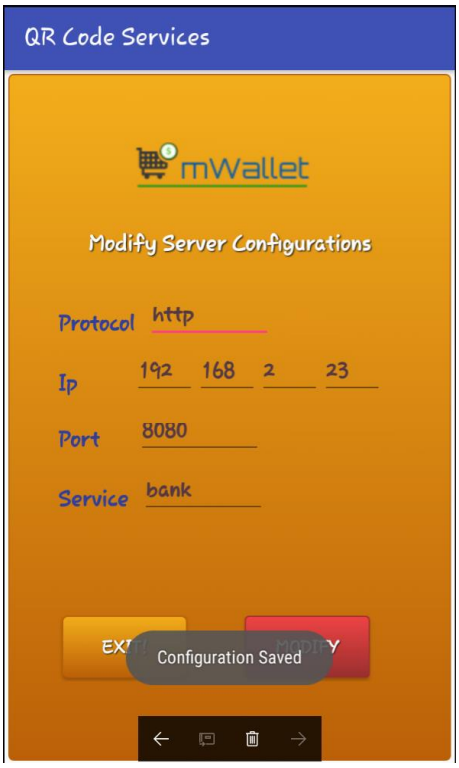
(εικ. 14).



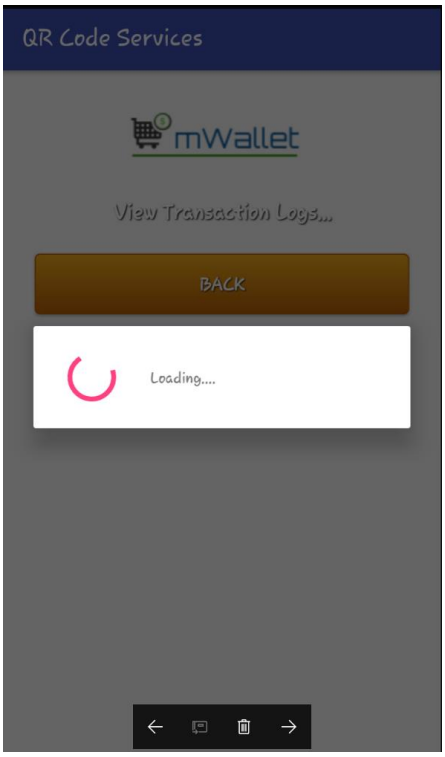
(εικ. 15)



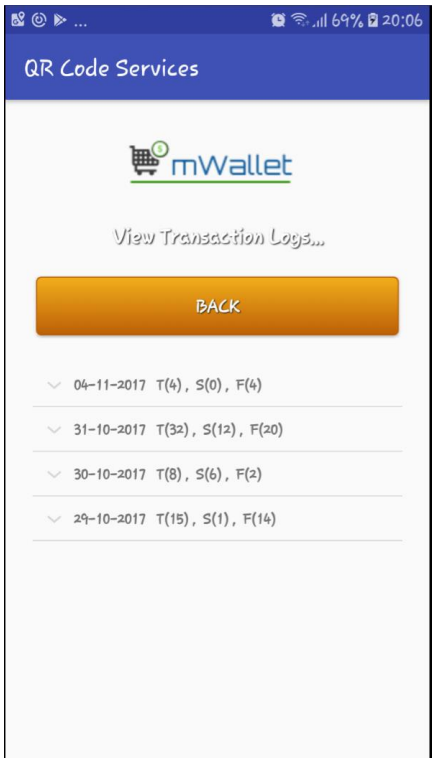
(εικ. 16)



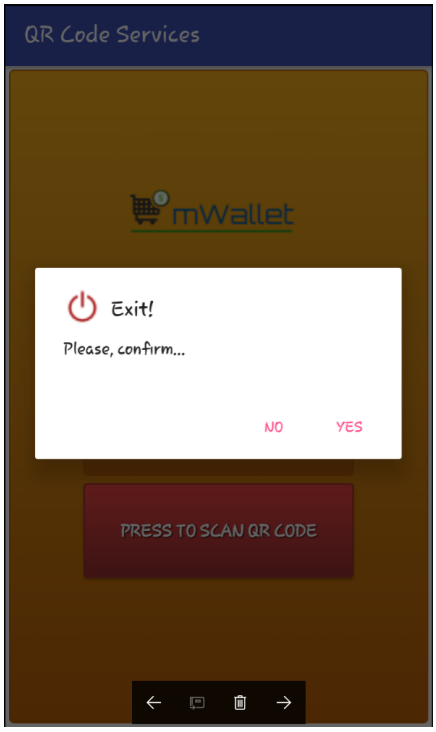
(εικ. 17)



(εικ. 18)

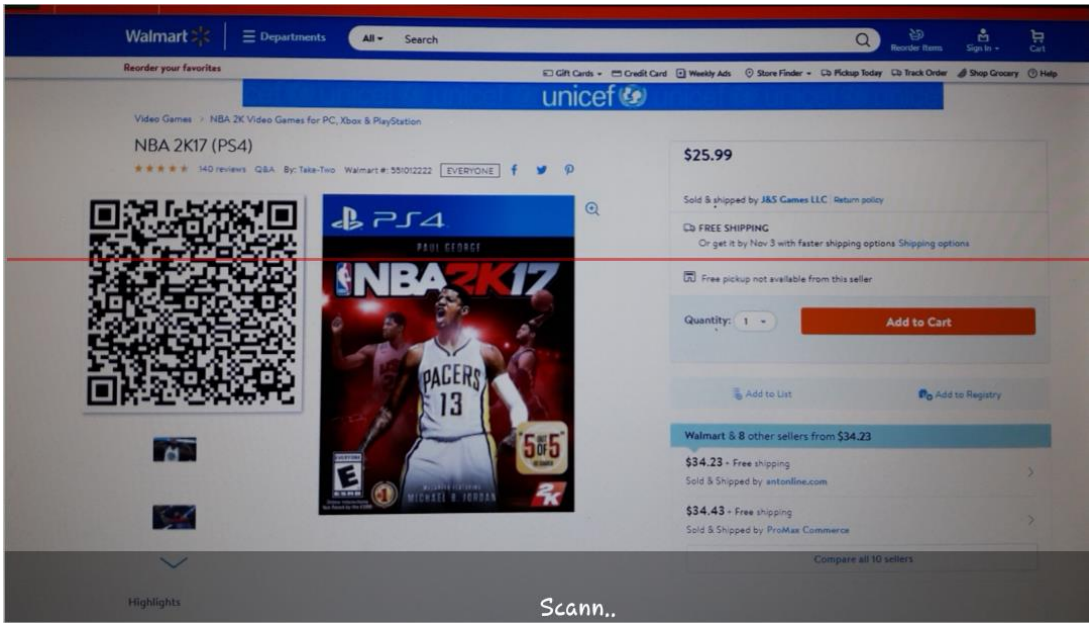


(εικ. 19)



(εικ. 20)

Στη συνέχεια, αν από την (εικ. 14) πατηθεί το πλήκτρο ‘PRESS TO SCAN QR CODE’, τότε αυτό που ακολουθεί είναι να ενεργοποιηθεί η κάμερα της συσκευής, η οποία εστιάζει αυτόματα, προκειμένου να εντοπίσει το QR Code, προσπαθώντας να την διαβάσει και να την αποκωδικοποιήσει. Το QR Code, εμπεριέχει πληροφορία για το προϊόν αλλά και τον επιτηδευματία, σε κωδικοποιημένη μορφή.
Στην παρακάτω εικόνα (εικ. 21), έχει σαρωθεί ένα QR Code ενός παιχνιδιού PS4, το οποίο βρίσκεται σε ιστοσελίδα της εταιρείας Walmart, μετά το πάτημα του προαναφερθέντος πλήκτρου.
Μετά την σάρωση, δηλαδή της αποκωδικοποιήσης της QR Code, η εφαρμογή προβάλει μια οθόνη με την αποκωδικοποιημένη πληροφορία που εξάγεται από την QR Code σήμανση (εικ. 22).

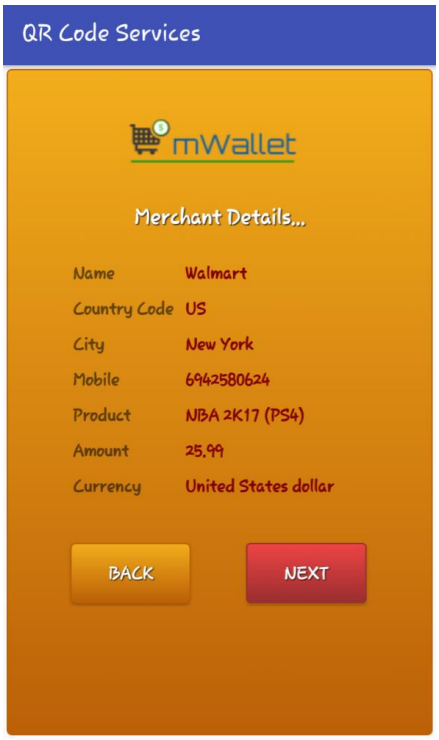


(εικ. 21)

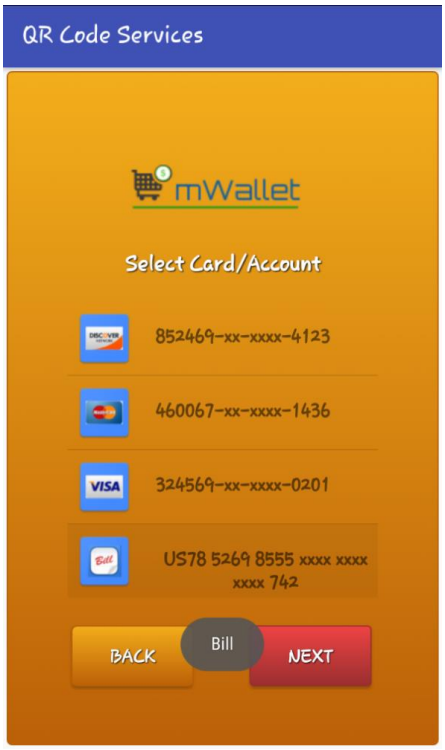
Πατώντας το πλήκτρο NEXT, ο χρήστης μεταφέρεται στην οθόνη (εικ. 23) στην οποία καλείται να επιλέξει μια εκ των τεσσάρων μεθόδων πληρωμής, δηλαδή είτε μία από τις χρεωστικές κάρτες, είτε από το λογαριασμό IBAN που προβάλλονται ως επιλογές και έχουν καταχωρηθεί και συνδεθεί με τον τραπεζικό του λογαριασμό, σε προηγούμενο στάδιο. Κατόπιν αυτού, από την (εικ.23), ο χρήστης έχει επιλέξει το IBAN, το οποίο έχει ελαφρώς μια σκίαση, προκειμένου να τονίσει οπτικά την επιλογή καθώς και ένα πτυσσόμενο μήνυμα με την ένδειξη Bill.

Το επόμενο στάδιο και αφού πατηθεί το πλήκτρο NEXT, είναι να καταχωρήσει το κρυφό και προσωπικό τετραψήφιο αριθμητικό κωδικό Pin. Ο αριθμός Pin, θα πρέπει να αποτελείται από τέσσερα αριθμητικά ψηφία και να έχει εκδοθεί από την τράπεζά του, ως είθισται. Σε περίπτωση που ο χρήστης δώσει εσφαλμένα κάποιο κωδικό που δεν πληρεί τους περιορισμούς, όπως κάποιο γράμμα, σχετικό μήνυμα προβάλλεται και η εφαρμογή εστιάζει στο συγκεκριμένο πεδίο

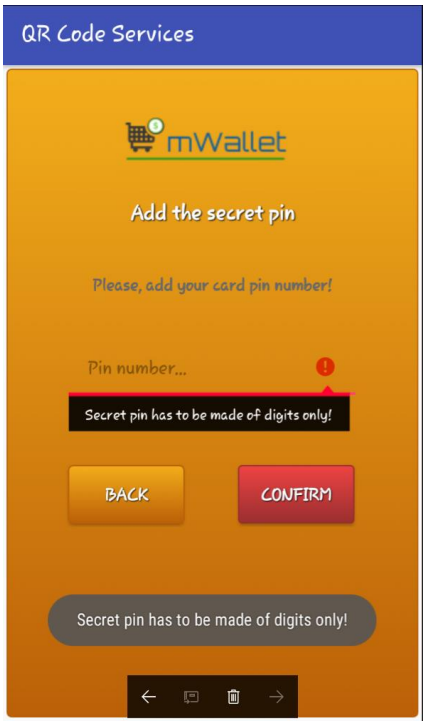
έως ότου ο χρήστης δώσει έναν κωδικό που συμμορφώνεται με τους περιορισμούς που το σύστημα έχει θέσει (εικ. 25).



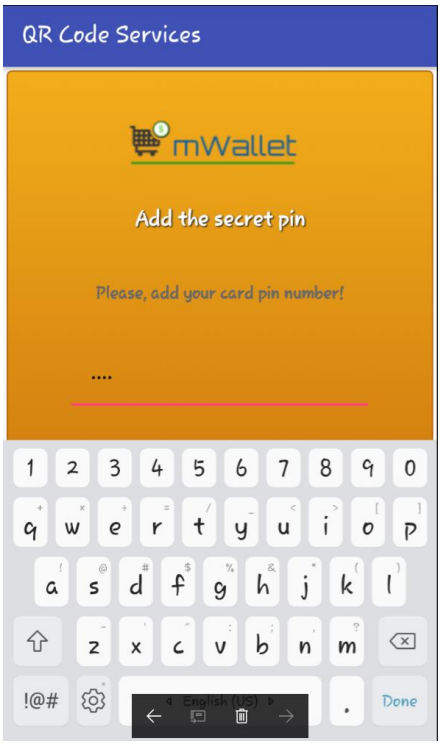
(εικ. 22).



(εικ. 23)



(εικ. 24).



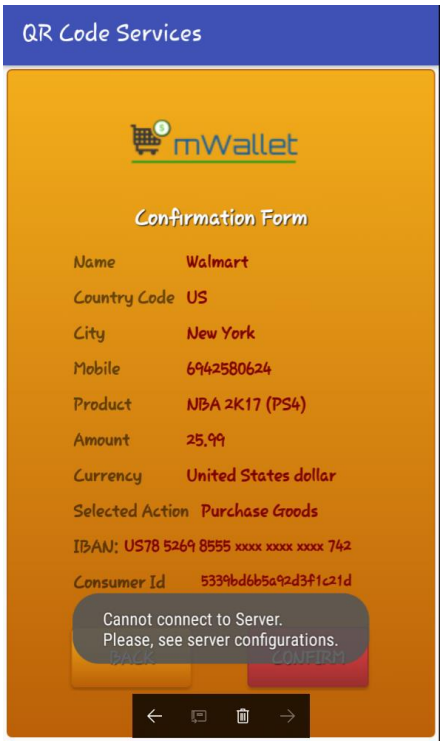
(εικ. 25)

Αφού πρώτα δώσει κάποιο σωστό Pin, ο χρήστης έρχεται αντιμέτωπος με τις (εικ. 24, 25). Η οθόνη αυτή είναι το επόμενο επίπεδο η οποία παρέχει στον χρήστη περισσότερη λεπτομέρεια για την επικείμενη αγορά. Ο χρήστης έχει την δυνατότητα να αποδεχθεί ή να απορρίψει την διαδικασία σε αυτό το στάδιο, ανάλογα με την ενέργεια την οποία επιλέγει. Αν επιλεχθεί BACK, ο χρήστης μεταφέρεται σε προηγούμενη οθόνη αλλιώς, αν πατήσει το NEXT, δίνει ουσιαστικά την συγκατάθεσή του να προβεί στην ολοκλήρωση της αγοράς. Αφού επιλεγεί το τελευταίο, ο χρήστης θα βρεθεί σε δύο καταστάσεις. Πρώτον, είτε να έχει θετική κατάληξη (εικ. 26) είτε, δεύτερον, αρνητική (εικ. 27). Σε περίπτωση που δεν έχει διαμορφωθεί σωστά την διεύθυνση (url) στην οθονη Server Configuration (εικ. 16, 17), ή/και η συσκευή δεν έχει άδεια (permissions) για πρόσβαση στο δίκτυο, η εφαρμογή θα ενημερώσει τον χρήστη για το συμβάν με ανάλογο και σχετικό μήνυμα. Κάποια μηνύματα εξ' αυτών, μπορούν να παρατηρηθούν στις ακόλουθες εικόνες (εικ. 26, 27).

