ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ MARKETPLACE ΑΥΤΟΚΙΝΗΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΠΑΡΜΠΑΣ ΧΡΗΣΤΟΣ

Επιβλέπων

ΒΟΛΟΓΙΑΝΝΙΔΗΣ ΣΤΑΥΡΟΣ

ΣΕΡΡΕΣ - ΦΕΒΡΟΥΑΡΙΟΣ 2023

Πρόλογος

Στο πλαίσιο της αναζήτησης θέματος για την εκπόνηση της μεταπτυχιακής διπλωματικής εργασίας χρειάστηκε να ανατρέξω στην μνήμη μου όλες τις γνώσεις και δεξιότητες που κατάφερα να συλλέξω τόσο την περασμένη χρονιά σαν φοιτητής του προγράμματος μεταπτυχιακών σπουδών στην «ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ» όσο και στα χρόνια που πέρασαν όντας φοιτητής στο ΕΑΠ όπου πήρα τις βάσεις και αγάπησα την επιστήμη της πληροφορικής. Ήταν πραγματικά ένα μακρύ και επίπονο ταξίδι που κατά την διάρκεια του συνάντησα εμπόδια, αναποδιές αλλά και στεναχώριες, όλα αυτά όμως κατάφεραν να με βγάλουν περισσότερο δυνατό και τώρα πλέον πλησιάζοντας κοντά στον τελευταίο ίσος σταθμό αυτού του ταξιδιού έχω να πω ότι ήταν ένα υπέροχο ταξίδι. Γνώρισα νέα γνωστικά αντικείμενα και απέκτησα καινούργιες δεξιότητες. Γνώρισα νέους φίλους, με κάποιους συνεργαστήκαμε στα πλαίσια διαφόρων μαθημάτων για τις ανάγκες εκπόνησης κάποιων εργασιών κατά καιρούς, με άλλους δημιουργήθηκαν στενοί δεσμοί που κρατάνε ακόμα μετά από τόσα χρόνια. Επίσης γνώρισα και πολλούς πολύ αξιόλογους καθηγητές οι οποίοι με την σειρά τους κατάφεραν με τις γνώσεις, τις δεξιότητες και την εμπειρία τους να με κάνουν να αγαπήσω αυτό το γνωστικό αντικείμενο. Η επιστήμη της πληροφορικής είναι τόσο «πολυπαραγοντική» και δανείζομαι επίτηδες αυτόν τον όρο γιατί πραγματικά πιστεύω ότι είναι ίσως ο μοναδικός που μπορεί να περιγράψει τόσο γλαφυρά τα τόσα γνωστικά αντικείμενα που διαθέτει στην φαρέτρα της αυτή η επιστήμη, μαθηματικά, λογική, αλγόριθμοι, γλώσσες προγραμματισμού, βάσεις δεδομένων, στατιστική, δίκτυα και άλλα πολλά που το καθένα χτίζει όλο το οικοδόμημα που ονομάζεται επιστήμη της πληροφορικής. Έτσι ψάχνοντας για το θέμα της μεταπτυχιακής διπλωματικής εργασίας έχω την αίσθηση ότι ήταν μονόδρομος η επιλογή ενός θέματος που θα έχει ως στόχο την δημιουργία κάποιας εφαρμογής, με σκοπό να εκμεταλλευτώ τις γνώσεις και τις δεξιότητες που αποκτήθηκαν όλα αυτά τα χρόνια.

Θα ήθελα να ευχαριστήσω την οικογένεια μου η οποία με στήριξε και με ενθάρρυνε όλα αυτά τα χρόνια. Επίσης θα ήθελα να ευχαριστήσω όλους τους καθηγητές που γνώρισα όλα αυτά τα χρόνια και ιδιαίτερα τον εισηγητή μου Κ. Βολογιαννίδη όπου στο πλαίσιο των μαθημάτων που γνωριστήκαμε κατάφερε άθελα του να φυτέψει τον σπόρο για αυτή την εργασία. Επίσης τους υπόλοιπους καθηγητές της τριμερούς επιτροπής Κ. Χειλά και Κ. Στρουθόπουλο, τους Κ. Βαρσάμη, Κ. Τσιμπίρη αλλά και τους υπόλοιπους καθηγητές του προγράμματος μεταπτυχιακών σπουδών στην «ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ» που με την εξαιρετική δουλειά που κάνουν κατάφεραν να κάνουν αυτό το τελευταίο σταθμό του ταξιδιού τόσο ευχάριστο και δημιουργικό.

Περίληψη

Στα πλαίσια του προγράμματος μεταπτυχιακών σπουδών και για τις ανάγκες της διπλωματικής μεταπτυχιακής εργασία με τίτλο «**Δημιουργία ενός marketplace αυτοκινήτων**» έχει υλοποιηθεί μια web based εφαρμογή η οποία έχει την δυνατότητα να διαχειρίζεται ένα σύστημα κρατήσεων για ενοικιάσεις αυτοκινήτων. Η εργασία έχει υλοποιηθεί με τις κατάλληλες γλώσσες προγραμματισμού Python, HTML, CSS, JavaScript, Βάσεις δεδομένων.

Υπάρχουν τρείς κατηγορίες χρηστών, ο διαχειριστής, η εταιρία (υπάλληλος) και ο τελικός χρήστης. Ο χρήστης της κάθε κατηγορίας έχει και διαφορετικά δικαιώματα στην εφαρμογή. Για παράδειγμα ο χρήστης «υπάλληλος – εταιρία» μπορεί να προσθέσει να ενημερώσει και να διαγράψει ένα αυτοκίνητο από την βάση δεδομένων. Επίσης έχει την δυνατότητα να δει όλες τις ενεργές και τις παλαιότερες ενοικιάσεις που έχουν γίνει. Ο τελικός χρήστης μπορεί να κάνει αναζήτηση με βάσει τις ημερομηνίες που θέλει να πραγματοποιήσει την ενοικίαση ή με βάση αναζήτησης με λέξεις κλειδιά πχ μάρκα οχήματος ή μοντέλο. Υπάρχει η δυνατότητα ο τελικός χρήστης να αποστείλει εκδήλωση ενδιαφέροντος για μια περίοδο ή κάποιο όχημα. Επίσης μπορεί να κάνει εγγραφή στο σύστημα και να πραγματοποιήσει κράτηση. Μέσω του προφίλ του χρήστη μπορεί να διαχειριστεί τις ενοικιάσεις που έχει κάνει όπως επίσης να ενημερώσει τα προσωπικά του στοιχεία. Τέλος ο διαχειριστής έχει όλες τις παραπάνω δυνατότητες όπως επίσης είναι ο μοναδικός που έχει πλήρη πρόσβαση στην βάση, μπορεί να δημιουργεί εταιρία ενοικιάσεων, να δίνει δικαιώματα στους χρήστες ανάλογα με το τι είναι επιθυμητό να μπορούν να κάνουν. Είναι ο μοναδικός που έχει το δικαίωμα να διαγράψει κάποιον χρήστη ή να του αλλάξει τα δικαιώματα. Τέλος έχει δημιουργηθεί ένα πλήρως λειτουργικό rest framework το οποίο με την σειρά του δίνει την δυνατότητα για μελλοντική αναβάθμιση και επέκταση των δυνατοτήτων που παρέχει η εφαρμογή, π.χ. δημιουργία εφαρμογής για κινητές συσκευές.

Λέξεις κλειδιά :

Python, HTML, CSS, JavaScript, Django, Pycharm, Βάσεις δεδομένων, marketplace, αυτοκίνητα, web-based, SQLITE3,

Summary

Within the framework of the postgraduate studies program and for the needs of the postgraduate thesis entitled “Creation of a car marketplace” a web based application has been implemented which has the ability to manage a reservation system for car rentals. The application has been implemented with the appropriate programming languages Python, HTML, CSS, JavaScript, database.

There are three categories of users, the administrator, the company (employee) and the end user. The user of each category has different rights in the application. For example, the user "employee - company" can add, update and delete a car from the database. It also has the ability to see all active and older rentals that have been made. The end user can search based on the dates he wants to rent or based on keywords such as vehicle brand or model. It is possible for the end user to send an expression of interest for a period or a vehicle. He can also register in the system and make a reservation. Through the user's profile he can manage the rentals he has made as well as update his personal information. Finally, the administrator has all the above possibilities as well as he is the only one who has full access to the database, he can create a rental company, give rights to the users depending on what they want to be able to do. He is the only one who has the right to delete a user or change his rights. Finally, a fully functional rest framework has been created, which in turn allows for future upgrades and expansion of the capabilities provided by the application, e.g. mobile app creation.

Keywords:

Python, HTML, CSS, JavaScript, Django, Pycharm, database, marketplace, car, web-based, SQLITE3,

Πίνακας Περιεχομένων

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της διπλωματικής

Στα πλαίσια αυτής της πτυχιακής θα υλοποιηθεί μια Web εφαρμογή η οποία θα έχει τα ακόλουθα χαρακτηριστικά:

- Θα υπάρχουν τρεις χρήστες, ο διαχειριστής, η εταιρεία και ο τελικός ανώνυμος χρήστης.
- Ο διαχειριστής θα μπορεί να προσθέτει εταιρείες στην πλατφόρμα αλλά και να βλέπει μετρικές για την συνολική χρήση του συστήματος.
- Μια εταιρεία θα μπορεί να εγγράφεται στην πλατφόρμα και μετά από αποδοχή του διαχειριστή, θα μπορεί να βάζει το στόλο της και την διαθεσιμότητά του κάθε αυτοκινήτου και να δέχεται αιτήσεις για κρατήσεις από χρήστες του διαδικτύου.
- Ο ανώνυμος χρήστης θα μπορεί να βρίσκει με διάφορα κριτήρια το αυτοκίνητο που τον ενδιαφέρει και να κάνει κράτηση.
- Ο διαχειριστής και η εταιρεία θα μπορεί να βλέπει κάποια συνολικά στατιστικά με την μορφή πινάκων αλλά και γραφημάτων.

Ο στόχος της διπλωματικής πτυχιακής εργασίας είναι η δημιουργία ενός marketplace ενοικίασης αυτοκινήτων. Ως απώτερος σκοπός είναι η εμβάθυνση στα γνωστικά αντικείμενα της ανάπτυξης λογισμικού. Η εφαρμογή έχει υλοποιηθεί με τις κατάλληλες γλώσσες προγραμματισμού Python, HTML, CSS, JavaScript, Βάσεις δεδομένων. Η υλοποίηση έγινε εξολοκλήρου με την βοήθεια του IDE Pycharm και σαν framework [1]χρησιμοποιήθηκε η DJANGO.

Έχει υλοποιηθεί μια βάση δεδομένων όπου υπάρχει δυνατότητα να διαχειρίζεται εταιρίες ενοικιάσεων, χρήστες με διαβάθμιση δικαιωμάτων, αυτοκίνητα, ενοικιάσεις. Επίσης έχει υλοποιηθεί και ένα Django-rest-framework [2] το οποίο μας επιτρέπει σε μελλοντική αναβάθμιση των υπηρεσιών που προσφέρει η web εφαρμογή και επέκταση αυτών σε κινητές συσκευές.

[1] https://en.wikipedia.org/wiki/Software_framework
[2] https://www.django-rest-framework.org/

1.2 Οργάνωση του τόμου

Στα κεφάλαια που θα ακολουθήσουν γίνεται μια αναλυτική παρουσίαση ολόκληρης της εφαρμογής σε θεωρητικό και σε πρακτικό επίπεδο. Κατά βάση η δομή που ακολουθείται περιλαμβάνει τρία βασικά κεφάλαια και το παράρτημα. Στην συνέχεια θα ακολουθήσει μια συνοπτική παρουσίαση των κεφαλαίων.

Το **ΚΕΦΑΛΑΙΟ 2 – Ανάλυση και σχεδίαση** πραγματεύεται την ανάλυση και τον σχεδιασμό της εφαρμογής. Είναι χωρισμένο σε δυο ενότητες, την ενότητα ***2.1 Θεωρητική ανάλυση*** στην οποία γίνεται μια ανάλυση του θεωρητικού πλαισίου που στηρίχτηκε η εφαρμογή, ποιες τεχνικές και ποια εργαλεία χρησιμοποιήθηκαν και οι γλώσσες και τεχνικές προγραμματισμού. Επίσης για κάθε γλώσσα και εργαλεία υπάρχει και μια μικρή παρουσίαση. Στην ενότητα ***2.2 Βάση δεδομένων*** γίνεται ανάλυση της βάσης δεδομένων, η αρχιτεκτονική που χρησιμοποιήθηκε όπως επίσης παρουσιάζεται και το μοντέλο οντοτήτων συσχετίσεων.

Το **ΚΕΦΑΛΑΙΟ 3 – Υλοποίηση** αποτελεί το κυρίως μέρος της εργασίας. Γίνεται αναλυτική παρουσίαση της εφαρμογής παρέχοντας περιγραφή της χρήση της εφαρμογής για τα μοντέλα που δημιουργήθηκαν καθώς και για τις τεχνικές υλοποίησης. Είναι χωρισμένο σε τέσσερις ενότητες. Στην ενότητα ***3.1 Μοντέλα (Βάση δεδομένων)*** όπου παρουσιάζονται τα μοντέλα της βάσης και η δομή των κλάσεων, την ενότητα ***3.2 Admin View*** όπου γίνεται παρουσίαση του εργαλείου διαχείρισης της βάσης δεδομένων, την ενότητα ***3.3 Ερωτήματα στην βάση «views»*** στην οποία παρουσιάζονται τα views που υλοποιούν την εφαρμογή και τέλος την ενότητα ***3.4 Παρουσίαση Template και δυνατοτήτων την εφαρμογής*** στην οποία γίνεται και η βασική παρουσίαση του συνόλου των δυνατοτήτων που παρέχει η εφαρμογή. Η ενότητα 3.4 είναι χωρισμένη σε 13 υποενότητες όπου κάθε μία αφορά ξεχωριστή σελίδα της εφαρμογής ή κάποια σημαντική δυνατότητα που προσφέρει η εφαρμογή.

Στο **ΚΕΦΑΛΑΙΟ 4 – ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΟΥΣΙΑΣΗ** δίνονται οδηγίες για την εγκατάσταση της εφαρμογής στην πλατφόρμα www.pythonanywhere.com , η οποία είναι μια web hosting υπηρεσία που προσφέρει λύσεις για ακαδημαϊκούς, φοιτητές αλλά και επαγγελματίες που επιθυμούν να παρουσιάσουν τα project τους.

Στο **ΚΕΦΑΛΑΙΟ 5 – ΕΠΙΛΟΓΟΣ** γίνεται μια σύνοψη της εργασίας και παρουσιάζονται τα συμπεράσματα. Αποτελείται από δυο ενότητες, την ενότητα ***5.1 Σύνοψη και συμπεράσματα*** όπου παρουσιάζονται τα συμπεράσματα και η ενότητα ***5.2 Μελλοντικές επεκτάσεις*** όπου αναφέρονται οι μελλοντικές επεκτάσεις.

Τέλος το **ΠΑΡΑΡΤΗΜΑ** όπου υπάρχει μέρος του κώδικα της εφαρμογής.

Για την χρήση της εφαρμογής μπορεί να χρησιμοποιηθεί ο σύνδεσμος:

http://cbar3.pythonanywhere.com/

Τον κώδικα της εφαρμογής μπορούμε να τον βρούμε στον παρακάτω σύνδεσμο:

https://github.com/cbar3/RAC

ΚΕΦΑΛΑΙΟ 2 – Ανάλυση και σχεδίαση

2.1 Θεωρητική ανάλυση

Για την υλοποίηση της εφαρμογής ο κώδικας που χρησιμοποιήθηκε περιλάμβανε τεχνικές από διάφορες γλώσσες προγραμματισμού οι οποίες χρησιμοποιούνται για την δημιουργία web-based εφαρμογών. Για το frontend έγινε χρήση της HTML για την δομή της εφαρμογής, CSS για την μορφοποίηση και την εμφάνιση, JavaScript για την δημιουργία εργαλείων μέσα στην εφαρμογή καθώς και για την δημιουργία εφέ. Για το backend έγινε χρήση της γλώσσας προγραμματισμού Python η οποία μας έδωσε την δυνατότητα να στηθεί ολόκληρη η λειτουργική δομή της εφαρμογής. Σαν βάση δεδομένων έγινε χρήση της SQLITE3. Τέλος για την υλοποίηση χρησιμοποιήθηκε το framework DJANGO το οποίο είναι ένα εξιδεικευμένο framework της γλώσσας προγραμματισμού Python το οποίο χρησιμοποιείται για την ανάπτυξη web-based εφαρμογών. Τέλος η υλοποίηση έγινε με την χρήση του IDE Pycharm.

Στην συνέχεια θα γίνει μια συνοπτική ανάλυση των βασικών εννοιών, των εργαλείων και των γλωσσών προγραμματισμού που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής στο πλαίσιο της διπλωματικής πτυχιακής εργασίας.

Όπως αναφέρθηκε και παραπάνω για την υλοποίηση του frontend[3], δηλαδή για την δημιουργία του επιπέδου της παρουσίασης της εφαρμογής έγινε χρήση διαφόρων εργαλείων προγραμματισμού και ένα μείγμα από τις πιο διαδεδομένες γλώσσες που χρησιμοποιούνται για frontend υλοποιήσεις web-based εφαρμογών. Για την βασική δομή της εφαρμογής έγινε χρήση της **HTML**[4], Η **HTML** (**H**yper**T**ext **M**arkup **L**anguage, Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, τα στοιχεία της οποίας είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από *ετικέτες* (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα `<html>`), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα `<h1>` και `</h1>`), με την πρώτη να ονομάζεται *ετικέτα έναρξης* και τη δεύτερη *ετικέτα λήξης* (ή σε άλλες περιπτώσεις *ετικέτα ανοίγματος* και *ετικέτα κλεισίματος* αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

[3] https://en.wikipedia.org/wiki/Frontend_and_backend
[4] https://el.wikipedia.org/wiki/HTML

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες όπου μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους τους ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές.

Για την σχεδίαση και παρουσίαση χρησιμοποιήθηκε η **CSS**[5]**,** Η **CSS** (***Cascading Style Sheets*** – **διαδοχικά φύλλα ύφους** ή **επάλληλα φύλλα ύφους**) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στιλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται απαραίτητη.

Τέλος για το frontend χρησιμοποιήθηκε και η πασίγνωστη γλώσσα προγραμματισμού **JavaScript**[6]**,** Η **JavaScript** (**JS**) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης.

[5] https://el.wikipedia.org/wiki/CSS
[6] https://el.wikipedia.org/wiki/JavaScript

Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self[7] και Scheme[8]. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστραφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js[9]) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται **ECMAscript[10]**.

Για τον σχεδιασμό και την υλοποίηση του backend[11] , έγινε κατά κύριο λόγο χρήση της γλώσσας προγραμματισμού **Python[12],** Η **Python** είναι διερμηνευόμενη (interpreted), γενικού σκοπού (general-purpose) και υψηλού επιπέδου, γλώσσα προγραμματισμού. Ανήκει στις γλώσσες προστακτικού προγραμματισμού (*Imperative programming*) και υποστηρίζει τόσο το διαδικαστικό (*procedural programming*) όσο και το αντικειμενοστραφές (*object-oriented programming*) προγραμματιστικό υπόδειγμα (*programming paradigm*).

Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (*garbage collection ή GC*).

Δημιουργήθηκε από τον Ολλανδό Γκίντο βαν Ρόσσουμ (Guido van Rossum[13]) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε για πρώτη φορά το 1991. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν

[7] https://en.wikipedia.org/wiki/Self_(programming_language)
[8] https://en.wikipedia.org/wiki/Scheme_(programming_language)
[9] https://en.wikipedia.org/wiki/Node.js
[10] https://en.wikipedia.org/wiki/ECMAScript
[11] https://en.wikipedia.org/wiki/Frontend_and_backend
[12] https://el.wikipedia.org/wiki/Python
[13] https://en.wikipedia.org/wiki/Guido_van_Rossum

έννοιες σε λιγότερες γραμμές κώδικα από ό,τι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της. Μειονεκτεί στο ότι επειδή είναι διερμηνευόμενη είναι πιο αργή από τις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C και η C++. Για αυτόν τον λόγο δεν είναι κατάλληλη για γραφή λειτουργικών συστημάτων.

Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Η Python αναπτύσσεται ως ανοιχτό λογισμικό (open source) και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation[14]. Ο κώδικας διανέμεται με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL[15]. Το όνομα της γλώσσας προέρχεται από την ομάδα των Άγγλων κωμικών Μόντυ Πάιθον και δεν έχει καμιά σχέση με το φίδι πύθωνα, παρότι το λογότυπό της παραπέμπει σε κάτι τέτοιο.

Η βάση δεδομένων υλοποιήθηκε σε **SQLITE3**[16] , η SQLite είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων το οποίο είναι σχεδιασμένο και βασισμένο στην γλώσσα προγραμματισμού C. Έχει δημιουργηθεί από τον D. Richard Hipp[17], την άνοιξη του 2000 δουλεύοντας για την General Dynamics για λογαριασμό του Αμερικανικού ναυτικού. Δεν αποτελεί αυτόνομη εφαρμογή αλλά κατά κύριο λόγο χρησιμοποιείται από προγραμματιστές σαν βιβλιοθήκη η οποία ενσωματώνεται στις εφαρμογές τους. Είναι το πιο διαδεδομένο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων στον κόσμο καθώς χρησιμοποιείται για υλοποιήσεις σε web εφαρμογές, σε εφαρμογές για κινητές συσκευές και σε λειτουργικά συστήματα. Κατά βάση χρησιμοποιεί την σύνταξη της PostgreSQL[18].

Τόσο η χρήση της Python σαν γλώσσα υλοποίησης του backend όσο και η SQLite σαν βάση δεδομένων υποστηρίζονται από το πολύ δημοφιλές web framework[19] Django[20] , το οποίο

[14] https://en.wikipedia.org/wiki/Python_Software_Foundation
[15] https://en.wikipedia.org/wiki/GNU_General_Public_License
[16] https://el.wikipedia.org/wiki/SQLite
[17] https://en.wikipedia.org/wiki/D._Richard_Hipp
[18] https://en.wikipedia.org/wiki/PostgreSQL
[19] https://en.wikipedia.org/wiki/Web_framework

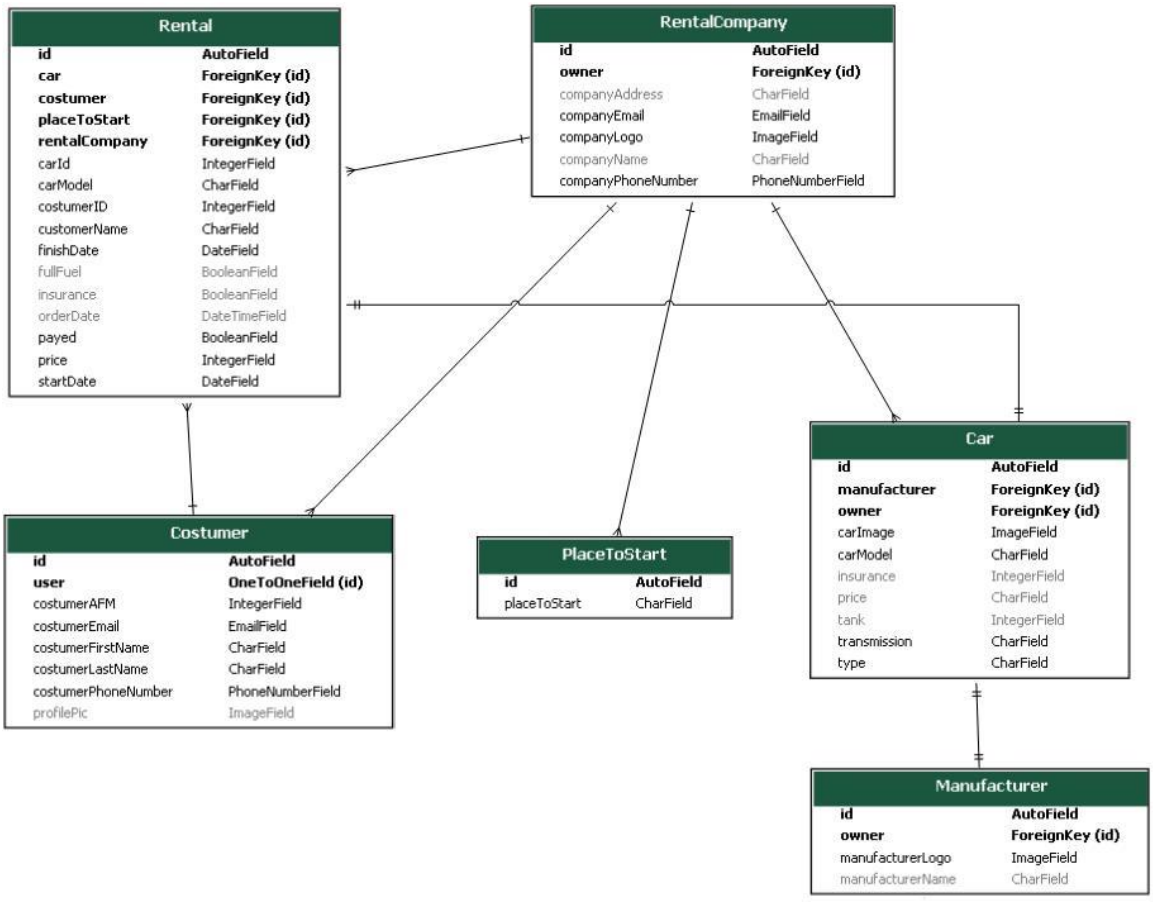
είναι ένα ελεύθερο λογισμικό ανοιχτού κώδικα που βασίζεται στην γλώσσα προγραμματισμού Python και χρησιμοποιείται για την κατασκευή web εφαρμογών. Ακολουθεί το αρχιτεκτονικό μοντέλο (MTV) model-template-views[21] , την υποστήριξη της Django έχει αναλάβει το Django Software Foundation (DSF)[22] το οποίο είναι ένας ανεξάρτητος μη κερδοσκοπικός οργανισμός που έχει έδρα στις ΗΠΑ. Ο κύριος σκοπός της Django είναι να απλουστεύσει την πολυπλοκότητα στην υλοποίηση web εφαρμογών οι οποίες βασίζονται σε βάσεις δεδομένων.

