ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Τμήμα Μηχανικών Πληροφορικής

Ανάπτυξη διαδικτυακής πλατφόρμας διαχείρισης πιστωτικών υπηρεσιών

Παντελή Άννα

Α.Μ 3870

Εισηγητής: Νικόλαος Παπαδάκης

Ηράκλειο, Ιούνιος 2017

Υπεύθυνη δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών Τμήματος Μηχανικών Πληροφορικής του Τ.Ε.Ι. Κρήτης.

Στην οικογένεια μου και σε όλους όσους στάθηκαν
δίπλα μου στη διάρκεια των σπουδών μου,
με ιδιαίτερη εκτίμηση και αγάπη.

Περίληψη

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάλυση, μελέτη και ανάπτυξη μιας διαδικτυακής πλατφόρμας διαχείρισης πιστωτικών υπηρεσιών. Η πλατφόρμα αυτή ενσωματώνει και προσφέρει μέσα από ένα απλό και φιλικό περιβάλλον όλες τις βασικές λειτουργίες αγοράς, διαχείρισης και πληρωμής πιστωτικών υπηρεσιών από τους πελάτες καθώς και εξειδικευμένες λειτουργίες διαχειρίσεις προς τους διαχειριστές της πλατφόρμας. Για την ανάπτυξη της πλατφόρμας αυτής αναζητήθηκαν και χρησιμοποιήθηκαν τα καταλληλότερα εργαλεία και τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών.

Abstract

The subject of this thesis is the analysis, research and development of an online financial service management platform. Through a simple and user friendly interface the platform integrates and offers all the basic functionalities (buy/manage/pay) to the customers, as well as advanced management tools to the platform managers. To develop this platform researched the most suitable tools and technologies for developing web applications.

Περιεχόμενα

Περίληψη iii
Abstract iv
Περιεχόμενα v
Λίστα Εικόνων vii
Λίστα Πινάκων ix
Ακρωνύμια x
Ευχαριστίες xi
1 Εισαγωγή 1
1.1 Στόχοι 1
1.2 Δομή 1
2 Θεωρητικό και Τεχνολογικό Υπόβαθρο 2
2.1 HTML 2
2.2 CSS 2
2.2.1 Semantic UI 3
2.3 JavaScript 3
2.3.1 jQuery 4
2.4 SQLite 4
2.5 Διαδικτυακές Εφαρμογές (Web Application) 4
2.6 Web Frameworks 5
2.7 Αρχιτεκτονική MVC (Model-View-Controller) 5
2.8 Αντικείμενο-Σχεσιακής Απεικόνισης (ORM) 6
3 Django Framework 8
3.1 Περιβάλλον εργασίας Django 8
3.2 Εγκατάσταση του Django 9
3.3 Δημιουργώντας μια Django εφαρμογή 11
3.3.1 Δημιουργία ενός Django Project 11
4 Διαδικτυακής πλατφόρμας διαχείρισης πιστωτικών υπηρεσιών 20
4.1 Προδιαγραφές 20
4.2 Δομή της Εφαρμογής 21
4.2.1 Accounts 21
4.2.2 Insurances 22
4.2.3 Loans 24
4.2.4 Reports 25
4.2.5 Dashboard 25
4.2.6 Website 25
4.3 Η Πλατφόρμα 26

4.3.1 Αρχική σελίδα 26
4.3.2 Αγορά νέου δανείου 26
4.3.3 Αγορά νέας ασφάλειας 28
4.3.4 Σύνδεση εγγεγραμμένου χρήστη 29
4.3.5 Διαχείριση ασφαλειών 29
4.3.6 Πληρωμή ασφάλειας 30
4.3.7 Διαχείριση δανείων 31
4.3.8 Πληρωμή δανείου 32
4.3.9 Διαχείριση Προφίλ 33
4.3.10 Περιβάλλον Διαχειριστή 33
5 Συμπεράσματα 39
6 Βιβλιογραφία 40

Λίστα Εικόνων

Εικόνα 2-1 Παράδειγμα HTML σελίδας 2
Εικόνα 2-2 Παράδειγμα CSS μορφοποίησης 3
Εικόνα 2-3 Παράδειγμα JavaScript κώδικα 4
Εικόνα 2-4 Αρχιτεκτονική MVC 6
Εικόνα 2-5 Παράδειγμα Αντικείμενο-Σχεσιακής Απεικόνισης (ORM) 7
Εικόνα 3-1 Εντολή επαλήθευσης της εγκατάστασης Python 9
Εικόνα 3-2 Εντολή επαλήθευσης της εγκατάστασης του pip 10
Εικόνα 3-3 Εντολή εγκατάστασης του Django 10
Εικόνα 3-4 Εντολή επαλήθευσης της εγκατάστασης του Django 11
Εικόνα 3-5 Δημιουργία ενός Django Project 11
Εικόνα 3-6 Εκτέλεση ενός Django Application 12
Εικόνα 3-7 Σελίδα καλωσορίσματος Django εφαρμογής 13
Εικόνα 3-8 Εκτέλεση ενός Django Application στην πόρτα 8080 13
Εικόνα 3-9 Εκτέλεση ενός Django Application στην διεύθυνση 0.0.0.0:8080 13
Εικόνα 3-10 Δημιουργία μιας Django εφαρμογής (app) 14
Εικόνα 3-11 Ρύθμιση SQLite βάσης δεδομένων στο settings.py 15
Εικόνα 3-12 Δημιουργία των πινάκων στην βάση δεδομένων 15
Εικόνα 3-13 Παράδειγμα View 15
Εικόνα 3-14 Παράδειγμα δημιουργίας URL 16
Εικόνα 3-15 Αποτελέσματα της σελίδας myView 16
Εικόνα 3-16 Παράδειγμα δημιουργίας URL με όρισμα 17
Εικόνα 3-17 Παράδειγμα View με όρισμα 17
Εικόνα 3-18 Αποτελέσματα της σελίδας myView με όρισμα 17
Εικόνα 3-19 Ρύθμιση του Template συστήματος στο settings.py 18
Εικόνα 3-20 Παράδειγμα Template αρχείου 18
Εικόνα 3-21 Παράδειγμα View με χρήση template 19
Εικόνα 3-22 Αποτελέσματα της σελίδας myView με χρήση template 19
Εικόνα 4-1 Δομή της εφαρμογής 21
Εικόνα 4-2 Μοντέλο λογαριασμού χρήστη 22
Εικόνα 4-3 Accounts URLs 22
Εικόνα 4-4 Μοντέλα ασφάλειας 23
Εικόνα 4-5 Insurances URLs 23
Εικόνα 4-6 Μοντέλα δανείου 24
Εικόνα 4-7 Loans URLs 24
Εικόνα 4-8 Reports URLs 25
Εικόνα 4-9 Dashboard URLs 25
Εικόνα 4-10 Website URLs 25
Εικόνα 4-11 Στιγμιότυπο αρχικής σελίδας 26
Εικόνα 4-12 Στιγμιότυπο αγοράς νέου δανείου 27
Εικόνα 4-13 Στιγμιότυπο αγοράς νέου δανείου (εγγεγραμμένος χρήστης) 27
Εικόνα 4-14 Στιγμιότυπο αγοράς νέας ασφάλειας 28
Εικόνα 4-15 Στιγμιότυπο αγοράς νέας ασφάλειας (εγγεγραμμένος χρήστης) 28
Εικόνα 4-16 Στιγμιότυπο σύνδεσης εγγεγραμμένου χρήστη 29
Εικόνα 4-17 Στιγμιότυπο διαχείρισης ασφαλειών 29
Εικόνα 4-18 Στιγμιότυπο διαχείρισης πληρωμών ασφάλειας 30
Εικόνα 4-19 Στιγμιότυπο διαχείρισης ασφαλειών (Εκπρόθεσμο) 30
Εικόνα 4-20 Στιγμιότυπο πληρωμής ασφάλειας 31
Εικόνα 4-21Στιγμιότυπο διαχείρισης δανείων 31

Εικόνα 4-22 Στιγμιότυπο διαχείρισης πληρωμών δανείου 32
Εικόνα 4-23 Στιγμιότυπο διαχείρισης δανείου (Εκπρόθεσμο) 32
Εικόνα 4-24 Στιγμιότυπο πληρωμής δανείου 33
Εικόνα 4-25 Στιγμιότυπο διαχείρισης προφίλ 33
Εικόνα 4-26 Στιγμιότυπο διαχείρισης χρηστών 34
Εικόνα 4-27 Στιγμιότυπο δημιουργίας νέου χρήστη 34
Εικόνα 4-28 Στιγμιότυπο διαγραφής χρήστη 35
Εικόνα 4-29 Στιγμιότυπο διαχείρισης στοιχείων του χρήστη 35
Εικόνα 4-30 Στιγμιότυπο διαχείριση κατηγοριών των ασφαλειών 36
Εικόνα 4-31 Στιγμιότυπο δημιουργίας κατηγορίας ασφάλειας 36
Εικόνα 4-32 Στιγμιότυπο επεξεργασίας κατηγορίας ασφάλειας 36
Εικόνα 4-33 Στιγμιότυπο αναφορών πελατών 37
Εικόνα 4-34 Στιγμιότυπο αναφορών ασφαλιστηρίων 37
Εικόνα 4-35 Στιγμιότυπο μηνιαίων αναφορών 38

Λίστα Πινάκων

Πίνακας 3-1 Βασικά αρχεία Django project 12
Πίνακας 3-2 Σύμβολα κανονικών εκφράσεων (regex) 17
Πίνακας 3-3 Παράδειγμα κανονικών εκφράσεων (regex) 17

Ακρωνύμια

HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
MVC	Model View Controller
ORM	Object Relational Mapping
IDE	Integrated Development Environment

Ευχαριστίες

Ιδιαίτερες ευχαριστίες προς τον επιβλέποντα καθηγητή μου κύριο Νικόλαο Παπαδάκη για την συνδρομή και την καθοδήγηση που μου προσέφερε καθ' όλη την διάρκεια συγγραφής της πτυχιακής εργασίας μου. Επιπλέον, σε όλους του καθηγητές και το προσωπικό του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Ηρακλείου Κρήτης, όντας πάντα πρόθυμοι να βοηθήσουν και να υποστηρίξουν τους φοιτητές με τον καλύτερο δυνατό τρόπο.

1 Εισαγωγή

Στην παρούσα πτυχιακή έγινε ανάλυση, μελέτη και ανάπτυξη μιας διαδικτυακής πλατφόρμας διαχείρισης πιστωτικών υπηρεσιών. Η πλατφόρμα αυτή ενσωματώνει και προσφέρει μέσα από ένα απλό και φιλικό περιβάλλον όλες τις βασικές λειτουργίες αγοράς, διαχείρισης και πληρωμής πιστωτικών υπηρεσιών από τους πελάτες καθώς και εξειδικευμένες λειτουργίες διαχειρίσεις προς τους διαχειριστές της πλατφόρμας. Αναλυτικότερα οι χρήστης της πλατφόρμας έχουν την δυνατότητα εγγραφής και στην συνέχεια μπορούν να προχωρήσουν στην αγορά κάποιου δανείου ή ασφάλειας. Ο χρήστης μπορεί μέσα από την πλατφόρμα να βλέπει την κατάσταση των δάνειων και των ασφαλών που έχει αγοράσει και να τα εξοφλήσει μέσα από αυτή καθώς η πλατφόρμα προσφέρει δυνατότητες πληρωμής. Αντίστοιχα οι διαχειριστές της πλατφόρμας έχουν την δυνατότητα μέσα από εξιδεικευμένα εργαλεία που προσφέρει να παρακολουθούν την κατάσταση των δανείων και των ασφαλειών του κάθε πελάτη.

Για τη δημιουργία της πλατφόρμας χρησιμοποιήθηκε το Django framework το οποίο είναι γραμμένο στη γλώσσα Python. Για την παρουσίαση των δεδομένων στον χρήστη έχει χρησιμοποιηθεί το CSS Framework Semantic-UI καθώς και η Javascript βιβλιοθήκη JQuery. Για την αποθήκευση των δεδομένων έγινε χρήση της βάσης δεδομένων SQLite. Οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας θα περιγράφουν αναλυτικά στο κεφάλαιο 2.

1.1 Στόχοι

Μέσα από την δημιουργία της συγκεκριμένης πλατφόρμας θα μου δοθεί η ευκαιρία να αποκτήσω εμπειρία πάνω στο κομμάτι της ανάπτυξης διαδικτυακών εφαρμογών και υπηρεσιών χρησιμοποιώντας νέες τεχνολογίες οι οποίες θα μου προσφέρουν γνώσεις και θα μου δώσουν τα κατάλληλα εφόδια για την ομαλότερη μετάβαση στην αγορά εργασία και την επαγγελματική αποκατάσταση.

1.2 Δομή

Η πτυχιακή εργασία ξεκινάει με την εισαγωγική παρουσιάζοντας το αντικείμενο το οποίο πραγματεύεται, στην συνέχεια στο κεφάλαιο 2 αναλύεται το θεωρητικό υπόβαθρο των τεχνολογιών και των εργαλείων που επιλέχτηκαν και χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας. Στο κεφάλαιο 3 θα γίνει αναλυτική παρουσίαση του Django Framework όπου είναι το κύριο Framework που χρησιμοποιήθηκε για την ανάπτυξη της πλατφόρμας. Στο κεφάλαιο 4 ακολουθεί αναλυτική παρουσίαση του σχεδιασμού και της υλοποίησης της πλατφόρμας. Το κεφάλαιο 5 αποτελεί μία σύνοψη της πτυχιακής εργασίας και παρατίθενται κάποια συμπεράσματα καθώς και κάποιες πιθανές επεκτάσεις της πλατφόρμας. Τέλος ακολουθεί η βιβλιογραφία και οι πηγές μελέτης.

2 Θεωρητικό και Τεχνολογικό Υπόβαθρο

Σε αυτό κεφάλαιο αναφέρονται βασικές πληροφορίες για τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας βοηθώντας τους αναγνώστες που δεν είναι εξοικειωμένοι με αυτά.

2.1 HTML

Η HTML είναι μια γλώσσα σήμανσης (Markup Language) η οποία χρησιμοποιείται κυρίως για την κατασκευή ιστοσελίδων. Η γλώσσα κάνει χρήση ενός συνόλου από ετικέτες (tags), δηλαδή λέξεις οι οποίες περικλείονται από τα σύμβολα < (μικρότερο από) και > (μεγαλύτερο από). Χρησιμοποιώντας αυτές τις ετικέτες μπορούμε να μορφοποιήσουμε κείμενο, να εισάγουμε πολυμεσικό περιεχόμενο όπως εικόνες και βίντεο η να δημιουργήσουμε συνδέσμους (links). Τα ονόματα των HTML αρχείων έχουν επέκταση .html και η δημιουργία τους μπορεί να γίνει με απλά προγράμματα επεξεργασίας κειμένου αλλά και με εξειδικευμένα προγράμματα ανάπτυξης ιστοσελίδων.

Εικόνα 2-1 Παράδειγμα HTML σελίδας

2.2 CSS

Η CSS είναι μια γλώσσα μορφοποίησης και χρησιμοποιείτε για να μορφοποιήσουμε μια HTML σελίδα. Το CSS ουσιαστικά περιγράφει τον τρόπο με τον οποίο θα γίνει η παρουσίαση της σελίδας, δίνοντας τα χαρακτηριστικά που θέλουμε να έχουν οι διάφορες ετικέτες της HTML σελίδας, όπως για παράδειγμα το χρώμα, την γραμματοσειρά και την διάταξη. Η δημιουργία τους μπορεί να γίνει με απλά προγράμματα επεξεργασίας κειμένου αλλά και με εξειδικευμένα προγράμματα ανάπτυξης ιστοσελίδων.

```css
h1 {
    font-family: courier, courier-new, serif;
    font-size: 20pt;
    color: blue;
    border-bottom: 2px solid blue;
}
p {
    font-family: arial, verdana, sans-serif;
    font-size: 12pt;
    color: #6B6BD7;
}
.red_txt {
    color: red;
}
```

Εικόνα 2-2 Παράδειγμα CSS μορφοποίησης

2.2.1 Semantic UI

Το Semantic-UI είναι μια από τις πιο εύχρηστες, ισχυρές και μοντέρνες CSS βιβλιοθήκες που χρησιμοποιούνται για την ανάπτυξης της εμφάνισης (Front-End) μιας διαδικτυακής εφαρμογής. Προσφέρει έναν μεγάλο αριθμό από έτοιμα στοιχεία που μπορούν να χρησιμοποιηθούν απευθείας και πολύ εύκολα στην εφαρμογή. Ο στόχος της βιβλιοθήκης είναι να βοηθήσει τους προγραμματιστές να δημιουργούν πολύ εύκολά και γρήγορα όμορφες εφαρμογές χωρίς να χρειάζεται να καταναλώσουν μεγάλο χρόνο στο εικαστικό κομμάτι της εφαρμογής.

2.3 JavaScript

Η JavaScript είναι μια γλώσσα η οποία χρησιμοποιείται για να μπορούμε να διαχειριζόμαστε δυναμικά το περιεχόμενο της ιστοσελίδας στην μεριά του πελάτη (client side). Το περιβάλλον εκτέλεσης της γλώσσας βρίσκετε στον περιηγητή ο οποίος αναλαμβάνει να εκτελέσει την κώδικα δίνοντας έτσι την δυνατότητα να δημιουργήσουμε διαδραστικότητας στις σελίδες μας. Για να γράψουμε κώδικα JavaScript κάνουμε χρήση της ετικέτας <script> γράφοντας μέσα σε αυτό τις εντολές που θέλουμε να εκτελεστούν.

```
<!DOCTYPE html>
<head>
<meta charset="utf-8">
<title>JavaScript Hello World</title>
</head>
<body>
<h1>First JavaScript</h1>
<hr>
<script>
  document.write("Hello World from JavaScript!");
</script>
</body>
</html>
```

Εικόνα 2-3 Παράδειγμα JavaScript κώδικα

2.3.1 jQuery

Η jQuery είναι μια από τις δημοφιλέστερες βιβλιοθήκες της JavaScript που χρησιμοποιείτε για να κάνει πιο εύκολη την χρήση της στις σελίδες. Η βιβλιοθήκη προσφέρει έναν πολύ μεγάλο αριθμό λειτουργιών και δυνατοτήτων απλοποιώντας κατά πολύ τον προγραμματισμό, καθώς με την χρήση της μπορούν να υλοποιηθούν πολύ πιο απλά και εύκολα εργασίες σε σχέση με την συμβατική χρήση της JavaScript.

2.4 SQLite

Η SQLite είναι ένα απλό και εύχρηστο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και έχει πολύ μικρές απαιτήσεις σε υπολογιστικούς πόρους. Η SQLite δεν ακολουθεί το κλασικό πελάτη-διακομιστή (client-server) μοντέλο διαχείρισης βάσεων δεδομένων. Κάνει χρήση αυτόνομων διεργασιών με τις οποίες επικοινωνεί με την εφαρμογή και αποθηκεύει όλα τα δεδομένα της βάσης ως ένα μοναδικό αρχείο στο μηχάνημα. Ένα πρόγραμμα με την χρήση της SQLite μπορεί καλώντας απλές συναρτήσεις να έχει πρόσβαση στα δεδομένα της βάσης πού εύκολα και γρήγορα.

2.5 Διαδικτυακές Εφαρμογές (Web Application)

Οι διαδικτυακές εφαρμογές ονομάζουμε το λογισμικό το οποίο εγκαθίστασαι σε έναν η περισσότερους υπολογιστές, συνήθως σε δικτυακούς εξυπηρετικές και είναι προσβάσιμο μέσω του διαδικτύου με την χρήση φυλλομετρητή (Web Browser). Το κύριο πλεονέκτημα των διαδικτυακών εφαρμογών σε σχέση με τις παραδοσιακές εφαρμογές είναι ότι η πρόσβαση σε αυτές μπορεί να γίνει απομακρυσμένα και χωρίς να είναι απαραίτητη η εγκατάσταση κάποιου προγράμματος τοπικά τον υπολογιστή. Μια διαδικτυακή εφαρμογή προσφέρει ένα περιβάλλον εργασίας όπου μπορεί να εκτελέσει διάφορες λειτουργίες να επεξεργαστεί δεδομένα αλλά και να πάρει πληροφορίες. Οι διαδικτυακές εφαρμογές αποτελούνται από 2 επίπεδα αρχιτεκτονική, το Front-End και το Back-End.

- **Front-End** είναι το τμήμα την εφαρμογής που τρέχει και είναι ορατή στον χρήστη και επιτρέπει σε αυτόν να αλληλεπιδράσει με την εφαρμογή. Για την ανάπτυξη του Front End χρησιμοποιούνται τεχνολογίες όπως HTML, CSS και JavaScript.