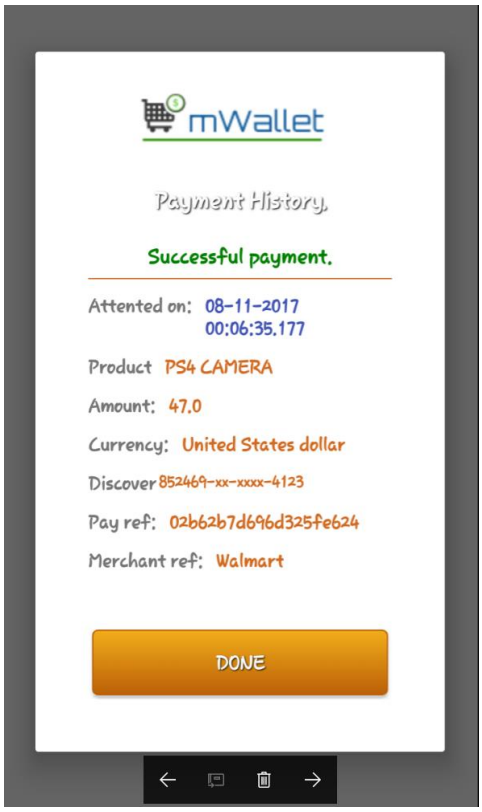
Η εφαρμογή μπορεί να συνδεθεί στο δίκτυο με ασύρματη ζεύξη wi-fi, είτε μέσω δικτύου κινητής τηλεφωνίας. Και στις δύο περιπτώσεις, η εφαρμογή δουλεύει χωρίς πρόβλημα και αντίστοιχο μήνυμα προβάλει τον τρόπο σύνδεσης.

Σε περίπτωση που όλα βαίνουν καλώς, έως το σημείο αυτό, τότε, η εφαρμογή ολοκληρώνει την αγορά και αντίστοιχη οθόνη (εικ. 28), προβάλλει τις λεπτομέρειες.

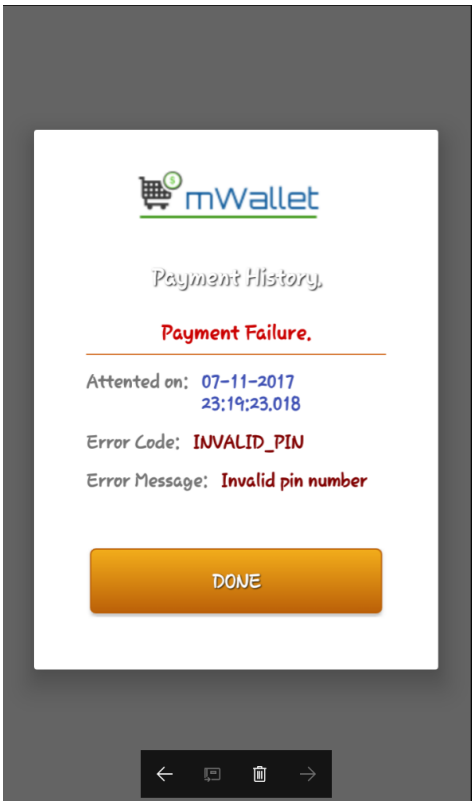
(εικ. 26)



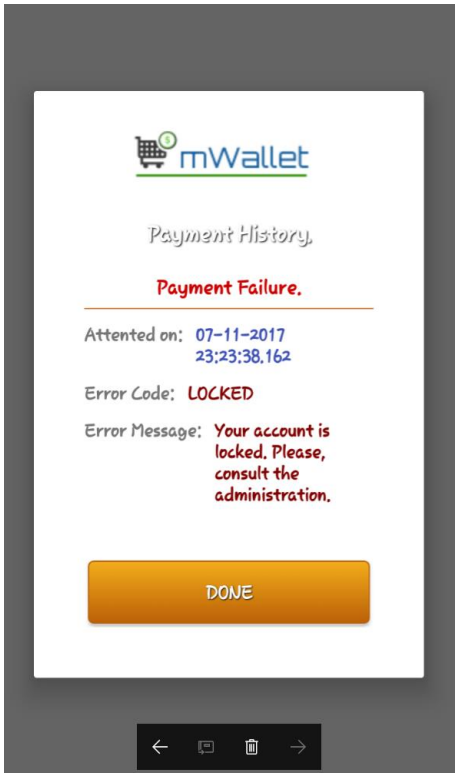
(εικ. 27)


mWallet
Payment History.
Successful payment.
Attented on: 08-11-2017
00:06:35.177
Product PS4 CAMERA
Amount: 47.0
Currency: United States dollar
Discover 852469-xx-xxxx-4123
Pay ref: 02b62b7d696d325fe624
Merchant ref: Walmart
DONE

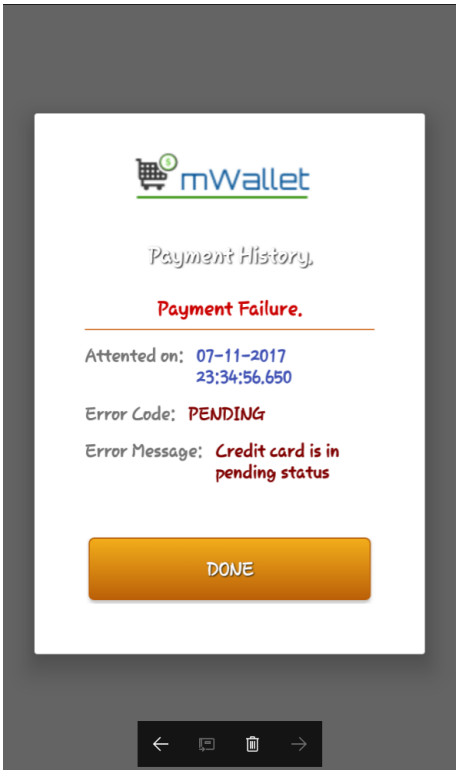
(εικ. 28)


mWallet
Payment History.
Payment Failure.
Attented on: 07-11-2017
23:19:23.018
Error Code: INVALID_PIN
Error Message: Invalid pin number
DONE

(εικ. 29)



(εικ. 30)



(εικ. 31)

χρωματισμούς η οποία τονίζει και οπτικά την θετική εξέλιξη τις επιχειρούμενης δράσης, σε αντίθεση με τους κόκκινους χρωματισμούς που συμβαίνουν στις περιπτώσεις αποτυχημένων συναλλαγών. Και στις επιτυχημένες αλλά και στις αποτυχημένες συναλλαγές, υπάρχει κάτι κοινό, το οποίο είναι η χρονική στιγμή στην οποία επιχειρήθηκε η συναλλαγή, απαραίτητο στοιχείο για μεταγενέστερες αναζητήσεις στη βάση δεδομένων, είτε για ιστορικά δεδομένα, τα οποία θα αναφερθούν παρακάτω, είτε για περαιτέρω επεξεργασία τους και πιθανή ανάλυση διαγραμμάτων.

Οι οθόνες των εικόνων (28-31), παραθέτουν ορισμένες καταστάσεις οι οποίες δείχνουν μέρος των επιτυχημένων και αποτυχημένων συναλλαγών. Η (εικ. 28), παρουσιάζει την εκδοχή της θετικής έκβασης, ενώ οι (εικ. 29 – 31), δείχνουν ορισμένα σφάλματα που έχουν συμβεί κατά την/τις προσπάθεια/ες των αγορών.

Στην περίπτωση της (εικ. 28), παρατηρείται πώς μια επιτυχημένη αγορά, συνοδεύεται από ένα αρχικό Label σε πράσινο χρώμα.

Ενώ μια επιτυχημέννη συναλλαγή αναγράφει το όνομα του προϊόντος, την αξία αυτού, η οποία αφαιρείται από τον πελάτη και προστίθεται στον επιτηδευματία, το συνάλλαγμα, την μέθοδο η οποία χρησιμοποιήθηκε για την αγορά, για παράδειγμα, κάρτα ή το IBAN λογαριασμού, την επωνυμία του καταστήματος, καθώς επίσης και ένα αναγνωριστικό της συγκεκριμένης συναλλαγής, μια αποτυχημένη συναλλαγή έχει κυρίως προβολή και επεξήγηση κωδικών και μηνυμάτων σφαλμάτων τα οποία ερμηνεύουν τα σφάλματα που έλαβαν χώρα κατά τη διαδικασία.

Σφάλματα τέτοιου είδους αποτελούν τα λεγόμενα Business Errors, τα οποία καταδεικνύουν το business logic της εφαρμογής. Για παράδειγμα, κωδικός σφάλματος INVALID_PIN (εικ. 29), ενημερώνει τον χρήστη για καταχώρηση εσφαλμένου κωδικού Pin. Ο τελευταίος, έχοντας δει τον κωδικό σφάλματος και το συνοδευτικό μήνυμα, είναι σε θέση να κατανοήσει για ποιο λόγο απέτυχε η συγκεκριμέννη συναλλαγή και προσαρμόζει ανάλογα την επόμενη ενέργειά του.

Αξίζει να αναφερθεί ο κωδικός σφάλματος LOCKED. Ο κωδικός σφάλματος LOCKED, εμφανίζεται κατά την περίπτωση που ο χρήστης επανειλημμένα έδωσε λάθος κωδικό Pin. Τέτοιου είδος σφάλμα, συμβαίνει διότι ο χρήστης ενδέχεται να έδωσε επανειλημμένα λάθος κωδικό Pin. Η παρούσα υλοποίηση, θέτει ένα ανώτατο όριο συνεχόμενων εσφαλμένων προσπαθειών έως και τρεις φορές. Αν κάποιος χρήστης δίνει είτε απο βιασύνη, είτε επειδή δεν θυμάται, τρεις συνεχόμενες φορές λάθος Pin, τότε ο χρήστης τίθεται αυτομάτως εκτός δυνατότητας να μπορεί να είναι σε θέση να κάνει κάτι ώστε να αποκαταστήσει το σφάλμα. Αντί αυτού, μπορει μόνο να δει τον Admnistrator του συστήματος, προφανώς την τράπεζα, προκειμένου να αποκαταστήσει ξανά την πρόσβαση.

Τέλος, ένα επόμενο σφάλμα που συνήθως συμβαίνει σε συναλλαγές, είναι αυτά της ανεπάρκειας υπολοίπου ικανού για την ολοκλήρωση μιας αγοράς. Το σφάλμα αυτό, εκτός τον κωδικό και το μήνυμα σφάλματος (error code, error message), επεκτείνει επιπλέον το μήνυμα σφάλματος, παραθέτοντας σε σύγκριση το κόστος του προϊόντος από τη μια πλευρά, με το ισοζύγιο τραπεζικού υπολοίπου από την άλλη, ενημερώνοντας έτσι τον χρήστη για αποφυγή σύγχυσής του.

6. Αρχιτεκτονική συστήματος

Η εφαρμογή, ως γνωστόν, έχει σχεδιαστεί και υλοποιηθεί σε δύο ξεχωριστά και αυτόνομα μέρη. Το ένα μέρος αφορά το Android λογισμικό και λειτουργεί στην συσκευή του χρήστη, ενώ το άλλο μέρος αφορά το λογισμικό που προσομοιώνει την λειτουργία κάποιας τράπεζας.

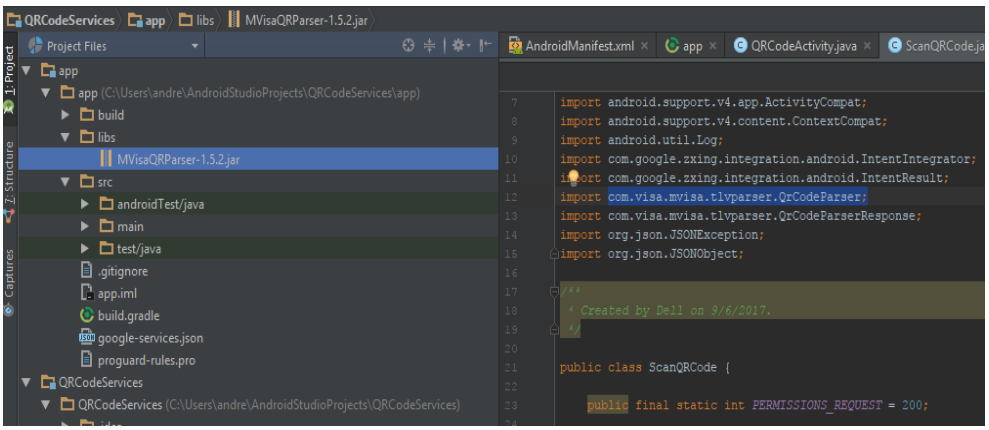
Όσον αφορά το πρώτο μέρος του λογισμικού, για την επίτευξη του αποτελέσματος, η εφαρμογή κάνει χρήση του Android SDK καθώς και third-parts βιβλιοθηκών. Μια σημαντική τέτοια βιβλιοθήκη είναι η XZING Core, η οποία διαθέτει τις κεντρικές (core) λειτουργίες για κωδικοποίηση/αποκωδικοποίηση των barcodes. Πρέπει να τονιστεί ότι γίνεται χρήση των binary distribution της παραπάνω βιβλιοθήκης αντί του jar αρχείου. Αυτό δηλώνεται στα dependencies της gradle με το @aar extension.

Επίσης, θα πρέπει να αναφερθεί άλλη μια third-part βιβλιοθήκη MVisaQRParser-1.5.2.jar, (εικόνα 32), της MasterCardVisa Parser. Όπως γίνεται εύκολα αντιληπτό, η βιβλιοθήκη αυτή προσφέρει λειτουργίες ανάγνωσης και μετατροπής των QR Code σημάνσεων σε πληροφορία η οποία είναι αναγνώσιμη από τον άνθρωπο.

Η τελευταία βιβλιοθήκη, προσαρτήθηκε στην εφαρμογή, για έναν και σημαντικό λόγο: διότι, χρησιμοποιείται για παραγωγή της QR Code σήμανσης που κάνει χρήση η εφαρμογή. Η ίδια η Mastercard διαθέτει ένα σχετικό website για την παραγωγή της παραπάνω QR Code(*). Η τελευταία βιβλιοθήκη, γνωρίζοντας πλέον τον τρόπο που παράγει το QR Code, γνωρίζει επίσης και τον τρόπο πώς να την διαβάσει και να την αποκωδικοποιήσει.

Οι προαναφερθείσες βιβλιοθήκες, απαιτούν την παραχώρηση αδειών (permissions) στο manifest αρχείο, για πρόσβαση και χρήση της κάμερας της συσκευής.

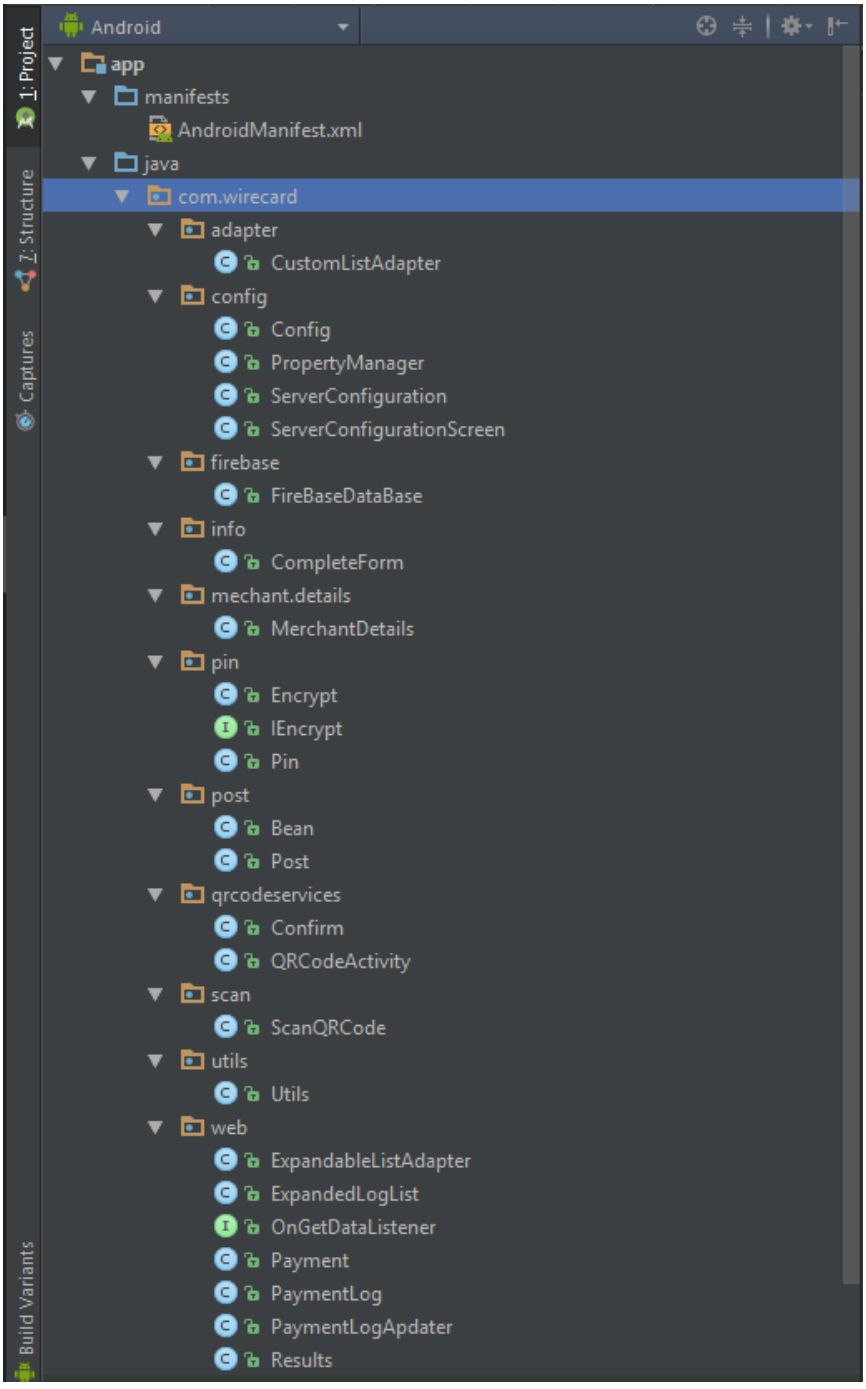
Μια άλλη βιβλιοθήκη, η οποία έχει προστεθεί, είναι αυτή της firebase toolkit. Η βιβλιοθήκη αυτή, (Firebase), είναι μια εργαλειοθήκη η οποία αντικαθιστά αρκετές υπάρχουσες τεχνολογίες της Google για προγραμματιστές και προσθέτει πολλές νέες υπηρεσίες.