Τέλος για την συγγραφή του κώδικα χρησιμοποιήθηκε κατάλληλο IDE[23] το πολύ δημοφιλές και διαδεδομένο Pycharm[24].

[20] https://en.wikipedia.org/wiki/Django_(web_framework)
[21] https://python.plainenglish.io/the-mvt-design-pattern-of-django-8fd47c61f582#117f
[22] https://en.wikipedia.org/wiki/Django_Software_Foundation
[23] https://en.wikipedia.org/wiki/Integrated_development_environment
[24] https://www.jetbrains.com/pycharm/

2.2 Βάση δεδομένων

Η βάση δεδομένων σχεδιάστηκε με βάση το σχεσιακό μοντέλο[25]. Οι σχέσεις μεταξύ των οντοτήτων της βάσης αποτυπώνονται στο παρακάτω Διάγραμμα Οντοτήτων – Συσχετίσεων[26].



Εικόνα 1 Entity Relationship Diagram – Διάγραμμα Οντοτήτων-Συσχετίσεων

Για τις ανάγκες του έργου έχουν δημιουργηθεί έξι πίνακες. Ο βασικός πίνακας είναι ο «RentalCompany» ο οποίος ουσιαστικά περιλαμβάνει τα στοιχεία της εταιρίας ενοικιάσεων. Αποτελείται από εφτά πεδία, το «id» το οποίο παράγει με αυτόματο τρόπο έναν αύξον ακέραιο αριθμό και έχει την θέση του πρωτεύοντος κλειδιού, ο «owner» το πεδίο αυτό είναι ξένο κλειδί από τον πίνακα user ο οποίος υπάρχει σε όλα τα μοντέλα τέτοιου είδους.

[25] https://el.wikipedia.org/wiki/%CE%A3%CF%87%CE%B5%CF%83%CE%B9%CE%B1%CE%BA%CE%AE_%CE%B2%CE%AC%CF%83%CE%B7_%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD

[26] https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CE%BD%CF%84%CE%AD%CE%BB%CE%BF_%CE%9F%CE%BD%CF%84%CE%BF%CF%84%CE%AE%CF%84%CF%89%CE%BD-%CE%A3%CF%85%CF%83%CF%87%CE%B5%CF%84%CE%AF%CF%83%CE%B5%CF%89%CE%BD

Στην συνέχεια τα υπόλοιπα πεδία αποτελούν τα στοιχεία της εταιρίας, «companyName» για το όνομα της εταιρίας, «companyAddress» για την διεύθυνση της εταιρίας, «companyEmail» για το email της εταιρίας, «companyPhoneNumber» για τον αριθμό τηλεφώνου της εταιρίας και τέλος το «companyLogo» όπου αποθηκεύεται το λογότυπο της εταιρίας.

Η επιλογή αυτής της ιεραρχίας στο μοντέλο έχει γίνει με γνώμονα την δυνατότητα επέκτασης της εφαρμογής στο μέλλον. Το μοντέλο αυτό παρέχει την δυνατότητα στον διαχειριστή του συστήματος να δημιουργήσει ξεχωριστές βάσεις για κάθε εταιρία ενοικίασης και απλά να αλλάζει το template για κάθε εταιρία ανάλογα και με τις ανάγκες της. Όλο το υπόλοιπο όμως στο backend μπορεί να παραμείνει το ίδιο. Αυτό δίνει πολλές δυνατότητες μελλοντικής επέκτασης της εφαρμογής.

Όλοι οι υπόλοιποι πίνακες ουσιαστικά βρίσκονται πιο κάτω σε ιεραρχία σε σχέση με τον πίνακα «RentalCompany». Οι τρείς βασικοί πίνακες στο επόμενο επίπεδο είναι οι «Rental», «Car» και «Costumer», υπάρχει επίσης και ο πίνακας «PlaceToStart». Οι πίνακες αυτοί περιέχουν και το μεγαλύτερο μέρος της πληροφορίας της βάσης, τον μεγαλύτερο όγκο δεδομένων. Η πληθικότητα εδώ σε σχέση με τον πίνακα «RentalCompany» είναι ένα προς πολλά, μια εταιρία «RentalCompany» μπορεί να έχει πολλές ενοικιάσεις «Rental», μπορεί να έχει πολλά αυτοκίνητα «Car», μπορεί να έχει πολλούς πελάτες «Costumer» και μπορεί να έχει πολλά σημεία εκκίνησης «PlaceToStart». Επίσης υπάρχει 1-1 σχέση μεταξύ των πινάκων «Rental» και «Car» με την λογική κάθε ενοικίαση αφορά (έχει) ένα αυτοκίνητο «Car».

Ο πίνακας «Rental» αποτελείται από δεκαέξι πεδία. Εδώ και πάλι το πρωτεύων κλειδί είναι το «id» ο οποίο παράγει με αυτόματο τρόπο έναν αύξον ακέραιο αριθμό. Τα πεδία «car», «costumer», «placeToStart» και «rentalCompany» αποτελούν ξένα κλειδιά για την σύνδεση με τους υπόλοιπους πίνακες του μοντέλου. Τα υπόλοιπα έντεκα πεδία αποτελούν και τα δεδομένα για την ενοικίαση. Τα «carId» και «carModel» το αναγνωριστικό κλειδί και το όνομα του μοντέλου για το όχημα που έχει γίνει ενοικίαση. Τα «costumerId» και «costumerName» το αναγνωριστικό κλειδί και το όνομα του πελάτη. Τα «startDate» και «finishDate» όπου αποθηκεύονται οι ημερομηνίες αναχώρησης της ενοικίασης και επιστροφής αντίστοιχα. Τα «fullFuel», «insurance», «payed» είναι τρεις Boolean μεταβλητές που ουσιαστικά ενημερώνουν την βάση αν τα τρία αυτά πεδία είναι ενεργά ή όχι στην συγκεκριμένη ενοικίαση. Αν ο πελάτης επιθυμεί να ξεκινήσει με γεμάτο ρεζερβουάρ «fullFuel» υπάρχει αντίστοιχη χρέωση για αυτή την παροχή και η τιμή διαμορφώνεται

ανάλογα με το αυτοκίνητο. Αν ο πελάτης επιθυμεί να έχει μικτή ασφάλεια «insurance» εδώ και πάλι υπάρχει χρέωση και είναι ανάλογη με την επιλογή του οχήματος. Το πεδίο «payed» μας ενημερώνει για τον αν η ενοικίαση έχει πληρωθεί ή όχι. Τέλος υπάρχει το πεδίο «price» όπου εκεί αποθηκεύεται η συνολική τιμή της ενοικίασης. Η τιμή διαμορφώνεται ανάλογα με το πλήθος των ημερών και ανάλογα για το αν ο πελάτης επιθυμεί να ξεκινήσει με γεμάτο ρεζερβουάρ ή αν επιθυμεί να έχει επιπλέον μικτή ασφάλεια.

Ο πίνακας «Car» είναι ο δεύτερος πιο βασικός πίνακας του μοντέλου μας. Εδώ βρίσκουμε επίσης και σύνδεση με πληθικότητα 1-1 με τον πίνακα του επόμενου επιπέδου «Manufacturer» κατασκευαστής αυτοκινήτου. Στον πίνακα «Car» αποθηκεύονται τα αυτοκίνητα που έχει η εταιρία «RentalCompany» προς ενοικίαση. Ο πίνακας αποτελείται από δέκα πεδία. Εδώ και πάλι το «id» παίρνει την θέση του πρωτεύοντος κλειδιού και αποτελείται από έναν αύξον ακέραιο αριθμό. Τα πεδία «manufacturer», «owner» είναι ξένα κλειδιά για την σύνδεση του πίνακα «Car» με τους πίνακες «Manufacturer», «User». Το πεδίο «carModel» που χρησιμοποιείται για το μοντέλο του αυτοκίνητου. Το πεδίο «carImage» που αποθηκεύεται η φωτογραφία του αυτοκινήτου. Το πεδίο «insurance» όπου αποθηκεύεται η τιμή ανά ημέρα για την επιλογή της επιπλέον παροχής μικτής ασφάλειας. Το πεδίο «price» όπου αποθηκεύεται η τιμή ενοικίασης ανά ημέρα. Το πεδίο «tank» όπου αποθηκεύεται η τιμή για την χρέωση της παροχής γεμάτο ρεζερβουάρ. Το πεδίο «transmission» όπου εδεώ υπάρχουν δυο επιλογές «Αυτόματο» ή «Μηχανικό» κιβώτιο το οποίο μας ενημερώνει για το αν το αυτοκίνητο είναι με αυτόματο κιβώτιο ταχυτήτων ή με μηχανικό. Τέλος το πεδίο «type» όπου αποθηκεύεται ο τύπος του οχήματος, μεταξύ των τύπων («Mini», «Medium», «Mid-range», «Sedan», «SUV»).

Ο πίνακας «Costumer» όπου αποθηκεύονται τα στοιχεία του πελάτη. Ο πίνακας «Costumer» συνδέεται με δύο πίνακες της βάσης «RentalCompany» και «Rental». Αποτελείται από οχτώ πεδία. Το πρωτεύον κλειδί και πάλι είναι το «id» και είναι ένας αύξον ακέραιος αριθμός. Υπάρχει σύνδεση και με έναν τρίτο πίνακα, τον πίνακα «User» με τον οποίο υπάρχει σχέση πληθικότητας 1-1. Το πεδίο «user» για την σύνδεση με τον πίνακα «User» και αποθηκεύει το id του user που γίνεται costumer. Το πεδίο «costumerFirstName» που αποθηκεύεται το όνομα του πελάτη και το πεδίο «costumerLastName» για το επίθετο του πελάτη. Τα πεδία «costumerAFM», «costumerEmail» και «costumerPhoneNumber» όπου αποθηκεύονται το ΑΦΜ, το email και ο αριθμός τηλεφώνου του πελάτη αντίστοιχα. Τέλος το πεδίο «profilePic» όπου αποθηκεύεται η φωτογραφία του προφίλ του πελάτη.

Ο πίνακας «PlaceToStart» που χρησιμοποιείται για το σημείο εκκίνησης. Αποτελείται από δυο πεδία. Το «id» που είναι το πρωτεύον κλειδί και το πεδίο «placeToStart» όπου υπάρχουν τρεις επιλογές, «Home» για παραλαβή από κάποιο κατάστημα της εταιρίας, «KTEL» για παραλαβή από το ΚΤΕΛ, «Airport» για παραλαβή από το αεροδρόμιο.

Τέλος ο πίνακας «Manufacturer» όπου αποθηκεύονται τα στοιχεία του κατασκευαστή του αυτοκινήτου. Υπάρχει σχέση πληθικότητας 1-1 με τον πίνακα «Car». Ο πίνακας «Manufacturer» αποτελείται από τέσσερα πεδία. Το «id» ως πρωτεύον κλειδί. Το «owner» ως ξένο κλειδί με τον user. Το πεδίο «ManufacturerName» όπου αποθηκεύεται το όνομα του κατασκευαστή και το πεδίο «manufacturerLogo» όπου αποθηκεύεται το λογότυπο του κατασκευαστή.

ΚΕΦΑΛΑΙΟ 3 – Υλοποίηση

3.1 Μοντέλα (Βάση δεδομένων)

Η βάση σχεδιάστηκε με βάση το σχεσιακό μοντέλο. Το web framework που χρησιμοποιήθηκε η Django ακολουθεί το αρχιτεκτονικό μοντέλο (MTV) model-template-views. Το αρχείο που περιέχει τον κώδικα των μοντέλων της βάσεις ονομάζεται "**models.py**" ο κώδικας είναι εξ' ολοκλήρου γραμμένος στης γλώσσα προγραμματισμού Python όπως και ολόκληρο το backend υλοποιήθηκε με την χρήση της Python. Οι κλάσεις που δημιουργήθηκαν είναι η "**RentalCompany**" και χρησιμοποιείται για να αποθηκεύσει τα στοιχεία της εταιρίας ενοικιάσεων, ο "**Costumer**" χρησιμοποιείται για την αποθήκευση των δεδομένων που αφορούν τον πελάτη, η κλάση "**Car**" δημιουργήθηκε για να αποθηκεύει τα δεδομένα για τα οχήματα, η κλάση "**PlaceToStart**" που αποθηκεύει τις επιλογές που έχει ο πελάτης για τον τόπο από τον οποίο μπορεί να παραλάβει το όχημα του, η κλάση "**Extras**" όπου αποθηκεύονται οι επιλογές με επιπλέον χρέωση για το προς ενοικίαση όχημα. Οι κλάσεις "Rental", "Costumer", "PlaceToStart", "Car" έχουν σχέση πολλά προς ένα με την κλάση "RentalCompany". Οι κλάση "Costumer" έχει σχέση ένα προς πολλά με την κλάση "Rental", επίσης η κλάση "Car" έχει σχέση ένα προς ένα με την κλάση ""Rental". Τέλος η κλάση "Manufacturer" έχει σχέση έναν και μόνο έναν με την κλάση "Car".

Ο κώδικας για το αρχείο "model.py" είναι ο παρακάτω :

```
from django.db import models
from django.contrib.auth.models import User
from phonenumber_field.modelfields import PhoneNumberField,
PhoneNumber

TYPE_CHOICES = [('MINI', 'mini'), ('Medium', 'medium'), ('MID_RANGE',
'mid-range'), ('SEDAN', 'sedan'), ('SUV', 'suv')]
TRANSMISSION_CHOICES = [('MANUAL', 'manual'), ('AUTOMATIC',
'automatic')]
PLACE_CHOICES = [('Home', 'home'), ('Airport', 'airport'), ('KTEL',
'ktel')]


class RentalCompany (models.Model):
    companyName = models.CharField(max_length=100, blank=True,
default='')
    companyAddress = models.CharField(max_length=100, blank=True,
default='')
    companyPhoneNumber = PhoneNumberField(null=False, blank=False,
unique=True)
    companyEmail = models.EmailField(max_length=254, null=False,
default='')
```

```
    owner = models.ForeignKey('auth.User', related_name='carRental',
on_delete=models.CASCADE, null=True)
    costumer = models.ManyToManyField('Costumer')
    car = models.ManyToManyField('Car')
    placeToStart = models.ManyToManyField('PlaceToStart')
    companyLogo = models.ImageField(null=True)

    def __str__(self):
        return self.companyName


class Costumer(models.Model):
    user = models.OneToOneField(User, null=True,
on_delete=models.CASCADE)
    costumerFirstName = models.CharField(max_length=80, null=False,
default='')
    costumerLastName = models.CharField(max_length=80, null=False,
default='')
    costumerEmail = models.EmailField(max_length=254, null=False,
default='')
    costumerPhoneNumber = PhoneNumberField(null=False, blank=False,
unique=True)
    costumerAFM = models.IntegerField(null=False, default='')
    profilePic = models.ImageField(default="avatar-
g90d8d966e_640.png", null=True, blank=True)

    def __str__(self):
        return self.costumerFirstName


class Car (models.Model):
    carModel = models.CharField(max_length=100, blank=False,
default='')
    manufacturer = models.ForeignKey('Manufacturer', null=True,
on_delete=models.CASCADE)
    type = models.CharField(choices=TYPE_CHOICES, default='',
max_length=15)
    transmission = models.CharField(choices=TRANSMISSION_CHOICES,
default='', max_length=15)
    carImage = models.ImageField(upload_to='images', null=True)
    owner = models.ForeignKey('auth.User', related_name='car',
on_delete=models.CASCADE, null=True)
    price = models.CharField(max_length=10, null=True, blank=True)
    insurance = models.IntegerField(null=True, blank=True)
    tank = models.IntegerField(null=True, blank=True)

    def __str__(self):
        return self.carModel


class PlaceToStart(models.Model):
    placeToStart = models.CharField(max_length=15,
choices=PLACE_CHOICES)

    def __str__(self):
        return self.placeToStart


class Manufacturer(models.Model):
    manufacturerName = models.CharField(max_length=100, blank=True,
default='')
```

```
    manufacturerLogo = models.ImageField(null=True)
    owner = models.ForeignKey('auth.User',
related_name='manufacturer', on_delete=models.CASCADE, null=True)

    def __str__(self):
        return self.manufacturerName


class Rental(models.Model):
    costumer = models.ForeignKey(Costumer, null=True,
on_delete=models.CASCADE)
    customerName = models.CharField(max_length=70, null=True)
    costumerID = models.IntegerField(null=True, blank=False)
    rentalCompany = models.ForeignKey(RentalCompany, null=True,
on_delete=models.CASCADE, default='1')
    car = models.ForeignKey(Car, null=True, on_delete=models.CASCADE)
    carModel = models.CharField(max_length=70, null=True)
    carId = models.IntegerField(null=True, blank=False)
    price = models.IntegerField(null=True, blank=False)
    startDate = models.DateField(auto_now_add=False, null=True)
    finishDate = models.DateField(auto_now_add=False, null=True)
    orderDate = models.DateTimeField(auto_now_add=True, null=True)
    placeToStart = models.ForeignKey(PlaceToStart, null=True,
on_delete=models.CASCADE)
    fullFuel = models.BooleanField(blank=True, null=True,
default=False)
    insurance = models.BooleanField(blank=True, null=True,
default=False)
    payed = models.BooleanField(null=True, default=False)

    def __str__(self):
        return str(self.id)


class Extras(models.Model):
    name = models.CharField(max_length=105, null=False)
    insurance = models.IntegerField(null=True, blank=True)
    fuel = models.IntegerField(null=True, blank=True)

    def __str__(self):
        return self.name

class CanceledOrders(models.Model):
    costumerID = models.CharField(max_length=70, null=True)
    carId = models.CharField(max_length=70, null=True)
    price = models.IntegerField(null=True, blank=False)
    payed = models.BooleanField(null=True, default=False)


class CarToDelete (models.Model):
    carId = models.CharField(max_length=70, null=True)
```

3.2 Admin View

Το web framework Django δημιουργεί ένα διαχειριστικό περιβάλλον για την ευκολότερη διαχείριση της βάσης δεδομένων. Το Django administration (admin view) βοηθάει στον διαχειριστή της εφαρμογής να διαχειριστεί με εύκολο και απλό τρόπο την βάση δεδομένων. Υπάρχει η δυνατότητα προσθήκης, ενημέρωσης και διαγραφής ενός στοιχείου της βάσης. Για τις ανάγκες της εφαρμογής έχουν δημιουργηθεί 3 χρήστες ο καθένας με διαφορετικά δικαιώματα. Ο superuser που είναι και ο administrator του συστήματος ο οποίος έχει δικαιώματα σούπερ χρήστη και ουσιαστικά είναι ο διαχειριστής του συστήματος.

Χρήστης Administrator/ Admin / Superuser / Διαχειριστής

Username: admin

Password: admin

Ο χρήστης υπάλληλος – εταιρία, ο χρήστης αυτός έχει δικαίωμα να εισάγει, ενημερώνει και να διαγράφει αυτοκίνητα. Μπορεί να διαγράψει έναν πελάτη, να δημιουργήσει μια ενοικίαση να την ενημερώσει και να την διαγράψει.

Χρήστης Staff / Υπάλληλος

Username: cbar

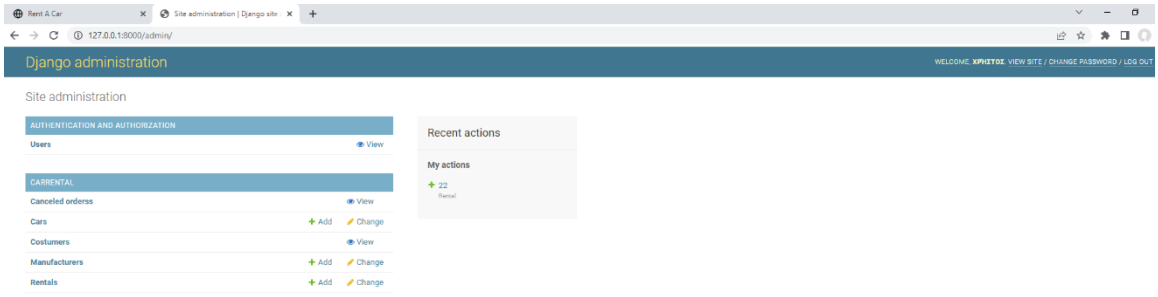
Password: aris1407

Ο χρήστης πελάτης, έχει δικαίωμα ενός απλού χρήστη της εφαρμογής. Ουσιαστικά του επιτρέπονται μόνο δικαιώματα που το δίνει το γραφικό περιβάλλον της εφαρμογής. Για παράδειγμα μπορεί να πραγματοποιήσει μια ενοικίαση, να δει τις ενοικιάσεις του, να δει και να ενημερώσει τα στοιχεία του.

Χρήστης Customer / Χρήστης / Πελάτης/

Username: miselser

Password: vivi2805



Εικόνα 2 Administration View Site – Διαχειριστικό περιβάλλον βάσης δεδομένων

3.3 Ερωτήματα στην βάση «views»

Για την επικοινωνία με την βάση η Django προτείνει την χρήση των views[27]. Ουσιαστικά πρόκειται για στιγμιότυπα views που μπορεί να δημιουργήσει ο χρήστης προς την βάση το οποία στην συνέχεια του δίνουν την δυνατότητα απεικόνισης των αποτελεσμάτων που επιστρέφουν. Ένα view είναι μια ρουτίνα γραμμένη σε Python η οποία δέχεται http requests[28] και επιστρέφει http responses [29].

Ο κώδικας εμπεριέχει επαρκή σχολιασμό ως προς την επεξήγηση των λειτουργιών που δημιουργεί. Στην συνέχεια και στο επόμενο κεφάλαιο όπου θα γίνει παρουσίαση του template της εφαρμογής θα υπάρξει και παράλληλα σύνδεση των αντίστοιχων HTML template με τα αντίστοιχα views. Η υλοποίηση βρίσκεται στο αρχείο «**views.py**»

Οι λειτουργίες που υλοποιεί ο κώδικας που περιέχουν τα «views» περιγράφονται στο επόμενο κεφάλαιο.

[27] https://www.w3schools.com/django/django_views.php
[28] https://sematext.com/glossary/http-requests/
[29] https://www.ibm.com/docs/en/cics-ts/5.2?topic=protocol-http-responses

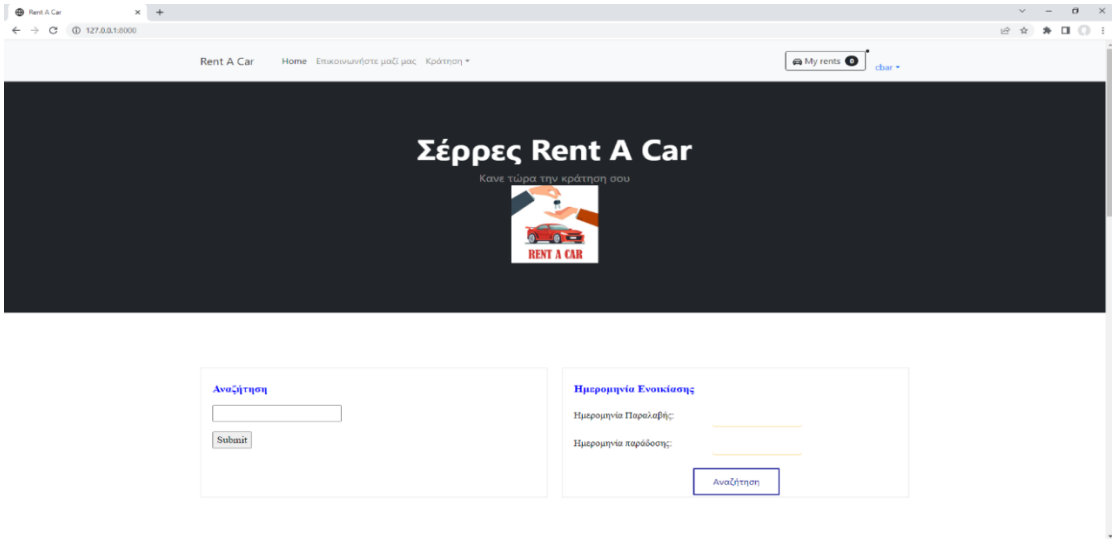
3.4 Παρουσίαση Template και δυνατοτήτων την εφαρμογής

Το template είναι ένα αρχείο κειμένου στο οποίο ο προγραμματιστής περιγράφει πως τα αποτελέσματα θα πρέπει να παρουσιάζονται. Της περισσότερες φορές είναι HTML[30] αρχεία τα οποία περιγράφουν την διάταξη της σελίδας, επίσης τα αρχεία αυτά μπορεί να είναι γραμμένα και σε άλλες μορφές π.χ. XML, CSV κ.α. Στην συγκεκριμένη εφαρμογή έχει γίνει χρήση κατά βάση της HTML σε συνδυασμό με JavaScript για πιο δυναμικό περιεχόμενο και CSS για την μορφοποίηση των σελίδων.