- **Back-End** είναι το τμήμα της εφαρμογής η οποία τρέχει στην μεριά του εξυπηρετική είναι υπεύθυνο για την διαχείριση των δεδομένων και εκεί υλοποιείται όλη η λογική της εφαρμογής.

2.6 Web Frameworks

Η διάδοσή των διαδικτυακών εφαρμογών δυναμικού περιεχομένου κυρίως με την χρήση της γλώσσας PHP συνέβαλλε στην εύκολη ανάπτυξη διαδικτυακών εφαρμογών προσφέροντας στους προγραμματιστές πολλές ευκολίες ανάπτυξης και σχετικά εύκολη και γρήγορη εκμάθηση. Ωστόσο η ανάπτυξη διαδικτυακών εφαρμογών είναι μία πολύπλοκη διαδικασία και μπορεί να οδηγήσει σε αρκετά σχεδιαστικά λάθη και σε μεγάλο όγκο επαναλαμβανόμενου κώδικα. Το πρόβλημα αυτό έρχονται να λύσουν τα Web Frameworks όπως το Django το οποίο και θα αναλύσουμε πλήρως στο κεφάλαιο 3. Σκοπός των Frameworks είναι να βοηθήσουν στην μείωση του χρόνου ανάπτυξης κάθε εφαρμογή παρέχοντας τα κατάλληλα εργαλεία στους προγραμματιστές. Τα κύρια πλεονεκτήματα της χρήση ενός Web Framework είναι τα παρακάτω:

- **Ταχύτερη ανάπτυξη**: Τα Frameworks περιέχουν μεγάλο αριθμό από έτοιμες βιβλιοθήκες και εργαλεία τα οποία μπορούνε να χρησιμοποιηθούν για την ταχύτερη ανάπτυξη των εφαρμογών. Για παράδειγμα παρέχει τις βιβλιοθήκες που χρειάζεται για να επικοινωνήσει η εφαρμογή με την βάση δεδομένων.
- **Ασφάλεια**: Η ανάπτυξη εφαρμογών από χρήστες που δεν έχουν μεγάλη εμπειρία στον προγραμματισμό μπορεί να δημιουργήσει πολλά προβλήματα και κενά στην ασφάλεια των εφαρμογών μας. Τα Frameworks παρέχουν αυξημένη ασφάλεια προσφέροντας έτοιμες βιβλιοθήκες οι οποίες μπορούν να χρησιμοποιηθούν για την πιστοποίηση των χρηστών, την διαχείριση των δικαιωμάτων καθώς και πλήθος άλλων χαρακτηριστικών όπως SQL Injections και άλλα συναφή.
- **Υποστήριξη**: Τα Frameworks, και ειδικότερα τα ανοιχτού κώδικα, υποστηρίζονται συνήθων από μεγάλες κοινότητες χρηστών. Σε περίπτωση που αντιμετωπίζεται κάποιο πρόβλημα είναι πολύ εύκολο να απευθυνθείτε στην κοινότητα και να λάβετε άμεσα βοήθεια στο πρόβλημα σας, η και να βρείτε λύση από τις γνωσιακές βάσεις των κοινοτήτων που συνήθων είναι διαθέσιμες. Οι κοινότητες των χρηστών συνήθως παρέχουν και ένα μεγάλο αριθμό από επεκτάσεις που μπορείτε να χρησιμοποιήσετε, επεκτάσεις που μπορεί να προσφέρουν διασύνδεση σε τρίτες εφαρμογές.

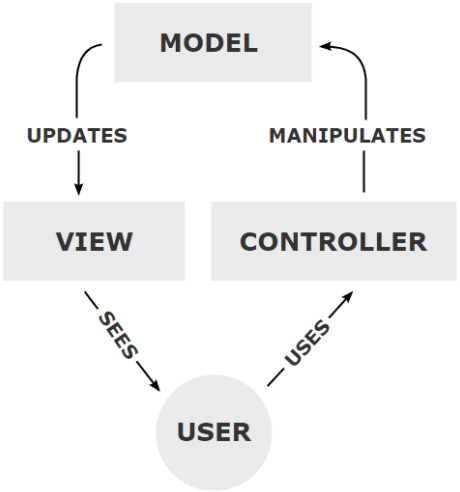
2.7 Αρχιτεκτονική MVC (Model-View-Controller)

Το MVC είναι ένα μοντέλο αρχιτεκτονικής ανάπτυξης λογισμικού το οποίο χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Η χρήση της συγκεκριμένης αρχιτεκτονικής διαχωρίζοντας τις οντότητες και την κατανομή της εργασίας συμβάλει σημαντικά στην ευκολότερη υλοποίηση και συντήρηση την εφαρμογής καθώς επίσης βοηθάει σημαντικά στον αποδοτικότερο σχεδιασμό και διευκολύνει την διαδικασία αποσφαλμάτωσης. Στο συγκεκριμένο μοντέλο η εφαρμογή αποτελείτε από 3 βασικά μέρη.

- **Model** - Στο Model τοποθετούνται όλες οι λειτουργίες της εφαρμογής που αφορούν την πρόσβαση στη βάση δεδομένων. Μέσα από το Model

μπορούμε να εκτελέσουμε λειτουργίες διαχείρισης των δεδομένων που λαμβάνουμε από τη βάση.

- **View** - Το View είναι υπεύθυνο για την παρουσίαση των δεδομένων, περιέχει τον HTML κώδικας της σελίδας της εφαρμογής.
- **Controller** - Ο Controller είναι αυτός που επικοινωνεί με το Model και το View, είναι αυτός που επεξεργάζεται τα δεδομένα και τα στέλνει στο View για να εμφανιστούν.



Εικόνα 2-4 Αρχιτεκτονική MVC

Όπως φαίνεται και στην Εικόνα 2-4 ο χρήσης βλέπει τα αποτελέσματα από το View και έχει την δυνατότητα να αλληλεπίδραση με τον Controller. Στην συνέχεια ο Controller μπορεί να ενημερώσει το Model βάσει της αλληλεπίδραση του χρήστη και τέλος το View να ενημερωθεί με αλλαγές που έχουν γίνει στο Model.

Η χρήση της αρχιτεκτονικής MVC κάνει πολύ πιο εύκολη και αποτελεσματική την παράλληλη ανάπτυξη εφαρμογών. Μπορεί πολύ εύκολα η λογική της ανάπτυξης της εφαρμογής να σπάσει σε επιμέρους κομμάτια. Ενας προγραμματιστής μπορεί να εργαστεί για την ανάπτυξη του View ενώ παράλληλα κάποιος άλλος να εργάζεται για την ανάπτυξη του Controller για να προγραμματίσει την λογική την εφαρμογής και κάποιος άλλος να αναπτύξει τα Models. Έτσι με αυτό τον τρόπο η ανάπτυξη των εφαρμογών μπορεί να γίνει πολύ πιο γρήγορη.

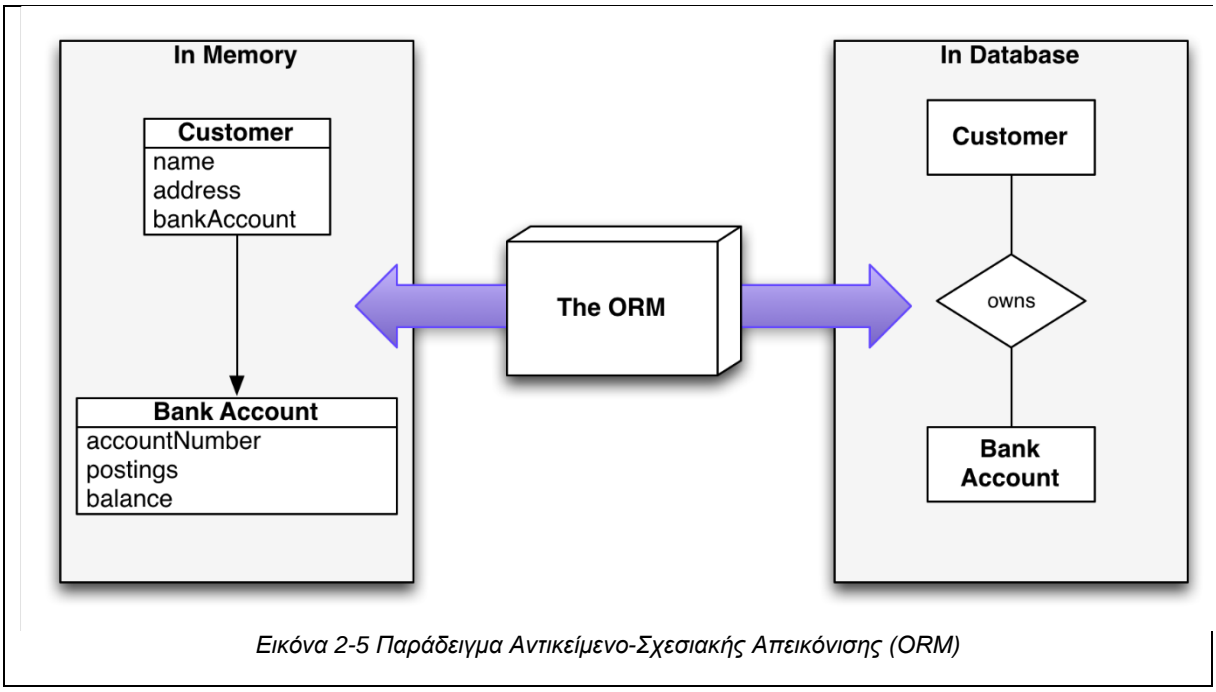
Το μοντέλο αυτό είναι πολύ αποτελεσματικό όταν έχουμε να κάνουμε με εφαρμογές οι οποίες έχουν αυξημένη πολυπλοκότητα και μας βοηθάει να έχουμε μια πολύ καλύτερη διαχείριση του κώδικα. Οι προγραμματιστές έχουν την δυνατότητα να επικεντρωθούν σε ένα μέρος του προγράμματος κάθε φορά όπως για παράδειγμα να συγκεντρωθούν στην ανάπτυξη του View χωρίς να εξαρτώνται από την λογική του προγράμματος η τα μοντέλα.

2.8 Αντικείμενο-Σχεσιακής Απεικόνισης (ORM)

Κατά την σχεδίαση μια εφαρμογής ένας από τους πιο σημαντικούς παράγοντές που λαμβάνουμε υπόψιν είναι ο τρόπος αποθήκευσης των δεδομένων. Ο πιο διαδεδομένος τρόπος αποθήκευσης δεδομένων είναι κάνοντας χρήση σχεσιακών βάσεων δεδομένων όπου αποτελούν έναν αποδοτικό και εύχρηστό τρόπο αποθήκευσης και διαχείρισης δεδομένων. Κατά την ανάπτυξη εφαρμογών, κάνοντας χρήση αντικειμενοστρέφειας όπου είναι η δημοφιλέστερη τεχνική ανάπτυξης. Η διαχείριση των δεδομένων σε μια σχεσιακές βάσεις αποτελεί μία αρκετά πολύπλοκη

διαδικασία. Ο πιο απλός τρόπος για να διαχειριστούμε τα δεδομένα σε μια σχεσιακή βάση είναι να δημιουργήσουμε στον κώδικα ερωτήματα SQL για κάθε αντικείμενο της εφαρμογής. Αυτός ο τρόπος όμως δεν είναι καθόλου αποδοτικός και έχει πολλά μειονεκτήματα, όπως τον χρόνο ανάπτυξης, περισσότερη συντήρηση αλλά και μειωμένη επεκτασιμότητα. Η χρήση των ORM εργαλείων έρχονται να δώσουν λύσει στα παραπάνω προβλήματα αυτοματοποιώντας την διασύνδεση των μοντέλων-αντικειμένων και της βάσης δεδομένων, επιτρέποντας έτσι την εύκολη αποθήκευση και ανάκτηση των δεδομένων που συσχετίζονται με τα αντικείμενα.

Τα πλεονεκτήματα της χρήσης ORM εργαλείων είναι πολλά. Με την χρήση τους έχουμε πολύ μεγάλη μείωση στον χρόνο ανάπτυξης της εφαρμογής αλλά και στην συντήρηση της. Το πιο σημαντικό χαρακτηριστικό είναι η αυτοματοποίηση της δημιουργίας των SQL ερωτημάτων προς την βάση για την ανάκτηση των δεδομένων αλλά και για την αποθήκευσή τους.



Εικόνα 2-5 Παράδειγμα Αντικείμενο-Σχεσιακής Απεικόνισης (ORM)

3 Django Framework

Το Django είναι ένα υψηλού επιπέδου web framework γραμμένο σε Python το οποίο παρέχει όλα τα απαραίτητα εργαλεία για την γρήγορη ανάπτυξη διαδικτυακών εφαρμογών. Διευκολύνει σημαντικά την ανάπτυξη σύνθετων διαδικτυακών εφαρμογών δίνοντας έμφαση στην επαναχρησιμοποίηση δομικών στοιχείων της εφαρμογής. Κάνει χρήση αντικείμενο-σχεσιακής απεικόνισης (ORM) αυτοματοποιώντας έτσι την διασύνδεση των μοντέλων της εφαρμογής με την σχεσιακή βάση δεδομένων επιτρέποντας την εύκολη ανάκτηση και αποθήκευση δεδομένων από την βάση. Επιπροσθέτως κάνει χρήση του μοντέλο αρχιτεκτονικής λογισμικού MVC κληρονομώντας όλα τα οφέλη ανάπτυξης εφαρμογών με την χρήση της συγκεκριμένης αρχιτεκτονικής. Μερικά από τα κύρια χαρακτηριστικά του, όπου το έχουν κάνει ένα από τα δημοφιλέστερα frameworks για την ανάπτυξη διαδικτυακών εφαρμογών είναι η ταχύτητα, η ασφάλεια αλλά και η επεκτασιμότητα που παρέχει.

Το Django συμβάλει σημαντικά στην ελαχιστοποίηση του κώδικα. Όσο περισσότερο κώδικα γράφουμε τόσο περισσότερες είναι οι πιθανότητες να έχουμε σφάλματα στον κώδικα μας. Αυτό σημαίνει πως μειώνοντας τον όγκο του κώδικα που πρέπει να γραφτεί για την ανάπτυξη μιας εφαρμογής, επιταχύνουμε σημαντικά την διαδικασία της ανάπτυξής της. Βοηθάει σημαντικά στην ελάττωση του επαναλαμβανόμενου κώδικα. Μια από τις αρχές ανάπτυξης λογισμικού είναι η αποφυγή κώδικα ο οποίος επαναλαμβάνεται συνέχεια στην εφαρμογή μας. Κάθε λειτουργία της εφαρμογής πρέπει να υπάρχει κάπου μόνο μια φορά και να μην επαναλαμβάνεται μέσα στην εφαρμογή μας. Με αυτό τον τρόπο ελαττώνεται σημαντικά ο όγκος του κώδικα και συμβάλει στην καλύτερη διαχείριση του.

3.1 Περιβάλλον εργασίας Django

Το Django είναι πολύ εύκολο να εγκατασταθεί σε οποιονδήποτε υπολογιστή. Ένα από τα βασικά εργαλεία που προσφέρει το Django αποτελείτε από ένα σύνολο Python Scripts με τα οποία μπορούμε πολύ εύκολα να δημιουργήσουμε λειτουργικά Django Projects, συνοδευόμενα από έναν απλό Webserver όπου χρησιμοποιείτε για την δοκιμή των εφαρμογών κατά την ανάπτυξη σε τοπικό περιβάλλον. Είναι εξαιρετικά ευέλικτο ότι αφορά την εγκατάσταση και στην ρύθμισή του. Η εγκατάσταση του μπορεί να γίνει σε όλα τα γνωστά λειτουργικά συστήματα όπως Windows, Linux και MacOS. Επίσης είναι συμβατό και με τις 2 εκδόσεις της Python (Python 3 και Python 2).

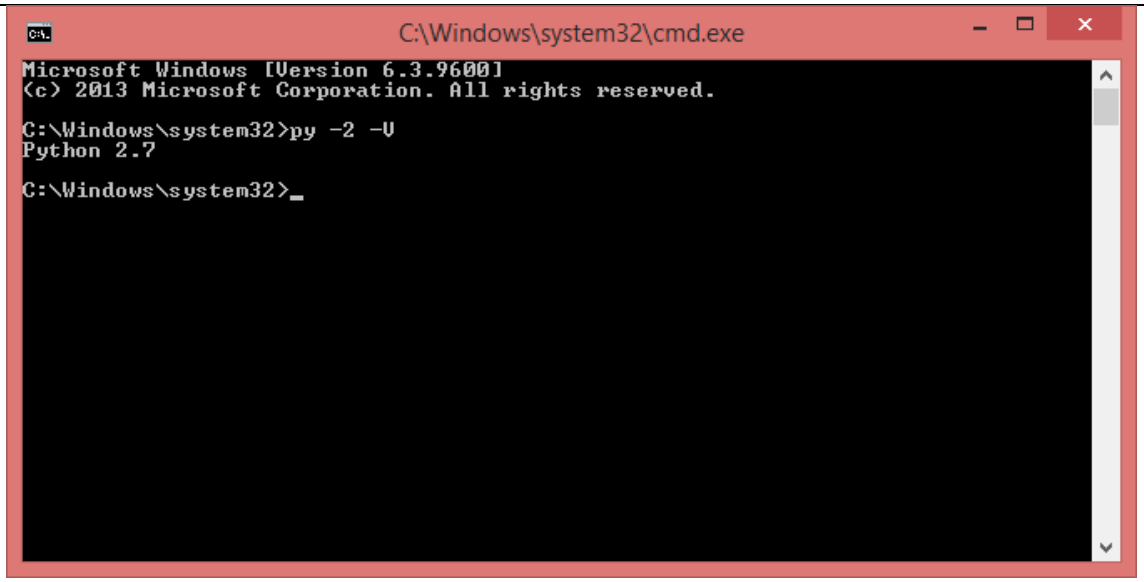
Για την δημιουργία μιας διαδικτυακής εφαρμογής είναι απαραίτητη η χρήση κάποιας βάσης δεδομένων. Η βάση δεδομένων θα πρέπει να είναι εγκατεστημένη στο σύστημά μας. Εξαίρεση αποτελεί η SQLite η οποία υποστηρίζεται απευθείας από το Django χωρίς την χρήση εξωτερικών βιβλιοθηκών η την εκτέλεση κάποιας υπηρεσίας. Το Django υποστηρίζει τις πιο διαδεδομένες βάσεις δεδομένων όπως την PostgreSQL, MySQL, Oracle και SQLite αλλά υπάρχει και πολύ μεγάλη υποστήριξη από την κοινότητα, διαθέτοντας βιβλιοθήκες για σχεδόν όλες τις γνωστές SQL και NOSQL βάσεις δεδομένων που υπάρχουν. Στα πλαίσια της πτυχιακής εργασίας η βάση δεδομένων που χρησιμοποιήθηκε είναι η SQLite.

Η ανάπτυξη μιας Django εφαρμογής μπορεί να γίνει κάνοντας χρήση ενός απλού κειμενογράφου και δεν είναι απαραίτητη η εγκατάσταση κάποια εξειδικευμένης εφαρμογής. Ωστόσο η ανάπτυξη εφαρμογών με απλό κειμενογράφο είναι σημαντικό μειονέκτημα αφού μπορεί να κάνει την ανάπτυξη και την αποσφαλμάτωση των εφαρμογών αρκετά πιο δύσκολη. Γι’ αυτό τον λόγο υπάρχει ένας μεγάλος αριθμός από πλατφόρμες ανάπτυξης Django εφαρμογών (IDE) όπου βοηθάνε σημαντικά. Μια από

τις πιο διαδεδομένες και ώριμες πλατφόρμες ανάπτυξής εφαρμογών Django είναι το PyCharm της JetBrains όπου χρησιμοποιήθηκε και στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας για την ανάπτυξη στην διαδικτυακής εφαρμογής. Η πλατφόρμα διατίθεται δωρεάν και μπορεί να χρησιμοποιηθεί από τον καθένα.