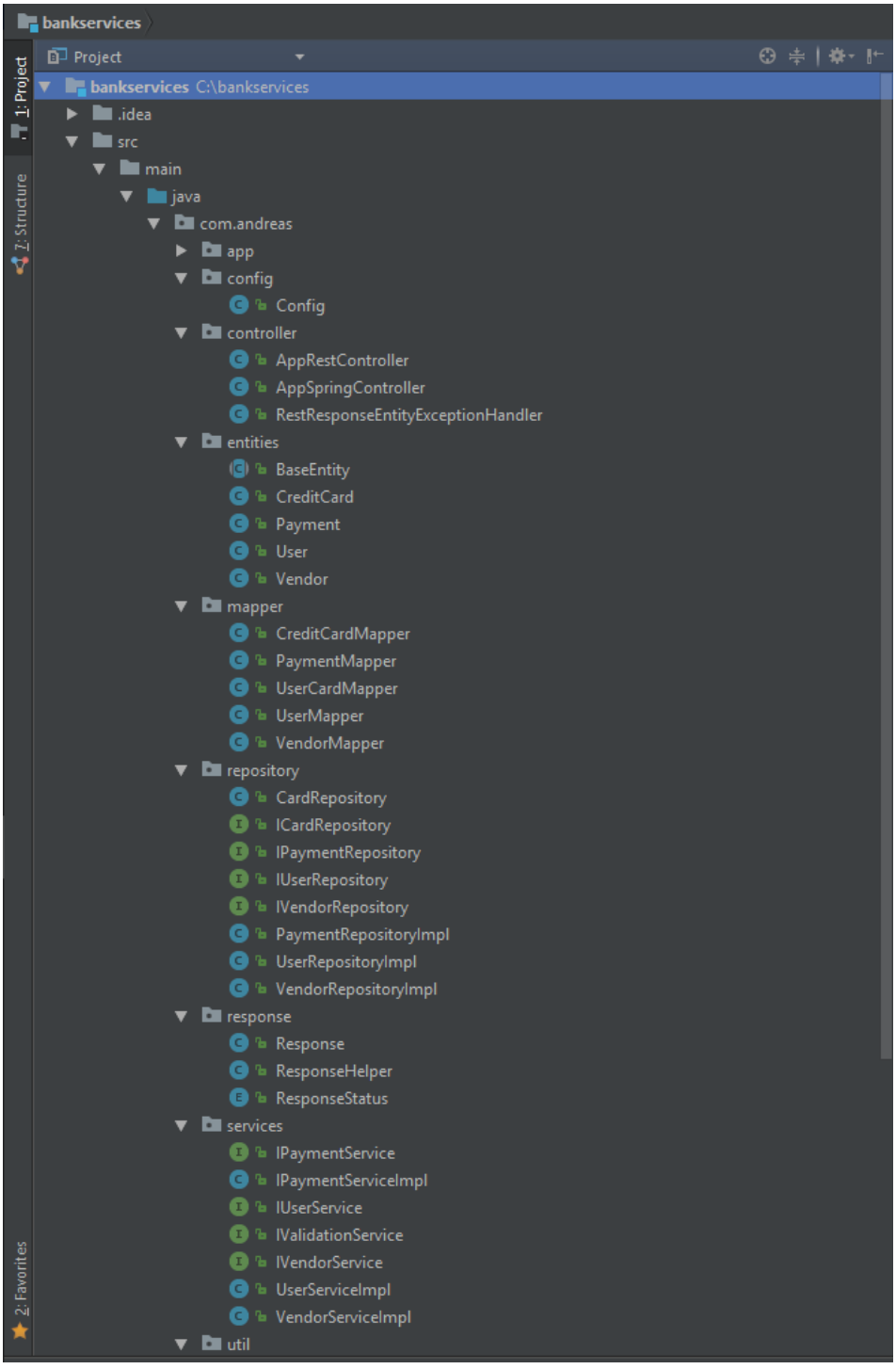
(εικ. 32)

Η δομή της εφαρμογής, όσον αφορά το πρώτο μέρος της, αυτό του Android λογισμικού, μπορεί να παρατηρηθεί παρακάτω (εικ. 33), ενώ το δεύτερο μέρος του λογισμικού, που αφορά την τράπεζα, φαίνεται στην επόμενη εικόνα (εικ. 34).

(* https://www.mastercardlabs.com/masterpass-qr/).



(εικ. 33)


bankservices
Project
bankservices C:\bankservices
.idea
src
main
java
com.andreas
app
config
Config
controller
AppRestController
AppSpringController
RestResponseEntityExceptionHandler
entities
BaseEntity
CreditCard
Payment
User
Vendor
mapper
CreditCardMapper
PaymentMapper
UserCardMapper
UserMapper
VendorMapper
repository
CardRepository
ICardRepository
IPaymentRepository
IUserRepository
IVendorRepository
PaymentRepositoryImpl
UserRepositoryImpl
VendorRepositoryImpl
response
Response
ResponseHelper
ResponseStatus
services
IPaymentService
IPaymentServiceImpl
IUserService
IValidationService
IVendorService
UserServiceImpl
VendorServiceImpl
util
1: Project
2: Structure
2: Favorites

(εικ. 34)

6.1. Πρώτο μέρος της εφαρμογής (Android Software Development).

Η τοποθεσία, (host, port, ip και service - εικ. 16, 17), η οποία αποτελεί την διεύθυνση που βρίσκεται η τράπεζα και οι υπηρεσίες της, ρυθμίζονται από τα activities: QRCodeActivity, ServerConfigurationScreen και το ServerConfiguration class. Το τελευταίο, λειτουργεί ως μια βοηθητική κλάση η οποία μεταξύ άλλων, διαθέτει τις μεθόδους για την αποθήκευση των προαναφερθέντων πληροφοριών στη συσκευή, μέσω των SharedPreferences, αλλά και για την δημιουργία της διεύθυνσης (url) στην πλήρη μορφή του: http://host:port/service.

Αρχικά, ρυθμίζεται η ServerConfiguration οθόνη δίνοντας τα σωστά επιμέρους συστατικά στοιχεία της διεύθυνσης (url). Στη συνέχεια, μέσω του κόκκινου πλήκτρου (εικ. 35-37), η εφαρμογή πρώτα επιβεβαιώνει ότι διαθέτει permissions πρόσβασης στην κάμερα της συσκευής και μετά ο έλεγχος της εφαρμογής μεταβαίνει στην ScanQRCode activity. Το τελευταίο, ενεργοποιεί την κάμερα και στη συνέχεια, αποκωδικοποιεί την QR Code σήμανση. Διαβάζει, εξάγει και αναλύει τα κωδικοποιημένα μηνύματα που αφορούν τις λεπτομέρειες για το προϊόν αλλά και τον επιτηδευματία.

```
public void requestIfNeedCameraPermissions() {
    if (ContextCompat.checkSelfPermission(getActivity(), Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED) {
      ActivityCompat.requestPermissions(getActivity(), new String[]{Manifest.permission.CAMERA}, PERMISSIONS_REQUEST);
    }
}
```

(εικ 35)

```
scanQRCode = new ScanQRCode(activity);
scanQRCode.requestIfNeedCameraPermissions();
pressToScan.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        scanQRCode.initScanner(activity);
    }
});
```

(εικ 36)

```
public void initScanner(Activity activity ){
    IntentIntegrator integrator = new IntentIntegrator(activity);
    integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES);
    integrator.setPrompt("Scanning..");
    integrator.setCameraId(0);
    integrator.setBeepEnabled(false);
    integrator.initiateScan();
}
```

(εικ 37)

Εν συνεχεία, το αποτέλεσμα της σάρωσης, το οποίο είναι τύπου String, μετατρέπεται σε JSON Object μέσω της προκαθορισμένης λειτουργίας που παρέχει η προσαρτημένη βιβλιοθήκη, απ'όπου γίνεται η ανάκτηση μόνο των προεπιλεγμένων επιθυμητών πληροφοριών.

Το αποτέλεσμα της σάρωσης και η πληροφορία που διαβάζεται από την QR Code σήμανση, έχει την παρακάτω μορφή:

Scan Content:

00020101021204154600678934521435204520453038405404 47.05802US5907Walmart6008New York62860126US5468464646844684864864680210694258062403 05A6008040300007084578431211 10PS4 CAMERA63049648

Μετά την εξαγωγή της παραπάνω πληροφορίας, η εφαρμογή την μετατρέπει σε JSON object όπου το περιεχόμενό του, προβάλλει παρακάτω (εικ. 38):


```
{
 "qrCodeData":{
     "payloadFormatIndicator":"01",
     "pointOfInitiation":"12",
     "masterCardPan1":"460067893452143",
     "merchantCategoryCode":"5204",
     "currencyCode":"840",
     "transactionAmount":"47.0",
     "countryCode":"US",
     "merchantName":"Walmart",
     "cityName":"New York",
     "billId":"US5468464646844684864864 68",
     "mobileNumber":"6942580624",
     "storeId":"A6008",
     "loyaltyNumber":"000",
     "terminalId":"45784312",
     "addDataMasterCard1":"PS4 CAMERA",
     "crc":"9648",
     "isPrimaryIdMandatory":false,
     "isSecondaryIdMandatory":false,
     "isBillIdMandatory":false,
     "isMobileNumberMandatory":false,
     "isStoreIdMandatory":false,
     "isLoyaltyNumberMandatory":false,
     "isReferenceIdMandatory":false,
     "isConsumerIdMandatory":false,
     "isTerminalIdMandatory":false,
     "isPurposeMandatory":false,
     "isAddDataMasterCard1Mandatory":false,
     "isAddDataMasterCard2Mandatory":false,
     "isAddDataNpci1Mandatory":false,
     "isAddDataNpci2Mandatory":false

 },
 "qrCodeError":[
 ]}
```


(εικ. 38)

Στο ίδιο Activity, δημιουργείται για πρώτη φορά ένα Bundle το οποίο θα χρησιμοποιηθεί ως ένας container στην μετάδοση δεδομένων μεταξύ διαφόρων δραστηριοτήτων (Activities) του Android διαμέσω των Intents.

Θα πρέπει να επισημανθεί ότι εάν ο χρήστης καλείτο να επιλέξει το action (εικ. 15), που επιθυμούσε να κάνει και μετά από την σχετική επιλογή, τότε μόνο η εφαρμογή θα επέτρεπε την συνέχεια της διαδικασίας. Δηλαδή, η σάρωση ενός QR Code, με σκοπό την αγορά ενός προϊόντος, θα επιτρεπόταν αν ο χρήστης δήλωνε την επιλογή του 'purchase goods' (εικ 15). Αυτή η δυνατότητα μεταφέρθηκε όμως σε ένα πλήκτρο στην πρώτη οθόνη (εικ. 14) για λόγους απλοποίησης. Παρόλα αυτά, στο παραπάνω bundle, προσαρτάται η επιλογή χρήστη ή η πρόθεσή του δηλαδή το 'purchase goods', όταν αυτό επιλεχθεί από το drop-down της πρώτης οθόνης για λόγους πληρότητας του request (εικ. 15).

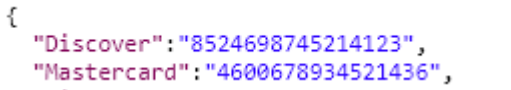
Η παραπάνω επιλογή, προσαρτάται στο προαναφερθέν Bundle μαζί με το JSON Object που κατέχει το qrCodeData και JSON τύπου.

Μετά την σάρωση της QR Code σήμανσης, και τα προαναφερθέντα, ο χρήστης, οδηγείται σε μια επόμενη οθόνη, αυτής του Merchant Details. (εικ. 22).

Όπως ο τίτλος της οθόνης προϊδεάζει, στην οθόνη αυτή, αναμένεται να προβληθούν στοιχεία σχετικά με τον επιτηδευματία (Merchant). Επιπλέον αυτής, προβάλλονται και στοιχεία σχετικά με την τιμή και την ονομασία του προϊόντος. Η οθόνη με τις λεπτομέρειες του Merchant, διαχειρίζεται από το activity MerchantDetails και το activity_merchant_detail.xml αρχείο.

Στο Merchant Details Activity, στο Bundle το οποίο καταφθάνει από το QRCodeActivity activity, προστίθενται στο φορτίο της και ο λογαριασμός (κάρτα ή το IBAN του χρήστη - εικονα 39), το billId του επιτηδευματία, το οποίο χρησιμοποιείται από το δεύτερο μέρος της εφαρμογής (το λογισμικό της τράπεζας), προκειμένου να συναθροιστούν τα εμβάσματα από διάφορες ροές οι οποίες θα προέρχονται από τις πωλήσεις προϊόντων. Επίσης προστίθεται και ένα request το οποίο φιλοδοξεί να αναπτύσσεται στην πορεία της διαδικασίας, καθώς περνά από τη μία φάση της εφαρμογής στην άλλη. Τέλος, το activity αυτό, προβάλλει στην οθόνη τις σχετικές λεπτομέρειες για τον Merchant.

Μία δευτερη επιλογή εκτός από το 'Purchase Goods' θα ήταν το 'Link your cards', το οποίο δεν έχει υλοποιηθεί στην παρούσα εργασία. Για λόγους συντόμευσης στην παρούσα εφαρμογή, οι κάρτες και οι λογαριασμοί του χρήστη, έχουν δοθεί ως hard coded (σε xml μορφή σε αρχείο της συσκευής), αλλά σε μια επαγγελματική εφαρμογή, θα πρέπει να δίνονται, όπως προαναφέρθηκε, δηλαδή, δυναμικά και σε ξεχωριστές οθόνες.



```
{
  "Discover":"8524698745214123",
  "Mastercard":"4600678934521436",
  "Visa":"3245698745210201",
  "Bill":"US78526985554255784126987 42"
}
```

(εικ. 39)

```
{
 "008":"Albanian lek",
 "012":"Algerian dinar",
 "032":"Argentine peso",
 "036":"Australian dollar",
 "044":"Bahamian dollar",
 "048":"Bahraini dinar",
 "050":"Bangladeshi taka",
 "051":"Armenian dram",
 "052":"Barbados dollar",
 "060":"Bermudian dollar",
 "064":"Bhutanese ngultrum",
 "068":"Boliviano",
 "072":"Botswana pula",
 "084":"Belize dollar",
 "090":"Solomon Islands dollar",
 "096":"Brunei dollar",
 "104":"Myanmar kyat",
 "108":"Burundian franc",
 "116":"Cambodian riel",
 "124":"Canadian dollar",
 "132":"Cape Verde escudo",
 "136":"Cayman Islands dollar",
 "840":"United States dollar",
    .....
}
```

(εικ. 40)

Μετά την σάρωση, συμπλήρωση και προβολή στην οθόνη των λεπτομερειών του επιτηδευματία, ο χρήστης οδηγείται στο επόμενο activity προκειμένου να επιλέξει την μέθοδο πληρωμής που επιθυμεί. Δηλαδή, την κάρτα ή τον λογαριασμό τον οποίο θα πρέπει να χρησιμοποιήσει για την διαφαινόμενη αγορά. Η επόμενη οθόνη, παρουσιάζει μια λίστα από κάρτες και λογαριασμούς, τα οποία έχουν εισαχθεί στην εφαρμογή σε προγενέστερη φάση (εικ. 23). Το activity CompleteForm, διαχειρίζεται την εν λόγω λίστα.