Στην συνέχεια θα γίνει μια αναλυτική παρουσίαση των δυνατοτήτων της εφαρμογής με παράλληλη παρουσίαση του κώδικα των template και των αντίστοιχων view και ανάλυση του περιεχομένου με περιγραφή των λειτουργιών της εκάστοτε σελίδας.

3.4.1 Αρχική σελίδα – Homepage

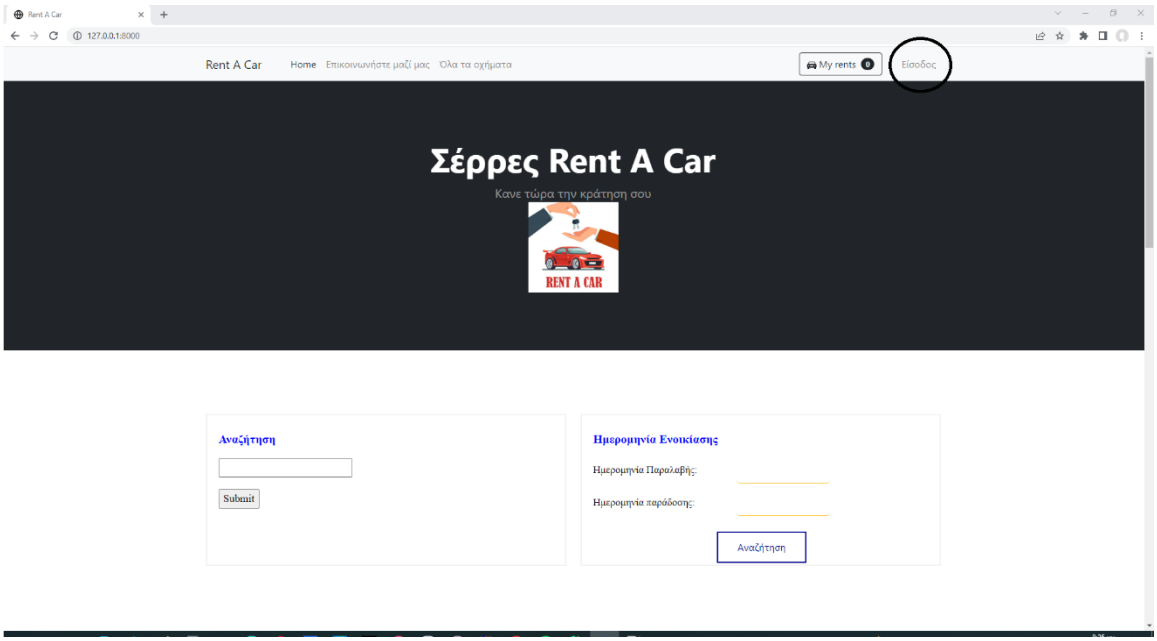
Η αρχική σελίδα της εφαρμογής είναι χωρισμένη σε 3 μέρη, την κεφαλίδα «header» το κυρίως μέρος «body» και το υποσέλιδο «footer» και αποτελείται από 5 αρχεία HTML, main.html, footer.html, navbar.html, mainheader.html, home.html. Ο λόγος που έχουν δημιουργηθεί 5 αρχεία είναι καθαρά για λόγους διαχείρισης του κώδικα γιατί η διάταξη με την οποία είναι στημένη η αρχική σελίδα είναι η βασική διάταξη για όλες τις σελίδες της εφαρμογής οπότε τα αρχεία main.html, footer.html, navbar.html και mainheader.html επαναλαμβάνονται σε ολόκληρη την εφαρμογή.



Εικόνα 3 Η αρχική σελίδα της εφαρμογής – Μπάρα πλοήγησης (Navbar)

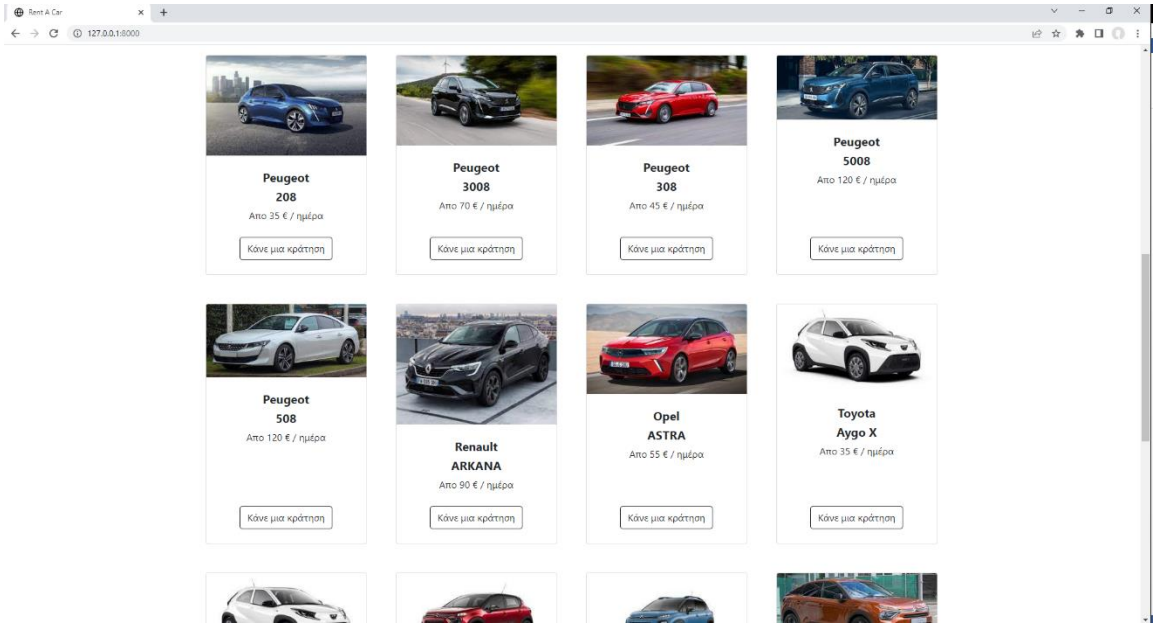
[30] https://el.wikipedia.org/wiki/HTML

Στην *Εικόνα 3 Η αρχική σελίδα της εφαρμογής* εμφανίζονται η μπάρα πλοήγησης της εφαρμογής η οποία έχει τον τίτλο της σελίδας «**Rent A Car**» που είναι και σύνδεσμος για την αρχική σελίδα. Υπάρχει η επιλογή «**Home**» η οποία είναι ανακατεύθυνση για την αρχική σελίδα, η επιλογή «**Επικοινωνήστε μαζί μας**» η οποία οδηγεί στο υποσέλιδο όπου υπάρχουν όλα τα στοιχεία επικοινωνίας, η επιλογή «**όλα τα οχήματα**» όπου μας μεταφέρει στην σελίδα που εμφανίζονται όλα τα οχήματα τα οποία περιέχονται στην βάση. Στην δεξιά γωνία εμφανίζεται αναγνωρίζοντας το *username – όνομα χρήστη* ο χρήστης που έχει συνδεθεί στην εφαρμογή. Αν δεν έχει συνδεθεί ακόμα κάποιος χρήστης τότε η επιλογή είναι η «**Είσοδος**» *Εικόνα 4 Η αρχική σελίδα της εφαρμογής* η οποία μας μεταφέρει στην σελίδα όπου ο χρήστης μπορεί να εισάγει σε ειδικά διαμορφωμένη φόρμα τα στοιχεία σύνδεσης του.



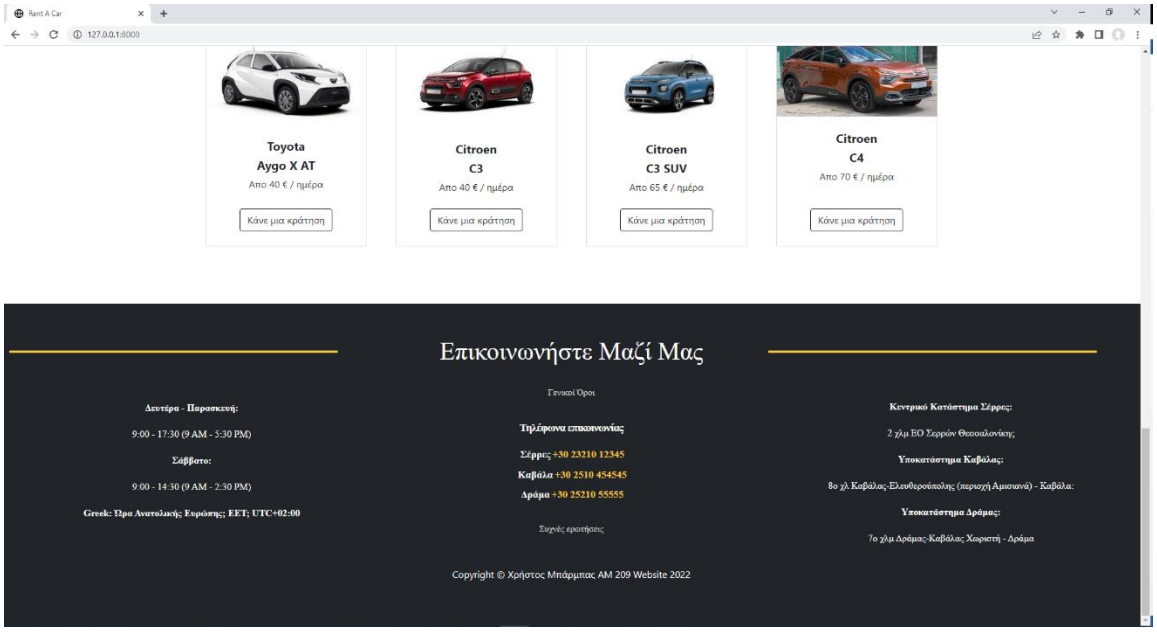
Εικόνα 4 Η αρχική σελίδα της εφαρμογής – Είσοδος στην εφαρμογή

Στο κυρίως σώμα της σελίδας *Εικόνα 5 Η αρχική σελίδα της εφαρμογής* βλέπουμε την παρουσίαση τον οχημάτων. Υπάρχουν τέσσερα οχήματα σε κάθε σειρά όπου το κάθε πλαίσιο περιέχει την φωτογραφία του οχήματος, το όνομα της μάρκας – κατασκευαστής, το μοντέλο του αυτοκινήτου την ελάχιστη τιμή ανά ημέρα και ένα κουμπί που μας προτρέπει να κάνουμε κράτηση το όχημα το οποίο επιλέγοντάς το μεταφερόμαστε στην σελίδα που υπάρχουν οι λεπτομέρειες του οχήματος.



Εικόνα 5 Η αρχική σελίδα της εφαρμογής

Τέλος στο υποσέλιδο *Εικόνα 6 Η αρχική σελίδα της εφαρμογής* παρουσιάζονται τα στοιχεία επικοινωνίας

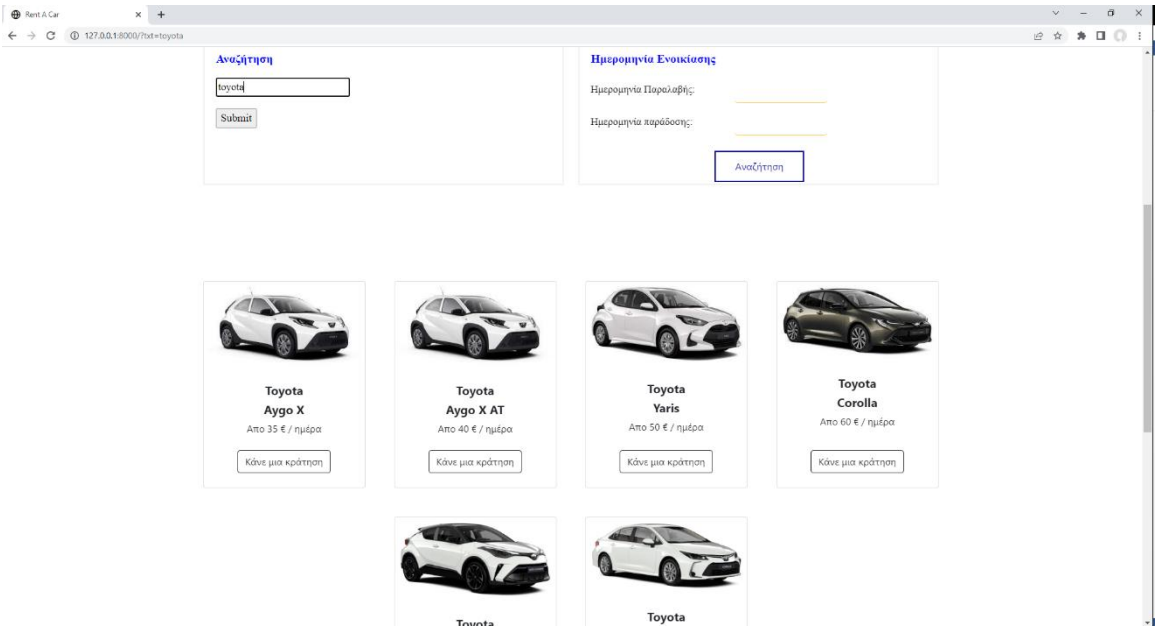


Εικόνα 6 Η αρχική σελίδα της εφαρμογής- Υποσέλιδο

Το template που περιέχει τον κώδικα για το κυρίως κορμό της αρχικής σελίδας είναι το αρχείο «**home.html**».

Το view που περιέχει τον κώδικα που παράγει τα αποτελέσματα για την αρχική σελίδα είναι το «**home**» και βρίσκεται στο αρχείο «**views.py**»

Τρία είναι τα κύρια σημεία αναφοράς σε αυτό το view. Αρχικά η ρουτίνα που αντιστοιχεί στη φόρμα αναζήτησης. Αν ο χρήστης δεν προσθέσει κείμενο στο πεδίο αναζήτησης τότε η εφαρμογή επιστρέφει σαν αποτέλεσμα δώδεκα οχήματα με αλφαβητική σειρά με βάση το όνομα του μοντέλου. Επίσης ο χρήστης μπορεί να βάλει στο πεδίο αναζήτησης οποιαδήποτε λέξη θέλει και η αναζήτηση στην βάση γίνεται με βάση το μοντέλο του οχήματος τον τύπο του οχήματος, τον τύπο του κιβωτίου και το όνομα του κατασκευαστή, η εφαρμογή επιστρέφει τα αντίστοιχα αποτελέσματα. *Εικόνα 7 Αναζήτηση*



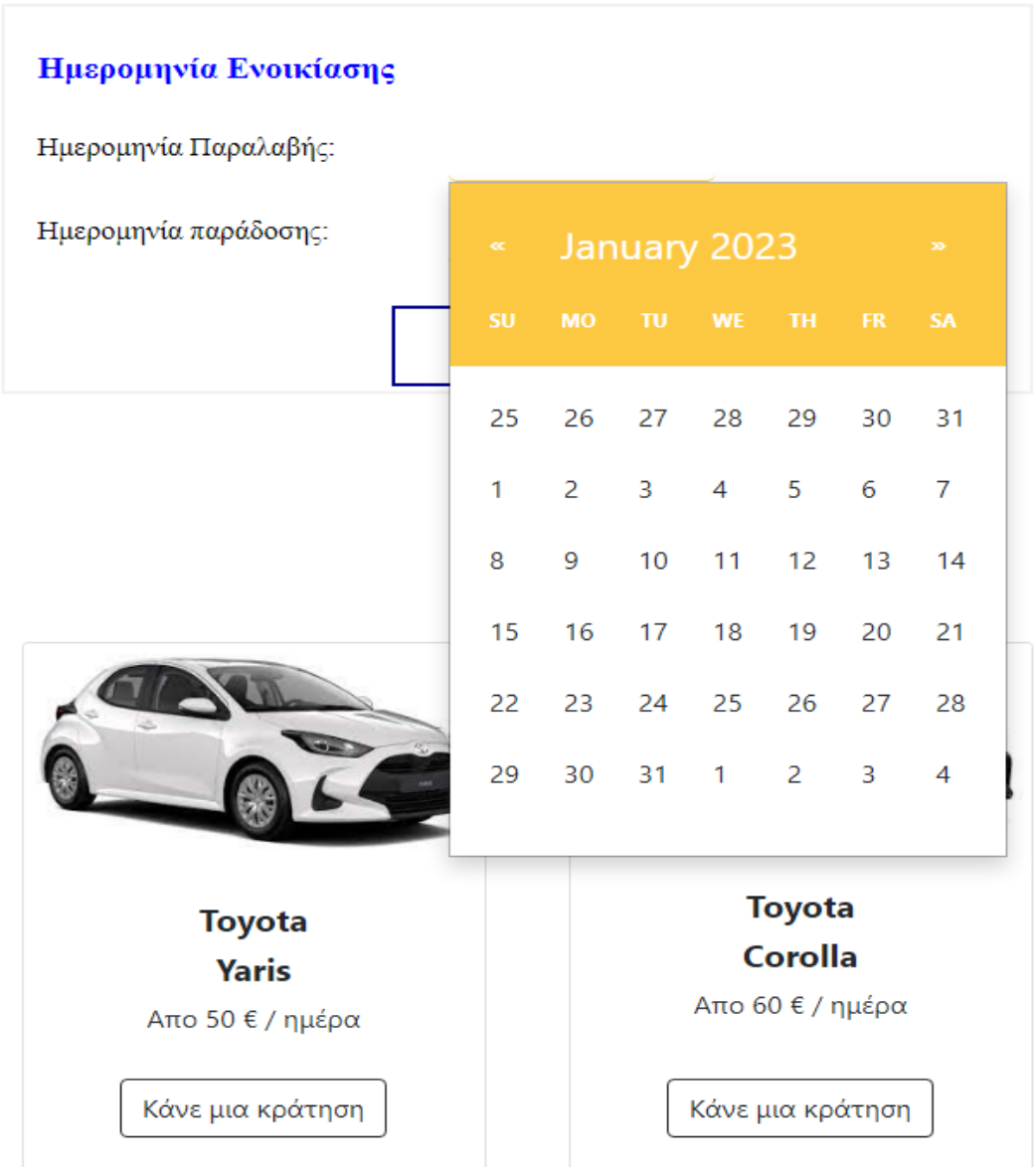
Εικόνα 7 Αναζήτηση

Η πιο σημαντική λειτουργία που υπάρχει στην αρχική σελίδα είναι η επιλογή για αναζήτηση με βάση την ημερομηνία. Εδώ υπάρχουν δύο κύρια σημεία. Το πρώτο είναι στο επίπεδο της παρουσίασης fronted και είναι τα δύο πεδία που έχει δυνατότητα ο χρήστης να επιλέξει την ημερομηνία παραλαβής του οχήματος και την ημερομηνία παράδοσης. Για την λειτουργία του συγκεκριμένων πεδίων έχει δημιουργηθεί ο παρακάτω κώδικας σε HTML και JavaScript.

Αρχείο home.html (datepicker)[31]

[31] Βλέπε παράστημα σελ. 62 -63

Με την χρήση του datepicker[32] η εφαρμογή δημιουργεί έναν πίνακα ημερολόγιο όπου ο χρήστης μπορεί με εύκολο τρόπο να επιλέξει την ημερομηνία παραλαβής και την ημερομηνία παράδοσης. *Εικόνα 8 Αναζήτηση με βάση ημερομηνία - Διαθεσιμότητα*



Εικόνα 8 Αναζήτηση με βάση ημερομηνία - Διαθεσιμότητα

Στο backend της εφαρμογής ο παρακάτω κώδικας είναι υπεύθυνος στο να επιστρέψει αποτελέσματα. **Έλεγχος διαθεσιμότητας – Availability check**[33]

Ο κώδικας παίρνει τις δυο ημερομηνίες που δίνει ο χρήστης και τις συγκρίνει με τις ημερομηνίες που είναι ήδη αποθηκευμένες στην βάση για την κάθε ενοικίαση. Υπάρχουν 4

[32] https://bootstrap-datepicker.readthedocs.io/en/latest/

[33] Βλέπε παράστημα σελ. 63

συγκρίσεις ουσιαστικά για να αποκλειστούν όλα τα ενδεχόμενα, τα αποτελέσματα που θα πρέπει να μας επιστρέψει η εφαρμογή θα πρέπει να είναι μόνο τα ελεύθερα προς ενοικίαση οχήματα. Στην συνέχεια η εφαρμογή επιστρέφει μια λίστα με τα διαθέσιμα οχήματα προς ενοικίαση.

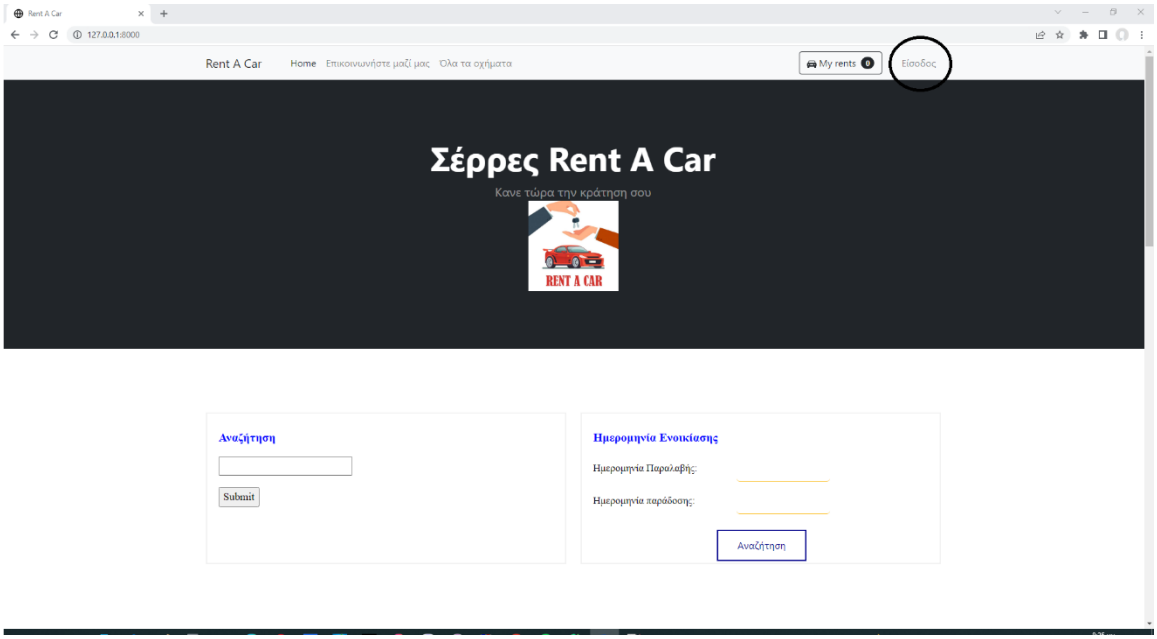
3.4.2 Όλα τα οχήματα – Car List

Η σελίδα αυτή εμφανίζει όλα τα οχήματα που υπάρχουν στην βάση. Η ανακατεύθυνση σε αυτή την σελίδα γίνεται από την μπάρα πλοήγησης και την επιλογή «**Όλα τα οχήματα**». Ο κώδικας παρουσίασης της σελίδας όπως επίσης και το view υλοποιούνται στα αρχεία «**carlist.html**» και **view carlist** αρχείο **«views.py»**

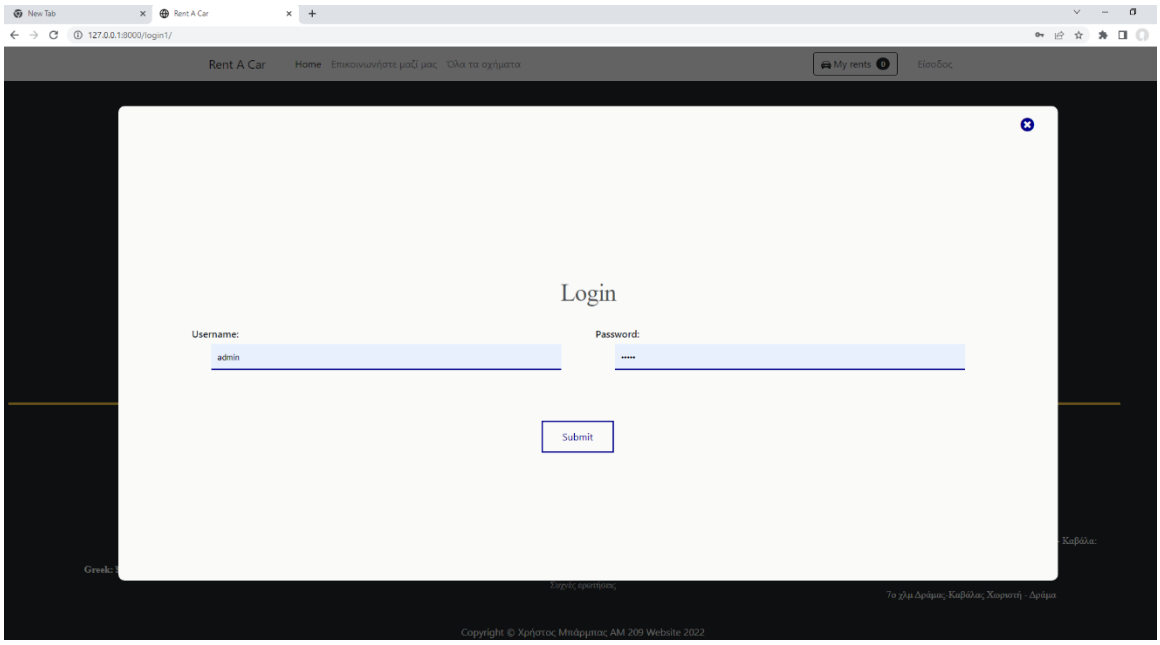
Ο κώδικας είναι σχεδόν ίδιος με την αρχική σελίδα. Οι επιλογές για αναζήτηση είναι οι ίδιες με την αρχική, δεν χρειάζεται να γίνει παραπάνω ανάλυση διότι έχει γίνει παραπάνω στην ανάλυση αυτών των δυνατοτήτων στο κεφάλαιο *3.4.1 Αρχική σελίδα – Homepage* . Ο βασικός λόγος που υπάρχει αυτή η σελίδα είναι για την παρουσίαση όλων των οχημάτων που έχουν καταχωρηθεί στην βάση σε μία και μόνο σελίδα.