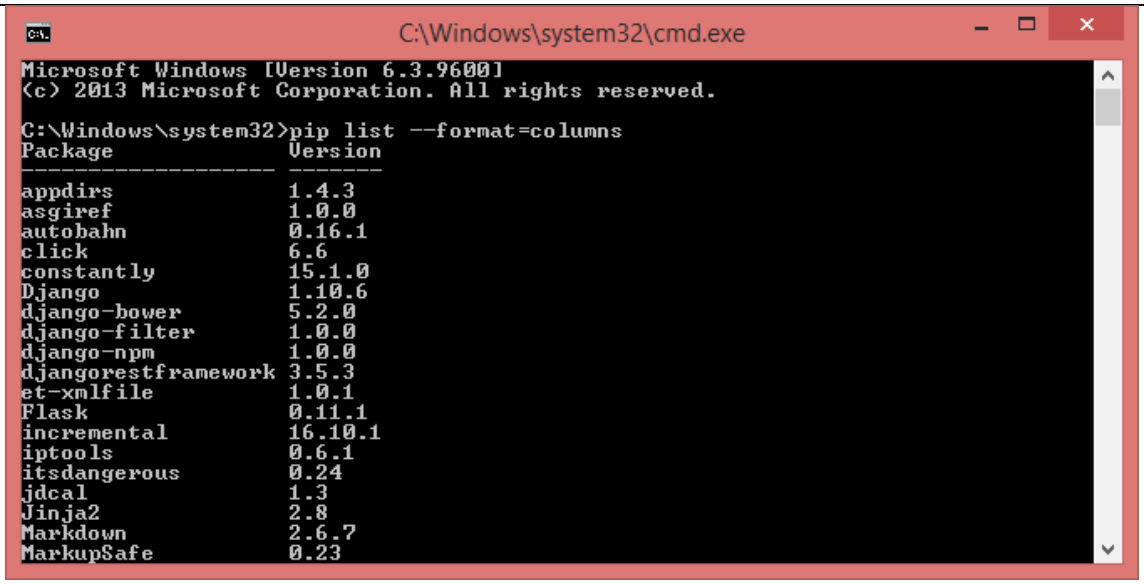
3.2 Εγκατάσταση του Django

Για την εγκατάσταση του Django είναι προαπαιτούμενο η εγκατάσταση της Python στο λειτουργικό σύστημα. Επίσης είναι απαραίτητη και τη εγκατάσταση του διαχειριστή πακέτων (βιβλιοθηκών) της Python “pip” (Python Package Index) με τον οποίο μπορούμε να διαχειριστούμε και να εγκαταστήσουμε βιβλιοθήκες της Python στο σύστημα μας. Στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας η ανάπτυξη της διαδικτυακής εφαρμογής έγινε σε περιβάλλον Windows. Τα Windows δεν συμπεριλαμβάνουν την γλώσσα Python, αλλά είναι πολύ εύκολο να την εγκαταστήσουμε κατεβάζοντας τον κατάλληλο εγκαταστάτη από τον επίσημο ιστότοπο της Python https://www.python.org/downloads/. Η έκδοση 2.7 της Python είναι αυτή που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής μας. Έτσι επιλέγοντας το κουμπί “Download Python 2.7.x” από τις διαθέσιμες εκδόσεις μπορούμε πολύ εύκολα να προχωρήσουμε στην εγκατάσταση, κατεβάζοντας και εκτελώντας τον εγκαταστάτη. Να σημειωθεί πως στην προκαθορισμένη εγκατάσταση εμπεριέχεται και ο διαχειριστή πακέτων της Python “pip”. Στην συνέχεια μπορούμε να επαληθεύσουμε ότι η Python εγκαταστάθηκε εκτελώντας την ακόλουθη εντολή στην γραμμή εντολών.



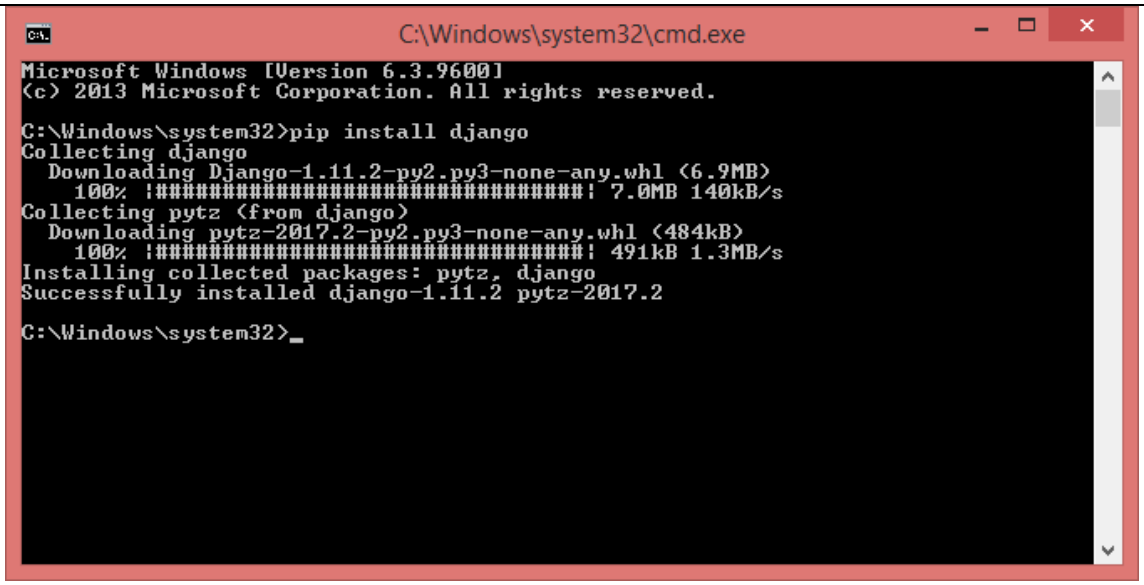
Εικόνα 3-1 Εντολή επαλήθευσης της εγκατάστασης Python

Με τον ίδιο τρόπο μπορούμε να επαληθεύσουμε την εγκατάσταση του διαχειριστή πακέτων της Python “pip” και να δούμε τις εγκατεστημένες βιβλιοθήκες του υπάρχουν στο σύστημά μας.



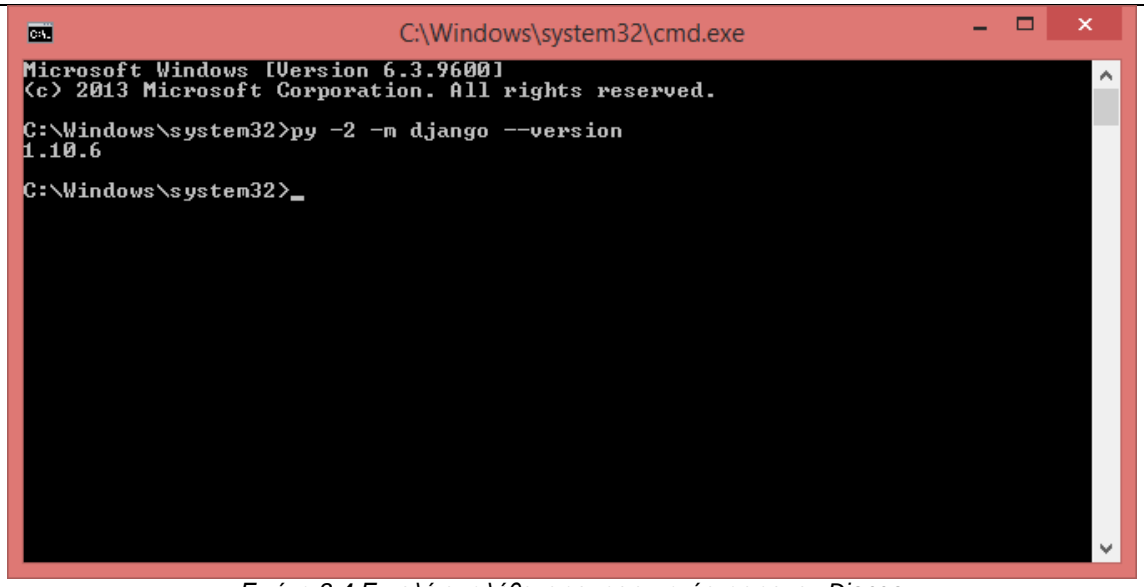
Εικόνα 3-2 Εντολή επαλήθευσης της εγκατάστασης του pip

Αφού ολοκληρώσουμε την εγκατάσταση της Python και του διαχειριστή πακέτων μπορούμε να προχωρήσουμε στην εγκατάσταση του Django χρησιμοποιώντας τον διαχειριστή πακέτων εκτελώντας την ακόλουθη εντολή στην γραμμή εντολών.



Εικόνα 3-3 Εντολή εγκατάστασης του Django

Ο διαχειριστή πακέτων θα εγκαταστήσει την τελευταία έκδοση του Django καθώς και όλες τις εξαρτήσεις που έχει για να λειτουργήσει. Για να επιβεβαιώσουμε την σωστή εγκατάσταση του Django εκτελούμε την ακόλουθη εντολή στην γραμμή εντολών.



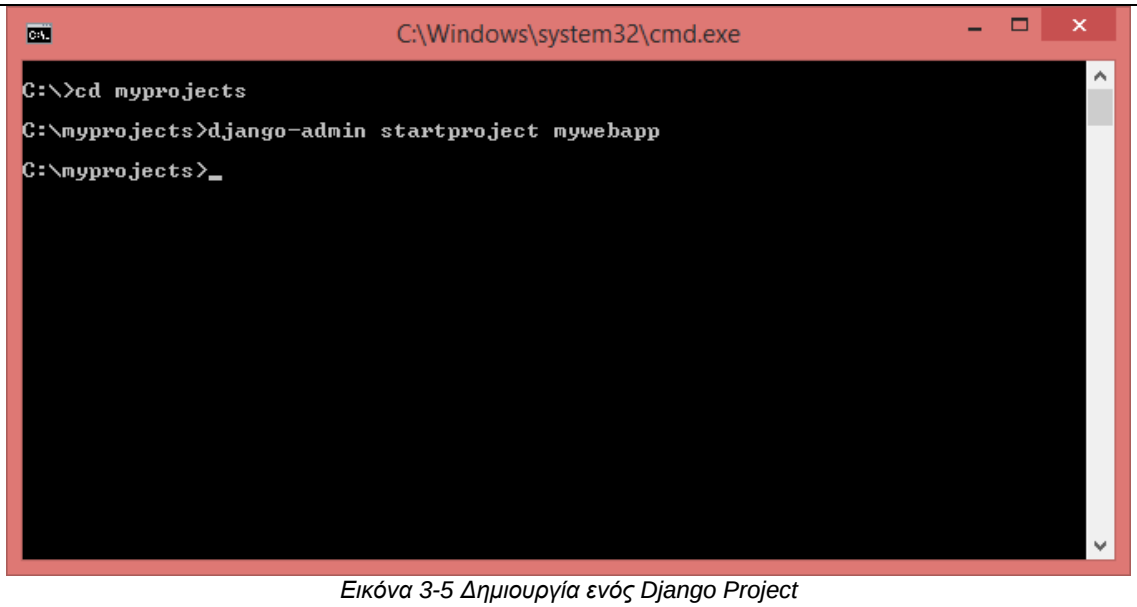
Εικόνα 3-4 Εντολή επαλήθευσης της εγκατάστασης του Django

3.3 Δημιουργώντας μια Django εφαρμογή

Για να ξεκινήσουμε την δημιουργία μιας Django εφαρμογής (Django Application) θα πρέπει να κάνουμε χρήση των εργαλείων που μας δίνει το Django για να δημιουργήσουμε ένα καινούριο project.

3.3.1 Δημιουργία ενός Django Project

Για να δημιουργήσουμε ένα καινούριο Django project θα πρέπει να κάνουμε χρήση του εργαλείου “django-admin” το οποίο είναι διαθέσιμο μαζί με το Django. Αυτό το εργαλείο δημιουργεί τον βασικό σκελετό ενός πλήρους λειτουργικού Django Application.



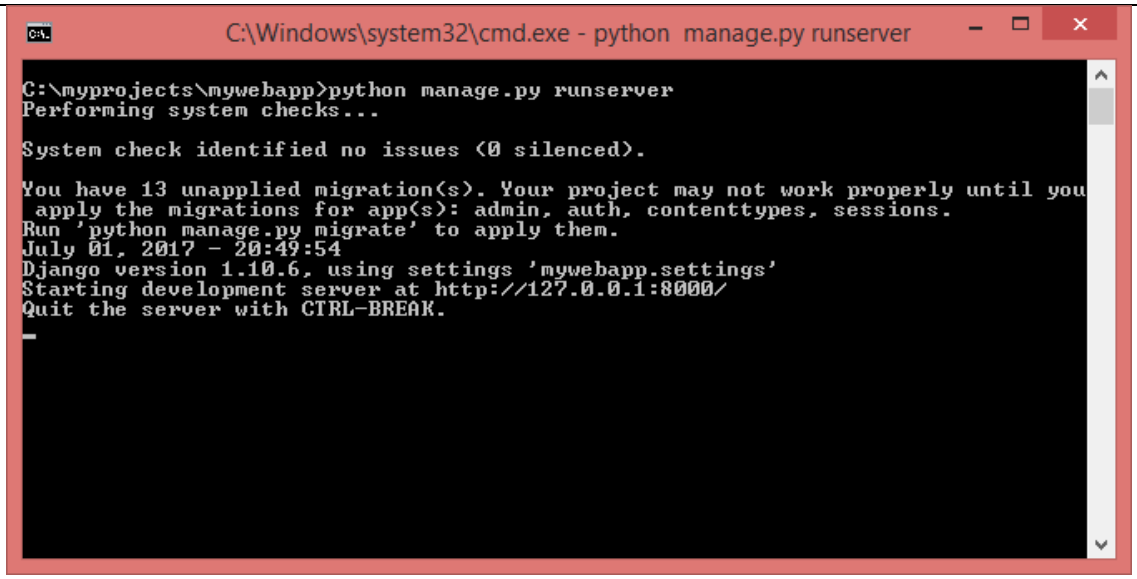
Εικόνα 3-5 Δημιουργία ενός Django Project

Η εντολή θα δημιουργήσει έναν καινούριο φάκελο με το όνομα του project ο οποίος περιέχει 4 βασικά αρχεία.

__init__.py	Είναι ένα κενό αρχείο το οποίο χρησιμοποιείται στην Python για να θεωρήσει τον φάκελο αυτό σαν ένα πακέτο της Python, δηλαδή είναι ένα φάκελος ο οποίος περιέχει κάποιο Python εφαρμογή.
manage.py	Αυτό το αρχείο χρησιμοποιείται για την αλληλεπίδραση με την Django εφαρμογή, ουσιαστικά είναι ένα εργαλείο με το οποίο μέσω της γραμμής εντολών μπορούμε να εκτελέσουμε διάφορες λειτουργίες που αφορούν το project.
settings.py	Σε αυτό το αρχείο βρίσκονται όλες οι ρυθμίσεις του project.
urls.py	Περιέχει τις δηλώσεις όλων των URL του project.

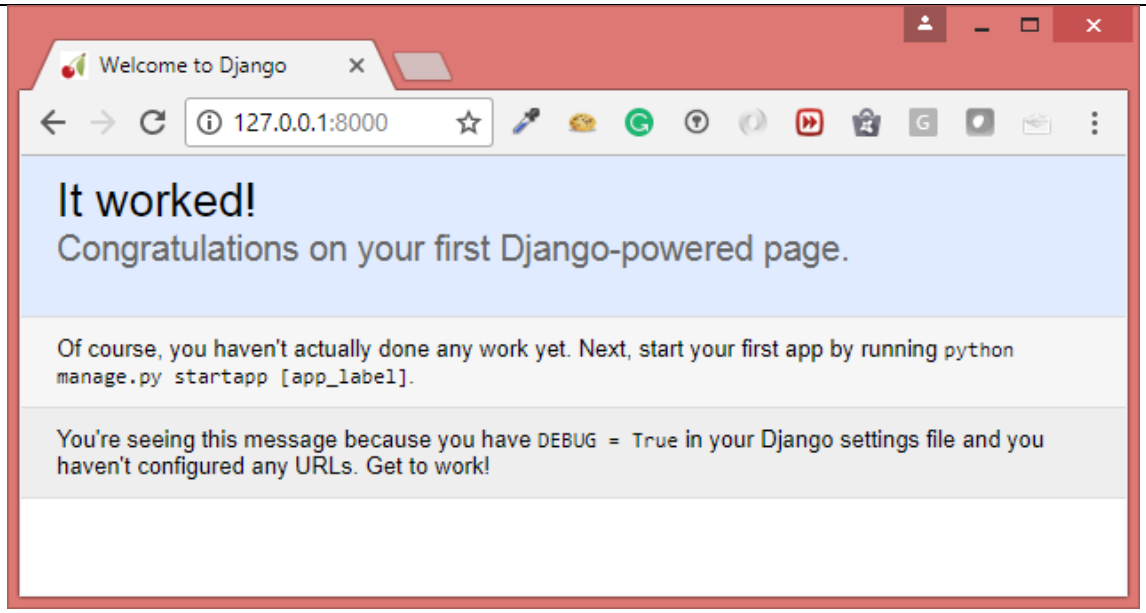
Πίνακας 3-1 Βασικά αρχεία Django project

Αυτά τα αρχεία δημιουργούνται αυτόματα από το Django και απαρτίζουν ένα πλήρως λειτουργικό Django Application. Χρησιμοποιώντας το **manage.py** μπορούμε να εκτελέσουμε διάφορες εντολές που αφορούν το project. Στην αρχή του project μόνο μία εντολή μπορούμε να εκτελέσουμε και αυτή είναι η “runserver” η οποία θα εκτελέσει τον webserver που έχει το Django για να τρέξει το project μας.



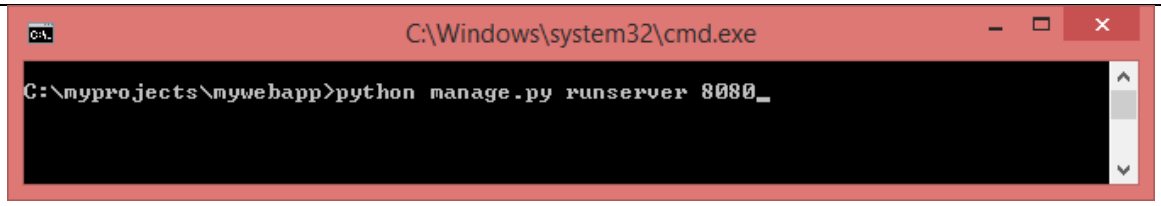
Εικόνα 3-6 Εκτέλεση ενός Django Application

Όπως φαίνεται και από την παραπάνω εικόνα η εφαρμογή μας τρέχει κανονικά και μπορούμε να δούμε πληροφορίες όπως την διεύθυνση IP και την πόρτα που χρησιμοποιεί ο webserver. Για να επιβεβαιώσουμε την λειτουργία της εφαρμογής μπορούμε να ανοίξουμε ένα φυλλομετρητή και να επισκεφτούμε την διεύθυνση που αναγράφεται.



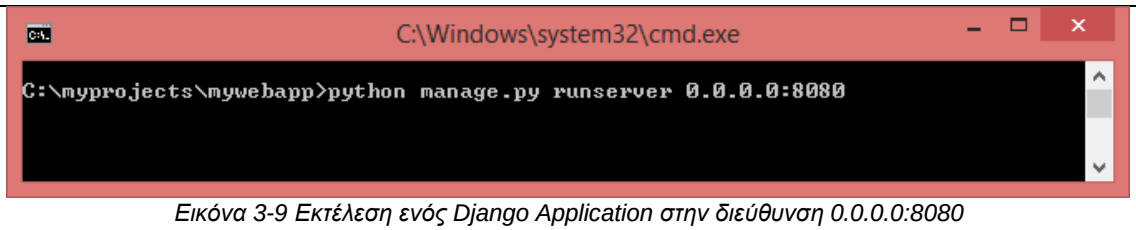
Εικόνα 3-7 Σελίδα καλωσορίσματος Django εφαρμογής

Η προεπιλεγμένες τιμές της εντολής "runserver" είναι η πόρτα 8000 και ακούει μόνο για τοπικές συνδέσεις καθώς χρησιμοποιεί την διεύθυνση IP 127.0.0.1 όπου είναι προσβάσιμη μόνο τοπικά, δηλαδή μόνο από τον ίδιο υπολογιστή απ' όπου τρέχει την εφαρμογή. Ο webserver μπορεί να χρησιμοποιήσει οποιαδήποτε πόρτα επιθυμούμε. Για να αλλάξουμε την πόρτα μπορούμε πολύ εύκολα να δώσουμε σαν όρισμα τον αριθμό της πόρτας.



Εικόνα 3-8 Εκτέλεση ενός Django Application στην πόρτα 8080

Εκτός από την πόρτα έχουμε την δυνατότητα να ορίσουμε και την IP διεύθυνση που θέλουμε να τρέξει ο webserver. Μπορούμε να δώσουμε για παράδειγμα την διεύθυνση 0.0.0.0 όπου είναι μια ειδική διεύθυνση που την χρησιμοποιούμε ούτως ώστε ο webserver να δέχεται συνδέσεις από όλα τα δίκτυα που έχει πρόσβαση.