Για την δημιουργία της λίστας καρτών, έχει χρησιμοποιηθεί ένα CustomListAdapter class το οποίο κάνει extend ή κληρονομεί ένα ArrayAdapter<String>. Στην κλάση αυτή, χρησιμοποιείται ο κατασκευαστής της (constructor) προκειμένου να αρχικοποιεί το Activity, εφοσον δεν είναι σε περιβάλλον activity, χρειάζεται να υπάρχει ένα reference (αναφορά) του, καθώς επίσης και δύο πίνακες εκ των οποίων ο ένας να είναι τύπου String για τα labels ή ονόματα της λίστας καρτών και ο άλλος, ένας πίνακας με Integers, οι οποίοι αντιπροσωπεύουν τα ids των resources - εικόνων για δυναμική προσάρτηση τους σε μια for loop διαδικασία. Για να επιτευχθεί αυτό, χρησιμοποιείται ένα custom xml layout με ένα ImageView και ένα TextView περιφραγμένα από ένα LinearLayout με οριζόντιο προσανατολισμό (horizontal orientation). Τέλος, χρησιμοποιείται ένα LayoutInflater προκειμένου να τοποθετηθούν τα προαναφερθέντα views.

Για την σύλληψη της επιλεγμένης κάρτας, αλλά και για να τονιστεί γραφικά, η όποια επιλογή του χρήστη, χρησιμοποιείται ένας ακροατής συμβάντων, list.setOnItemClickListener(new AdapterView.OnItemClickListener(), ο

οποίος πέρα από την κύρια χρησιμότητά του, η οποία είναι, όπως αναφέρθηκε, η σύλληψη της επιλογής του χρήστη, χρησιμοποιείται επίσης και για την σκιαγράφηση με ένα σκούρο χρώμα του παρασκηνίου της επιλεγμένης μεθόδου πληρωμής, με την ακόλουθη εντολή: parent.getChildAt(i).setBackgroundColor(Color.TRANSPARENT);. Δηλαδή, χρησιμοποιείται ένας κωδικός χρώματος #0D000000 και προστίθεται transparency αυτού (εικ. 41).

```
list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        setSelectedItem(itemName[+position]);
        setSelectedNumber(itemValues[+position]);
        for (int i = 0; i < parent.getChildCount(); i++) {
            if (position == i) {
                parent.getChildAt(i).setBackgroundColor(getResources().getColor(R.color.tr));
            } else {
                parent.getChildAt(i).setBackgroundColor(Color.TRANSPARENT);
```

(εικ. 41).

Προφανώς η επιλεγμένη κάρτα ή ο λογαριασμός, προστίθεται στο request, αφού υποστεί μια επεξεργασία απόκρυψης ή μάσκας της μορφής US78 5269 8555 xxxx xxxx xxxx 742, για ευνόητους λόγους ασφάλειας, καθώς αυτό είναι και το στοιχείο το οποίο θα προβάλλεται στην οθόνη της συσκευής. Στη συνέχεια, το request με τη σειρά του, προσαρτάται στο Bundle για περαιτέρω ανάπτυξη.

Στο συγκεκριμένο στάδιο της εφαρμογής, η μορφή του request, η οποία ‘χτίζεται’ σε κάθε βήμα της εφαρμογής (υποθέτουμε πως ο χρήστης έχει επιλέξει το bill/iban, για την συναλλαγή του και όχι κάποια χρεωστική κάρτα), είναι της παρακάτω μορφής:


```
{
    "cityName":"New York",
    "mobileNumber":"6942580624",
    "addDataMasterCard1":"PS4 CAMERA",
    "countryCode":"US",
    "transactionAmount":"47.0",
    "merchantName":"Walmart",
    "currencyCode":"United States dollar",
    "optionSelected":"Purchase Goods",
    "billId":"US5468464646844684864864 68",
    "merchantCategoryCode":"5204",
    "bill":"Bill",
    "ibanNumber":"US78 5269 8555 xxxx xxxx xxxx 742"
}
```


(εικ. 42)

Ο χρήστης, στην επόμενη φάση, καλείται να εισάγει ένα προσωπικό μυστικό τετραψήφιο Pin, το οποίο διαφέρει από κάρτα σε κάρτα και μέσω του Pin αυτού και σε συνδυασμό με τον αριθμό της κάρτας ή το λογαριασμού του,

γίνεται η αναγνώριση πελάτη (customer) από την πλευρά της τράπεζας (δεύτερο μέρος της εφαρμογής – εικ. 24, 25).

Η Pin υπηρεσία, διαχειρίζεται από το activity Pin (εικονα 43).

```
Button nextToPin = (Button) findViewById(R.id.nextToPin);
nextToPin.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (getSelectedItem() != null) {
            Intent intent = new Intent(CompleteForm.this, Pin.class);
            Bundle bundle = new Bundle();
            if (info != null) {
                bundle.putString("info", info.toString());
            }
```

(εικ. 43)

Για την παρακολούθηση της συμπλήρωσης του Pin το οποίο δίνεται από τον χρήστη, τίθεται ένα TextChangedListener με παράμετρο ένα αντικείμενο τύπου TextWatcher και γίνεται override η μέθοδος onTextChanged (εικ. 44). secretPin.addTextChangedListener(new TextWatcher() {….}

Όπως διαφαίνεται από την εικόνα (εικ. 44), το πεδίο Pin, υπόκειται μιας μικρής σειράς από ελέγχους για την ορθή εισαγωγή του σύμφωνα με τους περιορισμούς που θέτει η εφαρμογή. Αρχικά ελέγχεται αν ο χρήστης έδωσε κάποιο κενό ή κενή συμβολοσειρά και σε επόμενο στάδιο εάν έκανε εισαγωγή ψηφίων, διαφορετικά, γίνεται ανακατεύθυνση (redirect) και το πεδίο γίνεται focused (γραμμές 61, 68) καθώς και αναρτάται μια υπενθύμηση με το κατάλληλο μήνυμα.

```
50      secretPin.addTextChangedListener(new TextWatcher() {
51
52          @Override
53          public void beforeTextChanged(CharSequence s, int start, int count, int after) {
54          }
55
56          @Override
57          public void onTextChanged(CharSequence s, int start, int before, int count) {
58              if (s.toString() != null) {
59                  if (s.toString().length() != secretPinSize) {
60                      secretPin.setError("Secret pin is required!");
61                      secretPin.requestFocus();
62                      utils.showMessage("Secret pin is required!");
63                  }
64                  try {
65                      Integer.parseInt(s.toString());
66                  } catch (NumberFormatException e) {
67                      secretPin.setError("Secret pin has to be made of digits only!");
68                      secretPin.requestFocus();
69                      utils.showMessage("Secret pin has to be made of digits only!");
70                      Log.e("Number Format Exception", "Wrong secret digits", e);
71                  }
72              }
73          }
```

(εικ. 44)

Σε περίπτωση που όλα βαίνουν καλώς, το Pin κωδικοποιείται για λόγους ασφαλείας και για αποτροπή κλοπής του σε περίπτωση που το request γίνει αντικείμενο κακόβουλης επίθεσης (εικ. 45 line: 95).

```
93          if (allOk) {
94              info = utils.retrieveJsonFromString(infoString);
95              String encodedPin = utils.getEncodedPin(givenSecretPin);
96              info = utils.addOptToJSONObject(info, "secretPin", encodedPin);
97              info = utils.addOptToJSONObject(info, "consumerId", utils.splitRandomString(utils.randomString()));
98              bundle.remove("info");
99              bundle.putString("info", info.toString());
100             infoString = bundle.getString("info");
101             Intent intent = new Intent(Pin.this, Confirm.class);
102             intent.putExtras(bundle);
103             startActivity(intent);
104         }
```

(εικ. 45)

Παρατηρείται στην γραμμή 95 να καλείται μια μέθοδος getEncodedPin περνώντας της το Pin ως παράμετρο.

Αξίζει να σημειωθεί σε αυτό το σημείο πως η κλάση Util είναι μια βοηθητική (custom) κλάση η οποία περιέχει πολλές μεθόδους και κατά κάποιον τρόπο περικλείει και συγκεντρώνει το λεγόμενο business logic. Το γεγονός ότι είναι συγκεντρωμένες σε ένα σημείο όλες τις βοηθητικές και χρήσιμες λειτουργίες, βοηθά αφενός στην δομημένη διατήρηση του κώδικα και αφετέρου στην αποτροπή επανάληψης ίδιου κώδικα.

Η μέθοδος αυτή (εικ. 46), δέχεται το Pin σαν τύπο String, καλεί την Encrypt κλάση, και κωδικοποιεί κάνοντας χρήση της encodePin μεθόδου. Η Encrypt κλάση, έχει την παρακάτω υπογραφή:

public class Encrypt implements IEncrypt και κάνει χρήση της βιβλιοθήκης *android. util.Base64.*

```
512     public String getEncodedPin(String pin) {
513         String encryptedPin = null;
514         if ((pin != null) && (!pin.isEmpty())) {
515             Encrypt encrypt = new Encrypt();
516             byte[] encryptedBytes = encrypt.encodePin(pin);
517             encryptedPin = new String(encryptedBytes);
518         }
519         return encryptedPin;
520     }
```

(εικ. 46)

```java
package com.wirecard.pin;

/**
 * Created by
 * Andreas Karmenis
 * Software Engineer
 * on 9/25/2017.
 */

public interface IEncrypt {
    byte[] encodePin(String toBeEncoded);
    String decodePin(byte[] toBeDecoded);
}
```

(εικ. 47)

Η διεπαφή (interface) IEncrypt (εικ. 47), προβάλλει δύο μεθόδους: την encodePin & decodePin οι οποίες υλοποιούνται από την Encrypt κλάση.

```java
/**
 * Created by
 * Andreas Karmenis
 * Software Engineer
 * on 9/25/2017.
 */

public class Encrypt implements IEncrypt {

    private static final String PREFIX_SALT = "c0e81794";
    private static final String SUFFIX_SALT = "2cf24dba";

    public Encrypt() {}

    @Override
    public byte[] encodePin(String toBeEncoded) {
        String encoding = (PREFIX_SALT + toBeEncoded + SUFFIX_SALT);
        return Base64.encode(encoding.getBytes(), Base64.DEFAULT);
    }

    @Override
    public String decodePin(byte[] toBeDecoded) {
        byte[] decodedBytes = Base64.decode(toBeDecoded, Base64.DEFAULT);
        String decodedString = new String(decodedBytes);
        String withoutPrefix = decodedString.replace(PREFIX_SALT, "");
        String withoutSuffix = withoutPrefix.replace(SUFFIX_SALT, "");
        return withoutSuffix;
    }
}
```

(εικ. 48)

Η Encrypt κλάση, παρέχει δύο μέρη salt: την prefix & suffix salt. Τα προσθέτει στα άκρα του Pin που εδόθη από τον χρήστη και κωδικοποιεί με Base64 η οποία αποτελεί μια εσωτερική βιβλιοθήκη του Android.

Επίσης, για την αποκωδικοποίηση, ακολουθείται η αντίστροφη σειρά από εκείνη της κωδικοποίησης (γραμμή 26). Το κωδικοποιημένο Pin, προσαρτάται στο υπό κατασκευή request και επίσης προσαρτάται και ένα αναγνωριστικό του συγκεκριμένου πελάτη για τη συγκεκριμένη αγορά το οποίο είναι ένα τυχάιο (random) id που παράγεται μέσω του UUID.randomUUID().toString(); (εικ. 49). Το request φορτώνεται μέσω στην Bundle και αποστέλλεται για επόμενη στάση στο activity Confirm. *Intent intent = new Intent(Pin.this, Confirm.class);*

```
public String randomString() {
    return UUID.randomUUID().toString();
}
```

(εικ. 49)

Στην οθόνη αυτή, με έναν αυτοματισμό, προσθέτουμε, κάποια επιπλέον στοιχεία τα οποία είχαν εισαχθεί κατά τη διάρκεια της επιλογής κάρτας ή λογαριασμού όπου θα χρεωθούν τα χρήματα (εικ. 39). Ο αυτοματισμός, έχει να κάνει με την ανάκτηση μέσω του Java Reflection όλων τα δηλωμένων fields των κλάσεων και υπερκλάσεων (super classes) και των ids τα οποία είναι δηλωμένα στα διάφορα xml files. Ο λόγος που γίνεται αυτό είναι επειδή η οθόνη Confirm, δηλώνει ένα TableLayout που έχει για TableRows ids με την ακόλουθη μορφή: android:id="@+id/creditCardNumber". Αν παρατηρηθεί (εικ. 51), θα διαπιστωθεί ο αυτόματος μηχανισμός που χρησιμοποιείται για να αποδοθούν τιμές (values) σε επιλεγμένα views.

```
public Field[] declaredFields() {
    Class<R.id> c = R.id.class;
    Field[] fields = c.getDeclaredFields();
    return fields;
}
```

(εικ. 50)

```
int id = getResources().getIdentifier("creditCardNumber", "id", "com.wirecard.qrcodeservices");
((TextView) findViewById(id)).setText(value);
if (sardName != null) {
    credit.setText(sardName);
}
```

(εικ. 51)

Όπως προαναφέρθηκε (εικ. 26, 27), η οθόνη προσθέτει άλλα τρία στοιχεία. Μεταξύ άλλων και το IBAN ή credit card. Μπορεί να παρατηρηθεί η μάσκα στην οποία υπόκειται ο αριθμός λογαριασμού για λόγους ασφαλείας. Επίσης και το consumer id το οποίο επισυνάπτεται στον χρήστη ως αναγνωριστικό της εν δυνάμει αγοράς.

Με το πάτημα του πλήκτρου confirm, καλείται η κλάση Post, το περίγραμμα σε μπλέ χρώμα, πρώτα η εφαρμογή αιτείται προκειμένου να διαπιστώσει αν μπορεί να αποκτήσει σύνδεση δικτύου μέσω του ConnectivityManager. Η σύνδεση στο δίκτυο, μπορεί να αποκτάται με διάφορους τύπους όπως WIFI αλλά και MOBILE, ETHERNET και λοιπά. Σε περίπτωση θετικής έκβασης, ανακτάται από τα Preferences, και δημιουργείται το πλήρες url, στη doPost μέθοδο. Η συγκεκριμένη μέθοδος, καλεί την execute μέθοδο μιας εσωτερικής κλάσης στη Post κλάση (εικ. 53, 54).

```
Button post = (Button) findViewById(R.id.post);
post.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        info = utils.retrieveJsonFromString(infoString);
        Post post = new Post(activity, infoString, getBundle());
        boolean hasConnection = post.isConnected();
        disableConnectionReuseIfNecessary();
        String gotConnection = String.valueOf(hasConnection);
        if (hasConnection) {
            post.doPost();
            Log.i("Network OK: ", gotConnection);
            utils.showMessage(getString(R.string.okconnection));
        } else {
            Log.i("NO Network: ", gotConnection);
            utils.showMessage(getString(R.string.noconnection));
        }
    }
});
```

(εικ. 52)

```
public void doPost(){
    String url = serverConfig.getConfiguration();
    if ((url != null) && (!url.isEmpty())){
        Log.i("url to go: ",String.valueOf(url));
        new PostTask().execute(url);
    } else {
      utils.showMessage("Please, setup the server first.\n\tProtocol:\n\tHost:\n\tPort:\n\tService:");
    }
}
```

(εικ. 53)

```
private class PostTask extends AsyncTask<String, Void, String> {
```

(εικ. 54)

Η PostTask κάνει extend ή κληρονομεί την public abstract class AsyncTask, η οποία χρησιμοποιείται κατά κύριο λόγο ως η καταλληλότερη μέθοδος και η ευκολότερη χρήση σε UI threads. Η συγκεκριμένη μέθοδος, επιτρέπει την εκτέλεση λειτουργιών στο παρασκήνιο (background) και παρουσιάζει τα αποτελέσματα στο UI thread χωρίς να χρειαστεί η διαχείριση των νημάτων (threads) ή handlers. Ως εκ τούτου, η κλάση PostTask, διαθέτει ή καλύτερα, κάνει override, μια doInBackgroung μέθοδο η οποία παίρνει ως παράμετρο το προαναφερθέν url. Αυτό που αμέσως μετά γίνεται είναι το initialization της σύνδεσης. Σε αυτό το σημείο, προσπαθεί να ανοίξει έναν διαύλο με το server στον οποίο δείχνει το url απόσπασμα του κώδικα (εικ.55). Τίθενται διάφορες