3.4.3 Είσοδος – Login

Για την είσοδο στην εφαρμογή υπάρχει στην μπάρα πλοήγησης – «**navbar**» η επιλογή «**Είσοδος**» όπου επιλέγοντας την ο χρήστης της εφαρμογής ανοίγει σε ξεχωριστό παράθυρο η φόρμα εισόδου όπου μπορεί να συμπληρώσει τα στοιχεία εισόδου του «**credentials**». Βλέπουμε στην *Εικόνα 9 Είσοδος στην εφαρμογή* και στην *Εικόνα 10 Φόρμα εισόδου – Login form* τις επιλογές του χρήστη.



Εικόνα 9 Είσοδος στην εφαρμογή



Εικόνα 10 Φόρμα εισόδου – Login form

Ο κώδικας για την λειτουργία «**Είσοδος**» περιλαμβάνει το αντίστοιχο view με όνομα «**user_login1**» και βρίσκεται στο αρχείο «**views.py**», την φόρμα «**LoginFrom**» που βρίσκεται στο αρχείο «**forms.py**» και το αρχείο «**login1.html**».

view user_login1 κωδικας[34]

[34] Βλέπε παράρτημα σελ. 64

3.4.4 Είσοδος χρήστη με διαφορετικά δικαιώματα

Για τις ανάγκες της εφαρμογής έχουν δημιουργηθεί 3 χρήστες ο καθένας με διαφορετικά δικαιώματα. Ο superuser που είναι και ο administrator του συστήματος ο οποίος έχει δικαιώματα σούπερ χρήστη και ουσιαστικά είναι ο διαχειριστής του συστήματος.

Χρήστης Administrator/ Admin / Superuser / Διαχειριστής

Username: admin

Password: admin

Ο χρήστης υπάλληλος – εταιρία, ο χρήστης αυτός έχει δικαίωμα να εισάγει, ενημερώνει και να διαγράφει αυτοκίνητα. Μπορεί να διαγράψει έναν πελάτη, να δημιουργήσει μια ενοικίαση να την ενημερώσει και να την διαγράψει.

Χρήστης Staff / Υπάλληλος

Username: cbar

Password: aris1407

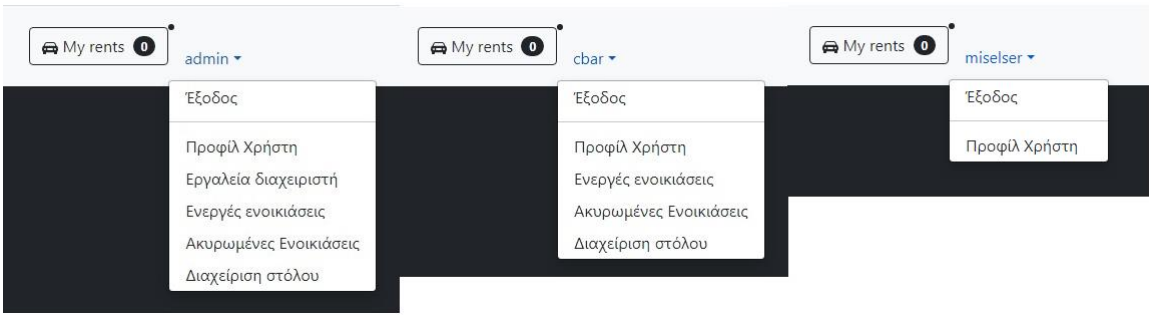
Ο χρήστης πελάτης, έχει δικαίωμα ενός απλού χρήστη της εφαρμογής. Ουσιαστικά του επιτρέπονται μόνο δικαιώματα που του δίνει το γραφικό περιβάλλον της εφαρμογής. Για παράδειγμα μπορεί να πραγματοποιήσει μια ενοικίαση, να δει τις ενοικιάσεις του, να δει και να ενημερώσει τα στοιχεία του.

Χρήστης Customer / Χρήστης / Πελάτης/

Username: miselser

Password: vivi2805

Τον ορισμό των δικαιωμάτων τον έχει αποκλειστικά και μόνο ο διαχειριστής του συστήματος. Η εφαρμογή αναγνωρίζει τον χρήστη που έχει συνδεθεί και πλέον η επιλογή «**είσοδος**» εμφανίζει το username του χρήστη που έχει πραγματοποιήσει είσοδο στην εφαρμογή.



Εικόνα 11 Είσοδος στην εφαρμογή με διαφορετικό χρήστη

Όπως βλέπουμε στην *Εικόνα 11 Είσοδος στην εφαρμογή με διαφορετικό χρήστη* η εφαρμογή ανάλογα με τον χρήστη που έχει κάνει είσοδο δημιουργεί και ένα μενού επιλογών dropdown list όπου ο χρήστης ανάλογα με τα δικαιώματα του, την διαβάθμιση που έχει βλέπει διαφορετικές επιλογές στο μενού επιλογών. Όπως βλέπουμε και στην *Εικόνα 11 Είσοδος στην εφαρμογή με διαφορετικό χρήστη* ο κάθε χρήστης έχει κάποιες κοινές επιλογές όπως για παράδειγμα η επιλογή «**Προφίλ Χρήστη**» με την οποία ο χρήστης μεταφέρεται στην σελίδα όπου μπορεί να δει το προφίλ του και να το επεξεργαστεί. Ο διαχειριστής και ο χρήστης υπάλληλος έχουν κοινές επιλογές τις «**Ενεργές ενοικιάσεις**» που τους μεταφέρει στην σελίδα όπου μπορούν να δουν όλες τις ενεργές ενοικιάσεις. Την επιλογή «**Ακυρωμένες ενοικιάσεις**» όπου μπορούν να δουν όλες τις ακυρωμένες ενοικιάσεις και την «**Διαχείριση στόλου**» όπου μπορούν να δουν όλα τα οχήματα που υπάρχουν στην βάση, να δουν λεπτομέρειες για τα οχήματα, να διαγράψουν ένα όχημα από την βάση και να ενημερώσουν τα στοιχεία του, όπως επίσης μπορούν να εισάγουν ένα νέο όχημα στην βάση. Τέλος ο διαχειριστής έχει την επιλογή «**Εργαλεία διαχειριστή**» όπου επιλέγοντας την μεταφέρεται στην σελίδα όπου υπάρχουν σύνδεσμοι Link για τις σελίδες του rest framework και εργαλεία για τον διαχειριστή του συστήματος.

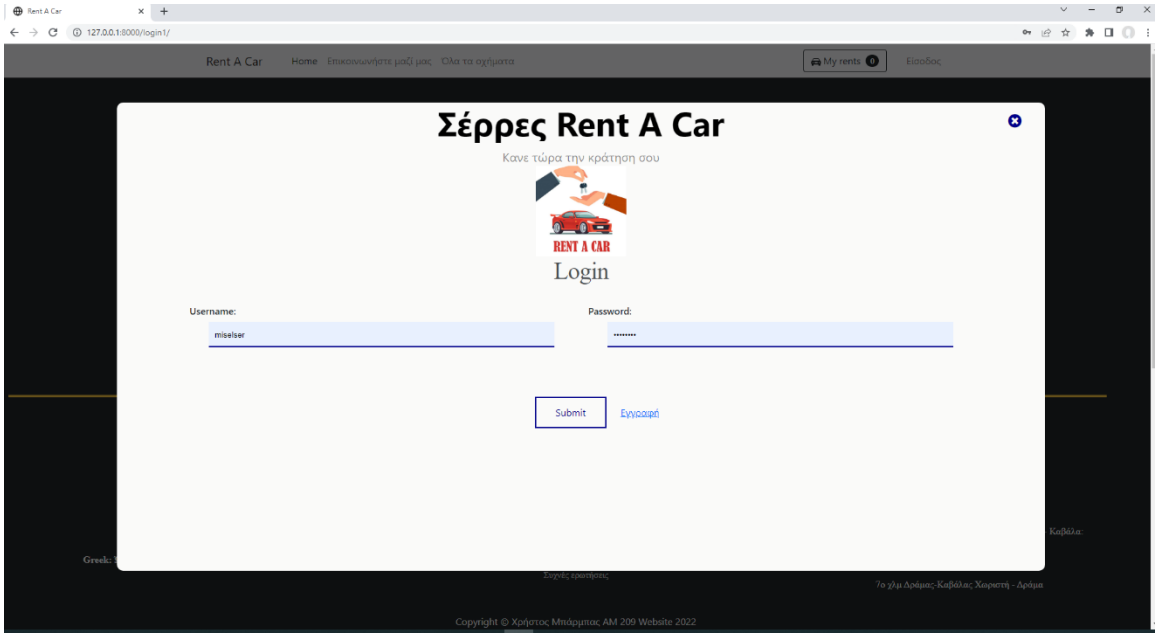
Ο κώδικας για την δημιουργία αυτού του μενού βρίσκεται στο αρχείο navbar.html.

Αρχείο mainheader.html – Μενού εισόδου – Login menu[35]

[35] Βλέπε παράρτημα σελ. 64 - 65

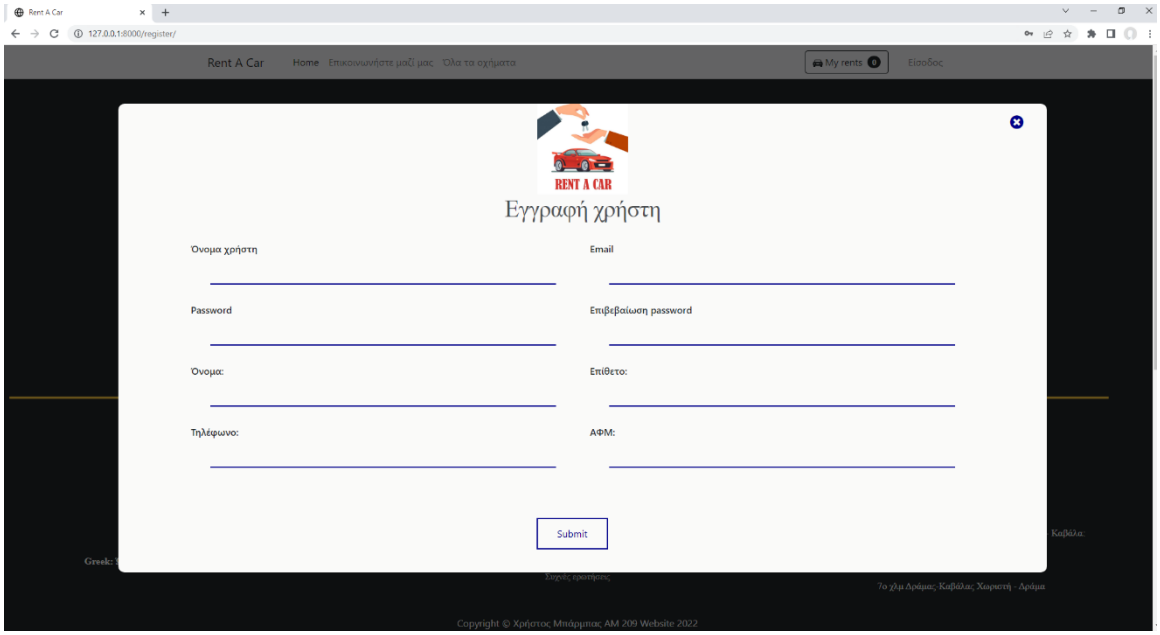
3.4.5 Εγγραφή χρήστη – Registration

Εκτός από την είσοδο – Login του χρήστη στην εφαρμογή αν ο χρήστης δεν έχει πραγματοποιήσει ακόμα εγγραφή στην εφαρμογή και δεν έχει στοιχεία σύνδεσης (username και password) τότε μπορεί να πραγματοποιήσει εγγραφή – registration. Η επιλογή αυτή δίνεται στον χρήστη στην φόρμα εισόδου. *Εικόνα 12 Φόρμα εισόδου – Επιλογή Εγγραφής χρήστη στην εφαρμογή*



Εικόνα 12 Φόρμα εισόδου – Επιλογή Εγγραφής χρήστη στην εφαρμογή

Επιλέγοντας ο χρήστης «**Εγγραφή**» ανοίγει νέα φόρμα, η φόρμα εγγραφής. *Εικόνα 13 Φόρμα εγγραφής* όπου ο χρήστης έχει την δυνατότητα να συμπληρώσει τα στοιχεία του στην φόρμα και να πραγματοποιήσει εγγραφή στο σύστημα.



Εικόνα 13 Φόρμα εγγραφής

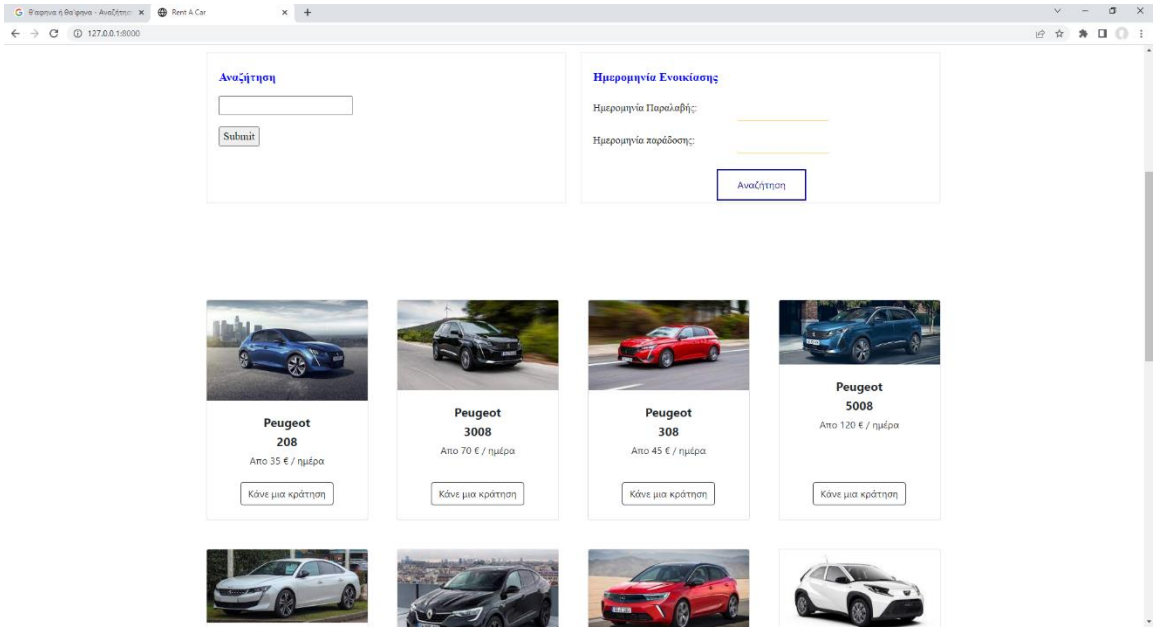
Ο κώδικας για την εγγραφή – registration του χρήστη στην εφαρμογή περιλαμβάνει το αντίστοιχο view «**user_register**» το οποίο βρίσκεται στο αρχείο «**views.py**», την φόρμα «**RegisterForm**» η οποία βρίσκεται στο αρχείο «**forms.py**»και το αρχείο «**register1.html».**

- **View user_register**[36]

3.4.6 Επιλογή οχήματος

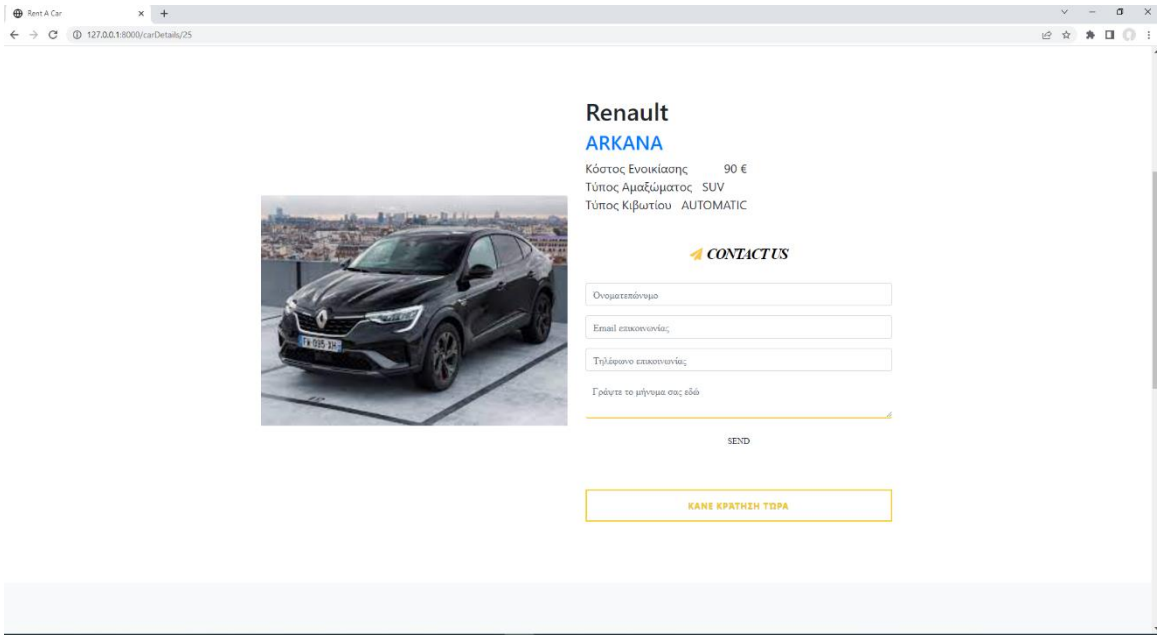
Στην αρχική σελίδα όπως έχουμε ήδη αναφέρει στην ενότητα *3.4.1 Αρχική σελίδα – Homepage* ο χρήστης έχει την δυνατότητα να δει κάποια από τα διαθέσιμα αυτοκίνητα που υπάρχουν, επίσης μπορεί να κάνει αναζήτηση και να επιλέξει τις ημερομηνίες που επιθυμεί να ενοικιάσει το όχημα. Στην *Εικόνα 14 Αρχική σελίδα – Εμφάνιση αποτελεσμάτων* βλέπουμε τις επιλογές που έχει ο χρήστης όπως επίσης και τα αποτελέσματα που εμφανίζονται. Τα αποτελέσματα εμφανίζονται το καθένα σε ένα ορθογώνιο πλαίσιο κάθετα προσανατολισμένο όπου ο χρήστης βλέπει την φωτογραφία του οχήματος την μάρκα και το μοντέλο, την τιμή που έχει το όχημα ανά ημέρα ενοικίασης και την επιλογή «**Κάνε μια κράτηση**» επιλέγοντας την ο χρήστης μεταφέρεται στην σελίδα όπου εμφανίζονται οι λεπτομέρειες του οχήματος.

[36] Βλέπε παράστημα σελ. 65 - 66

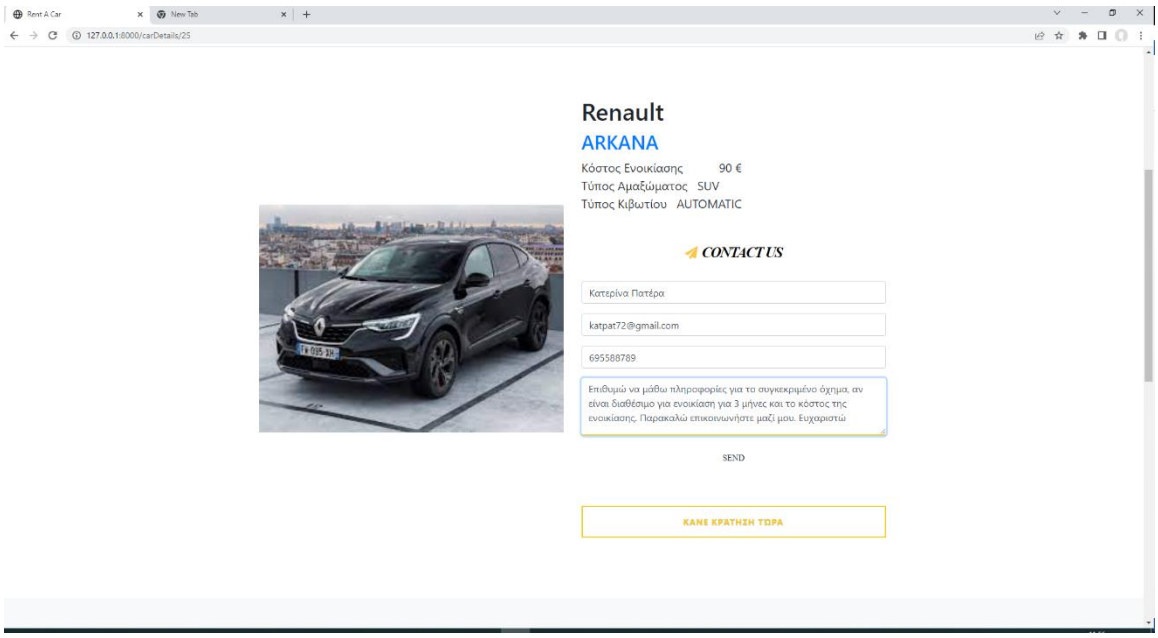


Εικόνα 14 Αρχική σελίδα – Εμφάνιση αποτελεσμάτων

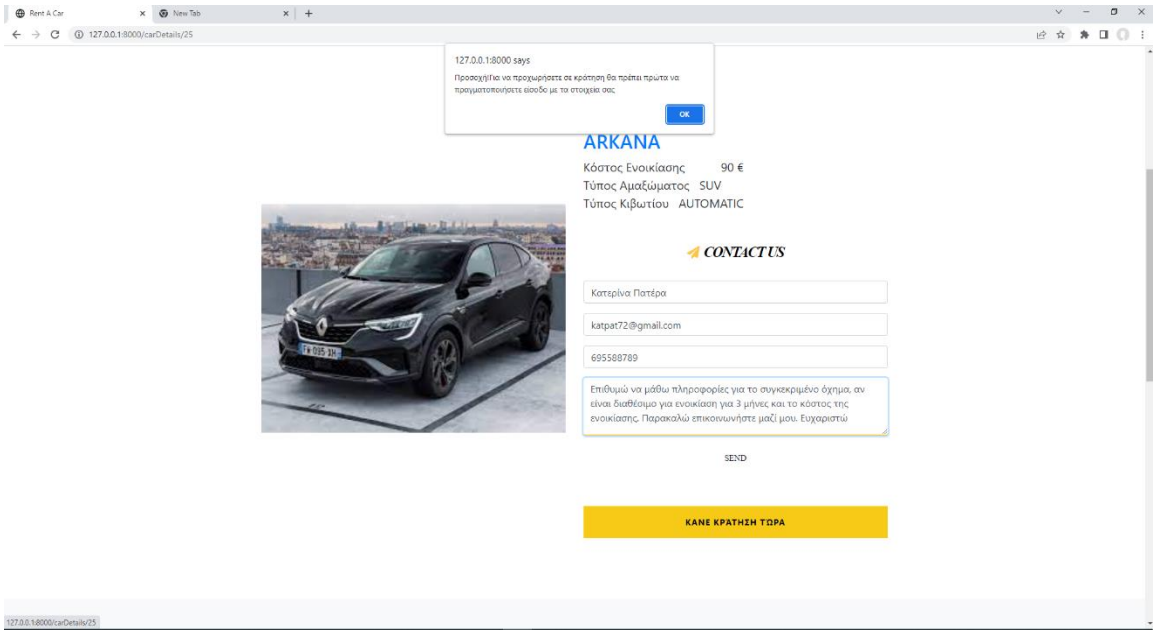
Στην σελίδα που εμφανίζονται οι λεπτομέρειες του οχήματος-car details όπως βλέπουμε και στην *Εικόνα 15 Λεπτομέρειες οχήματος Car details* εμφανίζονται η μάρκα και το μοντέλο του οχήματος, το κόστος ενοικίασης ανά ημέρα, ο τύπος αμαξώματος και ο τύπος του κιβωτίου. Παρακάτω υπάρχει η φόρμα επικοινωνίας , *Εικόνα 16 Λεπτομέρειες οχήματος – φόρμα επικοινωνίας* όπου ο χρήστης μπορεί να στείλει στην εταιρία τα στοιχεία του και το μήνυμα που επιθυμεί. Με την επιλογή «**Κάνε κράτηση τώρα**» ο χρήστης μεταφέρεται στην σελίδα της κράτησης, όμως όπως βλέπουμε στην *Εικόνα 17 Λεπτομέρειες οχήματος – Μήνυμα σφάλμα (Δεν έχει πραγματοποιηθεί σύνδεση χρήστη)*