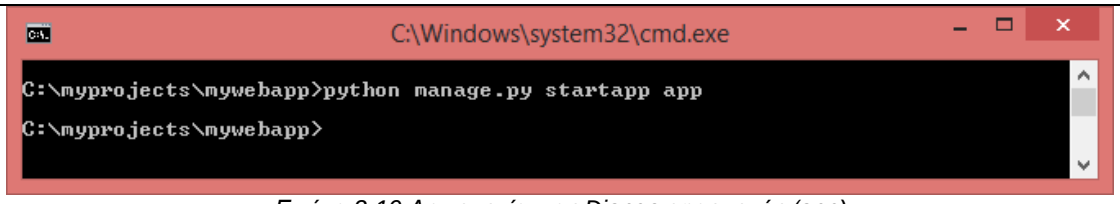
Εικόνα 3-9 Εκτέλεση ενός Django Application στην διεύθυνση 0.0.0.0:8080

3.3.1.1 Django app

Η διαφορά του "Project" και του "Application/Εφαρμογής" (app) είναι ότι οι εφαρμογές έχουν μία συγκεκριμένη λειτουργικότητα. Για παράδειγμα ένα project μπορεί να αποτελείτε από πολλές μικρές εφαρμογές που κάνουν μία συγκεκριμένη

δουλειά όπως μια εφαρμογή που εμφανίζει την μέρα και την ώρα ή μια άλλη που ανεβάζει ένα αρχείο στον server. Το project εμπεριέχει όλες αυτές τις εφαρμογές.

Για να δημιουργήσουμε μια καινούρια εφαρμογή μέσα στο project θα το χρησιμοποιήσουμε το **manage.py** και την εντολή “startapp”. Με αυτό τον τρόπο θα δημιουργήσουμε μια εφαρμογή με το όνομα “app”.



Εικόνα 3-10 Δημιουργία μιας Django εφαρμογής (app)

Η εκτέλεση της εντολής θα δημιουργήσει έναν καινούριο φάκελο μέσα στο project με το όνομα “app”, όπου περιέχει τέσσερα αρχεία τα οποία θα εξετάσουμε αναλυτικά στην πορεία.

- **__init__.py**
- **views.py**
- **models.py**
- **tests.py**

Για να είναι ορατό η καινούργια εφαρμογή που δημιουργήσαμε από το Django project θα πρέπει να δηλώσουμε την ύπαρξης της καινούριας εφαρμογής στης εγκατεστημένες εφαρμογές του project. Η επιλογή αυτή είναι διαθέσιμη στο αρχείο **settings.py** στο **INSTALLED_APPS**. Το όνομα της καινούρια εφαρμογής θα πρέπει να προστεθεί στην λίστα αυτή, στην δική μας περίπτωση το “app”. Η λίστα αυτή περιέχει ήδη κάποιες τιμές και ο λόγος είναι ότι το Django χρήση κάποιον προκαθορισμένων εφαρμογών που είναι συνήθων απαραίτητες στις περισσότερες εφαρμογές.

3.3.1.2 models.py

Η λογική των μοντέρνων διαδικτυακών εφαρμογών συνήθων περιλαμβάνει και την αλληλεπίδραση με βάσεις δεδομένων. Οι εφαρμογές συνδέονται με βάσεις δεδομένων για να ανακτήσουν δεδομένα και να τα εμφανίσουν σε κάποια σελίδα. Με το Django η διαδικασία αυτή είναι πολύ εύκολη, καθώς παρέχει πολύ ισχυρά εργαλεία για την εκτέλεση τέτοιου είδους λειτουργιών.

Το αρχείο **models.py** χρησιμοποιείται για να δημιουργήσουμε τα μοντέλα (Models), τα οποία χρησιμοποιούνται για να ορίσουμε τα μοντέλα στην βάση δεδομένων. Κάθε κλάση στο μοντέλο μεταφράζεται αν έναν πίνακας στην βάση δεδομένων. Ο σκοπός της χρήσης των μοντέλων είναι η σύνδεση τους με την βάση δημιουργώντας ένα επίπεδο διασύνδεσης δεδομένων. Κάνοντας χρήση αυτού του επιπέδου ορίζοντας τα δεδομένα στο μοντέλο αυτοματοποιεί την διαδικασία διαχείρισης των δεδομένων στην βάση. Πρακτικά αντί να γράφουμε SQL ερωτήματα για να πάρουμε η να γράψουμε δεδομένα στην βάση πλέων διαχειριζόμαστε όλα τα δεδομένα μέσω της χρήσης του μοντέλου.

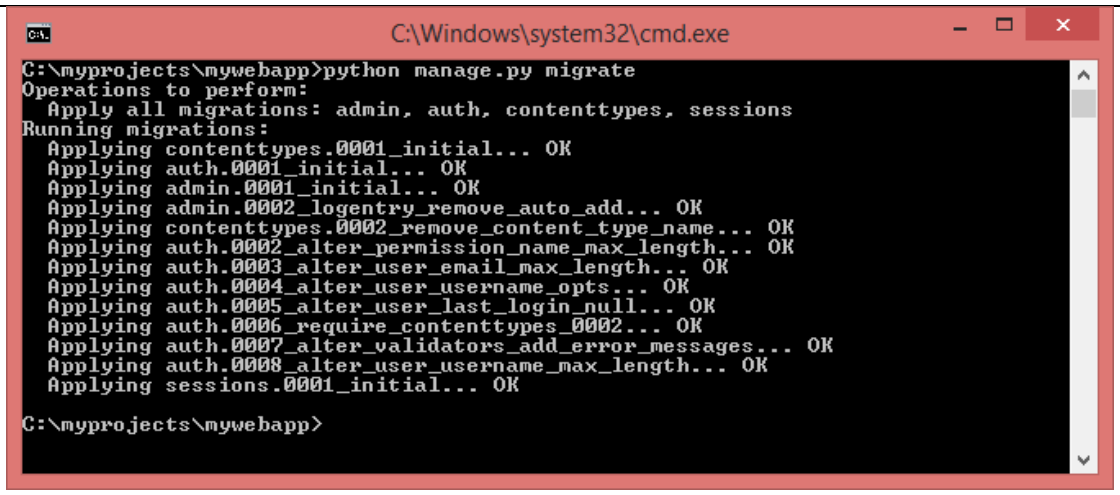
Για την εγκατάσταση της βάσης δεδομένων τα πρέπει να ρυθμίσουμε διάφορες επιλογές. Οι επιλογές αυτές θα πρέπει να δηλωθούν στο αρχείο **settings.py** του project. Στα πλαίσια της συγκεκριμένης πτυχιακής θα χρησιμοποιηθεί SQLite που όπως αναφέραμε σε προηγούμενο κεφάλαιο δεν απαιτεί την εγκατάσταση κάποια

άλλης εφαρμογής η υπηρεσίας, υποστηρίζεται απευθείας από το Django και αποθηκεύει όλα τα δεδομένα σε ένα αρχείο. Στην περίπτωση της SQLite θα πρέπει να δηλώσουμε την βάση δεδομένων που χρησιμοποιούμε και το όνομα της βάσης.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Εικόνα 3-11 Ρύθμιση SQLite βάσης δεδομένων στο settings.py

Αφού έχουμε ολοκλήρωση την ρύθμιση της βάσης δεδομένων θα πρέπει να εκτελέσομε την εντολή “migrate” για να δημιουργηθούν όλοι οι απαραίτητη πίνακες των εφαρμογών του project που έχουμε δημιουργήσει και δηλώσει στα INSTALLED_APPS. Οι πίνακες δημιουργούνται μόνο αν δεν υπάρχουν ήδη.



Εικόνα 3-12 Δημιουργία των πινάκων στην βάση δεδομένων

3.3.1.3 views.py

Σε αυτό το σημείο θα χρειαστούμε ένα αρχείο στο οποίο θα γράψουμε την συνάρτηση (View) η οποία θα είναι υπεύθυνη να εμφανίσει το περιεχόμενο της σελίδας. Γι’ αυτό τον σκοπό στο Django υπάρχει το αρχείο views.py το οποίο αρχικά δεν υπάρχει και θα πρέπει να δημιουργηθεί μέσα στον φάκελο της εφαρμογής (app). Η συνάρτηση αυτή παίρνει σας όρισμα έναν *HttpRequest* και επιστρέφει ένα *HttpResponse* αντικείμενο το οποίο περιέχει το περιεχόμενο της σελίδας που θέλουμε να εμφανιστεί.

```
from django.http import HttpResponse

def myView(request):
    return HttpResponse("This is a Hello World Django page!")
```

Εικόνα 3-13 Παράδειγμα View

Το παραπάνω View δημιουργεί ένα *HttpResponse* με το περιεχόμενο “*This is a Hello World Django page!*”. Το *HttpResponse* είναι μία κλάση του Django και πρέπει να εισαχθεί από το *django.http* module.

3.3.1.4 urls.py

Η τελευταία ενέργεια που πρέπει να κάνουμε για να είναι προσβάσιμο το view που δημιουργήσαμε προηγούμενος και να μπορέσουμε να δούμε την σελίδα είναι να δηλώσουμε ένα κατάλληλο URL το οποίο όταν καλείτε θα μας επιστρέφει τα αποτελέσματα του view. Για παράδειγμα έστω πως θέλουμε καλώντας το URL http://127.0.0.1:8000/app/ να βλέπουμε την σελίδα μας. Για να το κάνουμε αυτό θα πρέπει να τροποποιήσουμε το αρχείο **urls.py** και κάνοντας χρήση της συνάρτησης *url()* να προσθέσουμε ένα καινούριο URL στην εφαρμογή μας.

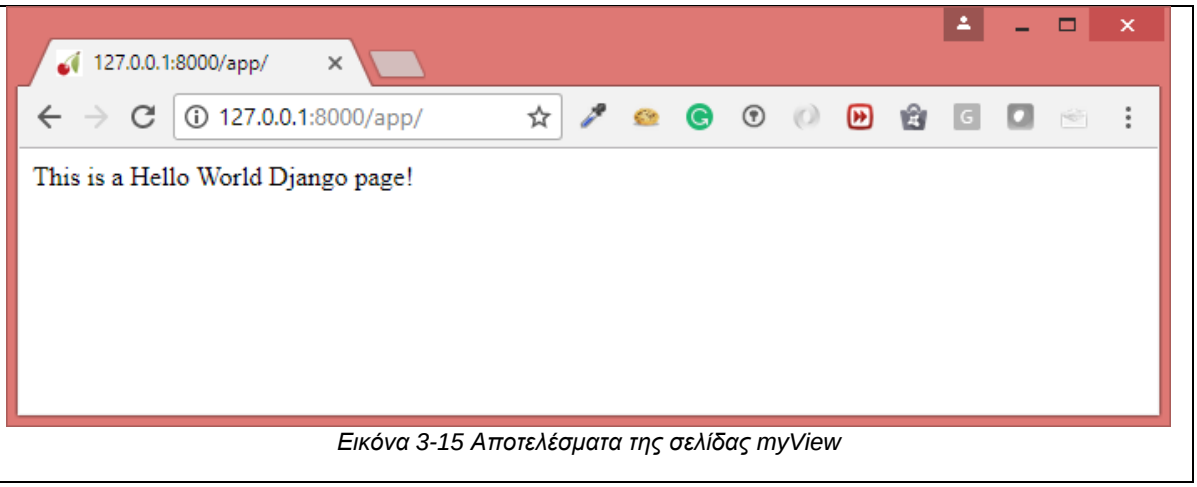
Η συνάρτηση *url()* έχει την ακόλουθη δομή, παίρνει σαν όρισμα κανονικές έκφρασής (regular expressions) το οποίο καθορίζει την διεύθυνση URL και ένα δεύτερο όρισμα το οποίο είναι η συνάρτηση η οποία θα κληθεί κατά της επίσκεψη της συγκεκριμένης διεύθυνσης. Όταν λοιπόν πληκτρολογήσουμε κάποια διεύθυνση URL στον φυλλομετρητή το Django ψάχνει την λίστα με τα URL που έχουν δηλωθεί από πάνω προς τα κάτω, αν το URL ταιριάζει με κάποιο από την λίστα τότε καλείτε η αντίστοιχη συνάρτηση, στην περίπτωσή μας η συνάρτηση *myView*.

```
from django.conf.urls import url

urlpatterns = [
    url(r'^app/', 'app.views.myView'),
]
```

Εικόνα 3-14 Παράδειγμα δημιουργίας URL

Με αυτό τον τρόπο όταν κάποιος επισκεφτεί το URL http://127.0.0.1:8000/app/ θα εκτελεσθεί το view “myView” που δημιουργήσαμε παραπάνω και θα επιστρέψει την σελίδα.



Εικόνα 3-15 Αποτελέσματα της σελίδας myView

Ο τρόπος με τον οποίο καθορίζουμε μια διεύθυνση URL γίνεται με την χρήση κανονικών εκφράσεων (regular expressions ή regex). Το regex είναι μία συμβολοσειρά η οποία περιγράφει ένα σύνολο συμβολοσειρών. Στο προηγούμενο παράδειγμα χρησιμοποιήσαμε το regex ***r’^app/$’***. Κάθε σύμβολο regex σημαίνει “οτιδήποτε άλλο” και αντιστοιχεί διαφορετικά URLs. Ο χαρακτήρας **r** μπροστά από το regex χρησιμοποιείται στην Python για να δηλώσουμε μία ακατέργαστη συμβολοσειρά, αυτό σημαίνει πως το δεν πρέπει να διαχειρίζεται χαρακτήρες όπως την ανάστροφη κάθετος (backslash) σαν ειδικούς χαρακτήρες.

Οι σημασίες ορισμένων συμβόλων παρουσιάζονται παρακάτω:

^	Αρχή της συμβολοσειράς
$	Τέλος της συμβολοσειράς
/συμβολοσειρά	Συγκεκριμένης συμβολοσειράς κειμένου που θα πρέπει να χρησιμοποιηθεί για την αντιστοίχιση διευθύνσεων URL
. (τελεία) ή \D	Οποιοσδήποτε χαρακτήρας
\d	Οποιοδήποτε μοναδικό ψηφίο
+	Μία ή περισσότερες από τις προηγούμενες εκφράσεις (η έκφραση \d+ αντιστοιχεί σε ένα ή περισσότερα ψηφία)

Πίνακας 3-2 Σύμβολα κανονικών εκφράσεων (regex)

Η έκφραση **^$** αντιστοιχίζεται σε μια κενή συμβολοσειρά, αντιστοιχεί με τη διεύθυνση URL http://127.0.0.1:8000/
Η έκφραση **r'^app/$'** αντιστοιχίζεται με το URL http://127.0.0.1:8000/app/, η αρχή της διεύθυνσης URL εξαρτάται από την διεύθυνση IP και τη θύρα στην οποία τρέχει ο webserver.
r'^\d+$' αντιστοιχίζεται με το URL http://127.0.0.1:8000/{αριθμός}
r'^app/\d+$' αντιστοιχίζεται με το URL http://127.0.0.1:8000/app/{αριθμός}

Πίνακας 3-3 Παράδειγμα κανονικών εκφράσεων (regex)

Με την χρήση των κανονικών εκφράσεων έχουμε την δυνατότητα να περάσουμε δυναμικά ορίσματα στο URL.

```
from django.conf.urls import url

urlpatterns = [
    url(r'^app/(?P<app_id>\d+)/$', 'app.views.myView')
]
```

Εικόνα 3-16 Παράδειγμα δημιουργίας URL με όρισμα

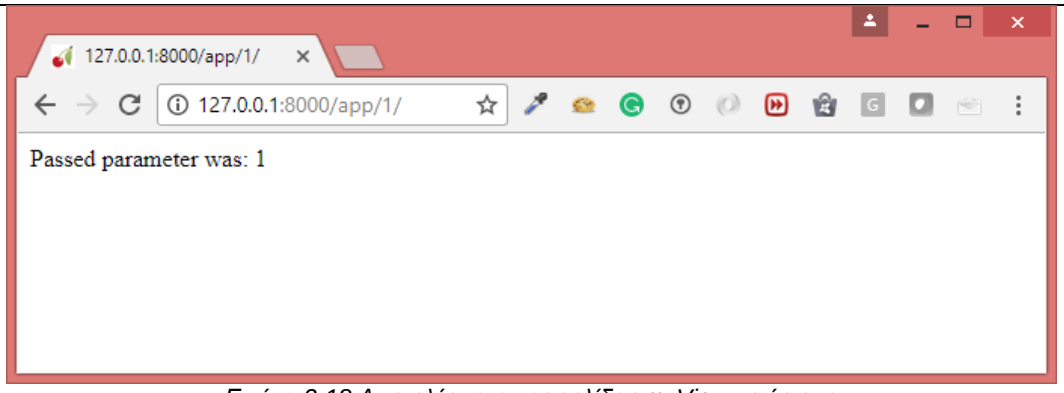
Για να πάρουμε το όρισμα από το URL θα πρέπει να τροποποιήσουμε την συνάρτηση view προσθέτοντάς άλλο ένα όρισμα στην συνάρτηση.

```
from django.http import HttpResponse

def myView(request, app_id):
    return HttpResponse("Passed parameter was: %s " % app_id)
```

Εικόνα 3-17 Παράδειγμα View με όρισμα

Προσθέτουμε την παράμετρο app_id και η τιμή της παραμέτρου χρησιμοποιείτε μέσα στην συνάρτηση του view.



Εικόνα 3-18 Αποτελέσματα της σελίδας myView με όρισμα

3.3.1.5 Templates

Τα templates χρησιμοποιούνται για να έχουμε views τα οποία είναι όμορφα και πλούσια σε περιεχόμενο. Για να το κάνει αυτό το Django κάνει χρήση ενός template συστήματος για να διαχωρίσει την σχεδίαση από το view. Τα templates αποτελούνται από HTML σελίδες τα οποία περιέχουν την σχεδίαση μια σελίδας όπως αυτή θέλουμε να εμφανιστεί όταν κληθεί το κατάλληλο view. Η τοποθεσία αποθήκευσης των αρχείων αυτόν πρέπει να οριστεί στο αρχείο **settings.py**.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
    },
]
```

Εικόνα 3-19 Ρύθμιση του Template συστήματος στο settings.py

Μέσα στο αρχείο, στην επιλογή TEMPLATES θα πρέπει να ορίσουμε τον τύπο του template συστήματος και την τοποθεσία του φακέλου όπου βρίσκονται τα template αρχεία. Στο project μπορεί να υπάρχουν πολλά διαφορετικά apps, και κάθε app μπορεί να έχει τα δικά του templates διαφορετικά έχουμε την δυνατότητα να έχουμε έναν κοινό φάκελο templates για όλο το project.

Αρχικά θα πρέπει να δημιουργήσουμε ένα καινούριο αρχείο template, δηλαδή ένα καινούριο HTML αρχείο το οποίο θα εμφανίζεται όταν καλείτε το *myView*. Το αρχείο αυτό χάριν παραδείγματος το ονομάζουμε **helloworld.html**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
</head>
<body>
    This is a Django Template!!!
</body>
</html>
```

Εικόνα 3-20 Παράδειγμα Template αρχείου

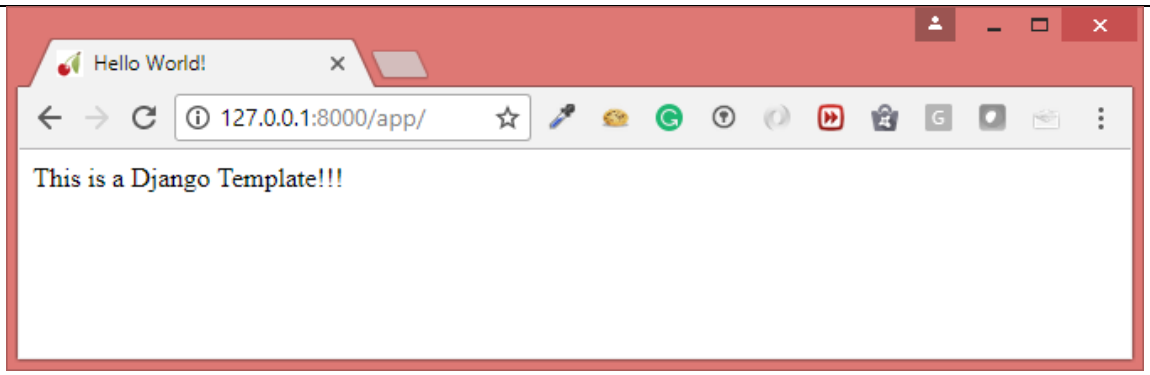
Το επόμενο βήμα είναι να τροποποιήσουμε το myView ούτως ώστε να εμφανίζει το template. Με την χρήση του *loader* δημιουργούμε ένα Template αντικείμενο και στην συνέχεια καλούμε την συνάρτηση *render()* η οποία παίρνει σαν όρισμα το request καθώς σύνολο μεταβλητών (context) που χρειαζόμαστε να εμφανίζονται δυναμικά στο template. Αν δεν υπάρχουν μεταβλητές το context μπορεί να είναι κενό.

```
from django.http import HttpResponse
from django.template import loader

def myView(request):
    t = loader.get_template('helloworld.html')
    context = {}
    return HttpResponse(t.render(context, request))
```

Εικόνα 3-21 Παράδειγμα View με χρήση template

Τέλος αν επισκεφτούμε το URL του app θα δούμε το περιεχόμενο του template που έχουμε δημιουργήσει.