ιδιότητες (connection properties), χαρακτηριστικά αναφέρεται το connection time out το οποίο είναι οριοθετημένο στα 15 δευτερόλεπτα.

```
203
204     public HttpURLConnection initConnection(String... params) {
205         HttpURLConnection connection = null;
206         try {
207             if ((params != null) && (params[0] != null)) {
208                 URL url = new URL(params[0]);
209                 connection = (HttpURLConnection) url.openConnection();
210                 connection.setRequestMethod(REQUEST_METHOD);
211                 connection.setReadTimeout(READ_TIMEOUT);
212                 connection.setConnectTimeout(CONNECTION_TIMEOUT);
213                 connection.setRequestProperty(TYPE, CONTENT_TYPE);
214                 connection.setDoOutput(true);
215                 connection.setDoInput(true);
216                 connection.setUseCaches(false);
217             }
218         } catch (Exception e) {
219             Log.e("Http URL Connection", "URL", e);
220         }
221
222         return connection;
223     }
```

(εικ. 55)

Μέσω της εντολής της γραμμής 62 (εικ. 56), προσπαθεί να ανοίξει διαύλο επικοινωνίας με τον server έχοντας ως παράμετρο το connection object, ενώ με την εντολή της γραμμής 64 στέλνει στον server το request το οποίο είναι τύπου String ονόματι json. Αν η αποστολή έγινε επιτυχώς, τότε ανακτάται μέσω του κώδικα της γραμμής 66 το αποτέλεσμα, δηλαδή το response. Βέβαια, το response είναι διαθέσιμο στην μέθοδο onPostExecute(String result) η οποία είναι ένας listener που αφουγκράζεται response events – (εικ. 57).

```
52          @Override
53          protected String doInBackground(String... params) {
54              InputStream inputStream = null;
55              String result = "";
56              try {
57                  for(String param : params){
58                      Log.i("xparamx", param);
59                  }
60                      connection = utils.initConnection(params);
61                      if (connection != null){
62                        OutputStream out = utils.getStream(connection);
63                        if (out != null){
64                            boolean sent = utils.sendRequest(out, json);
65                            if(sent){
66                              result = utils.getResponse(connection);
67                            }
68                        }
69                      } else {
70                          utils.showMessage("No network (wifi or mobile) exist!.");
71                          Log.e("No network", "exist");
72                      }
73              }catch (Exception e) {
74                  utils.showMessage("Connection to Server Has failed.");
75                  Log.e("Connection to Server", " Has failed.", e);
76                  e.printStackTrace();
77              }finally {
78                  if(connection != null) {
79                      connection.disconnect();
80                  }
81              }
```

(εικ. 56)

```
85          protected void onPostExecute(String result)  {
86              if ((result != null) && (!result.isEmpty())){
87                  Log.e("nPostExecute.result", result);
88                  Context context = getActivity().getApplicationContext();
89                  Intent intent = new Intent(context, Results.class);
90                  intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
91                  Bundle newBundle = getBundle();
92                  newBundle.putString("response", result);
93                  setBundle(newBundle);
94                  intent.putExtras(getBundle());
95                  context.startActivity(intent);
96              } else {
97                  utils.showMessage("Cannot connect to Server.\nPlease, see server configurations.");
98                  Log.e("Connection to Server", " Has failed.");
99              }
100         }
```

(εικ. 57)

Η Bundle, περικλείει το response και μαζί ‘συνταξιδεύουν’ με προρισμό το Result activity, γραμμή 89 (εικ. 57).

Το Result Activity ή η Result οθόνη, ασχολείται, κατά κύριο λόγο, με την εμφάνιση/απάντηση στην οθόνη του χρήστη των αποτελεσμάτων των διαφόρων αγορών που λαμβάνουν χώρα. Η απάντηση (response), που αποδέχεται από προηγούμενο στάδιο σε String μορφή (εικ. 57), μετατρέπεται

σε json object και από αυτό εξάγεται ένα πεδίο (tok- transaction ok) που αναπαριστά την έκβαση της συναλλαγής και είναι τύπου Boolean, με τιμές true – false, επιτυχής ή όχι συναλλαγή. Εν συνεχεία, ακριβώς βάση του πεδίου αυτού, η εφαρμογή μπορεί να αποφασίζει το περιεχόμενο που θα πρέπει να προβληθεί στην οθόνη με τις κατάλληλες πληροφορίες μιας πετυχημένης ή όχι συναλλαγής. Η απόφαση που λαμβάνει η εφαρμογή για την παρουσίαση των λεπτομερειών της συναλλαγής, μπορεί κάλλιστα να ληφθεί και από άλλα εξίσου σημαντικά πεδία του response object, όπως το status που είναι τύπου (HttpStatus), αλλά και το errorCode code το οποίο παίρνει τιμές-κωδικούς κυρίως από τους HttpStatus codes. Τέλος, υπάρχει και το πεδίο errorMessage που κατά περίπτωση, άλλοτε έχει τιμές που εξάγωνται από το HttpStatus object και άλλοτε δέχεται τιμές μιας custom enumeration κλάσης, η οποία περιλαμβάνει όλα τα business logic σφάλματα (errors) που σχετίζονται με το οικονομικό μέρος και πιθάνον να λαμβάνουν χώρα κατά την χρονική διάρκεια μιας συναλλαγής σε εξέλιξη. Η public enum ResponseStatus (εικ. 58), περιέχει τους κωδικούς των εν δυνάμει προαναφερθέντων σφαλμάτων.

```
/**
 * Created by andre on 10/28/2017.
 */
public enum ResponseStatus {

    EXPIRED(100, "Credit card has expired"),
    BLOCKED(101, "Credit card is blocked"),
    INACTIVE(102, "Credit card is inactive"),
    PENDING(103, "Credit card is in pending status"),
    CLOSED(104, "Credit card is closed"),
    BLACKLIST(105, "Credit card in blacklist"),
    INVALID_PIN(106, "Invalid pin number"),
    BALANCE_NOT_ENOUGH(107, "Balance not enough"),
    ACCOUNT_OK(108, "Account status is ok"),
    CARD_OK(109, "Card status is ok"),
    LOCKED(110, "Your account is locked. Please, consult the administration."),
    MAX_PIN_TRY_LIMIT(111, "Reached out the maximum tries number.");
```

(εικ. 58)

```
4
5   /**
6    * Created by andre on 10/18/2017.
7    */
8   public class Response {
9
10      private HttpStatus status;
11      private String errorMessage;
12      private String errorCode;
13      private String time;
14      private String paymentReference;
15      private String cityName;
16      private String mobileNumber;
17      private String addDataMasterCard1;
18      private String countryCode;
19      private String transactionAmount;
20      private String merchantName;
21      private String currencyCode;
22      private String optionSelected;
23      private String userName;
24      private String paymentType;
25      private Boolean pin;
26      private Double balance;
27      private String account;
28      private String consumerId;
29      private String product;
30      private String cardStatus;
31      private boolean tok;
32      private String secretPin;
```

(εικ. 59)

Επιπλέον, σε run time χρόνο στην οθόνη αυτή, αποφασίζεται ο χρωματισμός σε κάποια κύρια labels προκειμένου να τονίσει και να κάνει διάκριση μεταξύ μιας επιτυχούς από μια αποτυχημένη συναλλαγή (πράσσινο και κόκκινο χρώμα αντίστοιχα). Επίσης, με μπλέ χρώμα αποτυπώνεται και την χρονική στιγμή (timestamp) της συναλλαγής.

Σε περίπτωση μιας αποτυχημένης συναλλαγής λόγω τραπεζικού χρηματικού ισοζυγίου του χρήστη, εκτός από το τυπικό μήνυμα ‘balance not enough’, η εφαρμογή αναφέρει και τα δύο ποσά, αυτό του κόστους της συναλλαγής και αντίστοιχα αυτό του ισοζυγίου, προκειμένου να πλαισιώσει το μήνυμα σφάλματος, με επιπλέον σαφήνεια.

Πάντως, είτε θετική είτε αρνητική έκβαση έχει μια συναλλαγή, η εφαρμογή καταγράφει αυτές τις κινήσεις για ιστορικούς λόγους δίνοντας έτσι την δυνατότητα στον χρήστη της, να ανατρέξει σε συναλλαγή/ες παρελθούσας χρονικής στιγμής. Για τον λόγο αυτό, η εφαρμογή επιστρατεύει την Firebase βάση δεδομένων.

Η Firebase βάση, είναι μια μη σχεσιακή βάση η οποία παρέχεται από τον οργανισμό της Google. Θα μπορούσε κάλλιστα, η εφαρμογή να αποθηκεύσει τα στιγμιότυπα των συναλλαγών (logs) σε μια SQLite βάση εφόσον μια τέτοια βάση μπορεί να χρησιμοποιηθεί από το Android λογισμικό, αλλά και άλλου τύπου σχεσιακή βάση όπως αυτή της MySql για παράδειγμα, η οποία ήδη χρησιμοποιείται από το δεύτερο μέρος της εφαρμογής, στην οποία θα γίνει λεπτομερή αναφορά παρακάτω και σε μεταγενέστερο χρόνο αφού παρατεθεί η χρήση, η χρησιμότητα και η ανάπτυξη της Firebase βάσης.

Αρχικά θα πρέπει να αναφερθεί γιατί προτιμήθηκε η συγκεκριμένη βάση για την καταγραφή των δεδομένων. Επίσης, είναι γνωστό το γεγονός ότι για να χρησιμοποιηθεί μια τέτοια βάση, θα πρέπει η εφαρμογή να έχει αδιάκοπτα πρόσβαση στον παγκόσμιο ιστό (internet) προκειμένου να μπορέσει να κάνει χρήση της υπηρεσίας. Δεν θα ήταν καλύτερα ωστόσο, μια SQLite βάση η οποία να είναι τοπικά στη συσκευή με τα δεδομένα να είναι διαθέσιμα, μη παρακινδυνευμένα από κακόβουλες ενέργειες?

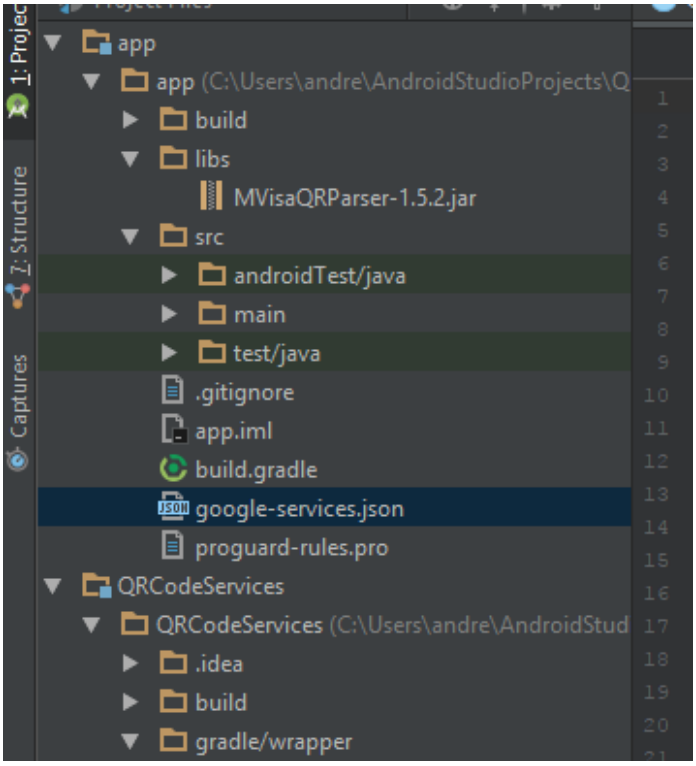
Ως απάντηση στα παραπάνω εύλογα ερωτήματα, θα πρέπει να αναφερθούν τα εξής: Πρώτον, η εφαρμογή από τη φύση της, δεν νοείται να λειτουργεί χωρίς την πρόσβαση στο παγκόσμιο ιστό. Επειδή η εφαρμογή ως επίκεντρό της έχει την συνδιαλλαγή με κάποια τράπεζα προκειμένου να πραγματοποιηθεί μια συναλλαγή, και επειδή η τράπεζα είναι ένας ανεξάρτητος οργανισμός, που μπορεί να βρίσκεται οπουδήποτε χωροταξικά, απαιτείται δικτυακή πρόσβαση σε αυτή. Ο μόνος τρόπος να ‘συνομιλήσει’ η εφαρμογή με οποιαδήποτε τράπεζα, είναι αυτός μέσω της πρόσβασης στον παγκοσμιο ιστό. Δεδομένου ότι το δεύτερο μέρος της εφσρμογής προσομειώνει μια τράπεζα, η πρόσβαση στον ιστό (web) είναι αναπόσπαστο, κεντρικό μέρος της εφαρμογής και μη διαπραγματεύσιμη.

Ως εκ τούτου, η χρήση μιας ‘on-line’ Firebase βάσης, δεν αποκλίνει από τη φύση της εφαρμογής. Ως επιπρόσθετο στοιχείο, προς ενίσχυση των παραπάνω, θα πρέπει να αναφερθεί και το γεγονός ότι στις ημέρες μας, η πρόσβαση στον παγκόσμιο ιστό αλλά και οι προσφερόμενες ταχύτητες, ενθαρρύνουν τις on-line υπηρεσίες και δεν καθιστούν απαγορευτική τη χρήση της Firebase βάσης.

Ένας άλλος λόγος που χρησιμοποιείται η Firebase βάση είναι η ανεξαρτησία της, δεδομένου ότι δεν υπάρχει ανάγκη διατήρησης μιας βάσης δεδομένων τοπικά (στην ευθύνη της εφαρμογής ή του χρήστη), αλλά αντίθετα αναλαμβάνεται και παρέχεται ως υπηρεσία από έναν αξιόπιστο οργανισμό, παγκόσμιου κύρους, όπως αυτόν της Google.

Επίσης, η Firebase είναι μια NO SQL, scheme-free και real time βάση δεδομένων και κάνοντας χρήση JSON Objects, ιδανική για RESTFUL HTTP API programming. Αυτό σημαίνει κυρίως μεγαλύτερες ταχύτητες επικοινωνίας.

Προκειμένου να γίνει χρήση της βάσης, αρχικά η Google απαιτεί κάποιο λογαριασμό χρήστη. Μετά ακολουθεί η διαμόρφωση ενός αρχείου σε μορφή JSON, η οποία προσαρτάται στα αρχεία της εφαρμογής κάτω από το src folder (εικ. 60).



(εικ. 60)

Το αρχείο αυτό, μεταξύ άλλων, περιέχει το project number, project id, μια firebase URL, ένα client id, ένα key και το configuration version.

Για την δημιουργία της Firebase βάσης, αρχικοποιούμε ένα instance της FirebaseDatabase, μέσω της static μεθόδου FirebaseDatabase fbInstance = FirebaseDatabase.getInstance()

(εικ. 61)

```
33      public void initFireBaseIstance(){
34          fbInstance = FirebaseDatabase.getInstance();
35          // get reference to 'users' node
36          fbDatabase = fbInstance.getReference("payments");
37          // store app title to 'app_title' node
38          fbInstance.getReference("app_title").setValue("QR Code Fire Base Database");
39      }
```

(εικ. 62)

Η εικόνα (εικ. 61), δείχνει το πραγματικό αποτέλεσμα που προέρχεται από τον κώδικα της (εικ. 62). Η εισαγωγή στη βάση μιας εγγραφής, επιχειρείται από τον κώδικα της (εικ. 63), στο Result activity.

```
Payment payment = new Payment(String.valueOf(afterResponseInfo));
fbDatabase = new FireBaseDataBase();
fbDatabase.createPaymentLog(payment);
```

(εικ. 63)

Η createPaymentLog μέθοδος δέχεται μια παράμετρο τύπου Payment και ο κώδικας της μεθόδου της (εικ. 64), αναλαμβάνει την εισαγωγή στη βάση.

Μέσω της εντολής *paymentId = fbDatabase.push().getKey();* διαβάζεται το key χωρίς να γίνει push κάποια τιμή ακόμη. Μεταγενέστερα, δημιουργείται ένα child με αυτό το key και γίνεται push ένα αντικείμενο τύπου payment, για το κλειδί (key) αυτό (εικ. 64).

Στην μέθοδο, *fbDatabase.child(paymentId).setValue(payment);*, το paymentId, χρησιμοποιείται να δεικτοδοτήσει την παράμετρο payment, που πρόκειται να σωθεί στη βάση.

```
public void createPaymentLog(Payment payment) {
    // TODO
    // In real apps this userId should be fetched
    // by implementing fire base auth
    if (TextUtils.isEmpty(paymentId)) {
        paymentId = fbDatabase.push().getKey();
    }
    fbDatabase.child(paymentId).setValue(payment);
    addLogChangeListener();
}
```

(εικ. 64)

Η FirebaseDatabase βιβλιοθήκη, διαθέτει δικούς του listeners εκ των οποίων, ο ένας εξ αυτών είναι ένα *addValueEventListener* το οποίο αφουγκράζεται συμβάντα (events) εισαγωγής οντοτήτων στη βάση.