Εικόνα 15 Λεπτομέρειες οχήματος Car details

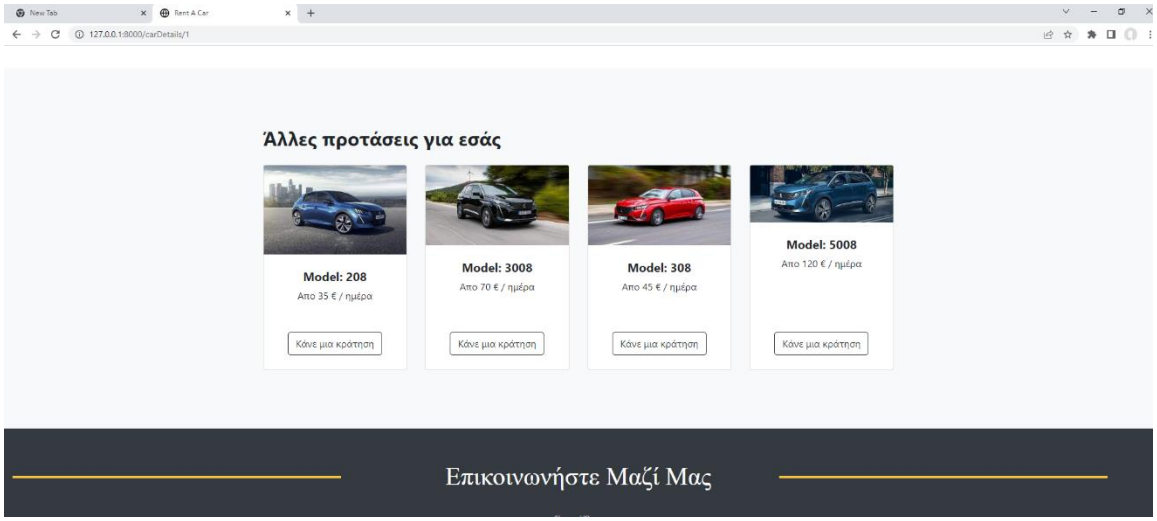


Εικόνα 16 Λεπτομέρειες οχήματος – φόρμα επικοινωνίας



Εικόνα 17 Λεπτομέρειες οχήματος – Μήνυμα σφάλμα (Δεν έχει πραγματοποιηθεί σύνδεση χρήστη)

αν δεν έχει γίνει είσοδος του χρήστη στην εφαρμογή, εμφανίζεται μήνυμα «σφάλμα» όπου προτρέπει τον χρήστη να πραγματοποιήσει πρώτα είσοδο με τα στοιχεία του και στην συνέχεια να προχωρήσει στην κράτηση. Τέλος στο κάτω μέρος της σελίδας υπάρχουν επιπλέον προτάσεις για τον χρήστη. *Εικόνα 18 Λεπτομέρειες οχήματος – Επιπλέον προτάσεις*

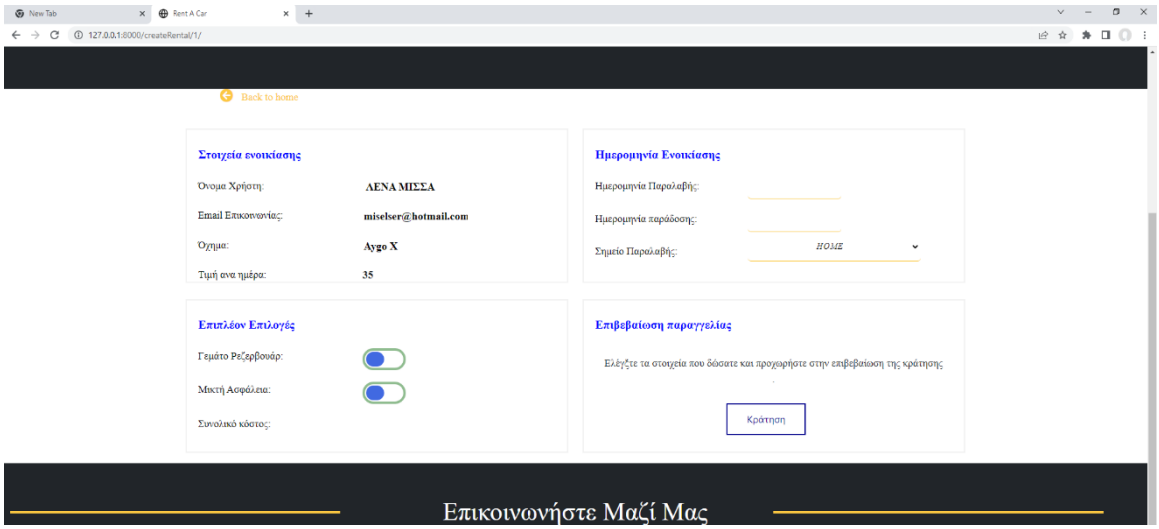


Εικόνα 18 Λεπτομέρειες οχήματος – Επιπλέον προτάσεις

Ο κώδικας για την σελίδα λεπτομέρειες οχήματος περιλαμβάνει το view «**Cardetail**» που βρίσκεται στο αρχείο «**views.py**» και επίσης το αρχείο «**cardetails.html».**

3.4.7 Κράτηση οχήματος – Επιβεβαίωση παραγγελίας

Στην σελίδα της κράτησης του οχήματος ο χρήστης μπορεί να επιβεβαιώσει τα στοιχεία της ενοικίασης, όνομα χρήστη, email επικοινωνίας, το όχημα που έχει επιλέξει και την τιμή ανά ημέρα. *Εικόνα 19 Κράτηση οχήματος – Στοιχεία ενοικίασης*

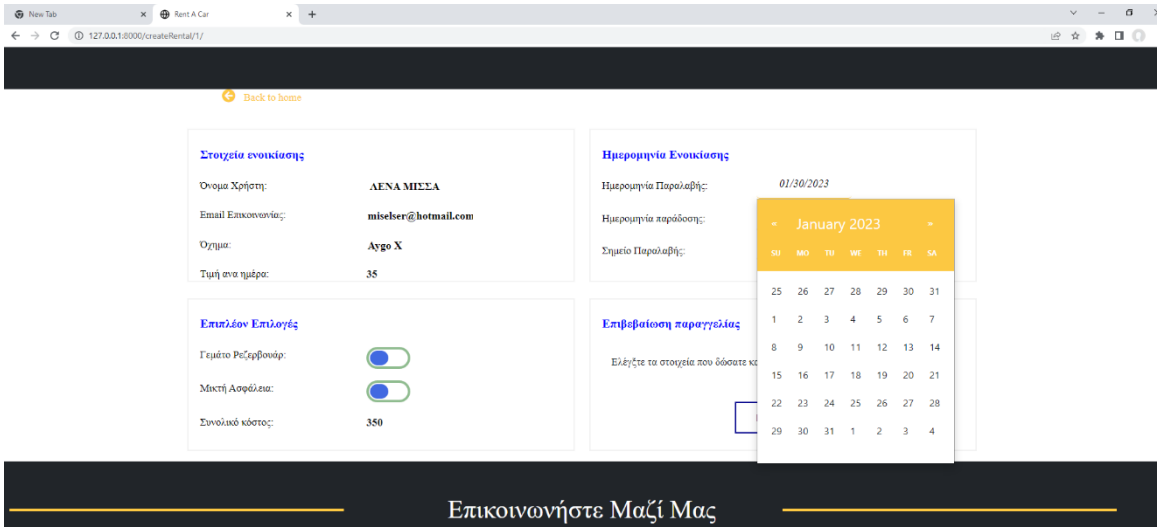


Εικόνα 19 Κράτηση οχήματος – Στοιχεία ενοικίασης

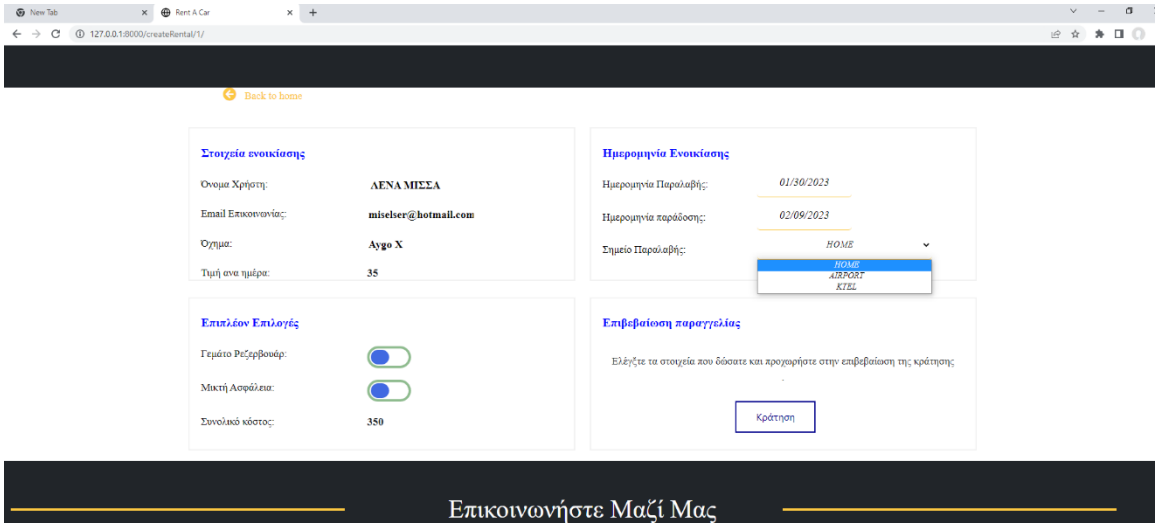
Η σελίδα είναι χωρισμένη σε τέσσερα πλαίσια.

- Στοιχεία ενοικίασης
  Εικόνα 19 Κράτηση οχήματος – Στοιχεία ενοικίασης
- Ημερομηνία Ενοικίασης
  Εικόνα 20 Κράτηση οχήματος – Ημερομηνία Ενοικίασης
  Εικόνα 21 Κράτηση οχήματος – Ημερομηνία Ενοικίασης – Σημείο παραλαβής
- Επιπλέον Επιλογές
  Εικόνα 22 Κράτηση οχήματος – Επιπλέον επιλογές
- Επιβεβαίωση παραγγελίας

Ο χρήστης μπορεί να δει απλά της πληροφορίες στο πλαίσιο «**Στοιχεία ενοικίασης**» στα επόμενα τρία όμως μπορεί να επιλέξει και ανάλογα με την επιλογή του διαμορφώνεται και το «**Συνολικό κόστος**» στο πλαίσιο «**Επιπλέον επιλογές**». Διαλέγοντας τις ημερομηνίες ενοικίασης γίνεται υπολογισμός τις τιμές με βάση την τιμή του αυτοκινήτου ανά ημέρα ενοικίασης και το πλήθος των ημερών που έχει διαλέξει ο χρήστης. Επίσης στο πλαίσιο «**Επιπλέον επιλογές**» δίνεται η δυνατότητα στον χρήστη να επιλέξει να επιθυμεί να ξεκινήσει την ενοικίαση του με γεμάτο ρεζερβουάρ στο καύσιμο, η επιλογή αυτή επιφέρει επιπλέον χρέωση που υπολογίζεται στο συνολικό κόστος. Επίσης ο χρήστης μπορεί να επιλέξει η ενοικίαση του να έχει μικτή ασφάλεια, η επιλογή αυτή επιφέρει επιπλέον χρέωση και υπολογίζεται και πάλι με βάση την τιμή ανά ημέρα για επιπλέον ασφάλεια και το πλήθος των ημερών ενοικίασης. Οι επιλογές του χρήστη διαμορφώνουν και το συνολικό κόστος της ενοικίασης.

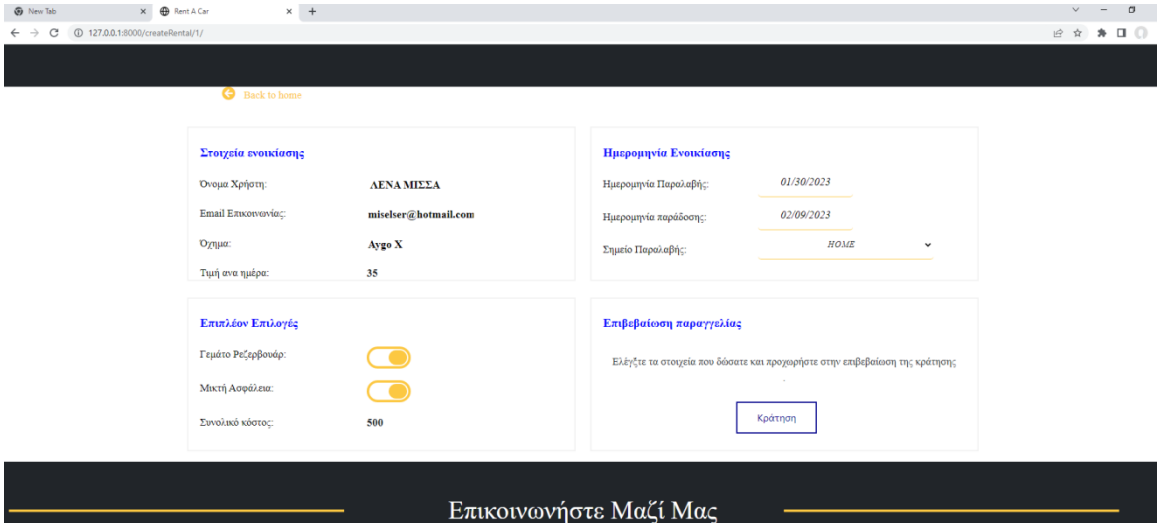


Εικόνα 20 Κράτηση οχήματος – Ημερομηνία Ενοικίασης



Εικόνα 21 Κράτηση οχήματος – Ημερομηνία Ενοικίασης – Σημείο παραλαβής

Στο τελευταίο πλαίσιο «**Επιβεβαίωση παραγγελίας**» ο χρήστης μπορεί να επιλέξει αφού έχει ελέγξει όλα τα πεδία που έχει συμπληρώσει να προχωρήσει στην κράτηση.



Εικόνα 22 Κράτηση οχήματος – Επιπλέον επιλογές

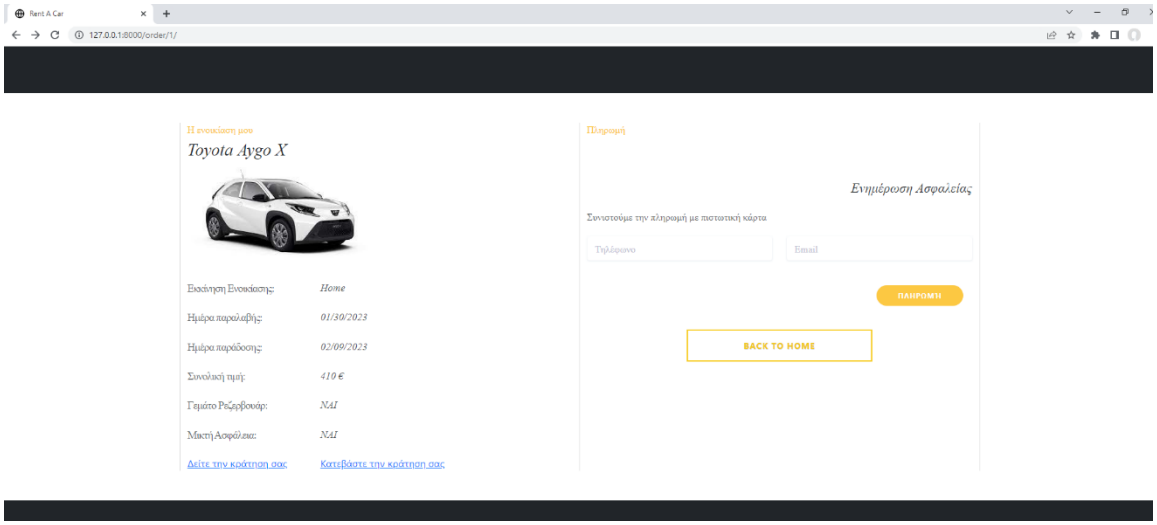
Ο κώδικας για την σελίδα κράτηση οχήματος περιλαμβάνει το view και το αρχείο «**rental.html».**

- **createRental view**[37]
- **Αρχείο rental.html κώδικας**[38]

[37] Βλέπε παράστημα σελ. 66-67
[38] Βλέπε παράρτημα σελ. 67 -72

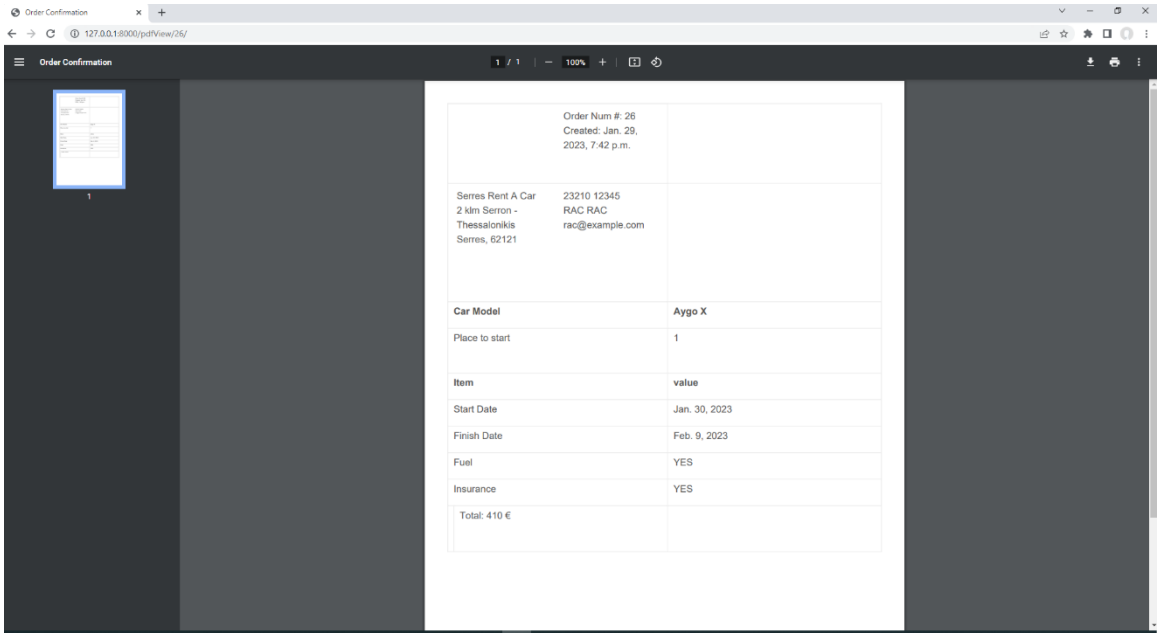
3.4.8 Επιβεβαίωση κράτησης

Στην σελίδα «**Επιβεβαίωση κράτησης**» ο χρήστης έχει την δυνατότητα να δει τα στοιχεία της κράτησης που έχει επιλέξει. Πλέον η κράτηση έχει αποθηκευτεί στην βάση. Εμφανίζονται δύο πλαίσια. Το αριστερό πλαίσιο με τίτλο «**Η ενοικίαση μου**» έχει όλες τις πληροφορίες για την κράτηση που έχει πραγματοποιήσει ο χρήστης. Στο δεξιό πλαίσιο υπάρχει μια φόρμα όπου ο χρήστης μπορεί να συμπληρώσει το τηλέφωνο και το email του, όπως επίσης και η επιλογή «πληρωμή» η οποία δεν έχει υλοποιηθεί καθώς και η επιλογή «back to home» η οποία επιστρέφει στην αρχική σελίδα. *Εικόνα 23 Επιβεβαίωση κράτησης*



Εικόνα 23 Επιβεβαίωση κράτησης

Επίσης ο χρήστης έχει την επιλογή «**Δείτε την κράτηση σας**» όπου επιλέγοντας την παράγεται ένα pdf αρχείο με τα στοιχεία της κράτησης. *Εικόνα 24 Επιβεβαίωση κράτηση – PDF invoice*



Εικόνα 24 Επιβεβαίωση κράτηση – PDF invoice

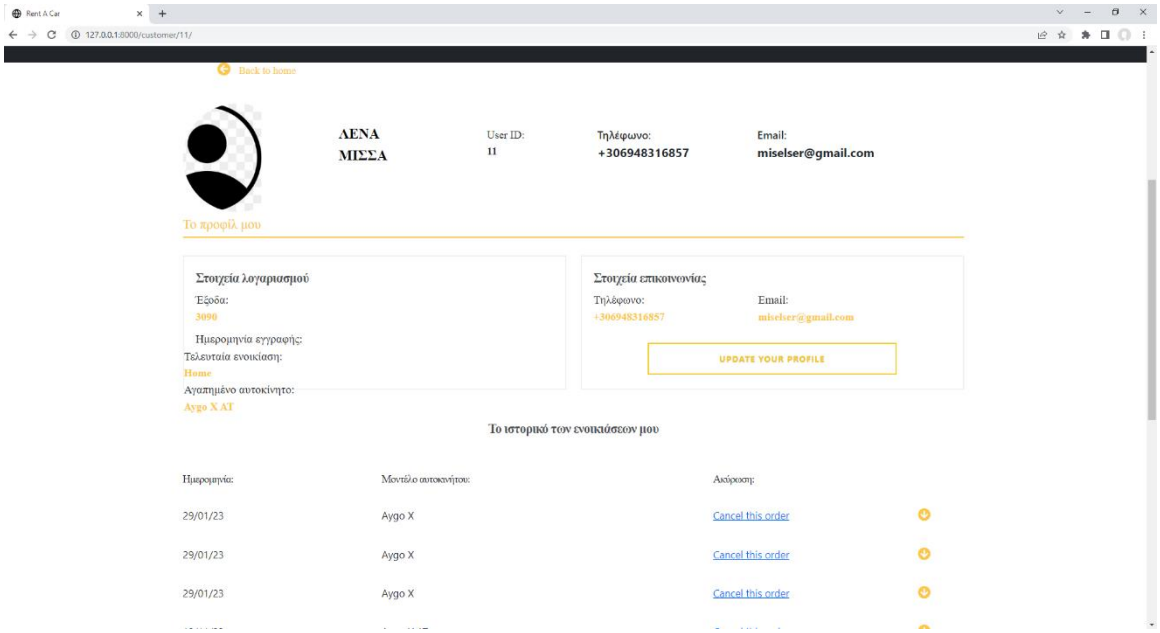
Ο κώδικας για την σελίδα «**Επιβεβαίωση κράτησης**» περιλαμβάνει τα views «**order**» το αρχείο «**order.html»** το αρχείο «**pdfViews.py»** και το αρχείο «**pdfInvoice.html»**.