Εικόνα 3-22 Αποτελέσματα της σελίδας myView με χρήση template

4 Διαδικτυακής πλατφόρμας διαχείρισης πιστωτικών υπηρεσιών

Η υλοποίηση της πλατφόρμας έγινε με την χρήση των μεθόδων και των εργαλείων που περιεγράφηκαν στα προηγούμενα κεφάλαια. Παρακάτω περιγράφεται η βασική λειτουργικότητα της εφαρμογής.

4.1 Προδιαγραφές

Οι πελάτες θα μπορούν να εγγράφονται στην πλατφόρμα δίνοντας τα στοιχεία τους όπως ονοματεπώνυμο, διεύθυνση, ημερομηνία γέννησης, τηλέφωνο, ημερομηνία απόκτησης του διπλώματος (σε περίπτωση ασφάλειας) καθώς και τον αριθμό ταυτότητας.

Από την πλατφόρμα οι πελάτες θα μπορούν να αγοράζουν ασφάλεια ζωής και δάνεια. Κάθε ασφαλιστήριο θα έχει τον κωδικό του, την ημερομηνία που έγινε και το μηνιαίο ποσό που θα δίνει ο πελάτης. Τα ασφαλιστήρια χωρίζονται σε κατηγορίες ανάλογα με το ποσό καταβολής και θα δημιουργούνται δυναμικά.

Κάθε δάνειο θα έχει τον κωδικό του, την ημερομηνία που έγινε, το ποσό που δανείζεται ο πελάτης, το μηνιαίο ποσό που θα καταβάλει ο πελάτης καθώς και το επιτόκιο. Το επιτόκιο μπορεί να είναι σταθερό ή κυμαινόμενο. Αν είναι κυμαινόμενο τότε ο πελάτης θα επιλέγει τον χρόνο αποπληρωμής και η δόση θα αναπροσαρμόζεται αυτόματα με κάθε αλλαγή του επιτοκίου. Το ίδιο θα ισχύει και για το σταθερό μόνο που η δόση θα είναι σταθερή.

Επίσης το σύστημα θα υποστηρίζει:

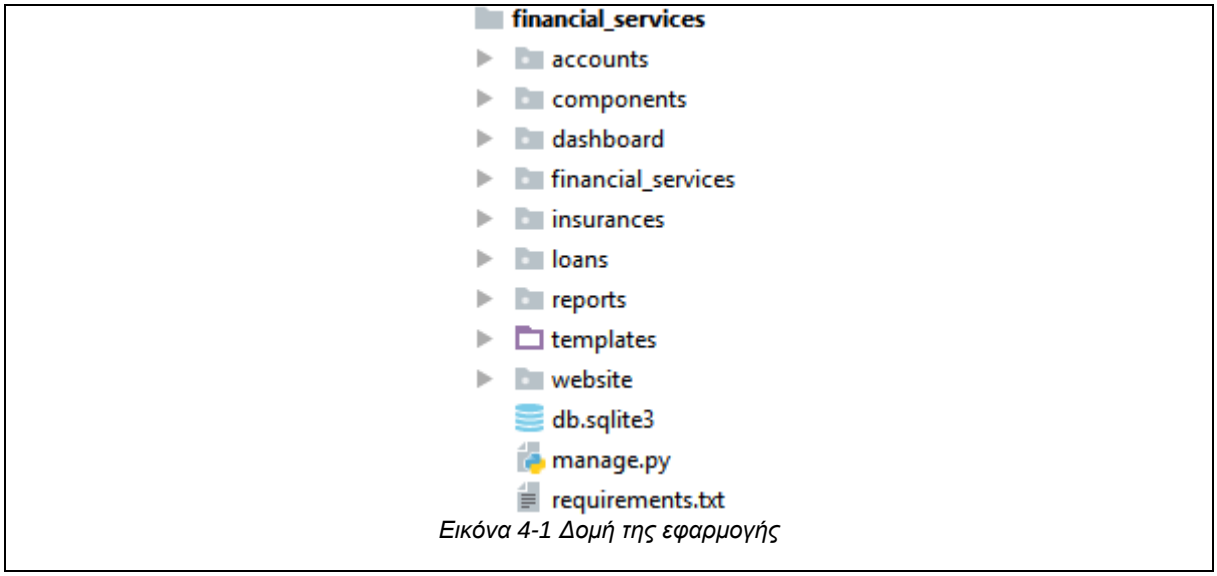
- Θα αποθηκεύομε πληροφορία για το αν ο πελάτης πληρώνει κανονικά ή όχι κάθε δόση. Αν κάποιος πελάτης έχει πάνω από τρία δάνεια ή ασφαλιστήρια τα οποία τα πληρώνει κανονικά τότε έχει έκπτωση 10% στα ασφάλιστρα και 5% στα επιτόκια των δανείων.
- Αν ο πελάτης δεν πληρώσει δύο συνεχόμενες δόσεις σε ένα ασφάλιστρο τότε θα έχει 10% επιπλέον σε κάθε δόση.
- Αν ο πελάτης δεν πληρώσει δύο συνεχόμενες δόσεις σε ένα δάνειο τότε θα έχει 5% επιπλέον σε κάθε δόση.
- Για να δοθεί ένα νέο δάνειο ή ασφαλιστήριο σε κάποιον πελάτη θα πρέπει να μην έχει χρωστάει καμία δόση.

Το σύστημα θα πρέπει να μπορεί να βγάνει συγκεντρωτικές αναφορές:

- Για τους πελάτες που έχουν καθυστερήσει δόσεις.
- Για τους πελάτες που είναι συνεπής.
- Για τους πελάτες που έχουν περισσότερα από κάποιο αριθμό ασφαλιστηρίων ή δανείων ή και συνδυασμό των δύο πιο πάνω. Θα πρέπει να δίνεται η δυνατότητα στον χρήστη να καθορίζει αυτά τα κριτήρια.
- Συγκεντρωτική αναφορά για τον αριθμό των πελατών ανά ασφαλιστήριο.
- Συγκεντρωτική αναφορά ανά μήνα(Ιούνιος, Ιούλιος, κτλ.) για τα ασφαλιστήρια και τα δάνεια που γίνονται.

4.2 Δομή της Εφαρμογής

Για την υλοποίηση της εφαρμογής αναπτυχθήκαν 6 βασικά Django apps με σκοπό την καλύτερη οργάνωση του κώδικα και την διαχείρισή του.



Εικόνα 4-1 Δομή της εφαρμογής

Η εφαρμογή αποτελείτε από τα 6 βασικά components που περιέχουν όλη την βασική λειτουργικότητα της εφαρμογής.

- accounts
- dashboard
- insurances
- loans
- reports
- website

Εκτός από τα βασικά apps στην δομή της εφαρμογής υπάρχουν και κάποιοι βοηθητικοί φάκελοι. Ο φάκελος *components* είναι ένας φάκελος ο οποίος περιέχει εσωτερικές javascript βιβλιοθήκες (π.χ JQuery) τις οποίες χρειαζόμαστε για τις σελίδες μας. Επίσης ο φάκελος *templates* περιέχει όλα τα HTML templates του χρησιμοποιούνται από την εφαρμογή μας. Τέλος ο φάκελος *financial_services* ο οποίος έχει ίδιο όνομα με το project και περιέχει αρχεία που αφορούν το project συνολικά όπως για παράδειγμα τις ρυθμίσεις του project.

4.2.1 Accounts

Το app accounts περιέχει κώδικα ο οποίος είναι υπεύθυνος για τους λογαριασμούς των χρηστών. Σε αυτό το app έχει προγραμματιστεί όλη η λογική για την διαχείριση και την δημιουργία λογαριασμών.

```python
from django.db import models
from django.contrib.auth.models import AbstractUser


class User(AbstractUser):
    id_number = models.CharField(max_length=7)
    address = models.CharField(max_length=120)
    birth_date = models.DateField(null=True, blank=True)
    driving_license_date_issued = models.DateField(null=True, blank=True)
```

Εικόνα 4-2 Μοντέλο λογαριασμού χρήστη

Στο μοντέλο του λογαριασμού κάνουμε χρήση ενός αφηρημένου (abstract) μοντέλου χρήστη που είναι διαθέσιμο από το Django και το επεκτείνουμε προσθέτοντας κάποια επιπλέων πεδία που είναι απαραίτητα.

```python
from django.conf.urls import url
from accounts import views

urlpatterns = [
    url(r'^$', views.index, name='accounts'),
    url(r'^new/$', views.account_new, name='account_new'),
    url(r'^(?P<account_id>[0-9]+)/$', views.account_edit, name='account_edit'),
    url(r'^(?P<account_id>[0-9]+)/delete/$', views.account_delete, name='account_delete'),
    url(r'^profile/$', views.profile, name='profile'),
]
```

Εικόνα 4-3 Accounts URLs

Στα URLs του app Accounts έχουν προγραμματιστεί τα κατάλληλα URLs ούτως ώστε να μπορούμε να βλέπουμε όλους τους λογαριασμούς να δημιουργούμε έναν καινούριο λογαριασμό, να τον επεξεργαζόμαστε καθώς και να τον διαγράφουμε. Το κάθε URL καλεί το αντίστοιχο view το οποίο είναι υπεύθυνο για να κάνει την συγκεκριμένη λειτουργία.

4.2.2 Insurances

Το app insurances περιέχει τον κώδικα που αφορά τις ασφάλειες. Αυτό το app είναι υπεύθυνο για την διαχείριση και την δημιουργία των ασφαλειών.

```python
from django.db import models
from accounts.models import User
from django.utils import timezone

from dateutil.relativedelta import relativedelta
from dateutil.rrule import rrule, MONTHLY

class InsuranceCategory(models.Model):
    name = models.CharField(max_length=256, unique=True)
    monthly_price = models.DecimalField(decimal_places=2, max_digits=10)

class Insurance(models.Model):
    account = models.ForeignKey(User)
    category = models.ForeignKey(InsuranceCategory)
    date_created = models.DateTimeField()

class InsurancePayment(models.Model):

    def get_total_paid(self):
        return self.amount_paid + self.penalty_amount_paid

    insurance = models.ForeignKey(Insurance)
    date_paid = models.DateTimeField()
    amount_paid = models.DecimalField(decimal_places=2, max_digits=10)
    penalty_amount_paid = models.DecimalField(decimal_places=2, max_digits=10)
```

Εικόνα 4-4 Μοντέλα ασφάλειας

Στο app insurances υπάρχουν 3 βασικά μοντέλα. Το μοντέλο **InsuranceCategory** το χρησιμοποιούμε για να δημιουργούμε και να κρατάμε τις διαφορετικές κατηγορίες των ασφαλειών. Το μοντέλο **Insurance** κρατάει όλες τις πληροφορίες που αφορούν μία ασφάλεια όπως τον χρήστη, την κατηγορία της ασφάλεια και την ημερομηνία που δημιουργήθηκε. Τέλος στο μοντέλο **InsurancePayment** κρατάμε τα στοιχεία που γίνονται για κάθε πληρωμή.

```python
from django.conf.urls import url
from insurances import views

urlpatterns = [
    url(r'^$', views.index, name='insurances'),
    url(r'^new/$', views.insurance_new, name='insurance_new'),
    url(r'^(?P<insurance_id>[0-9]+)/payments/$', views.insurance_payments, name='insurance_payments'),
    url(r'^(?P<insurance_id>[0-9]+)/payment/$', views.insurance_payment, name='insurance_payment'),
    url(r'^categories/$', views.insurance_categories, name='insurance_categories'),
    url(r'^categories/new/$', views.insurance_categories_new, name='insurance_categories_new'),
    url(r'^categories/(?P<category_id>[0-9]+)/$', views.insurance_categories_edit, name='insurance_categories_edit'),
    url(r'^categories/(?P<category_id>[0-9]+)/delete/$', views.insurance_categories_delete, name='insurance_categories_delete')
]
```

Εικόνα 4-5 Insurances URLs

Στα URLs του app Insurances έχουν προγραμματιστεί τα κατάλληλα URLs ούτως ώστε να μπορούμε να βλέπουμε όλες τις ασφάλειες και να δημιουργούμε καινούριες αλλά και για τις πληρωμές. Επίσης υπάρχουν τα κατάλληλα URLs για να δημιουργούμε και να διαχειριζόμαστε τις κατηγορίες των ασφαλειών. Το κάθε URL καλεί το αντίστοιχο view το οποίο είναι υπεύθυνο για να κάνει την συγκεκριμένη λειτουργία.

4.2.3 Loans

Το app loans περιέχει τον κώδικα που αφορά τα δάνεια. Αυτό το app είναι υπεύθυνο για την διαχείριση και την δημιουργία των δανείων.

```
from django.db import models
from django.conf import settings
from django.utils import timezone
from accounts.models import User

from dateutil.relativedelta import relativedelta
from dateutil.rrule import rrule, MONTHLY

INTEREST_RATE_TYPES = (
    ('S', 'static'),
    ('F', 'floating')
)
class Loan(models.Model):
    account = models.ForeignKey(User)
    interest_rate_type = models.CharField(max_length=1, default='S', choices=INTEREST_RATE_TYPES)
    interest_rate = models.DecimalField(decimal_places=2, max_digits=10, default=0)
    amount = models.DecimalField(decimal_places=2, max_digits=10)
    payoff_period = models.IntegerField()
    date_created = models.DateTimeField()

class LoanPayment(models.Model):

    def get_total_paid(self):
        return self.amount_paid + self.penalty_amount_paid

    loan = models.ForeignKey(Loan)
    date_paid = models.DateTimeField()
    amount_paid = models.DecimalField(decimal_places=2, max_digits=10)
    penalty_amount_paid = models.DecimalField(decimal_places=2, max_digits=10)
```

Εικόνα 4-6 Μοντέλα δανείου

Στο app loans υπάρχουν 2 βασικά μοντέλα. Το μοντέλο **Loan** όπου το χρησιμοποιούμε για να αποθηκεύομε πληροφορίες για τα δάνεια. Τα πεδία που έχει είναι ο χρήστης που παίρνει το δάνειο, τον τύπο του επιτοκίου, το επιτόκιο την περίοδο αποπληρωμής και την ημερομηνία δημιουργίας του δανείου. Τέλος στο μοντέλο **LoanPayment** κρατάμε τα στοιχεία που γίνονται για κάθε πληρωμή δόσης.

```
from django.conf.urls import url
from loans import views

urlpatterns = [
    url(r'^$', views.index, name='loans'),
    url(r'^new/$', views.loan_new, name='loan_new'),
    url(r'^(?P<loan_id>[0-9]+)/payments/$', views.loan_payments, name='loan_payments'),
    url(r'^(?P<loan_id>[0-9]+)/payment/$', views.loan_payment, name='loan_payment'),
]
```

Εικόνα 4-7 Loans URLs

Στα URLs του app Loans έχουν προγραμματιστεί τα κατάλληλα URLs ούτως ώστε να μπορούμε να βλέπουμε όλα τα δάνεια και να δημιουργούμε καινούρια αλλά και τις πληρωμές. Το κάθε URL καλεί το αντίστοιχο view το οποίο είναι υπεύθυνο για να κάνει την συγκεκριμένη λειτουργία.

4.2.4 Reports

Το app reports δεν περιέχει κάποιο μοντέλο, αλλά περιέχει τον κώδικα που είναι υπεύθυνος για την δημιουργία των αναφορών της εφαρμογής.

```
from django.conf.urls import url
from reports import views

urlpatterns = [
    url(r'^customers/$', views.customers, name='reports_customers'),
    url(r'^insurances/$', views.insurances, name='reports_insurances'),
    url(r'^monthly/$', views.monthly, name='reports_monthly'),
]
```

Εικόνα 4-8 Reports URLs

Στα URLs του app Reports έχουν προγραμματιστεί τα κατάλληλα URLs όπου μπορούμε να δούμε τις αναφορές του συστήματος για τους πελάτες, τις ασφάλειές καθώς και τις μηνιαίες αναφορές.

4.2.5 Dashboard

Το συγκεκριμένο app παίζει περισσότερο βοηθητικό ρόλο καθώς είναι αυτό που συνδέει όλα τα προηγούμενα apps. Παράλληλα περιέχει τον κώδικα για την σύνδεση και την αποσύνδεση των χρηστών από την εφαρμογή

```
from django.conf.urls import url, include
from dashboard import views

urlpatterns = [
    url(r'^$', views.dashboard, name='dashboard'),
    url(r'^accounts/', include('accounts.urls'), name='accounts'),
    url(r'^insurances/', include('insurances.urls'), name='insurances'),
    url(r'^loans/', include('loans.urls'), name='loans'),
    url(r'^reports/', include('reports.urls'), name='reports'),
]
```

Εικόνα 4-9 Dashboard URLs

4.2.6 Website

Το app αυτό είναι υπεύθυνο για την προβολή του site της εφαρμογής και ένας χρήστης μέσα από αυτό μπορεί να εγγραφή και να αγοράσει κάποιο δάνειο ή ασφάλεια.

```
from django.conf.urls import url, include
from website import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^purchase/insurance/$', views.insurance_purchase, name='insurance_purchase'),
    url(r'^purchase/loan/$', views.loan_purchase, name='loan_purchase'),
]
```

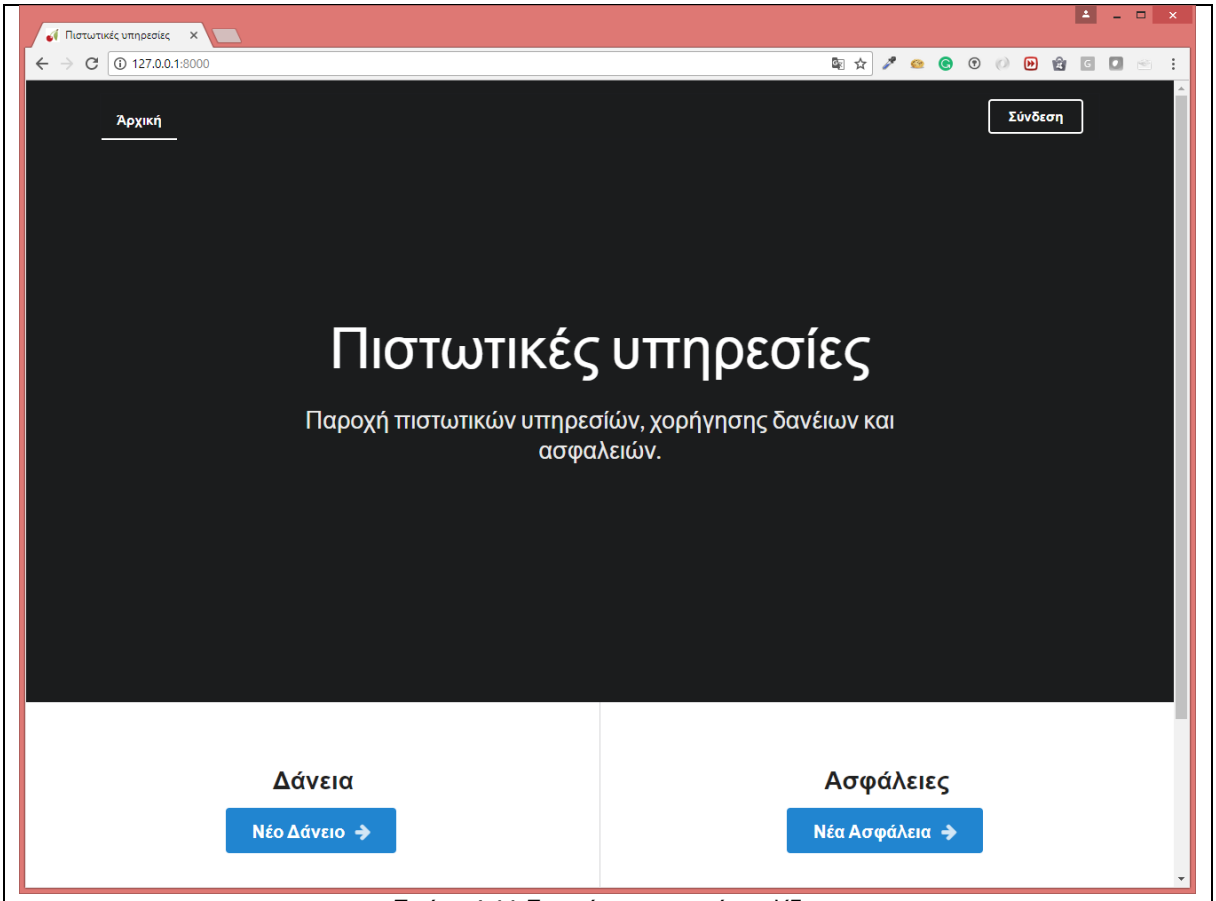
Εικόνα 4-10 Website URLs

4.3 Η Πλατφόρμα

Θα παρουσιαστούν βήμα-βήμα όλες οι δυνατότητες της πλατφόρμας.