Ενώ για την ανάκτηση δεδομένων από την βάση, χρησιμοπιείται ένας άλλος ακροατής συμβάντων (listener) και συγκεκριμένα ο

addListenerForSingleValueEvent. Η χρήση του συγκεκριμένου ακροατή συμβάντων, έχει επιλεχθεί για δύο λόγους. Πρώτον, λόγω του ότι η εφαρμογή δεν απαιτεί μια σταθερή ενημέρωση (constant updates) του GUI, η οποία είναι αρκετά δαπανηρή στην δικτυακή κυκλοφορία και δεύτερον, η μέθοδος χρειάζεται να διαβάσει δεδομένα μονάχα μια φορά, σε κάθε εισαγωγή στη βαση, ως εκ τούτου, ο ακροατής *addListenerForSingleValueEvent*, μοιάζει ιδανικός.

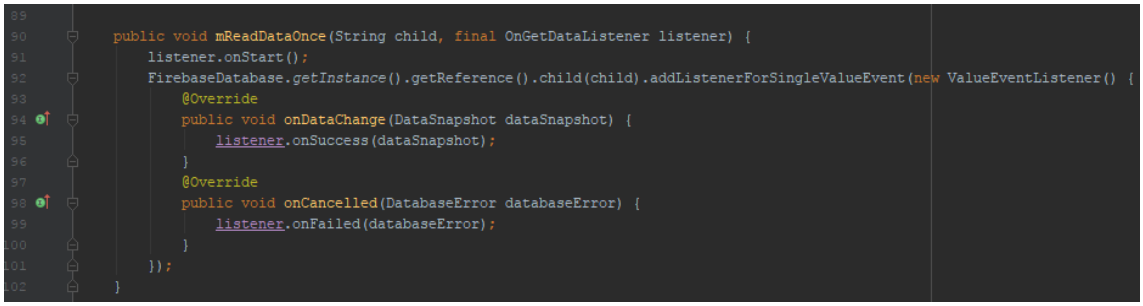
Τα δεδομένα, και πιο συγκεκριμένα μια λίστα δεδομένων, απαιτείται να προβληθεί ταξινομημένη κατά φθίνουσα σειρά, ομαδοποιημένη κατά ημερομηνία από το πιο πρόσφατο στο πιο παλαιό. Η εισαγωγή στη λίστα (ExpandableListView) για προβολή δεδομένων στον χρήστη, αν και μοιάζει μια ασήμαντη διαδικασία, ωστόσο αποτέλεσε μια πρόκληση κατά το στάδιο της υλοποίησης της λίστας για έναν και μοναδικό λόγο. Ο προαναφερθεν ακροατής συμβάντων, και γενικά η ανάκτηση δεδομένων με την βιβλιοθήκη της Firebase, είναι μια ασύγχρονη διαδικασία.

Αυτό σημαίνει ότι στον ασύγχρονο προγραμματισμό, δεν είναι δυνατόν να χρησιμοποιηθεί εκτός εμβέλειας του ακροατή κάποια αναφορά εξωτερικού αντικειμένου, είτε αυτό είναι μια global attribute, είτε κάποιο άλλο source. Η εκτέλεση του κώδικα στο σώμα του ακροατή, θέτει ή εκκινεί κάποια διαδικασία, όπως την εκχώρηση δεδομένων μιας λίστας, αλλά δεν περιμένει να τελειώσει η διαδικασία. Η εκτέλεση εξέρχεται του τελευταίου curly brace της μεθόδου προτού ολοκληρωθεί η διαδικασία που εκκινήθηκε, κι έτσι, η διαδικασία εκχώρησης της λίστας, δυστυχώς διακόπτεται χωρίς να προλάβει να τελειώσει. Αυτό θυμίζει πολύ τα γνωστά νήματα τα οποία έχουν απρόβλεπτη έκβαση ως προς τον συγχρονισμό. Όταν ένα νήμα περιμένει αποτέλεσμα από κάποιο άλλο, δεν υπάρχει εγγύηση ότι αυτό πράγματι θα συμβαίνει πάντα, δηλαδή είναι μια κατάσταση με απρόσμενα αποτελέσματα.

Προκειμένου να ξεπεραστεί το εμπόδιο αυτό, ορίστηκαν σε μια Interface τρείς μέθοδοι και υλοποιήθηκαν στην ExpandedLogList activity και στο εσωτερικό του ακροατή ετέθηκε ως μια παράμετρος με δηλωμένες ωστόσο τις μεθόδους εσωτερικά στις *public void onDataChange(DataSnapshot dataSnapshot)* και *public void onCancelled(DatabaseError databaseError)* αντίστοιχα (εικ. 65).

```
6   /**
7    * Created by
8    * Andreas Karmenis
9    * Software Engineer
10   * on 10/3/2017.
11   */
12
13  public interface OnGetDataListener {
14      void onStart();
15      void onSuccess(DataSnapshot data);
16      void onFailed(DatabaseError databaseError);
17  }
18
```

(εικ. 65)



```
public void mReadDataOnce(String child, final OnGetDataListener listener) {
    listener.onStart();
    FirebaseDatabase.getInstance().getReference().child(child).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            listener.onSuccess(dataSnapshot);
        }
        @Override
        public void onCancelled(DatabaseError databaseError) {
            listener.onFailed(databaseError);
        }
    });
}
```

(εικ. 66)

Αυτό το 'trick', έκανε τη πολυπόθητη δουλειά.

Πλέον, απομένει η απόδοση των ανακτώμενων δεδομένων σε μια *Expandable List View*. Στο ExpandedLogList activity, το οποίο είναι το activity που διαχειρίζεται την ExpandableListView, γίνονται override οι παραπάνω τρεις μεθόδοι. Αρχικά η *onStart* η οποία είναι μια μέθοδος που το σώμα της θα πρέπει να περιέχει κώδικα σχετικό με την αρχική εμφάνιση της λίστας.

Προτιμήθηκε να τοποθετηθεί μια *ProgressDialog* που δείχνει και αυξάνει μια progress bar για το διάστημα 'φόρτωσης' δεδομενών. Στο χρονικό αυτό διάστημα, προκειμένου να μην επιβαρυνθεί η λειτουργία της ανάκτησης δεδομένων από τη Firebase και η ανάθεσή τους στη λίστα, αλλά και η ίδια η λειτουργία της λίστας, το *ProgressDialog* υλοποιείται ή διαχειρίζεται εσωτερικά μιας *inner thread* κλάσης.

Η μέθοδος *public void onSuccess(DataSnapshot data)*, περιέχει κώδικα σχετικά με την ανάκτηση των δεδομένων από την real time Firebase βαση. Επίσης, μετά το πέρας της συγκεκριμένης λειτουργίας, σταματάει και κρύβει το *ProgressDialog thread*.

Πέρα από την προφανή κύρια λειτουργία της μεθόδου (ανάκτηση και απόδοση δεδομένων-λίστας με ιστορικά στοιχεία των αγορών, επιτυχημένων ή μη), είναι και η επιφόρτισή της με την επεξεργασία των στοιχείων της λίστας σχετικά με την ταξινόμηση. Η ταξινόμηση είναι φθίνουσα και περιέχει ομαδοποιημένες συναλλαγές που έχουν λάβει χώρα, ανά ημέρα. Αυτό γίνεται με σκοπό την προβολή πάντα ημερολογιακά της πιο πρόσφατης δραστηριότητας και της πιο πρόσφατης ημέρας.

Επιπλέον, σαν επικεφαλίδα (header) της Expandable List, πέρα από την προφανή αναφερόμενη ημέρα σαν ημερομηνία, αναγράφονται και επιπλέον πληροφοριακά στοιχεία που αναφέρονται σε αριθμητικά δεδομένα (αριθμός συνολικών συναλλαγών της συγκεκριμένης αναγραφόμενης ημέρας και των επιτυχυμένων και αποτυχυμένων επιχειρούμενων συναλλαγών). Έτσι, κατ' αυτόν τον τρόπο, προβάλλεται μια expandable λίστα, η οποία, διαθέτει σε οριζόντιες γραμμές, τις ημερομηνίες και τα προαναφερθέντα επιπλέον στοιχεία.

Η λίστα, όταν ο χρήστης επιλέγει μια συγκεκριμένη ημερομηνία και πατήσει πάνω της, προβάλλει μόνο τα εσωτερικά στοιχεία που αφορούν την επιλεγμένη μέρα. Κάθε στοιχείο της λίστας, είναι μια άλλη λίστα της οποίας κάθε γραμμή της, αποτελεί μια συλλογή από Android Views (xml) που είναι

ρυθμισμένα και στοιχισμένα ώστε στο σύνολό τους, να αποτελούν μια λογική εγγραφή. Κάθε λογική εγγραφή, έχει σχεδόν τον ίδιο αριθμό των Views που έχει και η οντότητα Payment (εικ. 74). Τέλος, θα πρέπει να τονισθεί πως ορισμένα στοιχεία της κάθε εγγραφής (record), έχουν πέρα από την προφανή ένδειξη επιτυχής ή όχι συναλλαγής, τα error codes και τα σχετικά συνοδευτικά μηνύματά τους με ανάλογο χρωματισμό για υποβοήθηση και διευκόλυνση του χρήστη προκειμένου να μπορεί να είναι σε θέση με μια γρήγορη ματιά, να διαπιστώσει αμέσως τα σημεία ενδιαφέροντος (εικ. 19).

6.2. Δεύτερο μέρος της εφαρμογής (Λογισμικό προσομείωσης της Τράπεζας).

Το δεύτερο μέρος της εφαρμογής, είναι ουσιαστικά ένα άλλο πρόγραμμα το οποίο έχει αναπτυχθεί με Java Spring Boot και Restful ως webservice, το οποίο προσομειώνει κάποια τυπική τράπεζα.

Το λογισμικό αυτό είναι ανεξάρτητο της πρώτης (Android QR Code) εφαρμογής και παρέχει υπηρεσίες ανάλογες με αυτές μιας οποιασδήποτε κοινής τράπεζας.

Τυπικές υπηρεσίες που προσφέρει είναι διάφοροι έλεγχοι σχετικά με το χρηματικό ισοζύγιο που διαθέτει ένας πελάτης, ελέγχους χρεωστικών καρτών, λογαριασμών όπως το IBAN, επίσης, κάνει χρεώσεις πελάτη και πιστώσεις επιτηδευματία.

Πρέπει σε αυτό το σημείο να αναφερθεί πως το λογισμικό ή η εφαρμογή σαν ολότητα, σκοπό έχει το concept proof στα πλαίσια επίδειξης μιας διπλωματικής εργασίας και όχι την ανάπτυξη μιας πραγματικής επαγγελματικής εφαρμογής. Σε μια πραγματική εφαρμογή, θα πρέπει να συνυπολογιστούν και να αναπτυχθούν και άλλοι παράγοντες όπως ενδεικτικά μπορεί να αναφερθεί το θέμα της ασφάλειας και άλλα τα οποία θα προβληθούν σε επόμενο κεφάλαιο.

Η εφαρμογή είναι δομημένη σε controllers, entities, services, mappers, repositories, responses και utilities. Η δομή της εφαρμογής, μπορεί να παρατηρηθεί στην (εικ. 34).

Ως μια ξεχωριστή εφαρμογή η οποία λειτουργεί ανεξάρτητα της πρώτης, το πρόγραμμα αυτό έχει μια δικό του σχήμα δεδομένων και το σχήμα της βάσης αυτής έχει υλοποιηθεί με *MySQL*.

Η εφαρμογή διαθέτει τέσσερα entities όπως τα*, User, Vendor, CreditCard* και *Payment*. Οι οντότητες αυτές, αποτελούν συνδυασμό Java και annotated κλάσεων και αναφέρονται στους πίνακες του σχήματος της παραπάνω βάσης δεδομένων.

Αν και οι ονομασίες των entities είναι προφανείς, θα περιγραφεί το ένα εξ' αυτών, το User entity, και τα υπόλοιπα έχουν ανάλογη ερμηνεία.

Η οντότητα (entity) User, όπως και κάθε άλλη, δεν αφορά απλώς μια Java class αλλά μια annotated Java class. Αυτό σημαίνει πως δεν είναι απαραίτητο να έχουμε κάποιο σχήμα της βάσης όπως ήδη είναι γνωστό από τις σχεσιακές

βάσεις, αλλά το σχήμα μπορεί δημιουργηθεί σε run time χρόνο και οι σχέσεις μεταξύ τους επίσης επιλύονται σε run time χρόνο την πρώτη φορά στην οποία η εφαρμογή θα ‘τρέξει’ (εικ. 67, 68).

Ακολουθεί μια περιγραφή των annotations, που φέρουν πάνω τους τα πεδία του entity User. Με αυτόν τον τρόπο, όλα τα υπόλοιπα entities, έχουν παρόμια annotations και ως εκ τούτου, οι παρακάτω περιγραφές, αναφέρονται στα κοινά τους στοιχεία περιγράφοντας το ίδιο και για τα υπόλοιπα entities.

Από την εικόνα αυτή (εικ. 67), μπορούμε να δούμε πως η Java κλάση έχει γίνει annotated με ένα hint *@Entity*. Ο ‘σχολιασμός’ με το *@Entity*, θα λέγαμε πως ‘σημειώνει’ ή ‘σημαδεύει’ την κλάση αυτή σαν ένα entity bean και ανήκει στην οικογένεια των JPA *javax.persistence* libraries. Αυτό το γεγονός, αυτομάτως προϋποθέτει η κλάση αυτή να έχει ρητά δηλωμένο ένα default empty constructor (εικ. 67, γραμμή 44). Τα beans χρησιμοπιούνται από το Spring framework και γίνονται γνωστά στην πλατφόρμα, μόλις η εφαρμογή ενεργοποιηθεί, (up and running). Μια Entity Bean είναι ένα component τύπου Enterprise JavaBean και αναπαριστά μόνιμα (persistent data) δεδομένα διατηρούμενα σε μια βάση δεδομένων.

Το annotation *@Table*, χρησιμοποιείται για καθορισμό ονόματος του πίνακα στην οποία θα διατηρηθούν τα δεδομένα. Η γραμμή 13 της (εικ. 67), προσδιορίζει ότι ο πίνακας για την annotated κλάση User θα είναι user.

Το *@Id* annotation, χρησιμοποιείται για να προσδιορίσει την ταυτότητα (identifier) ως ιδιότητα (property) της οντότητας Entity Bean.

Το *@GeneratedValue* annotation, χρησιμοποιείται για προσδιορισμό στρατηγικής της παραγωγής του πρωτεύοντα κλειδιού. Εν απουσία της αναφοράς της στρατηγικής, χρησιμοποιείται το default *AUTO*.

Το *@Column* annotation, χρησιμοποιείται για προσδιορισμό λεπτομεριών της στήλης του πίνακα στην οποία το field της Java κλάσης θα γίνει mapped.

Το *@OneToMany* annotation, χρησιμοποιείται για αυτόματη δημιουργία σχέσεων ένα προς πολλά. Αυτό σημαίνει πως ένας user διαθέτει πολλές κάρτες αλλά, ρητά, δεν ισχύει το αντίθετο. Δεν θα μπορούσε άλλωστε μια χρεωστική κάρτα να ανήκει ταυτόχρονα σε δύο χρήστες. Το ίδιο παρατηρείται και με τα payments. Ένας χρήστης κάνει πολλές πληρωμές που αναφέρονται στον ίδιο και από την άλλη, μια πληρωμή ανήκει μονάχα σε έναν χρήστη.

Τα τρία εναπομείναντα entities έχουν, σε γενικές γραμμές, την ίδια λειτουργία ως προς το σχολιασμό (annotations).

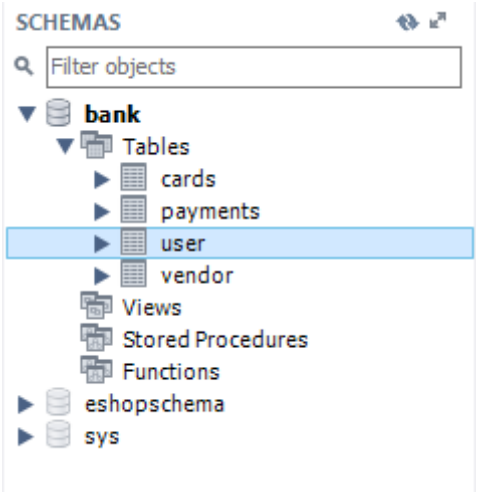
Θα πρέπει επίσης να αναφερθεί πως δεν χρησιμοποιήθηκε κάποιο *Object relational mapping (ORM),* όπως το *Hibernate* ή το *iBatis* για λόγους απλότητας, αλλα αντί αυτού χρησιμοποιήθηκε το *JdbcTemplate* το οποίο παρέχεται απο τα Spring libraries για πρόσβαση (access) στη βάση.

Το *JdbcTemplate*, απαιτεί ένα mapper class για κάθε οντότητα πίνακα, προκειμένου να αντλεί δεδομένα από και πρός τη βάση. Παρακάτω, θα ακολουθήσει η παρουσίαση ενός τέτοιου mapper για την οντότητα user και περίπου το ίδιο θα είναι και για τις τρεις άλλες οντότητες.