- **Αρχείο pdfViews.py κώδικας**[39]

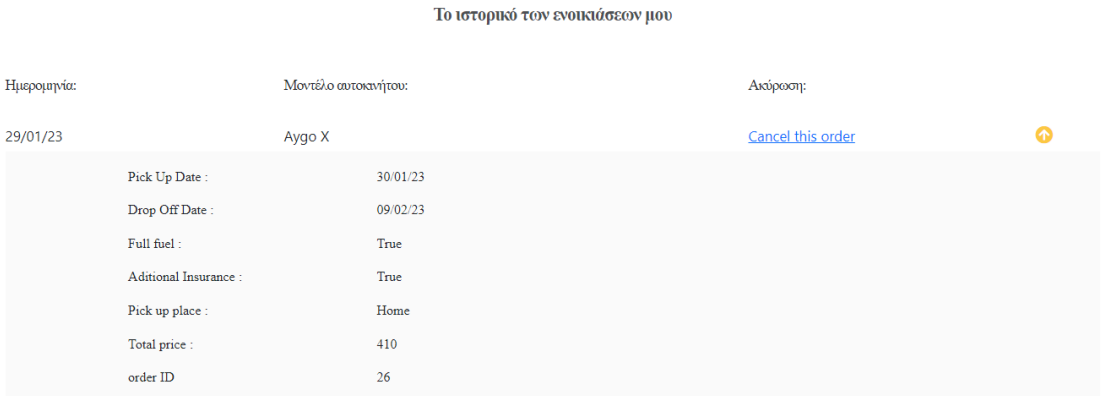
3.4.9 Προφίλ Χρήστη

Όπως ήδη έχουμε αναφέρει και στο κεφάλαιο *3.4.4 Είσοδος χρήστη με διαφορετικά δικαιώματα* οι χρήστες τις εφαρμογής έχουν διαφορετικά δικαιώματα ανάλογα με την διαβάθμιση που τους έχει δώσει ο διαχειριστής. Με την είσοδο του χρήστη η εφαρμογή αναγνωρίζει το όνομα του προφίλ που έχει κάνει είσοδο και η επιλογή «**Είσοδος**» αλλάζει ανάλογα με το προφίλ που συνδέθηκε και στην θέση του εμφανίζεται το όνομα του χρήστη, *Εικόνα 11 Είσοδος στην εφαρμογή με διαφορετικό χρήστη* . Επίσης δημιουργείται και μια λίστα επιλογών η οποία και πάλι είναι διαφορετική ανάλογα με την διαβάθμιση του χρήστη. Μια από τις επιλογές αυτής της λίστας είναι και το «**Προφίλ Χρήστη**» *Εικόνα 25 Προφίλ Χρήστη* το οποίο μας μεταφέρει στην σελίδα του προφίλ όπου ο χρήστης μπορεί να δει τα προσωπικά του στοιχεία, να δεί στοιχεία του λογαριασμού του ,όπως το συνολικό ποσό που έχει δαπανήσει σε ενοικιάσεις, πιο όχημα έχει ενοικιάσει τις περισσότερες φορές «**Αγαπημένο αυτοκίνητο**» , όπως επίσης και ένα ιστορικό με τις ενοικιάσεις του. *Εικόνα 26 Ιστορικό ενοικιάσεων*

[39] Βλέπε παράστημα σελ. 72-73

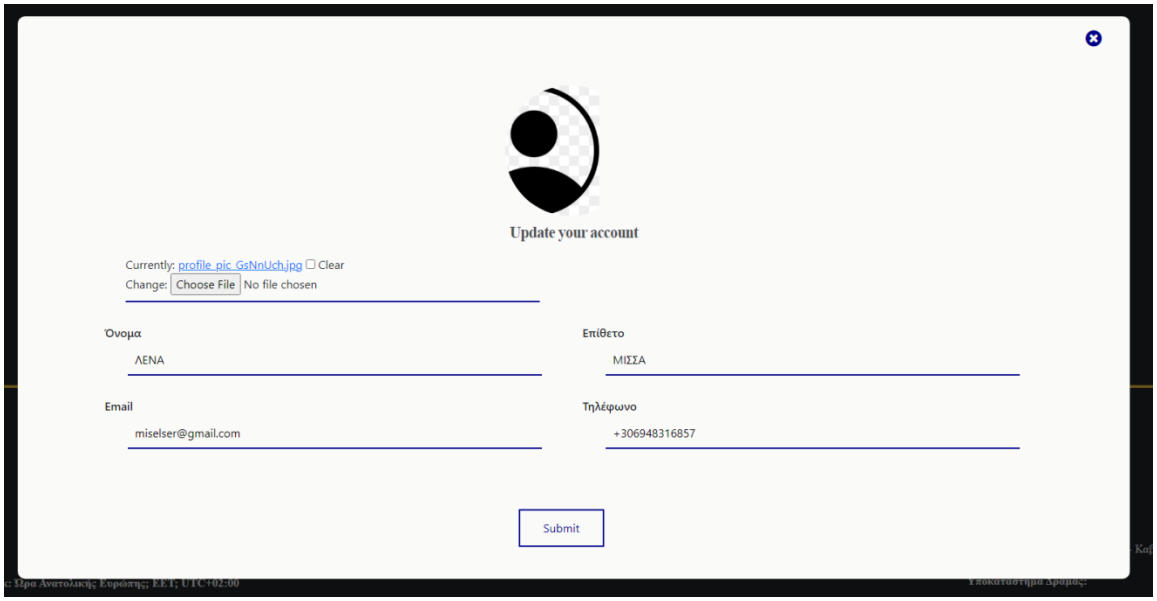


Εικόνα 25 Προφίλ Χρήστη



Εικόνα 26 Ιστορικό ενοικιάσεων

Επίσης ο χρήστης μπορεί να ακυρώσει και την παραγγελία του, επιλογή «**Ακύρωση**». Τέλος ο χρήστης έχει την επιλογή να ενημερώσει το προφίλ του αν τυχών υπάρχει κάποια αλλαγή στα προσωπικά του στοιχεία επιλέγοντας «Ενημέρωση προφίλ» *Εικόνα 27 Ενημέρωση προφιλ*



Εικόνα 27 Ενημέρωση προφιλ

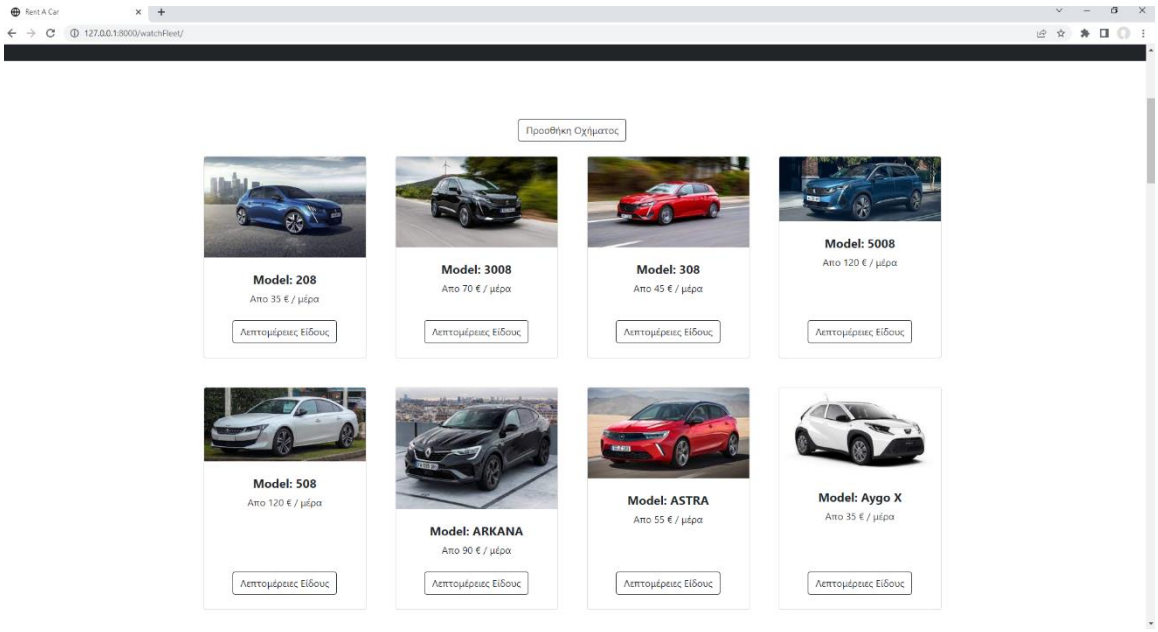
Ο κώδικας για την σελίδα «Προφίλ χρήστη» περιλαμβάνει τα αντίστοιχα view «**CustomerPage**», «**UpdateView**» και τα αρχεία **customer.html**, **toggle.js**, **updateCostumer.html**

- **Αρχείο toggle.js**[40]

3.4.10 Έλεγχος στόλου

Ο χρήστης υπάλληλος – εταιρία έχει διαβάθμιση “staff” στην εφαρμογή. Η διαβάθμιση αυτή του δίνει πρόσβαση σε διαφορετικό μενού στο προφίλ χρήστη. Μια από τις επιλογές είναι ο έλεγχος του στόλου. Στην σελίδα αυτή ο υπάλληλος της εταιρίας μπορεί να δεί όλα τα οχήματα που υπάρχουν καταχωρημένα στην βάση. *Εικόνα 28 Έλεγχος στόλου*

[40] Βλέπε παράστημα σελ. 73

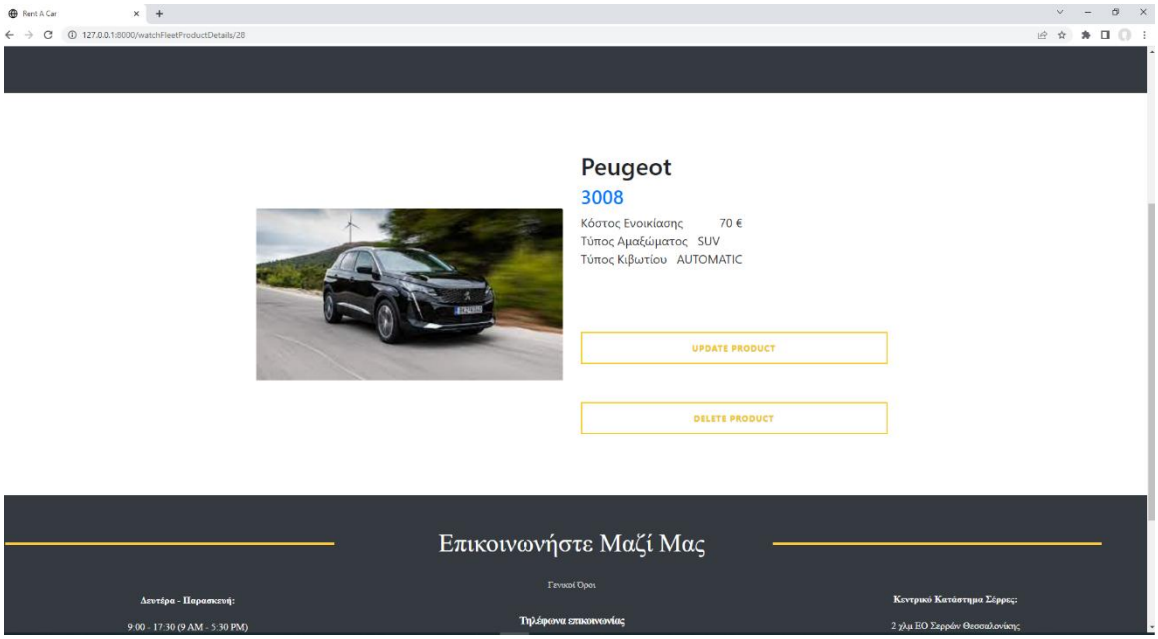


Εικόνα 28 Έλεγχος στόλου

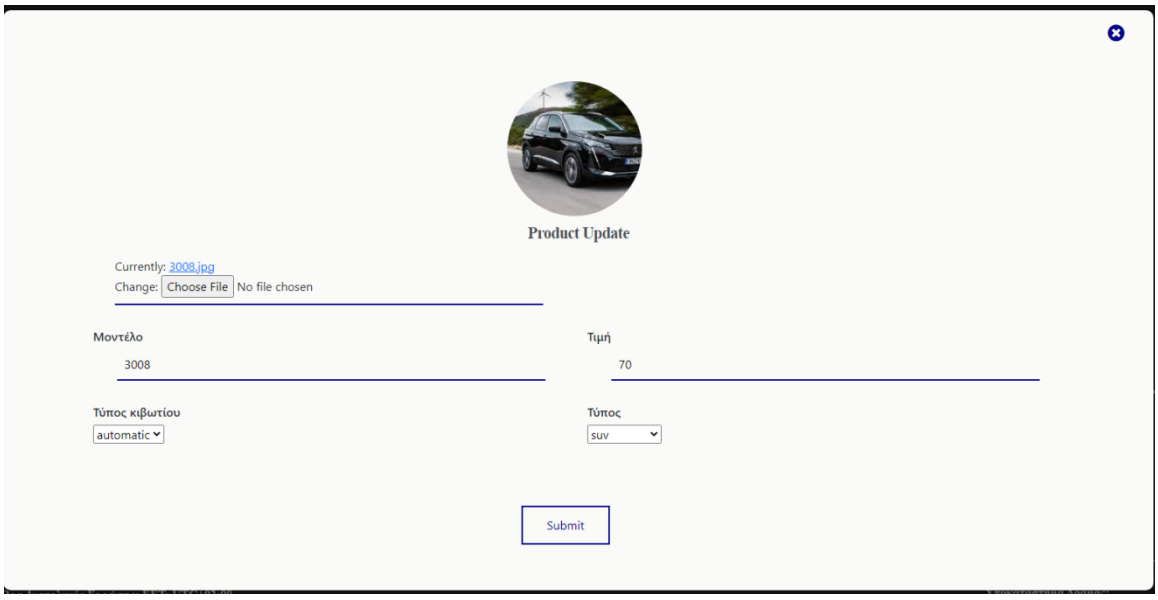
Ο χρήστης έχει την επιλογή «**Λεπτομέρειες είδους**» όπου τον μεταφέρει στην σελίδα με τις λεπτομέρειες του οχήματος, *Εικόνα 29 Λεπτομέρειες οχήματος* όπου μπορεί να δει τα χαρακτηριστικά του οχήματος. Επίσης υπάρχουν δύο επιλογές, η «Ενημέρωση προϊόντος» που επιλέγοντας την εμφανίζεται φόρμα όπου ο χρήστης μπορεί να ενημερώσει τα στοιχεία του οχήματος αν υπάρχουν τυχών αλλαγές. *Εικόνα 30 Ενημέρωση οχήματος* Τέλος υπάρχει η επιλογή «Διαγραφή» η οποία διαγράφει οριστικά το όχημα από την βάση δεδομένων.

Επιστρέφοντας στην σελίδα «**Έλεγχος στόλου**» υπάρχει η επιλογή «**Προσθήκη οχήματος**» η οποία μας μεταφέρει σε μια φόρμα όπου ο υπάλληλος της εταιρίας έχει την δυνατότητα να προσθέσει ακόμα ένα όχημα στην βάση δεδομένων. *Εικόνα 31 Προσθήκη οχήματος*

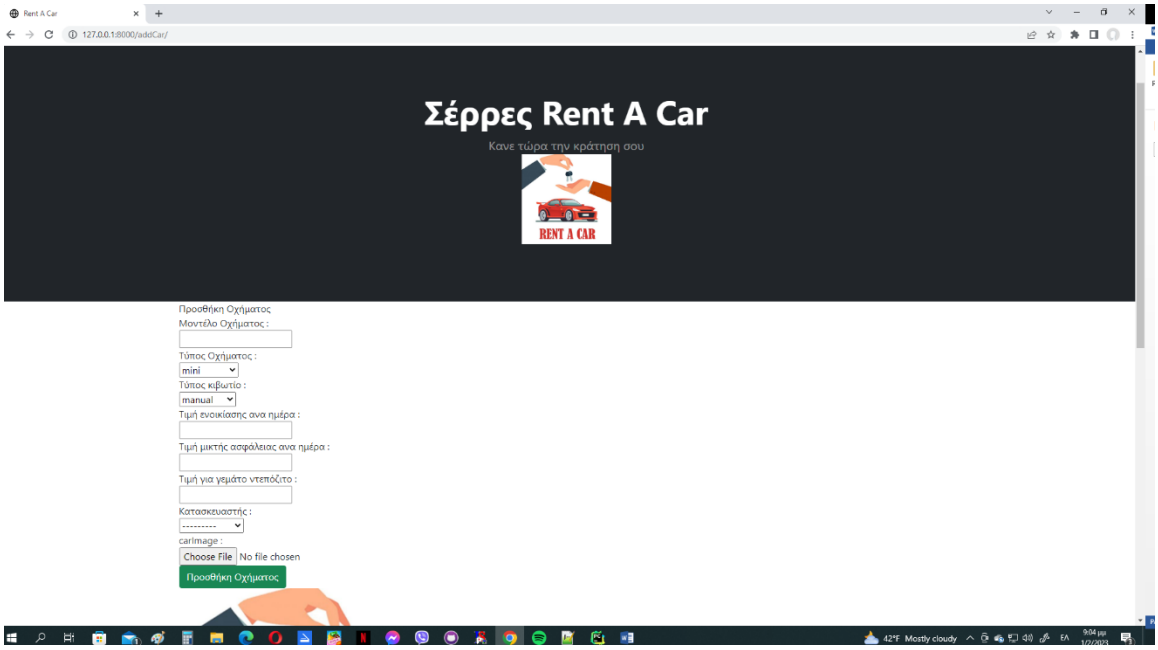
Ο κώδικας για την σελίδα «Έλεγχος στόλου» περιλαμβάνει τα αντίστοιχα view «**watchFleet**», «**watchFleetProductDetails**», «**updateProduct**», «**addCar**» και επίσης τα αρχεία «**watchFleet.html», «watchFleetProductDetails.html», «updateProduct.html», «addCar.html»**.



Εικόνα 29 Λεπτομέρειες οχήματος



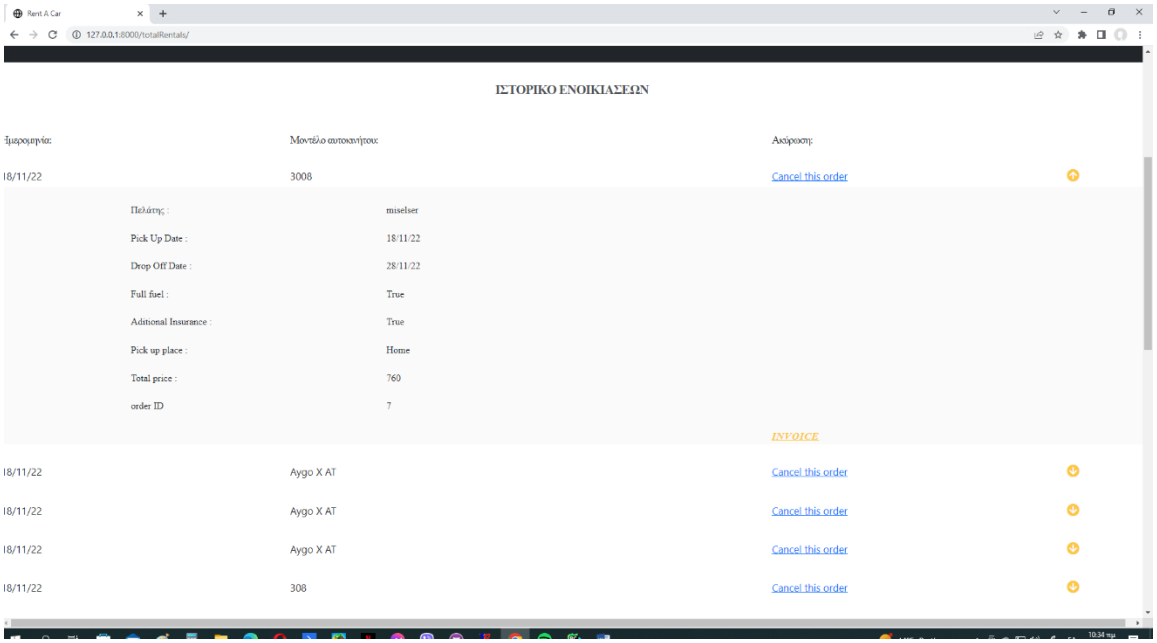
Εικόνα 30 Ενημέρωση οχήματος



Εικόνα 31 Προσθήκη οχήματος

3.4.11 Ιστορικό Ενοικιάσεων

Ο χρήστης υπάλληλος – εταιρία έχει ακόμα μια επιλογή στο μενού του προφίλ του. Η επιλογή αυτή είναι το «Ιστορικό ενοικιάσεων» όπου έχει την δυνατότητα να παρακολουθεί όλες τις ενοικιάσεις που έχουν γίνει, *Εικόνα 32 Ιστορικό ενοικιάσεων* . Επίσης έχει την δυνατότητα να ακυρώσει κάποια ενοικίαση.

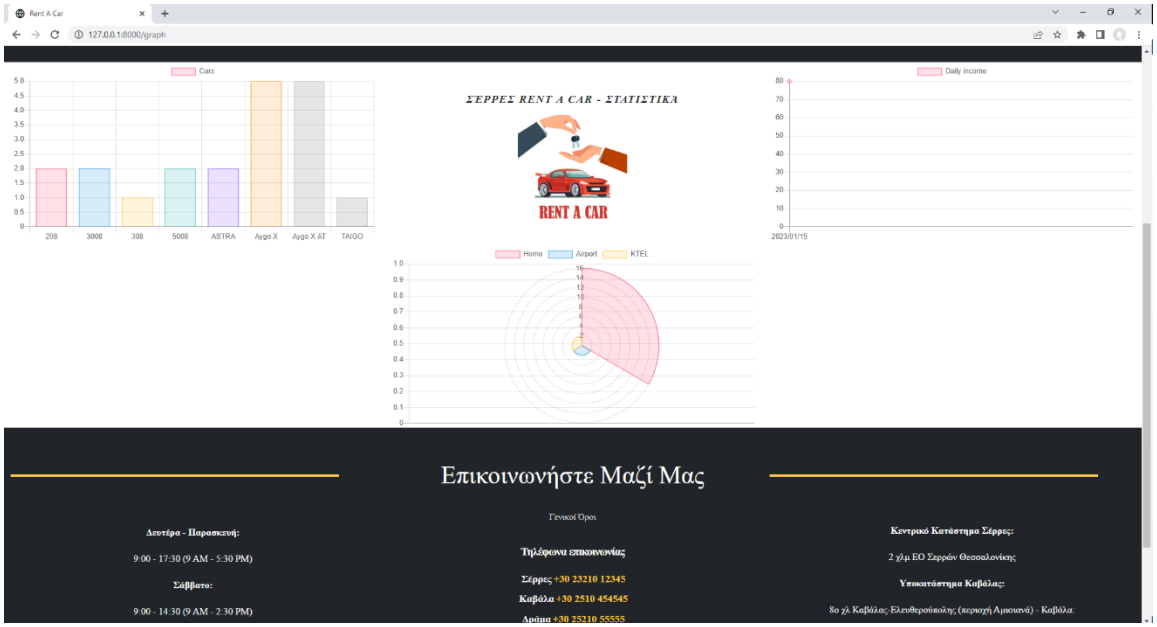


Εικόνα 32 Ιστορικό ενοικιάσεων

Ο κώδικας περιλαμβάνει το αντίστοιχο view «**totalRentals**»και το αρχείο «**totalRentals.html**».

3.4.12 Στατιστικά

Η σελίδα «Στατιστικά» παρέχει στους διαχειριστές της εφαρμογής την δυνατότητα να ελέγχουν με την βοήθεια γραφημάτων κάποια χρήσιμα στατιστικά στοιχεία για τις ενοικιάσεις που έχουν πραγματοποιηθεί. Υπάρχουν συνολικά τρία γραφήματα, ένα που μας δείχνει το πλήθος των ενοικιάσεων ανά μοντέλο οχήματος, το δεύτερό το πλήθος των ενοικιάσεων ανά σημείο εκκίνησης και το τρίτο τον ημερήσιο τζίρο σε ευρώ. *Εικόνα 33 Στατιστικά*



Εικόνα 33 Στατιστικά

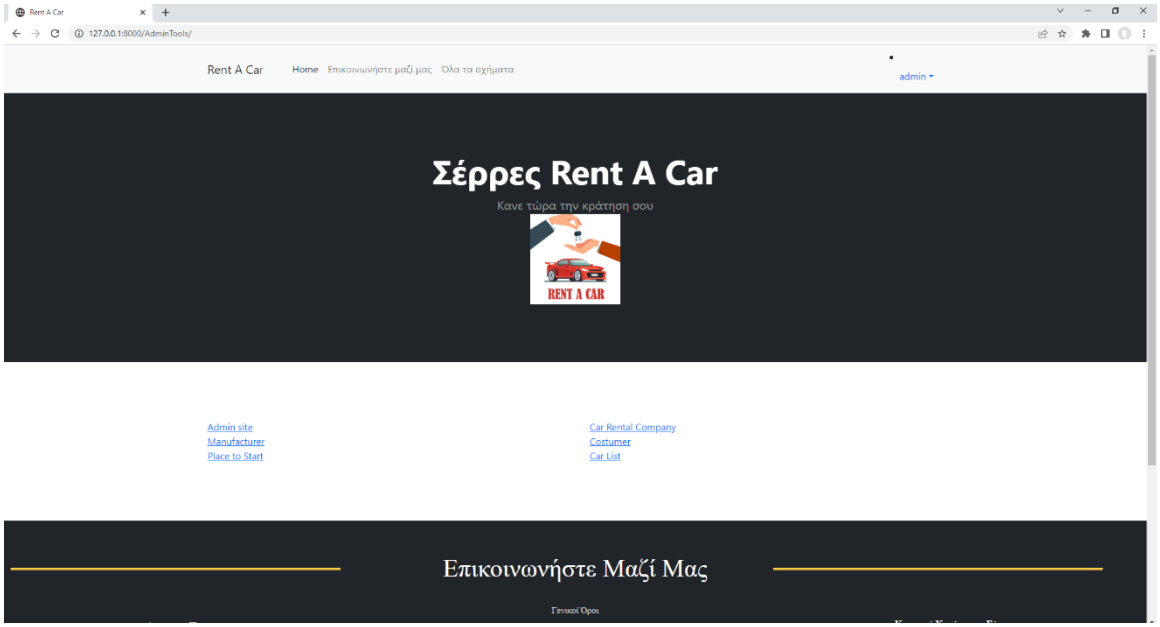
Ο κώδικας για στην σελίδα «Στατιστικά» περιλαμβάνει τα αρχεία:

- **Αρχείο graphViews.py**[41]
- **Αρχείο graph.html**[42]

3.4.13 Εργαλεία διαχειριστή

Η σελίδα «Εργαλεία διαχειριστή» είναι αποκλειστικά προσπελάσιμη από τον διαχειριστή του συστήματος. Ο σύνδεσμος υπάρχει στην λίστα με τις επιλογές κάτω από το προφίλ διαχειριστή. Η σελίδα περιέχει συνδέσμους για τα εργαλεία του διαχειριστή. Το admin view για την διαχείριση της βάσης δεδομένων, όπως έχουμε αναφέρει στο κεφάλαιο 3.2 Admin View, όπως επίσης και σύνδεση με το rest framework που έχει δημιουργηθεί.