4.3.1 Αρχική σελίδα

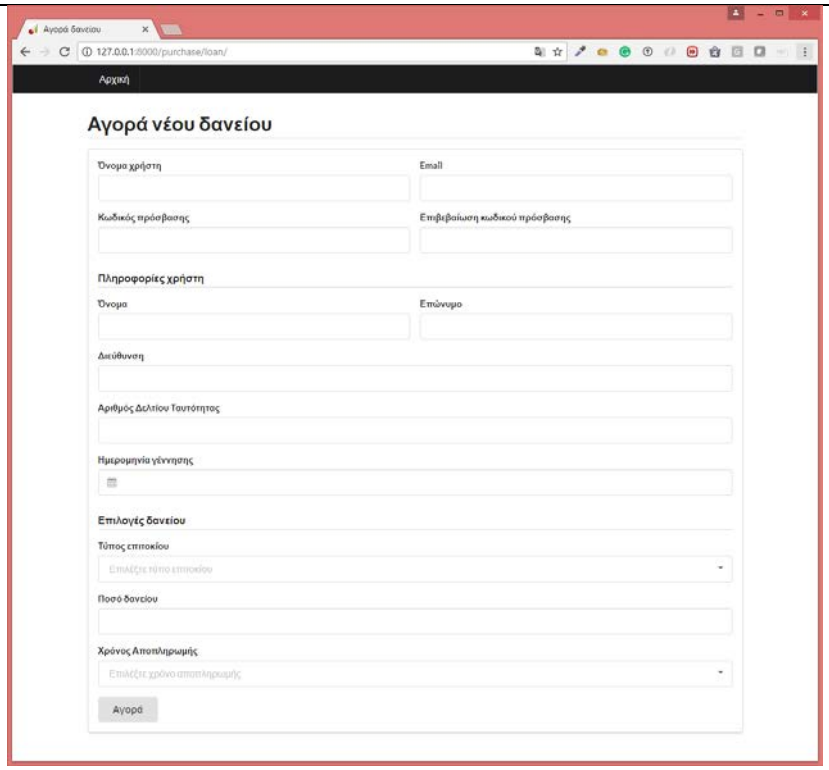
Ένας χρήστης μπαίνοντας στην αρχική σελίδα μπορεί να αγοράσει έναν νέο δάνειο η ασφάλεια. Επίσης έχει την δυνατότητα από το κουμπί σύνδεση να συνδεθεί στο περιβάλλον διαχείρισης και να διαχειριστεί τις υπηρεσίες του σε περίπτωση που είναι έναν ήδη εγγεγραμμένος χρήστης.



Εικόνα 4-11 Στιγμιότυπο αρχικής σελίδας

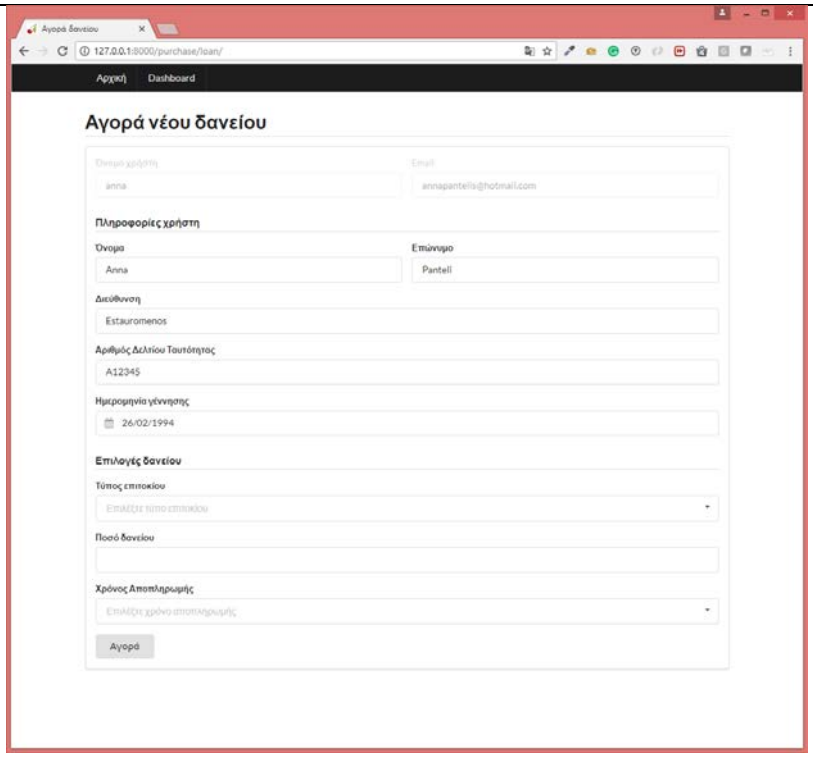
4.3.2 Αγορά νέου δανείου

Επιλέγοντας την αγορά νέου δάνειου ο χρήστης μεταφέρεται στην φόρμα αγοράς νέου δανείου όπου θα πρέπει να συμπληρώσει όλα τα απαραίτητα προσωπικά του στοιχεία καθώς και στοιχεία που αφορούν το δάνειο, όπως τον τύπου του επιτοκίου, το ποσό που θέλει να δανειστεί αλλά και τον χρόνο αποπληρωμής.



Εικόνα 4-12 Στιγμιότυπο αγοράς νέου δανείου

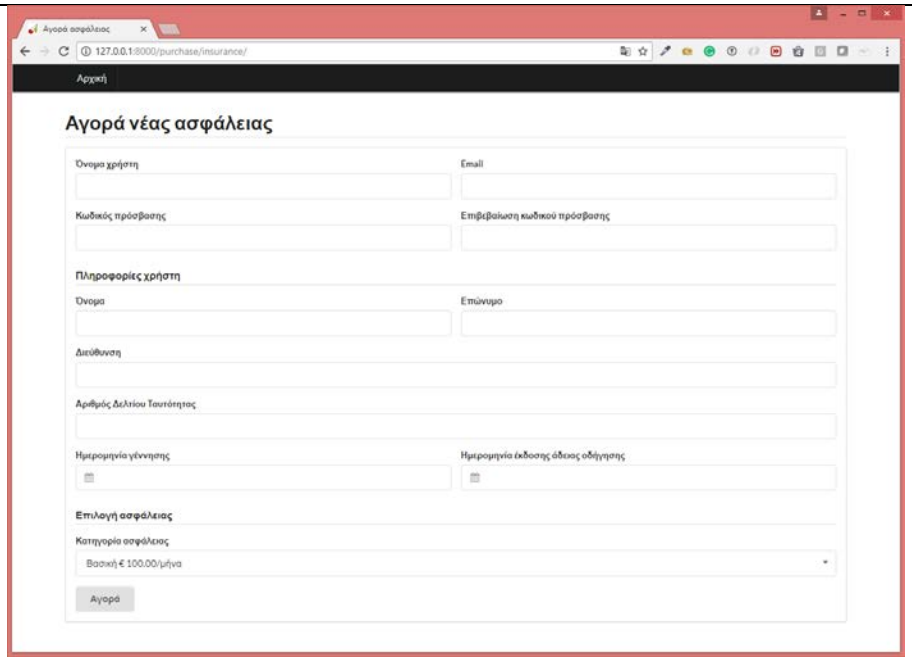
Στην περίπτωση που ο χρήστης θέλει να πάρει ένα καινούριο δάνειο και είναι ένας ήδη εγγεγραμμένος χρήστης τα προσωπικά του στοιχεία συμπληρώνονται αυτόματα και έχει την δυνατότητα να αλλάξει κάποια αν αυτό απαιτείται.



Εικόνα 4-13 Στιγμιότυπο αγοράς νέου δανείου (εγγεγραμμένος χρήστης)

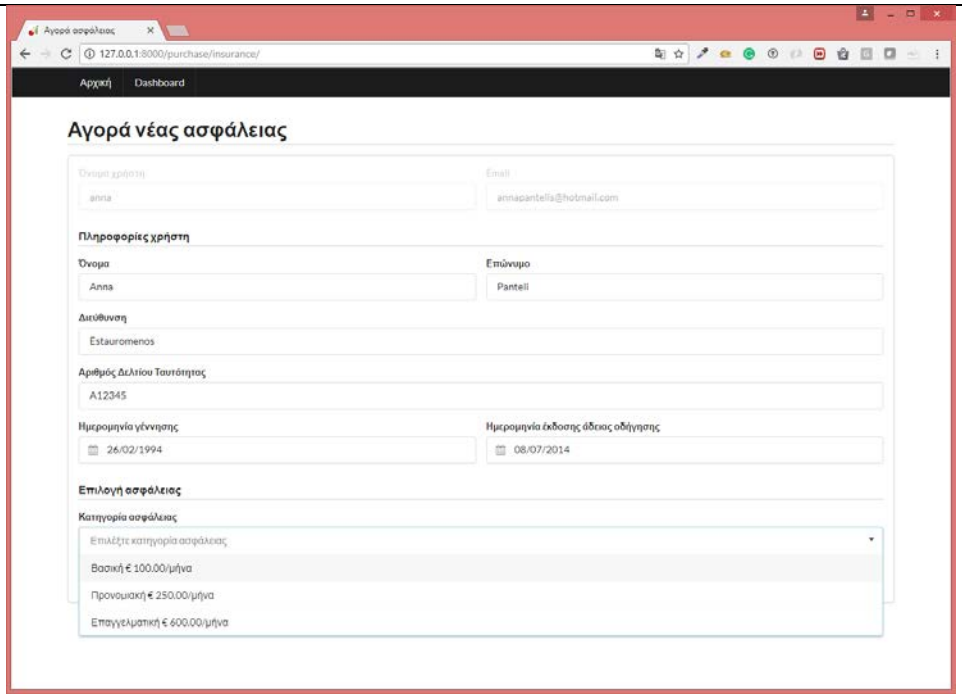
4.3.3 Αγορά νέας ασφάλειας

Επιλέγοντας την αγορά νέας ασφάλειας ο χρήστης μεταφέρεται στην φόρμα αγοράς νέας ασφάλειας όπου θα πρέπει να συμπληρώσει όλα τα απαραίτητα προσωπικά του στοιχεία καθώς και να επιλέξει την κατηγορίας ασφάλειας που επιθυμεί.



Εικόνα 4-14 Στιγμιότυπο αγοράς νέας ασφάλειας

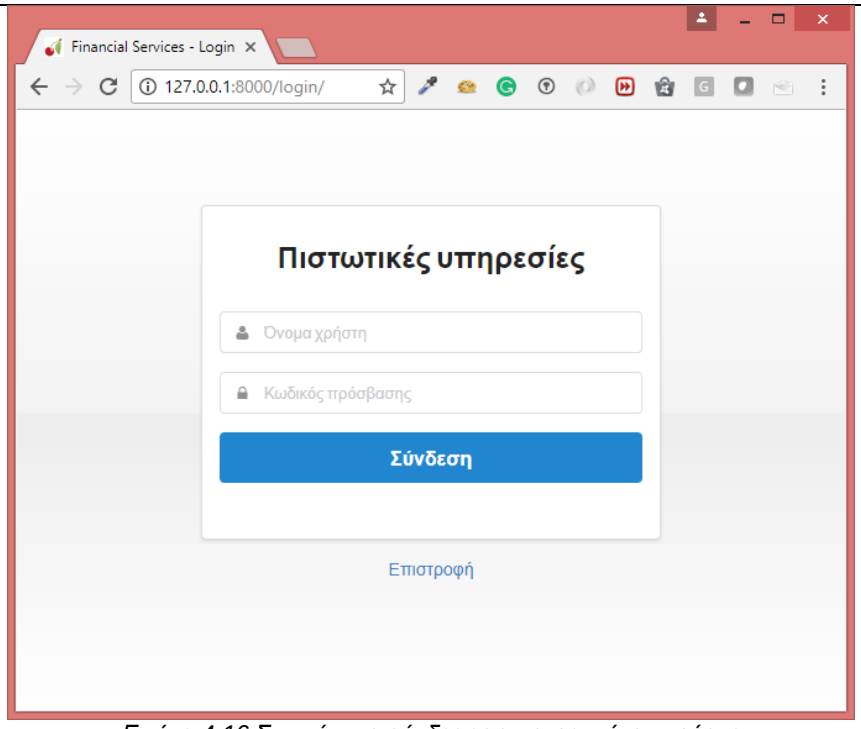
Στην περίπτωση που ο χρήστης θέλει να πάρει μια καινούργια ασφάλεια και είναι ένας ήδη εγγεγραμμένος χρήστης τα προσωπικά του στοιχεία συμπληρώνονται αυτόματα και έχει την δυνατότητα να αλλάξει κάποια αν αυτό απαιτείται.



Εικόνα 4-15 Στιγμιότυπο αγοράς νέας ασφάλειας (εγγεγραμμένος χρήστης)

4.3.4 Σύνδεση εγγεγραμμένου χρήστη

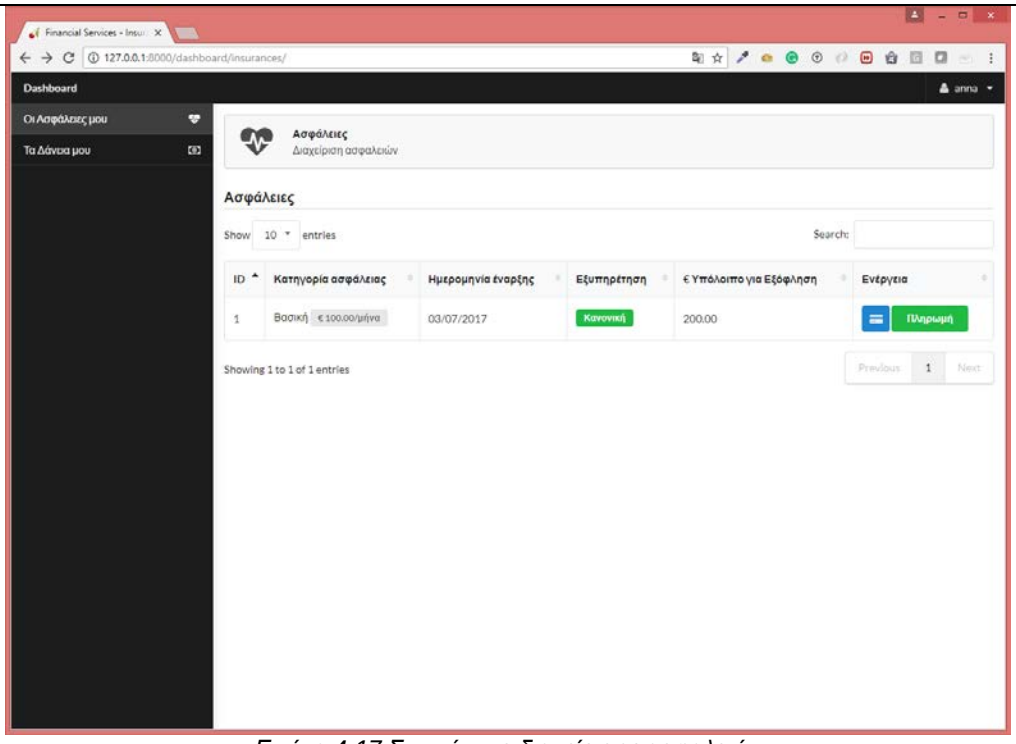
Οι εγγεγραμμένοι χρήστες έχουν την δυνατότητα να συνδεθούν στην πλατφόρμα για να διαχειριστούν της υπηρεσίες τους.



Εικόνα 4-16 Στιγμιότυπο σύνδεσης εγγεγραμμένου χρήστη

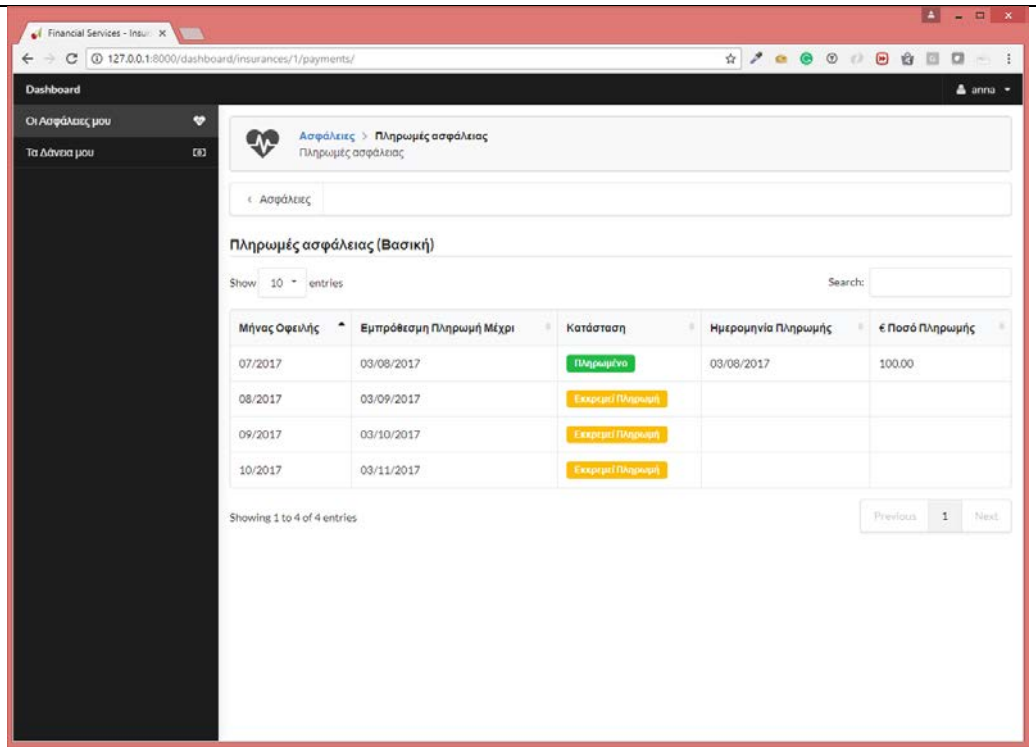
4.3.5 Διαχείριση ασφαλειών

Μετά την αγορά κάποιας ασφάλειας ο χρήστης μεταφέρεται αυτόματα στο περιβάλλον διαχείρισης όπου μπορεί να δει την κατάσταση των υπηρεσιών του.



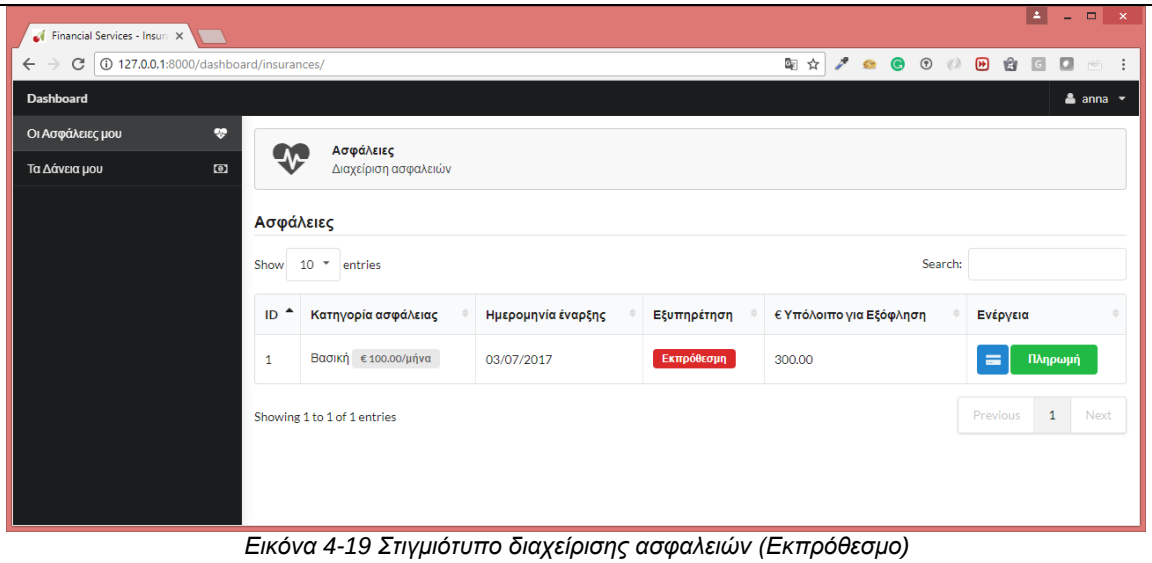
Εικόνα 4-17 Στιγμιότυπο διαχείρισης ασφαλειών

Μέσα από το περιβάλλον διαχείρισης ο χρήστης μπορεί να δει τις πληρωμές που έχει κάνει γι' αυτή την ασφάλεια η τις πληρωμές που εκκρεμούν.