Στην (εικ. 69) και στη γραμμη 16, παρατηρείται πως η *UserMapper class* υλοποιεί την *RowMaper Interface* η οποία δέχεται μια παράμετρο τύπου User. Το *RowMaper Interface*, παρέχεται απο το Spring framework και επιτρέπει να κάνει ένα mapping μεταξύ μιας γραμμής (row) της σχέσης (relation), με την instance παράμετρο *User* που δέχεται. Εσωτερικά, κάνει ένα iteration στο αντικείμενο *ResultSet* που κάποιο query προηγουμένως έχει αιτηθεί και φέρει και τα προσθέτει στο collection. Επομένως, δεν απαιτείται μεγάλος όγκος κώδικα από τον προγραμματιστή. Η κλάση, *UserMapper*, όπως δείχνει η (εικ. 69), κάνει override μια μέθοδο *mapRow* και το μόνο που γίνεται είναι η πρόσθεση στη συλλογή του entity, των δεδομένων που εξάγονται από το *ResultSet* και ασφαλώς η επιστροφή του entity στο σημείο από που κλήθηκε.

```
/**
 * Created by andre on 10/18/2017.
 */

@Entity
@Table(name = "user")
public class User implements Serializable {

    @Id
    @GeneratedValue
    private Integer id;

    @Column(nullable = false)
    private String name;

    @Column(nullable = false)
    private Double balance;

    @Column(nullable = false)
    private String secretPin;

    @Column(nullable = false)
    private String number;

    @Column(nullable = false)
    private Integer tries;

    @Column(nullable = false)
    private String status;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private Set<CreditCard> cards    = new HashSet<>();

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private Set<Payment> payments    = new HashSet<>();

    public User() {}
```

(εικ. 67)

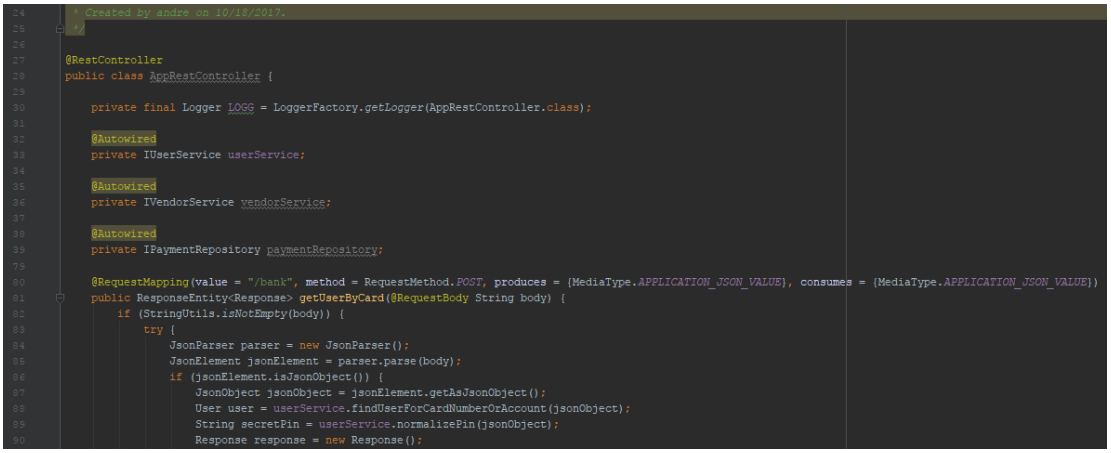


(εικ. 68)

```
/**
 * Created by andre on 10/22/2017.
 */
public class UserMapper implements RowMapper<User> {
    @Override
    public User mapRow(ResultSet resultSet, int i) throws SQLException {
        User user = new User();
        user.setId(resultSet.getInt( columnLabel: "id"));
        user.setName(resultSet.getString( columnLabel: "name"));
        user.setNumber(resultSet.getString( columnLabel: "number"));
        user.setBalance(resultSet.getDouble( columnLabel: "balance"));
        user.setSecretPin(resultSet.getString( columnLabel: "secretPin"));
        user.setStatus(resultSet.getString( columnLabel: "status"));
        user.setTries(resultSet.getInt( columnLabel: "tries"));

        Payment payment = new Payment();
        if (resultSet.getInt( columnLabel: "p.id") >= 0){
            payment.setId(resultSet.getInt( columnLabel: "p.id"));
            payment.setTime(resultSet.getString( columnLabel: "p.time"));
            payment.setProduct(resultSet.getString( columnLabel: "p.product"));
            payment.setReference(resultSet.getString( columnLabel: "p.reference"));
            payment.setPrice(resultSet.getDouble( columnLabel: "p.price"));
        }

        CreditCard creditCard = new CreditCard();
        creditCard.setId(resultSet.getInt( columnLabel: "c.id"));
        creditCard.setName(resultSet.getString( columnLabel: "c.name"));
        Set<CreditCard> cards = new HashSet<>();
        cards.add(creditCard);

        payment.setUser(user);

        Set<Payment> payments = new HashSet<>();
        payments.add(payment);
        user.setCards(cards);
        user.setPayments(payments);

        return user;
    }
}
```

(εικ. 69)

Ένα request το οποίο στέλνεται από την συσκευή μετά την σάρωση κάποιου QR Code προϊόντος, έχει την μορφή της (εικ. 42). Η url διεύθυνση, στην οποία αποστέλλεται, ρυθμίζεται από την Server Configuration οθόνη και έχει την μορφή: http://host:port/bank. Το σημείο άφιξης του request είναι ο controller της εφαρμογής της τράπεζας και πιο συγκεκριμένα το *AppRestController* (εικ. 70) και η μέθοδος *ResponseEntity<Response> getUserByCard(@RequestBody String body)* (εικ. 70, γραμμή 80).



```
24  * Created by andre on 10/18/2017.
25  */
26
27  @RestController
28  public class AppRestController {
29
30      private final Logger LOGG = LoggerFactory.getLogger(AppRestController.class);
31
32      @Autowired
33      private IUserService userService;
34
35      @Autowired
36      private IVendorService vendorService;
37
38      @Autowired
39      private IPaymentRepository paymentRepository;
79
80      @RequestMapping(value = "/bank", method = RequestMethod.POST, produces = {MediaType.APPLICATION_JSON_VALUE}, consumes = {MediaType.APPLICATION_JSON_VALUE})
81      public ResponseEntity<Response> getUserByCard(@RequestBody String body) {
82          if (StringUtils.isNotEmpty(body)) {
83              try {
84                  JsonParser parser = new JsonParser();
85                  JsonElement jsonElement = parser.parse(body);
86                  if (jsonElement.isJsonObject()) {
87                      JsonObject jsonObject = jsonElement.getAsJsonObject();
88                      User user = userService.findUserForCardNumberOrAccount(jsonObject);
89                      String secretPin = userService.normalizePin(jsonObject);
90                      Response response = new Response();
```

(εικ. 70)

Αρχικά παρατηρείται πως η κλάση *AppRestController*, είναι μια annotated κλάση με την σήμανση:

@RestController , η σήμανση αυτή προέρχεται από το Spring framework και είναι ένα annotation το οποίο είναι και το ίδιο σχολιασμένο (annotated) με τους *@Controller* και *@ResponseBody*.

@Controller annotation , χρησιμοποιείται για αυτόματη αναγνώριση των beans από το Spring framework χρησιμοποιώντας το classpath.

Από την άλλη, το *@ResponseBody* annotation, κάνει map το *HttpRequest body* σε ένα μεταφερούμενο αντικείμενο, ενεργοποιώντας αυτόματα το deserialization του ενσωματωμένου *HttpRequest body* σε ένα Java object.

Το *@Autowired annotation*, χρησιμοποιείται για dependency injections. Αυτό το annotation, επιτρέπει στο Spring framework να επιλύει σε run time και να κάνει inject τα συνεργαζόμενα beans στο τρέχον bean.

Το *@RequestMapping* annotation, χρησιμοποιείται για mapping των web requests σε μια συγκεκριμένη μέθοδο handler ή/και κλάση.

@RequestMapping(

value = "/bank",

method = RequestMethod.POST,

produces = {MediaType.APPLICATION_JSON_VALUE},

consumes = {MediaType.APPLICATION_JSON_VALUE})

value = "/bank" : κάνει mapping ένα request με κατάληξη το /bank

method = RequestMethod.POST : σημαίνει ότι μόνο η μέθοδος POST επιτρέπεται να διαχειριστεί αυτή τη μέθοδο/handler.

produces, consumes : σημαίνει ότι η αυτή μέθοδος/handler, δέχεται και παράγει μόνο δεδομένα ίδιου τύπου με αυτό που ακολουθεί το MediaType, δηλαδή τύπου JSON στη συγκεκριμένη περίπτωση.

Για τα *@Service* και *@Repository*, ισχύει ότι ισχύει και για το *@Controller* annotation.

Μετά από την περιγραφή των παραπάνω, μπορούν να κατανοηθούν καλύτερα τα παρακάτω.

Επομένως όσον αφορά την διεύθυνση http://host:port/bank , θα πρέπει να φαίνεται περισσότερο οικείο στο ερώτημα σε ποιο handler θα κάνει mapping στην κλάση controller. Παρατηρώντας το value = "/bank", το request θα κάνει στάση στη μέθοδο getUserByCard.

RequestMapping(value = "/bank", method = RequestMethod.POST, produces = {MediaType.APPLICATION_JSON_VALUE}, consumes = {MediaType.APPLICATION_JSON_VALUE})

public ResponseEntity<Response> getUserByCard(@RequestBody String body) {...}

Από το request της (εικ. 70), η πρώτη δουλειά είναι η μετατροπή του *RequestBody* , το οποίο είναι τύπου String σε τύπο json και αυτό γίνεται μέσω των παρακάτω δύο γραμμών κώδικα.

JsonElement jsonElement = parser.parse(body);

JsonObject jsonObject = jsonElement.getAsJsonObject();

Μετά την μετατροπή σε json, μπορούμε πλέον να κάνουμε μια ερώτηση στη βάση δεδομένων της MySQL για την ανάκτηση του χρήστη διαμέσου του credit card ή του αριθμού του λογαριασμού, ανάλογα της μεθόδου που επιλέχθηκε (εικ. 23) για να γίνει η αγορά. Η μέθοδος (εικ. 70 και γραμμή 88) έχει ως ακολούθως:

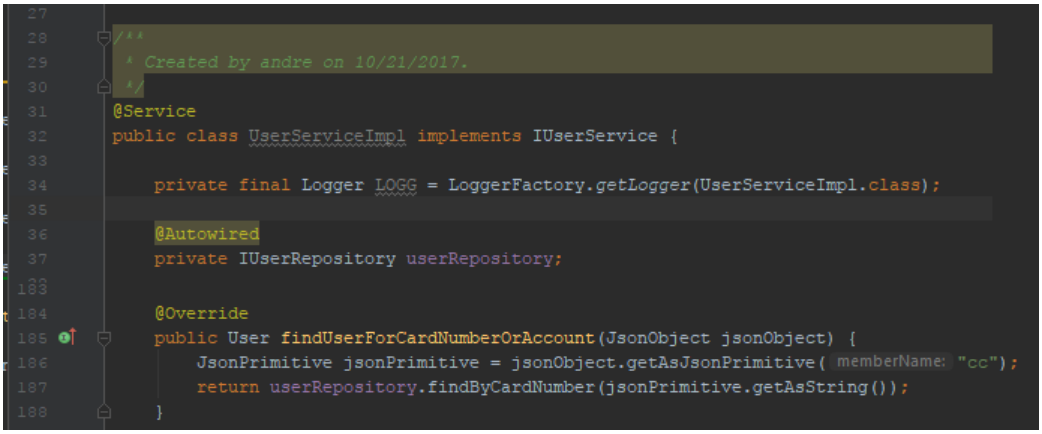
User user = userService.findUserForCardNumberOrAccount(jsonObject); (α)

Το userService, γίνεται inject στον controller με τον παρακάτω τρόπο.

@Autowired

private IUserService userService;

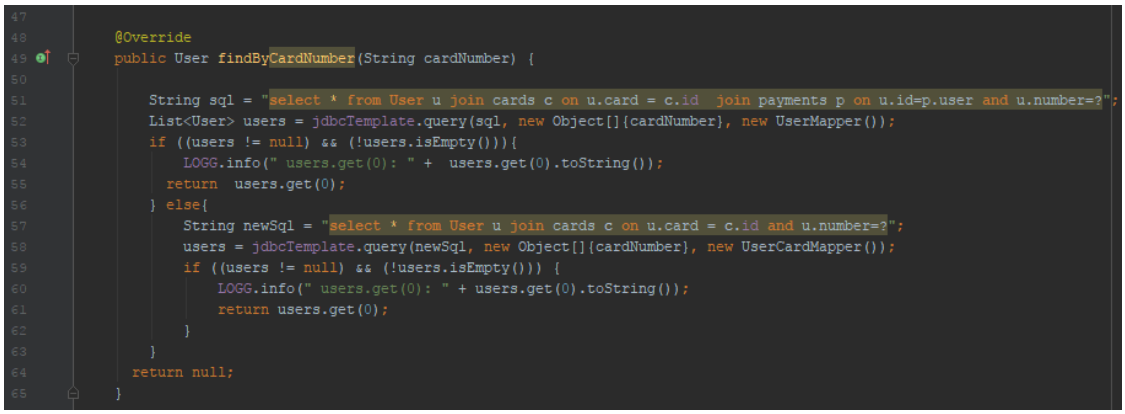
Το *userService*, είναι ένα Interface στο οποίο δηλώνονται μεταξύ άλλων μεθόδων και η ανωτέρω μέθοδος (α) και υλοποιείται (implemented) από το Service (εικ. 71).



(εικ. 71)

Στην παραπάνω (εικ. 71) και στη γραμμή 37, γίνεται autowired ένα άλλο bean τύπου repository και στην *findByCardNumber*. Πρόκειται για ένα άλλο Interface το οποίο υλοποιείται από την *UserRepository* class. Η *UserRepository*, είναι annotated με τη σειρά της με το *@Repository* annotation το οποίο αναφέρθηκε παραπάνω. Πρόκειται για μια DAO (Data Access Object) κλάση η οποία παρέχει πρόσβαση στη βάση δεδομένων.

Το σώμα της μεθόδου, φαίνεται στην παρακάτω (εικ. 72). Επίσης, μπορεί να προσεχθεί και το *UserMapper* στη γραμμή 53.



(εικ. 72)

Με την προϋπόθεση ότι κατά την ανάκτησή του δεν υπήρχε κάποιο σφάλμα, ο επόμενος έλεγχος είναι αυτός του Pin με την ακόλουθη μέθοδο: *response = userService.checkPin(responseHelper, user, secretPin);*

Αρχικά, αυτό που ελέγχεται είναι ο αριθμός των αποτυχημένων προσπαθιών. Έχοντας ήδη τον χρήστη από το προηγούμενο βήμα, ελέγχεται αν ο αριθμός των προηγούμενων αποτυχημένων προσπαθιών, δηλαδή ο αριθμός των συνεχόμενων λανθασμένων εισαγωγών Pin, είναι μεγαλύτερος ή ίσος με το τρία. Τρεις προσπάθειες, είναι ο ανώτερος αριθμός αποτυχημένων προσπαθειών που μπορεί το σύστημα να 'ανεχθεί' προτού ο χρήστης να απορριφθεί και το σύστημα απαντά με ένα κωδικό σφάλματος LOCKED και ένα σχετικό μήνυμα προσαρτάται στην απάντηση που δέχεται πίσω (response).

Σε περίπτωση που δίνει λάθος pin αλλά συνυπολογίζοντας με τον προηγούμενο αριθμό αποτυχυμένων προσπαθιών δεν έχει φθάσει το όριο των τριών προσπαθειών, τότε δέχεται ως απάντηση (response) ένα κωδικό σφάλματος INVALID_PIN και ένα αντίστοιχο/σχετικό και επεξηγητικό μήνυμα για ενημέρωση, ενώ παράλληλα ο συνολικός αριθμός αποτυχημένων προσπαθειών ενημερώνεται στη βάση (update), αυξημένος κατά ένα. Βέβαια, αξίζει να σημειωθεί πως αν δεν έχει φθάσει ο χρήστης το όριο των αποτυχημένων προσπαθειών και δώσει σωστό pin, ο συνολικός αριθμός αποτυχημένων προσπαθειών, μηδενίζεται.

Και στις δύο παραπάνω υποθέσεις, πέρα από τα προαναφερθέντα, η συναλλαγή υποχρεούται σε καταγκαναστική αποτυχία tok = false, pin = false καθώς επίσης και *transaction status = HttpStatus.BAD_REQUEST* διασφαλίζοντας κατά αυτόν τον τρόπο, πως η συναλλαγή δεν κινδυνεύει να θεωρηθεί επιτυχυμένη από λάθος εκτίμηση. Διαφορετικά, η εφαρμογή μεταφέρει το request στον επόμενο έλεγχο.