[41] Βλέπε παράρτημα σελ. 74-75
[42] Βλέπε παράρτημα σελ. 75-78



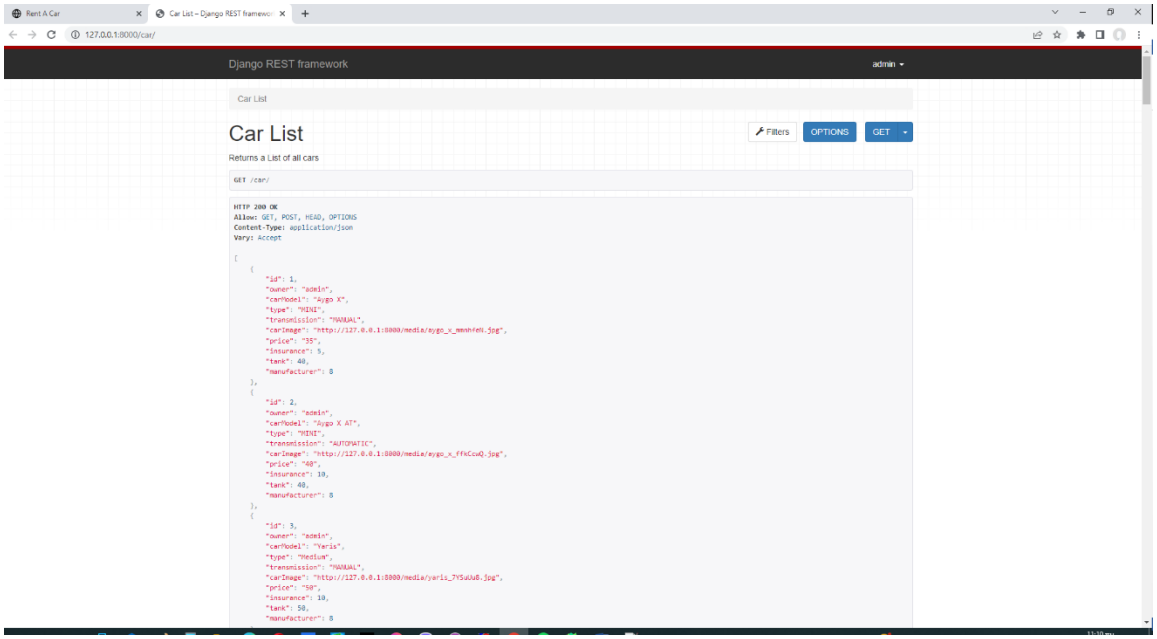
Εικόνα 34 Εργαλεία διαχειριστή

Για μελλοντική χρήση και επέκταση της εφαρμογής έχει υλοποιηθεί και ένα πλήρως λειτουργικό REST framework[43] το οποίο μας παρέχει την δυνατότητα διαχείρισης της βάσης δεδομένων μέσω ενός api[44] όπου μελλοντικά μπορεί να γίνει επέκταση της εφαρμογής και σε κινητές συσκευές ή tablet. Το framework μετατρέπει και μας παρουσιάζει σε μορφή json[45] ***Εικόνα 35 Django REST framework – Car List*** τα δεδομένα της βάσης, τα οποία μπορούμε να τροποποιήσουμε, να τα διαγράψουμε ή να προσθέσουμε μια εγγραφή ,***Εικόνα 36 Django REST framework –Add car from post***. Επίσης μας παρέχει την δυνατότητα ενσωμάτωσης φίλτρων για ευκολότερη διαχείριση και προσπέλαση των δεδομένων. ***Εικόνα 37 Django REST framework - tools, Εικόνα 38 Django REST framework – filters***

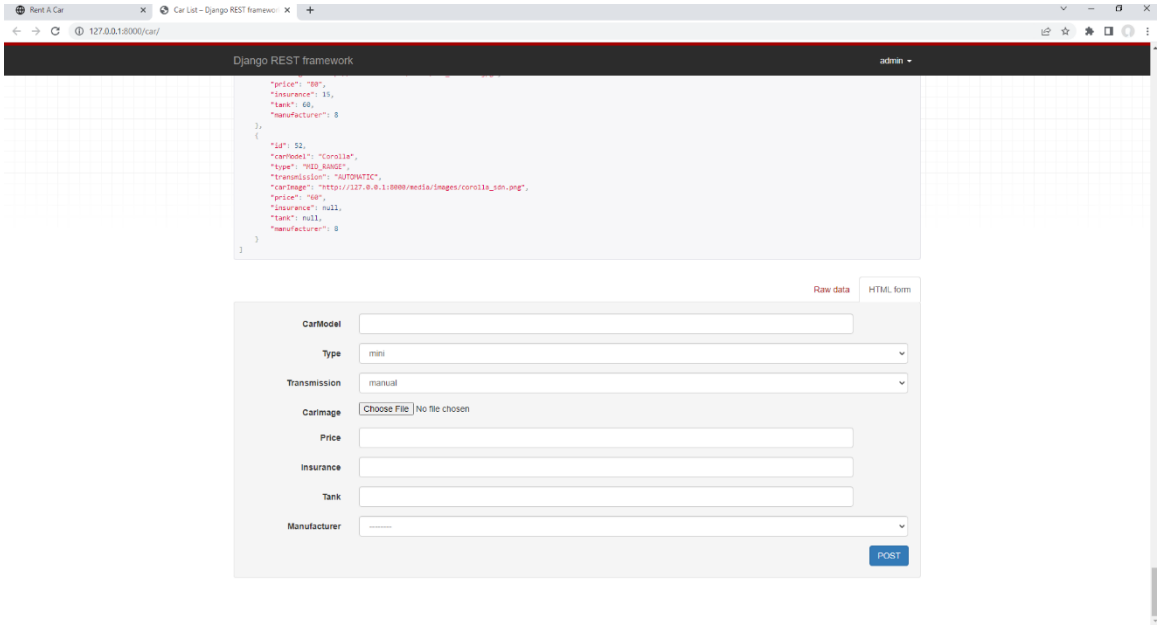
[43] https://www.django-rest-framework.org/

[44] https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B5%CF%80%CE%B1%CF%86%CE%AE_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D_%CE%B5%CF%86%CE%B1%CF%81%CE%BC%CE%BF%CE%B3%CF%8E%CE%BD

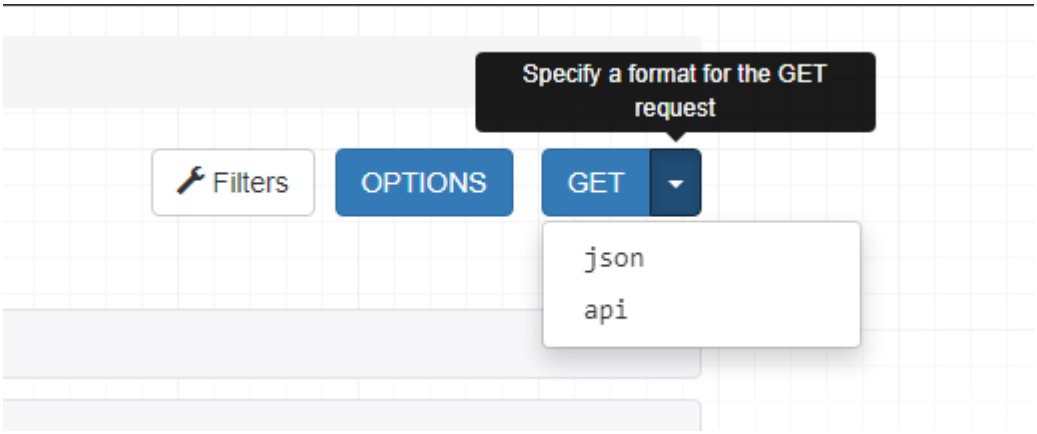
[45] https://el.wikipedia.org/wiki/JSON , https://en.wikipedia.org/wiki/JSON



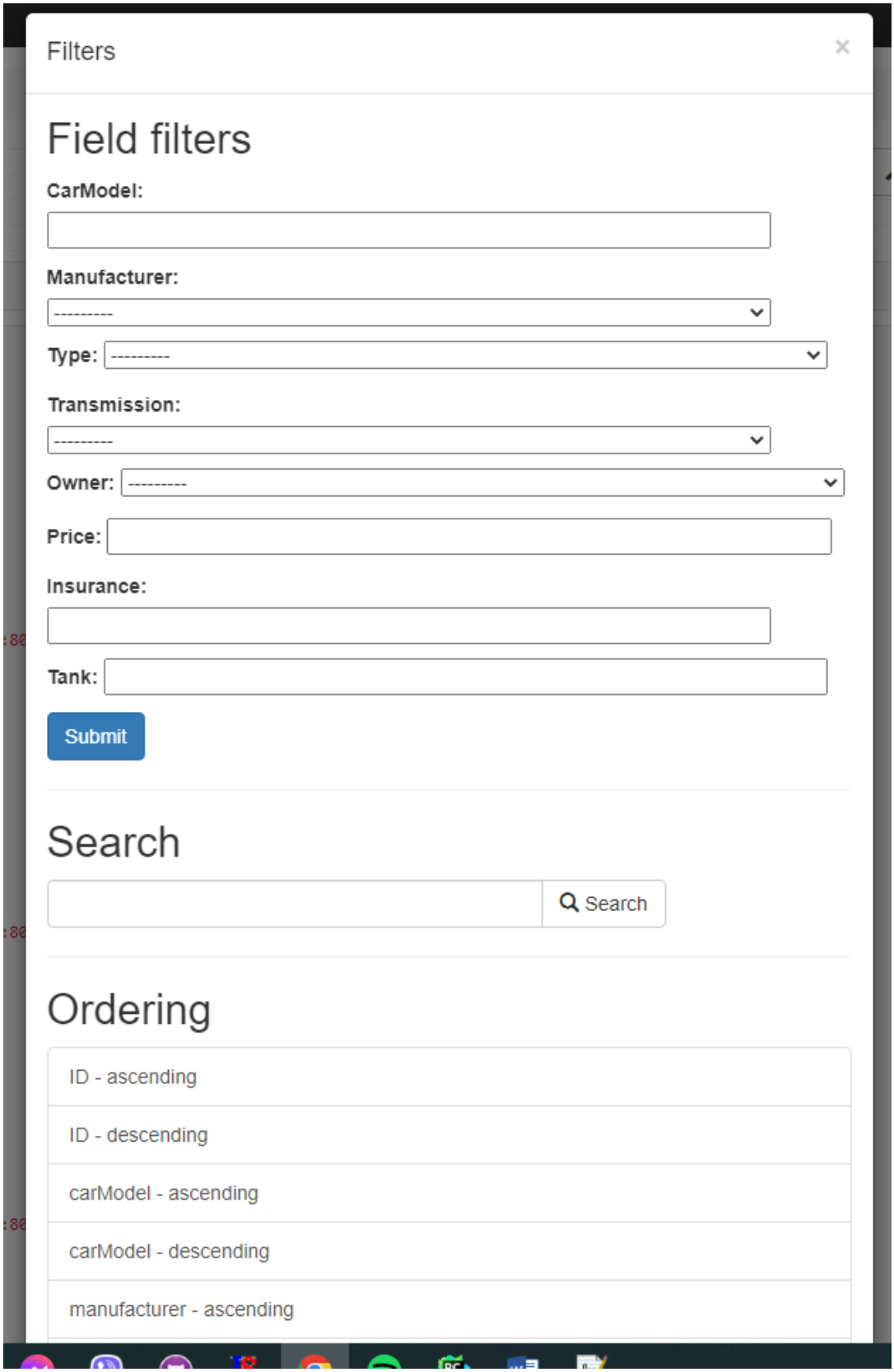
Εικόνα 35 Django REST framework – Car List



Εικόνα 36 Django REST framework –Add car from post



Εικόνα 37 Django REST framework - tools



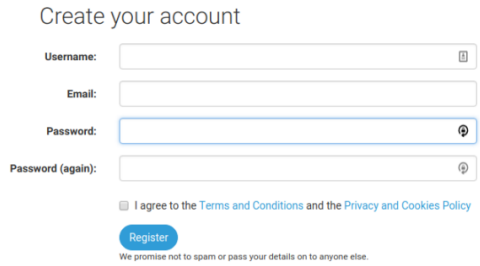
Εικόνα 38 Django REST framework – filters

ΚΕΦΑΛΑΙΟ 4 – ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΟΥΣΙΑΣΗ

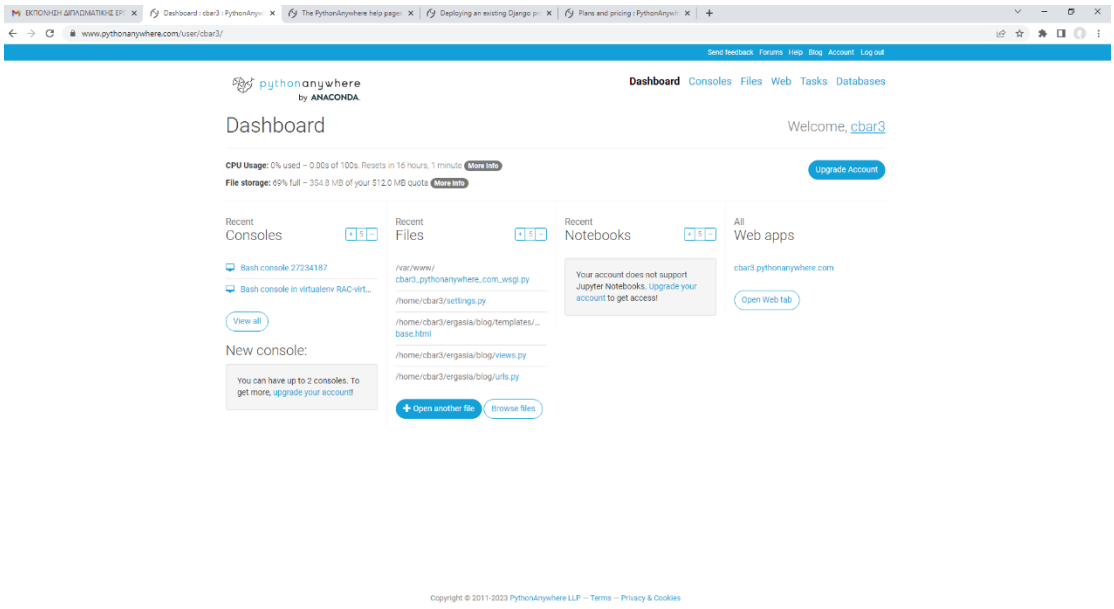
Για τις ανάγκες τις παρουσίασης της μεταπτυχιακής διπλωματικής εργασίας, έγινε χρήση του PythonAnywhare, το οποίο είναι μια διαδικτυακή πλατφόρμα που επιτρέπει στον χρήστη να ανεβάσει τα project που έχει δημιουργήσει. Η πλατφόρμα ουσιαστικά λειτουργεί σαν web hosting server ο οποίος φιλοξενεί την σελίδα για ένα χρονικό διάστημα 3 μηνών, στην συνέχεια ζητάει ανανέωση. Επίσης υπάρχουν δωρεάν παροχές όπως επίσης και παροχές επί πληρωμής. Για την συγκεκριμένη εργασία επιλέχθηκε ο δωρεάν λογαριασμός.

Αρχικά ο χρήστης θα πρέπει να δημιουργήσει κάποιον λογαριασμό για να μπορέσει στην συνέχεια να συνδεθεί στην πλατφόρμα. *Εικόνα 39 Εγγραφή PythonAnywhere*

- https://www.pythonanywhere.com



Εικόνα 39 Εγγραφή PythonAnywhere



Εικόνα 40 PythonAnywhere Dashboard

Ο χρήστης αφού έχει συνδεθεί στην πλατφόρμα, *Εικόνα 40 PythonAnywhere Dashboard* , βλέπει τον πίνακα ελέγχου της πλατφόρμας. Εδώ υπάρχουν διάφορες επιλογές και εργαλεία. Για να μπορέσει κάποιος χρήστης να ανεβάσει ένα project του στην πλατφόρμα υπάρχουν δύο τρόποι. Αν ο χρήστης έχει αποθηκευμένα τα αρχεία του μόνο στον υπολογιστή του, τότε θα πρέπει να συμπίεση σε ένα αρχείο της μορφής zip ή rar με το αντίστοιχο πρόγραμμα τον φάκελο του project και στην συνέχεια επιλέγοντας από τον πίνακα ελέγχου την επιλογή files να ανεβάσει το συμπιεσμένο αρχείο που έχει δημιουργήσει. Ενναλακτικά αν τα αρχεία του project υπάρχουν ήδη σε κάποια git πλατφόρμα τότε ο χρήστης επιλέγει από τον πίνακα ελέγχου να ανοίξει μια bash κονσόλα και να δώσει την παρακάτω εντολή

```
# for example
$ git clone https://github.com/myusername/myproject.git
```

Στην συνέχεια όποια και από τις δυο μεθόδους που περιεγράφηκαν παραπάνω ο χρήστης θα πρέπει να δημιουργήσει ένα virtualenv για την εφαρμογή. Θα πρέπει να δώσει τις παρακάτω εντολές με την ακόλουθη σειρά.

```
$ mkvirtualenv --python=/usr/bin/python3.10 mysite-virtualenv
(mysite-virtualenv)$ pip install django
# or, if you have a requirements.txt:
(mysite-virtualenv)$ pip install -r requirements.txt
```

Στην τελευταία εντολή όπως φαίνεται παραπάνω ο χρήστης κάνει εγκατάσταση το αρχείο requirements.txt . Το αρχείο αυτό περιέχει όλα πακέτα που χρειάστηκε για την εφαρμογή που ανεβάζει στην πλατφόρμα. Το αρχείο αυτό μπορεί να το εξάγει από το pycharm δίνοντας στην κονσόλα την εντολή:

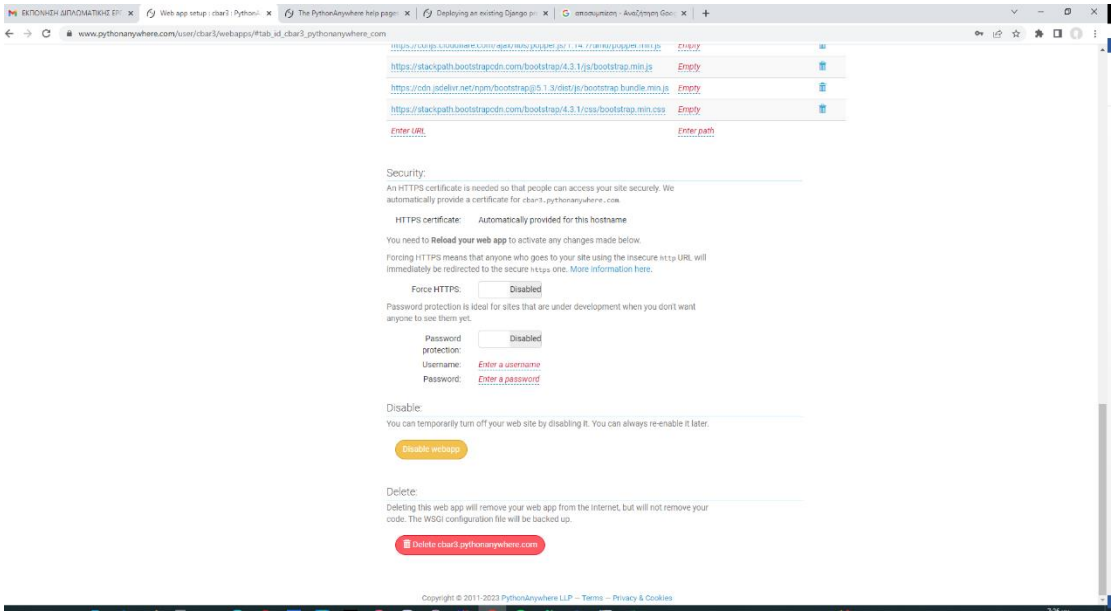
```
pip3 freeze > requirements.txt
```

Με την εντολή αυτή δημιουργείτε ένα αρχείο της μορφής txt όπου περιέχει όλα τα πακέτα και τις εκδόσεις που χρησιμοποιήθηκαν κατά την υλοποίησης της εφαρμογής και που είναι αναγκαία για να δουλεύει η εφαρμογή.