Εικόνα 4-18 Στιγμιότυπο διαχείρισης πληρωμών ασφάλειας

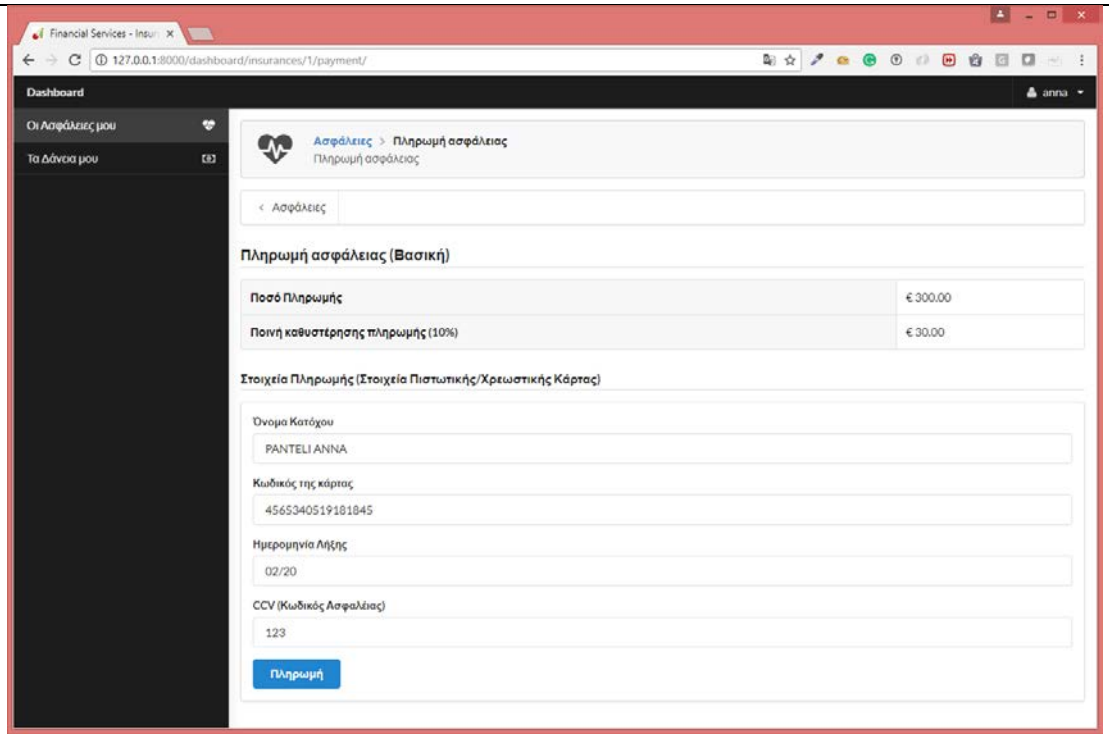
Όταν κάποιος πελάτης καθυστερήσει την πληρωμή περισσοτέρων από δύο δόσεων τότε η κατάσταση της ασφάλειας αλλάζει σε εκπρόθεσμη.



Εικόνα 4-19 Στιγμιότυπο διαχείρισης ασφαλειών (Εκπρόθεσμο)

4.3.6 Πληρωμή ασφάλειας

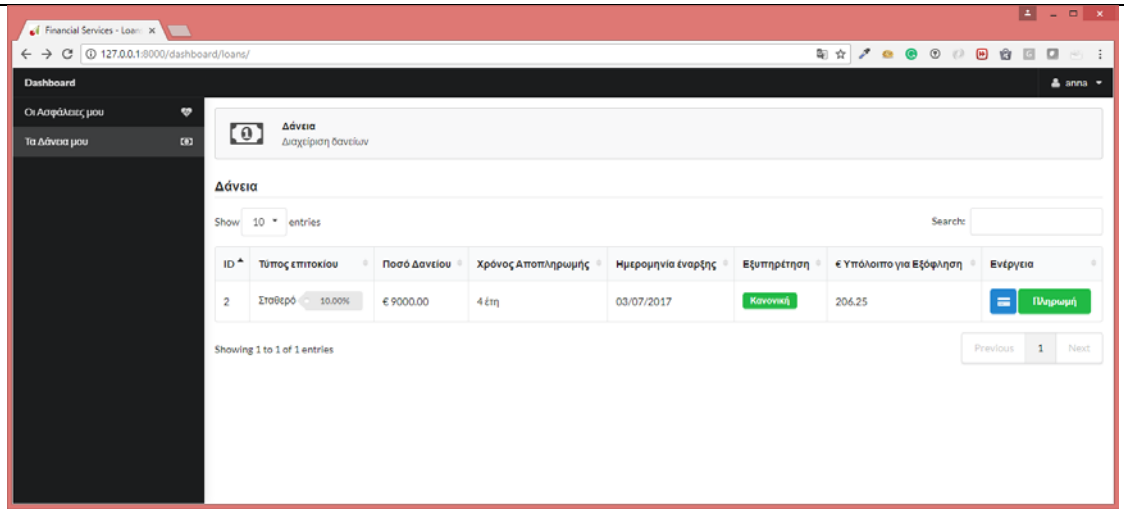
Ο πελάτης έχει την δυνατότητα από το περιβάλλον διαχείρισης να προχωρήσει στην εξόφληση των δόσεων με την χρήση πιστωτικής κάρτας. Σε περίπτωση που έχουμε εκπρόθεσμή εξυπηρέτηση των δόσεων ο χρήστης επιβαρύνεται με ένα 10% στο αρχικό ποσό πληρωμής.



Εικόνα 4-20 Στιγμιότυπο πληρωμής ασφάλειας

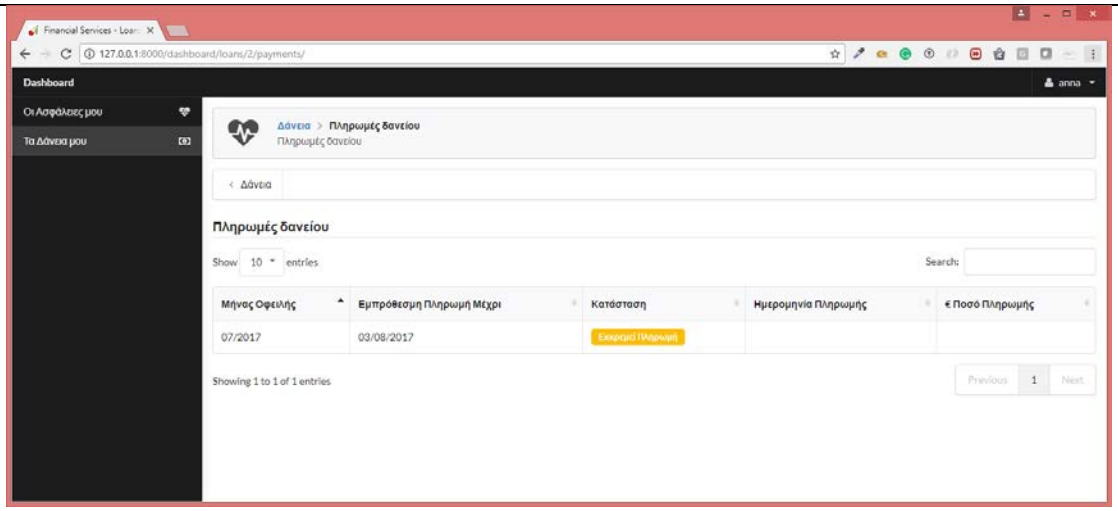
4.3.7 Διαχείριση δανείων

Μετά την αγορά κάποιας ασφάλειας ο χρήστης μεταφέρεται αυτόματα στο περιβάλλον διαχείρισης όπου μπορεί να δει την κατάσταση των υπηρεσιών του, όπου μπορεί να δει πληροφορίες για την κατάσταση των δανείων.



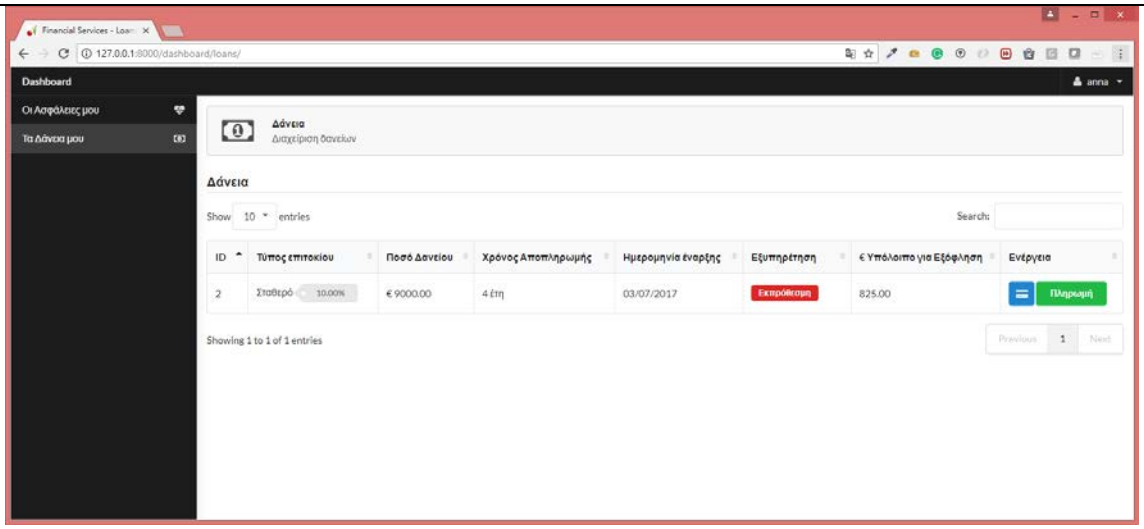
Εικόνα 4-21Στιγμιότυπο διαχείρισης δανείων

Μέσα από το περιβάλλον διαχείρισης ο χρήστης μπορεί να δει τις πληρωμές που έχει κάνει γι' το δάνειο η τις πληρωμές που εκκρεμούν.



Εικόνα 4-22 Στιγμιότυπο διαχείρισης πληρωμών δανείου

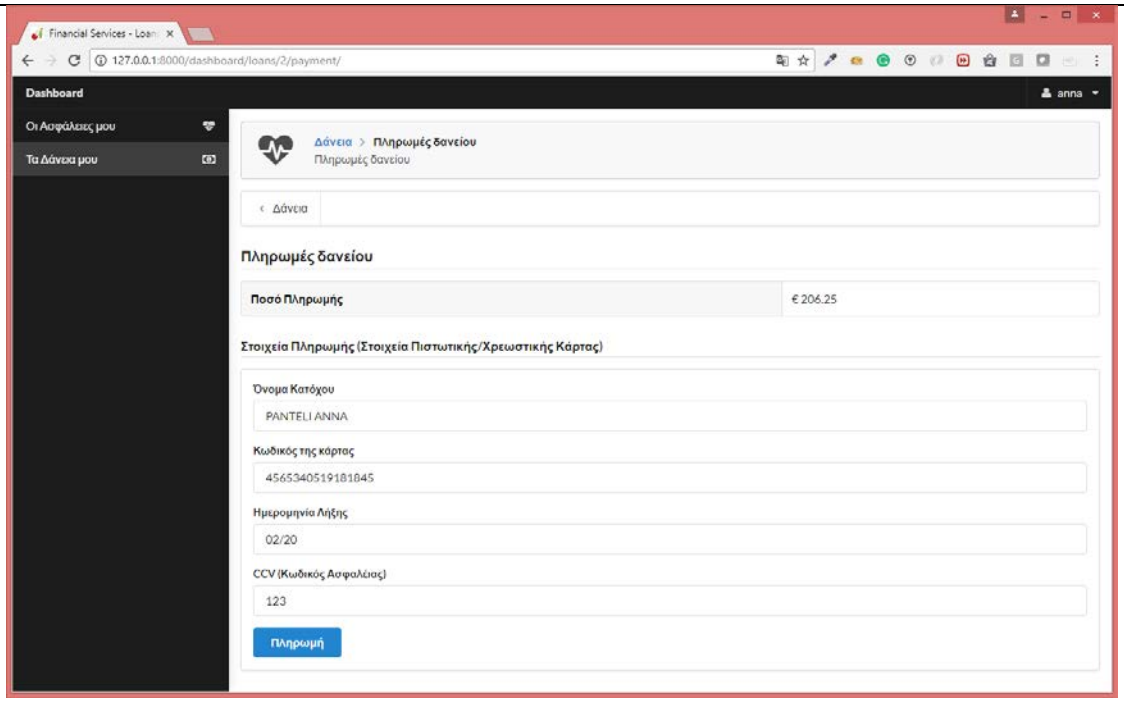
Όταν κάποιος πελάτης καθυστερήσει την πληρωμή περισσοτέρων από δύο δόσεων τότε η κατάσταση του δανείου αλλάζει σε εκπρόθεσμη.



Εικόνα 4-23 Στιγμιότυπο διαχείρισης δανείου (Εκπρόθεσμο)

4.3.8 Πληρωμή δανείου

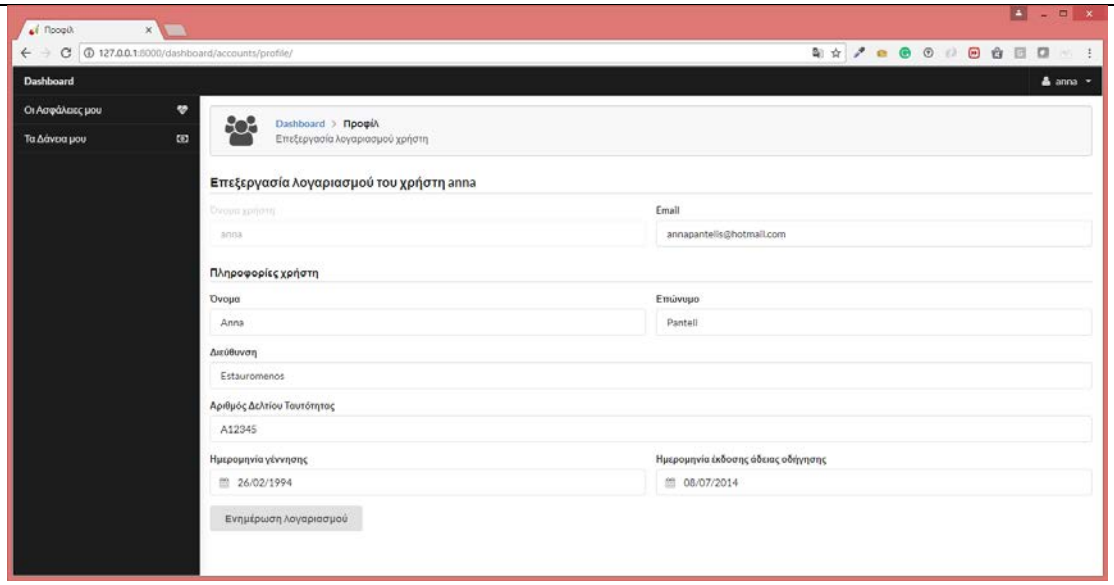
Ο πελάτης έχει την δυνατότητα από το περιβάλλον διαχείρισης να προχωρήσει στην εξόφληση των δόσεων με την χρήση πιστωτικής κάρτας. Σε περίπτωση που έχουμε εκπρόθεσμή εξυπηρέτηση των δόσεων ο χρήστης επιβαρύνεται με ένα 5% στο αρχικό ποσό πληρωμής.



Εικόνα 4-24 Στιγμιότυπο πληρωμής δανείου

4.3.9 Διαχείριση Προφίλ

Ο κάθε χρήστης έχει την δυνατότητα από το περιβάλλον διαχείρισης να κάνει αλλάξει στα στοιχεία της καρτέλας του, να αλλάζει στοιχεία από το προφίλ του.



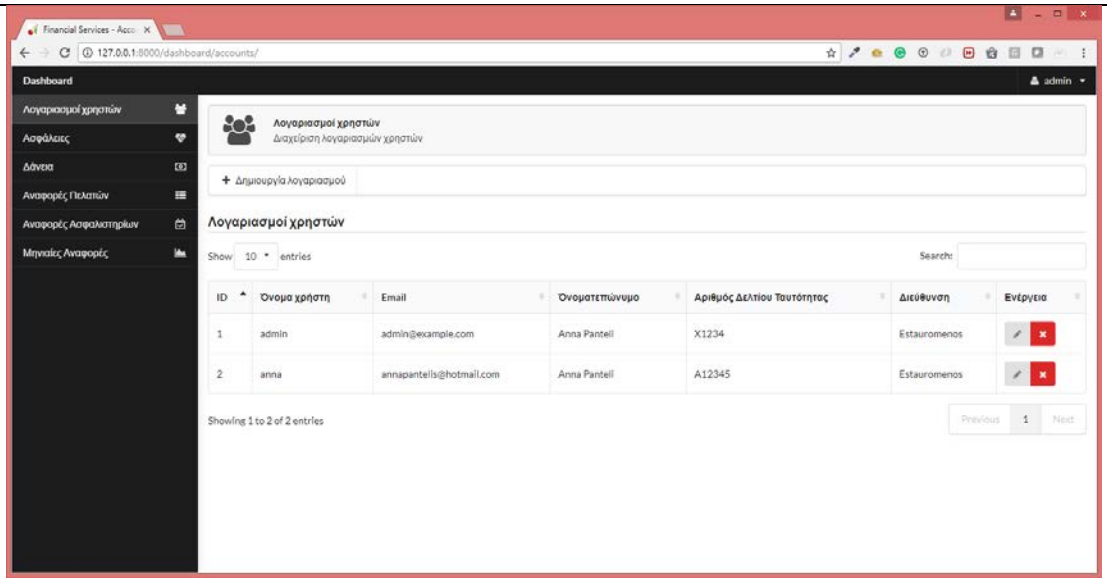
Εικόνα 4-25 Στιγμιότυπο διαχείρισης προφίλ

4.3.10 Περιβάλλον Διαχειριστή

Ο διαχειριστής της πλατφόρμας έχει διαθέσιμα μια σειρά από εργαλεία για διαχειρίζεται του χρήστες, τα δάνεια και τις ασφάλειες των χρηστών καθώς και σε εργαλεία που του δίνουν την δυνατότητα να παρακολουθεί την κατάσταση των πληρωμών και σε στατιστικά στοιχεία.

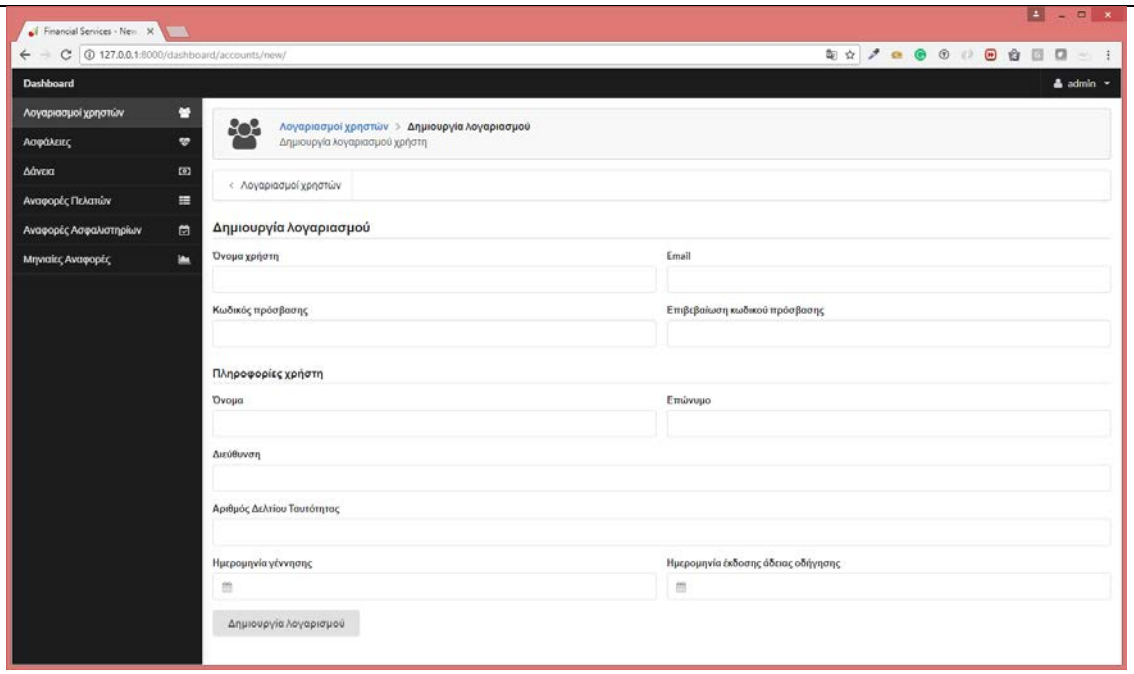
4.3.10.1 Διαχείριση χρηστών

Ο διαχειριστής της πλατφόρμας έχει την δυνατότητα να μέσα από το περιβάλλον διαχείρισης να δει όλους τους εγγεγραμμένου χρήστης στην πλατφόρμα.