Επόμενο βήμα στην αλυσίδα του κύκλου ζωής του request, είναι ο έλεγχος του τραπεζικού χρηματικού ισοζυγίου του χρήστη. Στο βήμα αυτό, ελέγχεται αν το ποσό του balance είναι ικανό και αρκετά μεγάλο, τόσο ώστε να μπορεί να πληρωθεί τουλάχιστον το προϊόν προς αγορά. Αυτό, γίνεται με μια απλή πράξη αφαίρεσης της τιμής προς αγορά από το balance. Αν είναι η διαφορά ένας θετικός αριθμός, τότε, θα θεωρηθεί θετική ένδειξη ολοκλήρωσης της παραγγελίας και ο έλεγχος προχωρά παρακάτω, αλλιώς, ως απάντηση το response περικλείει ένα κωδικό σφάλματος *BALANCE_NOT_ENOUGH* αλλά και το σχετικό μήνυμα σφάλματος. Επίσης, η συναλλαγή αποτυχαίνει με τα ακόλουθα tok = false, pin = false αλλά και το transaction status τίθεται με τιμή *HttpStatus.BAD_REQUEST*.

Ο τελευταίος έλεγχος, πριν την πραγματοποίηση της συναλλαγής, είναι αυτός των χρεωστικών καρτών, αλλά και των άλλων πιθανών λογαριασμών του χρήστη, ανάλογα από ποια κάρτα ή λογαριασμό του έχει επιλέξει να κάνει την χρέωση. Σε περίπτωση κάποιου σφάλματος, ο κωδικός που αντιστοιχεί μπορεί να είναι κάποιος εκ των τύπων *PENDING, CLOSED, BLOCKED, BLACKLISTED* και *INACTIVE*.

Μια περίπτωση στο να βρεθεί μια κάρτα σε μια από τις παραπάνω καταστάσεις (status), είναι για παράδειγμα να εκρεμμεί η ενεργοποίηση κάποιας κάρτας και ο χρήστης πιθανότατα, να επιχείρησε κάποια συναλλαγή. Στην περίπτωση αυτή το σύστημα απαντά με *INACTIVE* σαν κωδικό σφάλματος. Μια άλλη περίπτωση, είναι η κάρτα να έχει γίνει blocked για κάποιο λόγο και αυτό είναι ένδειξη στον χρήστη ότι δεν θα μπορεί να κάνει, τουλάχιστον προσωρινά, καμία συναλλαγή με αυτή την κάρτα. Τέλος, όλες οι τράπεζες διαθέτουν κάποια λίστα με ανεπιθύμητους χρήστες όπως αυτή του terror list και ο χρήστης ελέγχεται αν εμφανίζεται το όνομά του στην blacklist και σε περίπτωση που διαπιστωθεί κάτι τέτοιο, τότε ένας κωδικός σφάλματος *BLACKLISTED* επιστρέφεται.

Για λόγους όμως επίδειξης των μόλις προαναφερθέντων, στην παρούσα εφαρμογή που δεν αφορά σε πραγματική τράπεζα, αλλά λειτουργεί

περισσότερο σαν ένα demo, έχει οριστεί τυχαία κάποια κάρτα χρήστη ρητά με ένδειξη PENDING ή BLOCKED ή μία εξ αυτών.

Σε περίπτωση θετικής έκβασης του request έως την παρούσα στιγμή, πλέον, είναι η κατάλληλη στιγμή να πραγματοποιηθεί η συναλλαγή. Η σειρά με την οποία πραγματοποιείται, είναι η ακόλουθη:

Αρχικά, αφαιρείται το χρηματικό ποσό από τον χρήστη και στη συνέχεια πιστώνεται ο επιτηδευματίας. Για αποφυγή ανεπιθύμητων καταστάσεων, όπως πιθανή χρέωση του χρήστη και μη πληρωμή του επιτηδευματία λόγω κάποιου σφάλματος ή το αντίστροφο, η μέθοδος η οποία είναι στην ευθύνη της η χρηματική συναλλαγή, δηλώνεται ως transactional μέθοδος, annotated με το *@Transactional annotation* (εικ. 73).

```
@Override
@Transactional
public Response purchase(ResponseHelper responseHelper, User user, JsonObject jsonObject) {
    JsonPrimitive jsonPrimitive = jsonObject.getAsJsonPrimitive( memberName: "transactionAmount");
```

(εικ. 73)

Το *@Transaction* annotation, είναι μια 'σήμανση' του Spring framework και αυτό που καταφέρνει είναι να διασφαλίσει ότι όλες οι κινήσεις στο εσωτερικό του σώματος της μεθόδου, είτε θα επιτυγχάνουν όλα σαν σύνολο βημάτων, είτε θα αποτυγχάνουν όλα τα βήματα, ακόμα και εκείνα που είχαν επιτύχει μέχρι τη στιγμή της συνάντησης του προβλήματος. Αυτό είναι το γνωστό rollback το οποίο προσφέρεται αυτόματα από το Spring framework αντί να αφεθεί στην διακριτική ευχέρεια του προγραμματιστή, διασφαλίζοντας και θωρακίζοντας έτσι την ασφάλεια της συναλλαγής.

Τέλος, η όλη διαδικασία/συνναλλαγή καταγράφεται, στην βάση του τραπεζικού συστήματος η οποία είναι ένα MySQL σχήμα. Καταγράφεται ποιος χρήστης εκκίνησε την συναλλαγή, τα ποσά, ο αριθμός της κάρτας ή του λογαριασμού και η χρονική στιγμή (timestamp) της αγοράς. Τα παραπάνω, αποθηκεύονται σε ένα Payment entity (εικ. 74).

```
10
11      @Entity
12      @Table(name = "payments")
13      public class Payment implements Serializable {
14
15          @Id
16          @GeneratedValue(strategy = GenerationType.AUTO)
17          private Integer id;
18
```

(εικ. 74)

Αυτή ήταν μια σύντομη αλλά και περιεκτική περιγραφή του δεύτερου μέρους της εφαρμογής.

Στη συνέχεια, θα αναφερθούν κάποια συμπεράσματα και προτάσεις για μελλοντικές επεκτάσεις/βελτιστοποίησεις.

7. Συμπεράσματα και μελλοντικές επεκτάσεις.

7.1. Συμπεράσματα

Με μεγάλη έκπληξη παρατηρούμε την εξέλιξη των έξυπνων συσκευών τηλεφωνίας. Οι δυνατότητές τους είναι αναμφισβήτητες. Πλέον, όχι μόνο θεωρούνται υπολογιστικά συστήματα, αλλά θα τολμούσε να πει κάποιος ότι έχουν ήδη ξεπεράσει σε δυνατότητες αλλά και σε χρήση τους προσωπικούς υπολογιστές.

Αυτό, άλλωστε, μαρτυρά και η παρούσα υλοποιημένη εφαρμογή. Η έξυπνη συσκευή, πέρα των υπολοίπων δυνατοτήτων, είναι σε θέση να χρησιμοποιηθεί όπως τα κοινά πορτοφόλια (wallets), παρέχοντας την δυνατότητα μιας απλής σάρωσης ενός κωδικοποιημένου πλαισίου, QR Code, κάτι που τελευταία βρίσκει μια άνθιση και ευρεία εφαρμογή. Μπορεί να βρεθεί πλέον τόσο σε υλικά, αλλά και σε άϋλα προϊόντα, είτε στην φυσική τους υπόσταση, είτε στο διαδίκτυο ή κάποιο περιοδικό, διαφημιστικό σποτ, σε κάποια κοινή θέα ακόμη και σε κάποια εικόνα, φωτογραφία κ.ο.κ. Τελευταία, η χρήση τους παρατηρήθηκε και στο λαϊκό λαχείο (εικ. 75).

Το πρόβλημα στις ημέρες μας είναι κυρίως η έλλειψη χρόνου. Εφαρμογές σαν και αυτή, επιδιώκουν να αμβλύνουν το πρόβλημα αυτό. Να απαλλάξουν τον χρήστη της συσκευής από χρονοβόρες διαδικασίες πληρωμών. Η σάρωση ενός QR Code, διαβάζει την κωδικοποιημένη πληροφορία, εξάγει και χρησιμοποιεί ένα επιλεγμένο μέρος της, μειώνοντας ή εξαλείφοντας την ανάγκη και το χρόνο που απαιτείται για πληκτρολόγηση, η οποία στις περισσότερες περιπτώσεις λειτουργεί ως αποτρεπτικός παράγοντας για την ολοκλήρωση κάποιας αγοράς.

Επίσης, η αποφυγή πληκτρολόγησης προσωπικών στοιχείων, είναι ένας άλλος λόγος αποθάρρυνσης του χρήστη να προχωρήσει στην ολοκλήρωση μιας συναλλαγής. Η εν λόγω εφαρμογή, συγκεντρώνει πολλά από τα ζητούμενα θετικά στοιχεία και αποφεύγει ακόμη περισσότερα από τα προαναφερθέντα αρνητικά, ελπίζοντας να αφήσει τον χρήστη της, ευχαριστημένο, δίνοντάς του μια αρκετά καλή εμπειρία χρήσης και προσδοκεί να γίνει ένα αναπόσπαστο και απαραίτητο μέρος της καθημερινότητάς του.

7.2. Μελλοντικές επεκτάσεις/Βελτιστοποίησεις

Σίγουρα, η εφαρμογή δεν μπορεί να θεωρηθεί πως είναι έτοιμη να αναρτηθεί στο play store της Google, τη δεδομένη χρονική στιγμή. Αυτό δε σημαίνει ότι η εφαρμογή δεν ανταποκρίνεται στις ανάγκες για τις οποίες δημιουργήθηκε, αλλά περισσότερο θα λέγαμε ότι είναι ένα proof concept, δηλαδή, πραγματοποιήθηκε ο αρχικός σχεδιασμός, υλοποιήθηκε και δουλεύει στην πράξη.

Σίγουρα, για να μπορεί να είναι σε θέση να υπάρχουν σκέψεις για επαγγεματική προώθηση, θα μπορούσαν να αλλάξουν αρκετά κυρίως σε επίπεδο διεπαφής χρήστη και σε design. Καλύτερα γραφικά από επαγγελματίες σχεδίων (designers), οπωσδήποτε θα βοηθούσαν πολύ, καθώς ο καλλωπισμός είναι το πρώτο στοιχείο το οποίο οι χρήστες προσέχουν.

Πέρα των επαγγελαμτικών interfaces, υπάρχουν και κάποια άλλα στοιχεία τα οποία θα πρέπει να προσεχθούν. Τέτοια είναι για παράδειγμα, το θέμα της ασφάλειας και της ιδιωτικότητας. Επειδή πρόκειται για χρηματικές συναλλαγές, θα πρέπει να προστεθούν ασφαλή πρωτόκολλα επικοινωνίας, όπως το πρωτόκολλο https, αντί του υλοποποιημένου http, καθώς επίσης, πρωτόκολλα κρυπτογράφισης τα οποία παρέχουν ασφάλεια επικοινωνίας πάνω από ένα δίκτυο υπολογιστών όπως το Transport Layer Security (TLS ή SSL). Αν και στη παρούσα υλοποίηση χρησιμοποιείται κρυπτογραφία για κάλυψη κάποιων στοιχείων, όπως είναι η μάσκα των αριθμών των καρτών και των λογαριασμών, αλλά και των pins, ωστόσο, με τα παραπάνω πρωτόκολλα, γίνεται κρυπτογράφηση όλης της επικοινωναίς που λαμβάνει χώρα στο διαδίκτυο.

Επιπλέον, για την πραγματική επικοινωνία με την τράπεζα, θα χρειαστεί να προσαρτηθούν κάποια modules, τα οποία αναλαμβάνουν την επικοινωνία με την τράπεζα. Ουσιαστικά, τα modules αυτά, υλοποιούν τα services που κάνει expose η τράπεζα. Συνήθως, τέτοια modules, βρίσκονται υλοποιημένα και έτοιμα για downloads έναντι μιας συνήθως μικρής αμοιβής (http://www.webit.bz/). Εκτός αυτού όμως, μπορούν να υλοποιηθούν και από τον ίδιο τον προγραμματιστή.

Κάποια επιπλέον λειτουργία, η οποία μπορεί να υποστηριχθεί από την εφαρμογή, είναι η μεταφορά χρημάτων, δηλαδή εμβάσματα, από κάποιον λογαριασμό σε κάποιον άλλο, είτε του ιδίου του χρήστη είτε σε λογαριασμό τρίτου. Cashout χρήματα σε κάποιο ταμείο και πληρωμές από έμπορα σε έμπορα και έμπορα σε προμηθευτή, καθώς και τακτοποίηση αποθήκης για τον επιτηδευματία με τακτική ενημέρωση αποθεμάτων (logistics).

Όσον αφορά το firebase database, τη δεδομένη στιγμή δεν έχουν τεθεί αυστηρά μέτρα ασφάλειας, κάτι το οποίο θα πρέπει οπωσδήποτε να προσεχθεί σε μια επαγγελματική έκδοση.

Τέλος, για την (εικ. 15), αναφέρθηκαν ορισμένα ζητήματα βελτιστοποίησης όπως οι επιλογές purchase goods και link your cards. Το πρώτο, θα μπορούσε να έχει την λειτουργία του πλήκτρου ‘Press to scan’ και όσον αφορά το δεύτερο, όπως προειπώθηκε, θα μπορούσε να οδηγούσε σε ξεχωριστές οθόνες για εισαγωγή και σύνδεση χρεωστικών καρτών με τον κύριο τραπεζικό

λογαριασμό του χρήστη, όπως και μια άλλη οθόνη για εισαγωγή και διαχείρηση χρηστών.

8. Βιβλιογραφία.

Pay with your bank mobile app using QR code
This is the fastest and most convenient payment method available.
https://tpay.com/en/mobile/qr-payment

What is Qr Code?
https://www.marketingtochina.com/the-ultimate-guide-to-qr-codes-in-china/

Masterpass QR Generation
https://www.mastercardlabs.com/masterpass-qr/

Masterpass QR How it works
https://developer.mastercard.com/product/masterpass-qr

Android QR Scan SDK
https://developer.mastercard.com/documentation/masterpass-qr-sdks-for-originating-institution#android-qr-scan-sdk

Looking for a barcode scanner to read QR codes and other 2D barcodes
How you can use Qr Codes in your business
http://www.waspbarcode.com/buzz/5-ways-qr-codes/

Are QR Codes useful for merchant payments in emerging markets?
https://www.gsma.com/mobilefordevelopment/programme/mobile-money/decoding-qr-codes-are-they-useful-for-merchant-payments-in-emerging-markets

How to Market with a QR Code Strategy
http://www.scanlife.com/blog/2013/07/how-to-market-with-a-qr-code-strategy/#sthash.E9bMNCVG.2dhEvFnX.dpbs

QR Codes for Marketing:
A Unique Way to Bridge Offline and Online Media
https://www.hswsolutions.com/services/mobile-web-development/qr-code-marketing/

Creative Ways to Use QR Codes for Marketing

https://www.fastcompany.com/1720193/13-creative-ways-use-qr-codes-marketing

Welcome to QR Pay
http://www.qrpay.com/

Use Amazon’s Barcode Scanner to Easily Buy Anything from Your Phone
https://www.howtogeek.com/97347/use-amazons-barcode-scanner-to-easily-buy-anything-from-your-phone/

Google Chrome gets its own QR code & barcode scanner
https://techcrunch.com/2017/02/02/google-chrome-gets-its-own-qr-code-barcode-scanner/

QR Code Reader and Scanner
This app is only available on the App Store for iOS devices.
https://itunes.apple.com/us/app/qr-code-reader-and-scanner/id388175979?mt=8

BHIM – Making India Cashless
https://www.bhimupi.org.in
https://play.google.com/store/apps/details?id=in.org.npci.upiapp&hl=en
Axis Bank mVisa - Merchant
http://apk-dl.com/axis-bank-mvisa-merchant

Bharat Interface for Money (BHIM) is a payment app that lets you make simple, easy and quick transactions using Unified Payments Interface (UPI).
Direct bank payments to anyone on UPI scanning their QR with the BHIM app.
https://www.bhimupi.org.in/what-we-do

Axis Bank QR Payment App
https://play.google.com/store/apps/details?id=com.axis.mobile&hl=en
https://www.axisbank.com/

Similar Apps
https://play.google.com/store/apps/collection/similar_apps_me.scan.android.client?clp=qgEeChwKFm1lLnNjYW4uYW5kcm9pZC5jbGllbnQQARgD%3AS%3AANO1ljJnpvE&hl=en

Firebase database.
https://firebase.google.com/docs/android/setup

E-Payments modules for banks.
http://www.webit.bz/