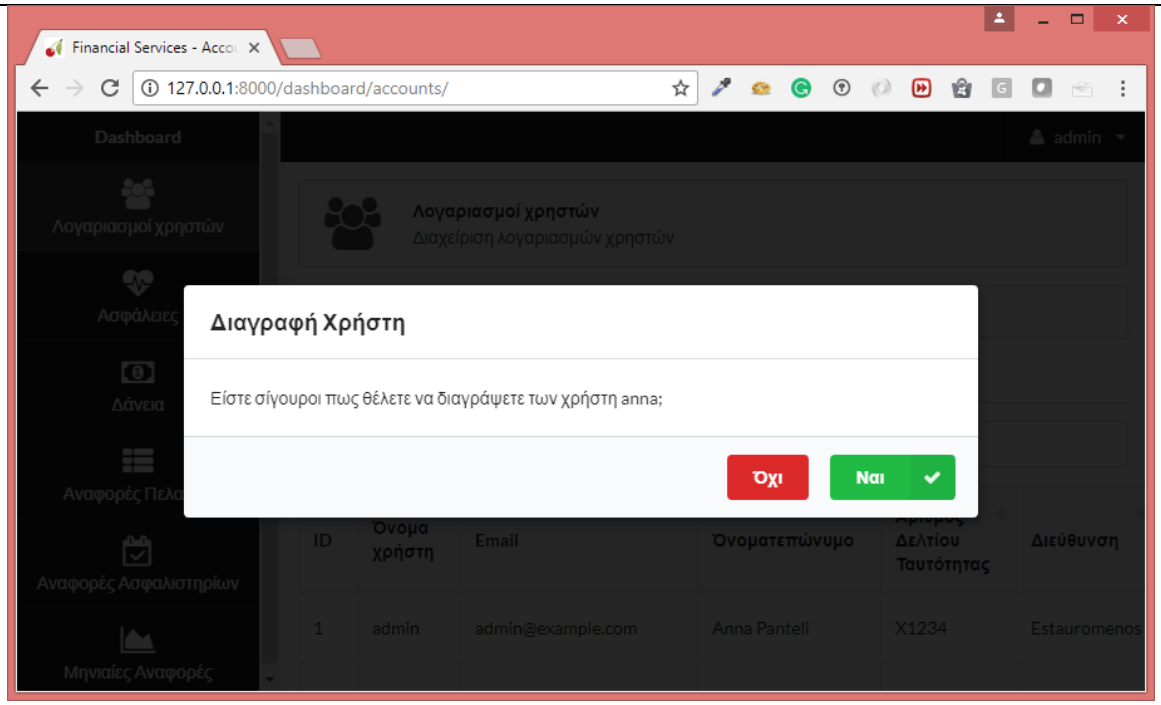
Εικόνα 4-26 Στιγμιότυπο διαχείρισης χρηστών

Επίσης έχει την δυνατότητα να δημιουργήσεις νέους χρήστες



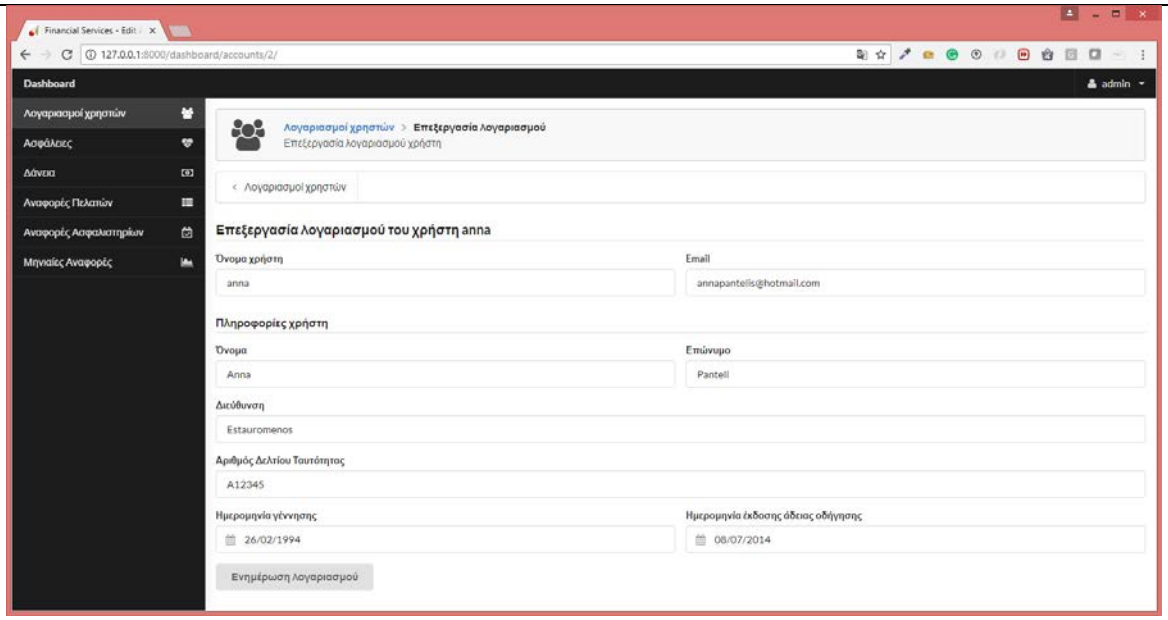
Εικόνα 4-27 Στιγμιότυπο δημιουργίας νέου χρήστη

Ο διαχειριστής μπορεί επίσης να προχωρήσει στην διαγραφή κάποιου χρήστη



Εικόνα 4-28 Στιγμιότυπο διαγραφής χρήστη

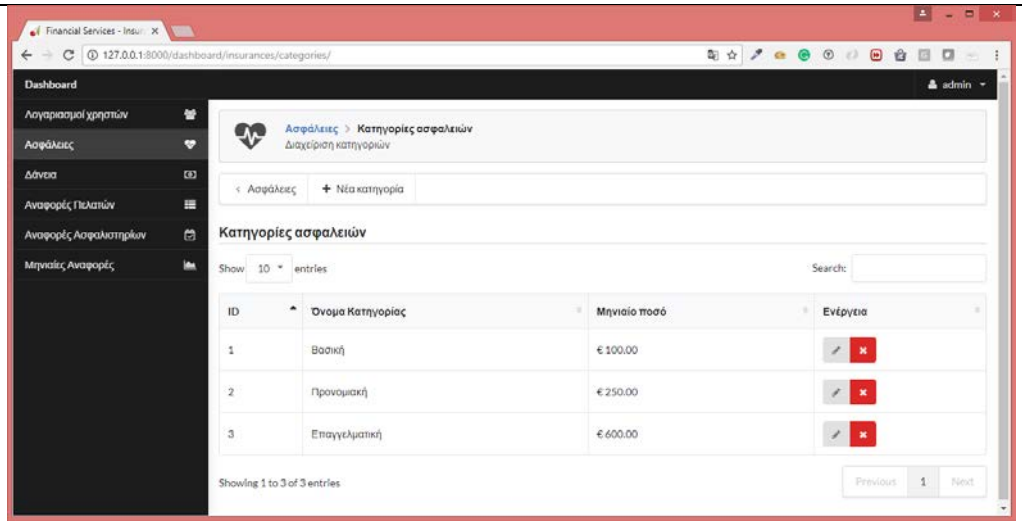
Μπορεί επίσης να διαχειριστή τα στοιχεία κάποιου χρήστη



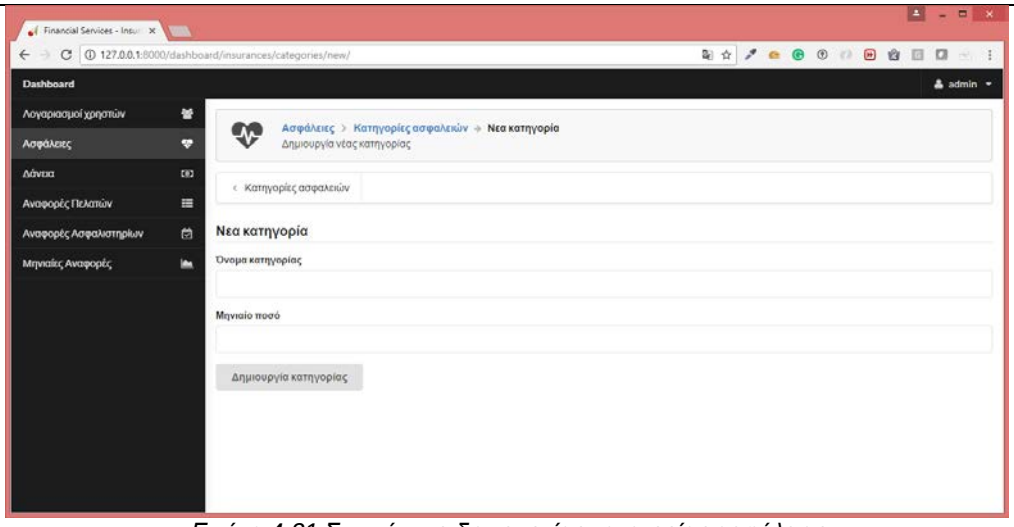
Εικόνα 4-29 Στιγμιότυπο διαχείρισης στοιχείων του χρήστη

4.3.10.2 Διαχείριση Κατηγοριών των Ασφαλειών

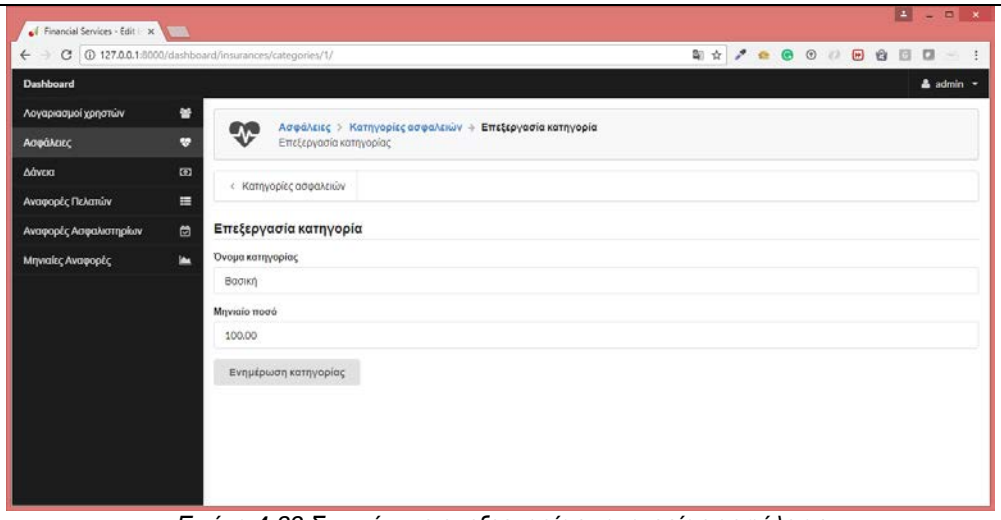
Μέσα από την το περιβάλλον διαχείρισης υπάρχει η δυνατότητα της δημιουργίας των κατηγοριών των ασφαλειών και η επεξεργασία των κατηγοριών που υπάρχουν ήδη. Μπορεί να αλλάξει το όνομα και την τιμή.



Εικόνα 4-30 Στιγμιότυπο διαχείριση κατηγοριών των ασφαλειών



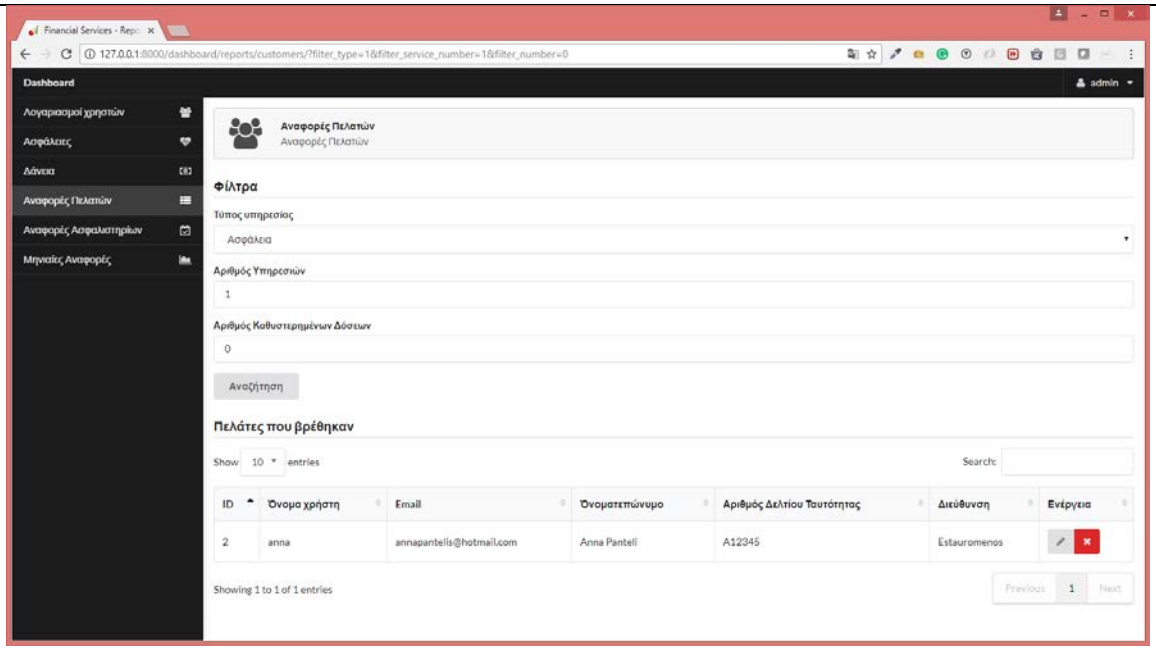
Εικόνα 4-31 Στιγμιότυπο δημιουργίας κατηγορίας ασφάλειας



Εικόνα 4-32 Στιγμιότυπο επεξεργασίας κατηγορίας ασφάλειας

4.3.10.3 Αναφορές Πελατών

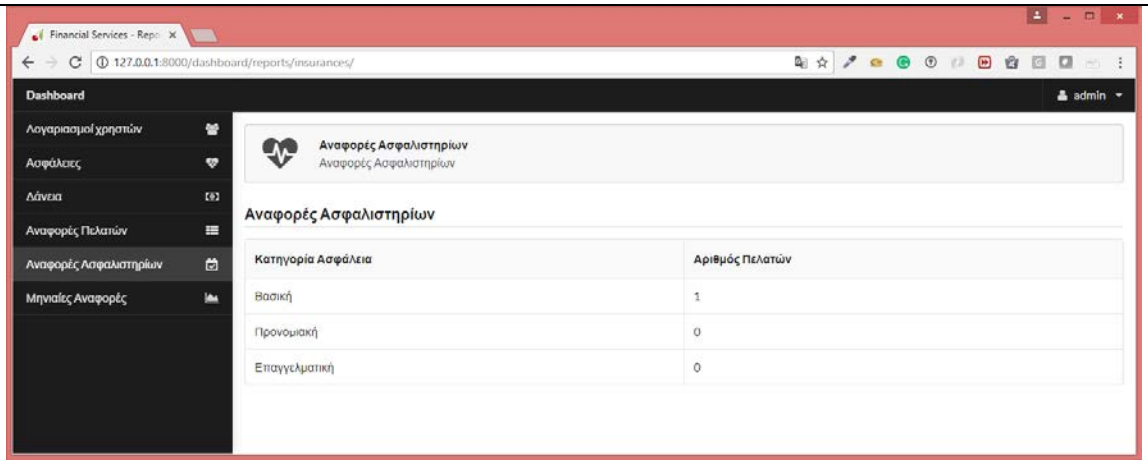
Ο διαχειριστής της πλατφόρμας πηγαίνοντας στην σελίδα “Αναφορές Πελατών” έχει την δυνατότητα να αναζητήσει πελάτες με βάσει την υπηρεσία που έχουν αγοράσει, τον αριθμό των υπηρεσιών που έχουν αλλά και τον αριθμό των δόσεων που οφείλουν.



Εικόνα 4-33 Στιγμιότυπο αναφορών πελατών

4.3.10.4 Αναφορές Ασφαλιστηρίων

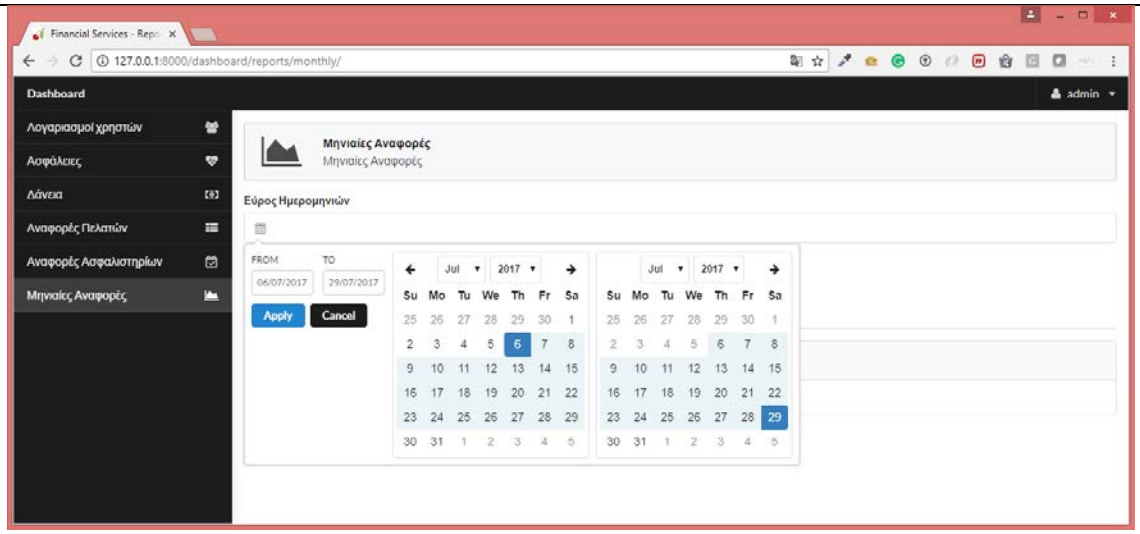
Από την σελίδα “Αναφορές Ασφαλιστηρίων” μπορεί να πάρει μια γενική εικόνα για του πελάτες που έχει κάθε κατηγορία ασφαλιστηρίου.



Εικόνα 4-34 Στιγμιότυπο αναφορών ασφαλιστηρίων

4.3.10.5 Μηνιαίες Αναφορές

Από την σελίδα “Μηνιαίες Αναφορές” ο διαχειριστής επιλέγοντας ένα εύρος ημερομηνιών έχει την δυνατότητα να δει τον αριθμό τον ασφαλειών και των δάνειων που έγιναν στο εύρος ημερομηνιών.



Εικόνα 4-35 Στιγμιότυπο μηνιαίων αναφορών

5 Συμπεράσματα

Η ιδέα της ανάπτυξης πλατφόρμας αυτής έγινε με την χρήση μεθόδων, εργαλείων και μοντέρνων τεχνολογιών που χρησιμοποιούνται σήμερα. Όπως έχουμε αναφέρει και στην εισαγωγή η ανάπτυξη αυτής της εφαρμογής επιδεικνύει τον συνδυασμό όλων των μεθόδων που χρησιμοποιούνται και σε εφαρμογές μεγαλύτερης κλίμακας όπου η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί σαν βάση. Η προσωπική μου εμπειρία είναι πολύ θετική καθώς είχα την ευκαιρία να ασχοληθώ με τεχνολογίες που σίγουρα θα μου είναι χρήσιμες στο μέλλον. Πολύ σημαντικό είναι η βελτίωση των γνώσεων μου στον προγραμματισμό αλλά και η εκμάθηση ενός μεγάλου αριθμού από νέες τεχνολογίες. Από την εργασία αυτή αποκόμισα πολύ σημαντικές γνώσεις το οποίο είναι πολύ σημαντικό για την μελλοντική επαγγελματική μου αποκατάσταση, επίσης είχα την ευκαιρία να εντοπίσω τις αδυναμίες που είχα και να βελτιωθώ.

6 Βιβλιογραφία

[1] https://www.w3schools.com/html/

[2] https://en.wikipedia.org/wiki/JavaScript

[3] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[4] https://semantic-ui.com/introduction/getting-started.html

[5] https://docs.djangoproject.com/en/1.11/

[6] https://media.readthedocs.org/pdf/django/latest/django.pdf

[7] https://jquery.com/

[8] https://www.sqlite.org/

[9] https://datatables.net/