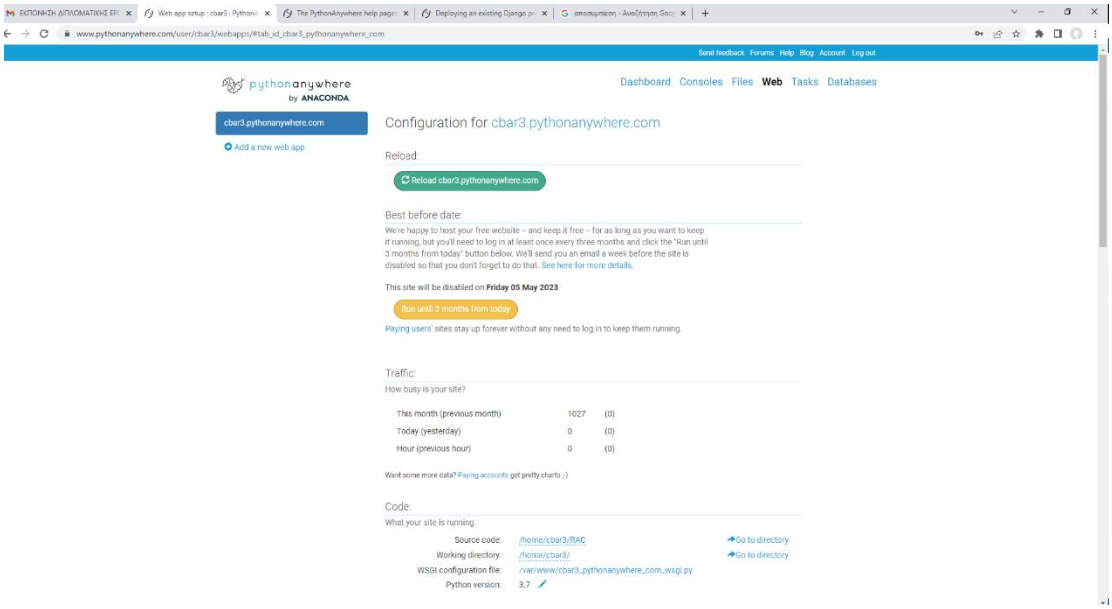
Αν ο χρήστης έχει αποθηκεύσει αποκλειστικά τοπικά στον υπολογιστή του τα αρχεία του project και έχει κάνει upload το συμπιεσμένο αρχείο όπως περιεγράφηκε παραπάνω τότε θα πρέπει να κάνει αποσυμπίεση τα αρχεία του δίνοντας την εντολή :

```
# for example
$ (mysite-virtualenv)$ unzip myproject.zip
```

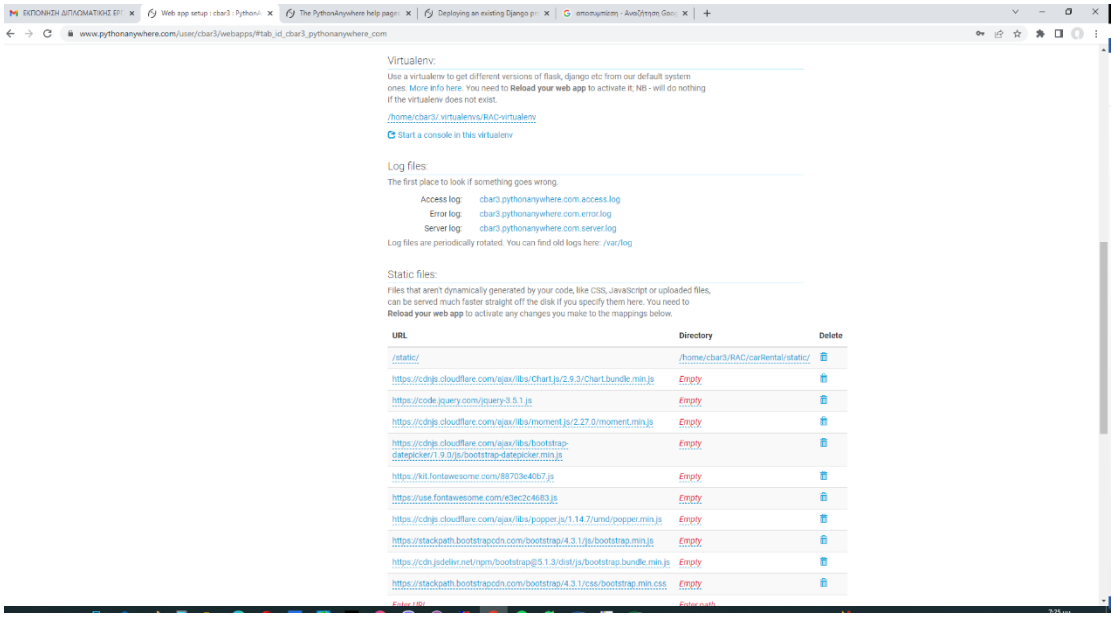
Η συνέχεια είναι κοινή είτε έχουμε ανεβάσει το συμπιεσμένο αρχείο στην πλατφόρμα είτε το ανεβάσαμε μέσω του git. Ο χρήστης επιλέγει από τον πίνακα ελέγχου το web tab, *Εικόνα 41 PythonAnywhere – Web tab 1, Εικόνα 42 PythonAnywhere – Web tab 2,*



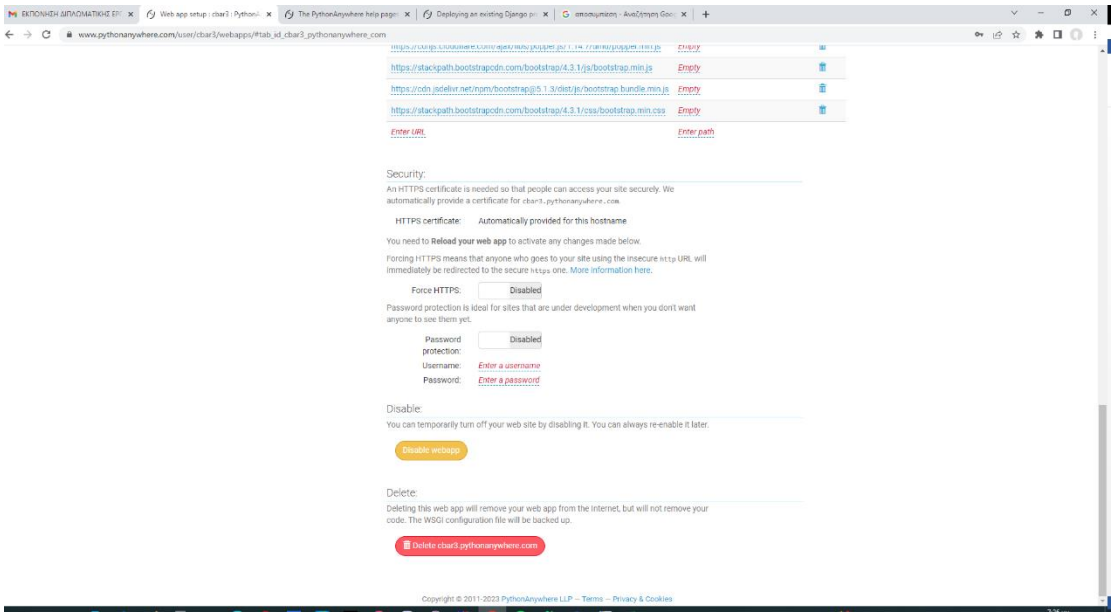
Εικόνα 43 PythonAnywhere – Web tab 3



Εικόνα 41 PythonAnywhere – Web tab 1

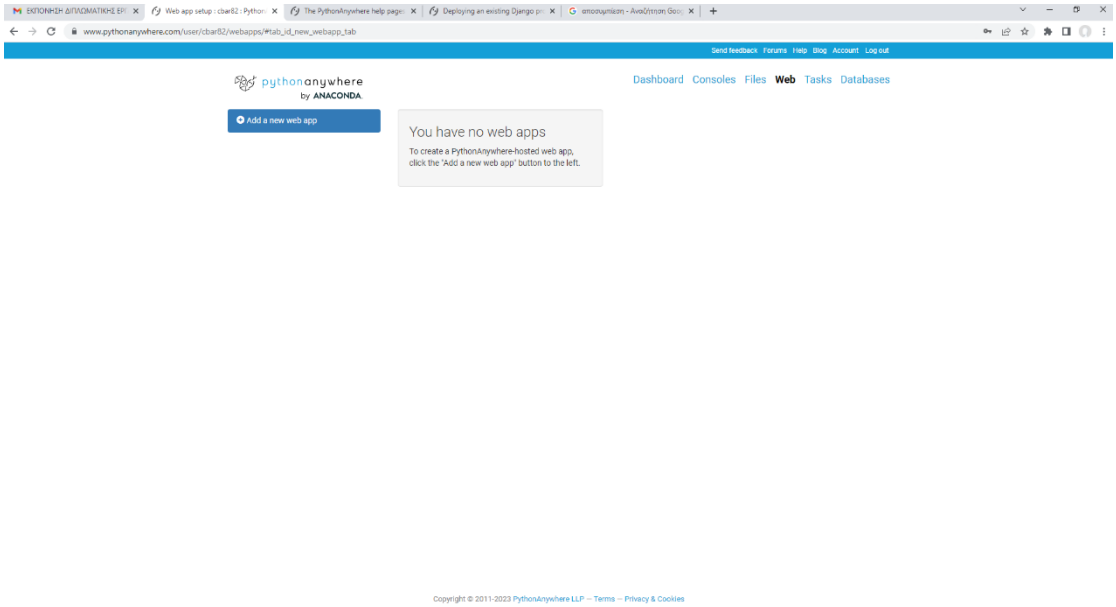


Εικόνα 42 PythonAnywhere – Web tab 2



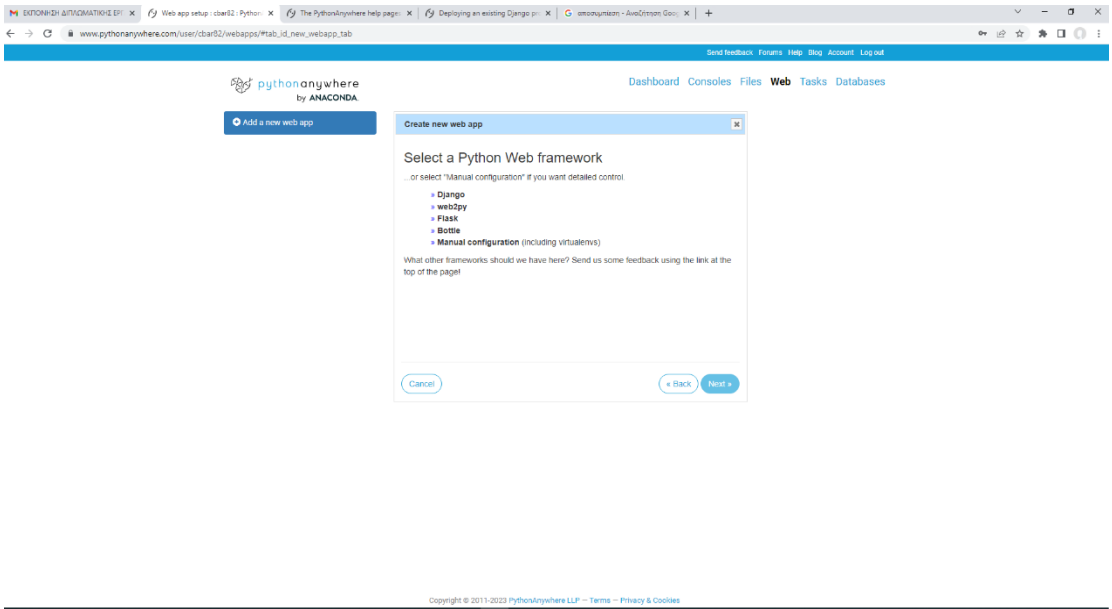
Εικόνα 43 PythonAnywhere – Web tab 3

Επιλέγει να δημιουργήσει ένα νέο web app, *Εικόνα 44 PythonAnywhere – Add a new web app*



Εικόνα 44 PythonAnywhere – Add a new web app

Εδώ ο χρήστης επιλέγει manual configuration για να μπορέσει να καθορίσει ο ίδιος τις παραμέτρους του συστήματος. Στην συνέχεια θα πρέπει να ορίσει την διαδρομή όπου έγινε η εγκατάσταση του virtualenv. *Εικόνα 42 PythonAnywhere – Web tab 2, Εικόνα 46 PythonAnywhere- Enter a virtualenv name*



Εικόνα 45 PythonAnywhere – Add a new web app (Manual configuration)

Εικόνα 46 PythonAnywhere- Enter a virtualenv name

Προαιρετικά μπορεί να κάνει το ίδιο και με την διαδρομή για το source code.

Στην συνέχεια θα πρέπει να επεξεργαστεί το WSGI file, μέσα στην δομή των αρχείων του project υπάρχει και ένα αρχείο με το όνομα wsgi.py, δεν πρέπει να αλλαχθεί αυτό. Ο χρήστης πρέπει να επιλέξει στο web tab → κάτω από το Code: → την επιλογή WSGI configuration file. *Εικόνα 41 PythonAnywhere – Web tab 1*

Θα ανοίξει μια σελίδα με έναν editor με τον κώδικα του αρχείου wsgi.py, ο χρήστης θα πρέπει να κάνει διαγράψει ότι υπάρχει στο αρχείο να μεταφέρει τον παρακάτω κώδικα στον editor. Στην συνέχεια να κάνει τις απαραίτητες αλλαγές

path = '/home/myusername/mysite'

και στο

```
os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'
```

με τα ονόματα του project που έχει δημιουργήσει.

```
# ++++++++++++ DJANGO ++++++++++++
# To use your own Django app use code like this:
import os
import sys

# assuming your Django settings file is at '/home/myusername/mysite/mysite/
settings.py'
path = '/home/myusername/mysite'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

## Uncomment the lines below depending on your Django version
###### then, for Django >=1.5:
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
###### or, for older Django <=1.4
```

```
#import django.core.handlers.wsgi
#application = django.core.handlers.wsgi.WSGIHandler()
```

Γυρίζουμε και πάλι στο web tab και επιλέγουμε reload. Η εφαρμογή δουλεύει πλέον χωρίς static files. Στην συνέχεια ανοίγουμε την κονσόλα του virtualenv και δίνουμε τις παρακάτω εντολές :

- python manage.py migrate
- python manage.py collectstatic

Στην συνέχεια πάμε web tab → κάτω από το Static: → δίνουμε την διαδρομή για τον φάκελο όπου βρίσκονται τα στατικά αρχεία. (Φωτογραφίες, CSS, Javascript κ.α.)

Τέλος αλλάζουμε το αρχείο settings.py της εφαρμογής μας και προσθέτουμε:

- ALLOWED_HOSTS = [’127.0.0.1’, ’.pythonanywhere.com’]

Κάνουμε ξανά Reload και πλέον η εφαρμογή μας δουλεύει πλήρως μορφοποιημένη και με φωτογραφίες.

ΚΕΦΑΛΑΙΟ 5 – ΕΠΙΛΟΓΟΣ

5.1 Σύνοψη και συμπεράσματα

Στο πλαίσιο υλοποίησης της διπλωματικής μεταπτυχιακής εργασίας δημιουργήθηκε με web based εφαρμογή ενός marketplace ενοικίασης αυτοκινήτων. Απώτερος σκοπός ήταν η εμβάθυνση στα γνωστικά αντικείμενα της ανάπτυξης λογισμικού, ο οποίος επιτεύχθηκε στο σύνολο του. Για τις ανάγκες που προέκυψαν κατά την διάρκεια της υλοποίησης και δημιουργίας της εφαρμογής επιστρατεύτηκαν όλες οι γνώσεις και οι δεξιότητες που αποκτήθηκαν από την παρακολούθηση του μεταπτυχιακού προγράμματος σπουδών. Το πλέον σημαντικό όμως ήταν το γεγονός ότι αυτές οι γνώσεις και οι δεξιότητες εμπλουτίστηκαν και αναπτύχθηκαν περεταίρω ούτως ώστε να ξεπεραστούν όλες οι δυσκολίες κατά την υλοποίηση. Κατά κύριο λόγο έγινε μια περαιτέρω εμβάθυνση σε τεχνικές προγραμματισμού, σε αυτό κατά βάση βοήθησε το ανεξάντλητο υλικό που υπάρχει στο διαδίκτυο όπως και οι κοινότητες και τα φόρουμ που με την βοήθεια τους λύθηκαν οι όποιες απορίες και τεχνικές δυσκολίες προέκυψαν. Ήταν ένα πολύ δημιουργικό εγχείρημα το οποίο έδωσε από μεριά του την δική του βαρύτητα στο σύνολο του μεταπτυχιακού προγράμματος σπουδών.

5.2 Μελλοντικές επεκτάσεις

Όπως ήδη έχει αναφερθεί στο πλαίσιο της υλοποίησης της διπλωματικής εργασίας αναπτύχθηκε και ένα επιπλέον εργαλείο το οποίο μας δίνει την δυνατότητα για μελλοντική επέκταση της εφαρμογής. Ουσιαστικά έχει υλοποιηθεί η προεργασία που χρειάζεται από την πλευρά του backend για την χρήση της εφαρμογής με μελλοντικές επεκτάσεις όπως η υλοποίηση μιας εφαρμογής για κινητά τηλέφωνα ή tablet όπου μέσω του REST framework και του api μπορεί να επικοινωνήσει η εφαρμογή της κινητής συσκευής με την ήδη υπάρχουσα βάση δεδομένων και κατ’ επέκταση με την εφαρμογή. Ουσιαστικά με κατάλληλες μεθόδους και τεχνικές μπορεί η εφαρμογή να έχει ακριβώς τις ίδιες δυνατότητες που παρέχει ήδη μέσω της web based μορφής της.

ΒΙΒΛΙΟΓΡΑΦΕΙΑ - ΑΝΑΦΟΡΕΣ

https://www.w3schools.com/

https://www.geeksforgeeks.org/

https://stackoverflow.com/

https://www.djangoproject.com/

https://www.django-rest-framework.org/

https://startbootstrap.com/template/shop-homepage

https://icons.getbootstrap.com/

https://github.com/Darshan4114/Django_HMS

https://github.com/WinterOdin/car-rental-with-Django

https://bootstrap-datepicker.readthedocs.io/en/latest/

https://stripe.com/docs/api

https://docs.djangoproject.com/en/4.1/ref/csrf/

https://docs.djangoproject.com/en/4.1/topics/forms/

https://docs.djangoproject.com/fr/2.2/topics/forms/modelforms/

https://docs.djangoproject.com/en/4.1/ref/forms/widgets/#selector-and-checkbox-widgets

https://docs.djangoproject.com/en/4.1/ref/forms/validation/

https://docs.djangoproject.com/en/4.1/ref/forms/api/

https://www.w3schools.com/howto/howto_js_alert.asp

https://stackoverflow.com/questions/7347786/html-anchor-tag-with-javascript-onclick-event

https://getbootstrap.com/docs/4.0/layout/grid/

https://stackoverflow.com/questions/59081451/django-filter-to-check-if-there-are-any-other-booking-between-the-given-dates

https://stackoverflow.com/questions/30458037/how-to-redirect-to-a-particular-section-of-a-page-in-html

https://docs.djangoproject.com/en/4.1/ref/templates/language/#templates

https://www.tutorialspoint.com/http/http_responses.htm

https://www.codecademy.com/article/http-requests

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΑΡΑΡΤΗΜΑ

Αρχείο home.html (datepicker)

```
                        <div class="row">
                            <div class="col-5">
                                <div class="textUserData">
                                    Ημερομηνία Παραλαβής:
                                </div>
                            </div>
                            <div class="col-7">

                                <label>
                                    <input type="text" class="dateSelect
startDateJq " required="required" name='startDate' />
                                </label>
                            </div>
                        </div>
                        <div class="row">
                            <div class="col-5">
                                <div class="textUserData">
                                    Ημερομηνία παράδοσης:
                                </div>
                            </div>
                            <div class="col-7">
                                <label>
                                    <input type="text" class="dateSelect
endDateJq" required="required" name='endDate' />
                                </label>
                            </div>
                        </div>
                        <div class="row ">
                            <div class="col-md-4 col-4"></div>
                                <div class="col-md-4  col-4" >
                                    <div class="orderButton">
                                        <button type='submit'
formaction="{% url 'home' %}" formmethod="post"
                                                value="Αναζήτηση"
class="raise">Αναζήτηση
                                        </button>
                                    </div>
                                </div>
                            <div class="col-md-4 col-4"></div>
                        </div>


<script>
    var data            = JSON.parse("{{dataClean|escapejs}}");
    var currentDate     = new Date();

    $('.dateSelect').datepicker({
        format: 'mm/dd/yyyy',
        autoClose:true,
        datesDisabled:data,
        startDate: currentDate ,
    });
    $('.dateSelect').datepicker()
    .on('changeDate', function() {
```

```
        var startDate    = new Date($(".startDateJq").val());
        var endDate      = new Date($(".endDateJq").val());
        var diffDate     = (endDate  - startDate) / (1000 * 60 * 60 *
24);
        var days         = Math.round(diffDate);
        console.log(days)
    });
</script>
```

Έλεγχος διαθεσιμότητας – Availability check

```
# Ο έλεγχος που γίνεται για να επιστρέψει μόνο τα διαθέσιμα οχήματα.
Ο χρήστης επιλέγει ημερομηνία αναχώρησης
    # και ημερομηνία επιστροφής και η σελίδα επιστρέφει μόνο τα
αποτελέσματα - τα οχήματα που είναι διαθέσιμα.
    format = "%m/%d/%Y"
    if request.method == 'POST':
        filterStartDate = request.POST['startDate']
        filterEndDate = request.POST['endDate']

        formattedFilterStartDate = datetime.strptime(filterStartDate,
format)
        formattedFilterEndDate = datetime.strptime(filterEndDate,
format)

        # Φίλτρα ελέγχου για τη διαθεσιμότητα. Σύγκριση της
ημερομηνία που επιλέγει ο χρήστης με τις ημερομηνίες των ήδη
        # νοικιασμένων οχημάτων που είναι αποθηκευμένα στη βάση.
        rentals = Rental.objects.filter(
            (
                    Q(startDate__lte=formattedFilterEndDate)
                    & Q(finishDate__gte=formattedFilterStartDate)
            ) | (
                    Q(startDate__lte=formattedFilterEndDate)
                    & Q(finishDate__gte=formattedFilterStartDate)
            ))

        # Αν υπάρχουν διαθέσιμο οχήματα επέστρεψε μια λίστα με τα
διαθέσιμα
        for x in carData:
            available = True

            for y in rentals:
                if y.carId == x.id:
                    available = False
            if available:
                available_cars.append(x.id)

        carsCopy = []

        for x in carData:
            if available_cars.__contains__(x.id):
                carsCopy.append(x)

        cars = carsCopy
```

view user_login1 κωδικας

```
def user_login1(request):
    # view για τη φόρμα εισόδου του χρήστη στην εφαρμογή
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            # Γίνεται επεξεργασία του request αν υπάρχουν, έχουν
γίνει POST δεδομένα. Username/ password
            username = request.POST['username']
            password = request.POST['password']
            # Έλεγχος αν ο συνδυασμός username και password είναι
σωστός
            user = authenticate(username=username, password=password)
            if user is not None:
                # Σώζει το session σε cookie για να μπορέσει να
πραγματοποιήσει την είσοδο του χρήστη
                login(request, user)
                # Επιτυχία εισόδου, επιστροφή στη home page
                return redirect('home')
        else:
            # Μήνυμα λάθος σε περίπτωση που χρησιμοποιηθούν λάθος
στοιχεία εισόδου.
            return render(request, 'login1.html', {'error_message':
'Incorrect username and / or password.'})
    else:
        form = LoginForm
        # Όταν δεν έχουν μπει στοιχεία εισόδου στη φόρμα τότε
επιστρέφει ξανά τη σελίδα εισόδου με τη φόρμα.
    return render(request, 'login1.html', {'loginForm': form})
```

Αρχείο mainheader.html – Μενού εισόδου – Login menu

```
<!-- Ο παρακάτω κώδικας αναγνωρίζει ποιος χρήστης έχει συνδεθεί στην
εφαρμογή και ανάλογα με τα δικαιώματα
                που έχει ο εκάστοτε χρήστης η εφαρμογή παρουσιάζει
μια διαφορετική λίστα με επιλογές κάτω απο το όνομα
                του χρήστη-->
                    </div>
                {% if user.is_authenticated %}
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle"
id="navbarDropdown" href="#" role="button" data-bs-toggle="dropdown"
aria-expanded="false">{{ request.user.username}}</a>
                            <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">
                                <li><a class="dropdown-item"
href="{% url 'logout' %}">Εξοδος</a></li>
                                <li><hr class="dropdown-divider"
/></li>
                                {% if request.user.is_active %}
                                    <li><a class="dropdown-item"
href="{% url 'customer' pk=user.id %}">Προφίλ Χρήστη</a></li>
                                {% endif  %}
                                {% if request.user.is_superuser
%}
                                    <li><a class="dropdown-item"
href="{% url 'pathUrlsAdmin' %}">Εργαλεία διαχειριστή</a></li>
                                {% endif  %}
                                {% if request.user.is_staff %}
```

```
                                        <li><a class="dropdown-item"
href="{% url 'totalRentals' %}">Ενεργές ενοικιάσεις</a></li>
                                        <li><a class="dropdown-item"
href="{#% url 'customer' pk=user.id %#}">Ακυρωμένες
Ενοικιάσεις</a></li>
                                        <li><a class="dropdown-item"
href="{% url 'watchFleet' %}">Διαχείριση στόλου</a></li>
                                            {% endif  %}
                                    </ul>
                            </li>
                    {% else  %}
                            <div>
                                <ul class="navbar-nav me-auto mb-2 mb-lg-
0 ms-lg-4">
                                    <li class="nav-item"><a class="nav-
link" href="{% url 'login1' %}">Είσοδος</a></li>
                                    </ul>
                            </div>
                    {% endif  %}
```

View user_register

```
def user_register(request):
    # Το view για τη σελίδα της εγγραφής του χρήστη
    # template = 'register.html'
    template = 'register1.html'
    # Έλεγχος αν είναι μέθοδος POST το request και στη συνέχεια να
επεξεργαστούν τα στοιχεία της φόρμας
    if request.method == 'POST':
        # Δημιουργία ενός στιγμιότυπου της φόρμας το οποίο θα
τροφοδοτηθεί με δεδομένα απο το POST request
        form = RegisterForm(request.POST)
        # Έλεγχος αν η φόρμα είναι έγκυρη
        if form.is_valid():
            # Έλεγχος αν το username που χρησιμοποιήθηκε υπάρχει στη
βάση. Αν έχει ήδη γίνει χρήση απο άλλον χρήστη.
            if
User.objects.filter(username=form.cleaned_data['username']).exists():
                return render(request, 'register1.html', {
                    'form': form,
                    'error_message': 'Username already exists.'
                })
            # Έλεγχος αν το email που χρησιμοποιείται για την εγγραφή
υπάρχει ήδη στην βάση.
            # Αν έχει ήδη γίνει χρήση απο άλλον χρήστη.
            elif
User.objects.filter(email=form.cleaned_data['email']).exists():
                return render(request, 'register1.html', {
                    'form': form,
                    'error_message': 'Email already exists.'
                })
            # Έλεγχος αν το password έχει συμπληρωθεί και στα 2 πεδία
που απαιτείται είναι το ίδιο.
            elif form.cleaned_data['password'] !=
form.cleaned_data['password_repeat']:
                return render(request, 'register1.html', {
                    'form': form,
                    'error_message': 'Passwords do not match.'
                })
```

```
            else:
                # Αν περάσει απο όλους τους ελέγχους τότε δημιούργησε
τον χρήστη - πελάτη.
                user = User.objects.create_user(
                    form.cleaned_data['username'],
                    form.cleaned_data['email'],
                    form.cleaned_data['password']
                )
                user.first_name = form.cleaned_data['first_name']
                user.last_name = form.cleaned_data['last_name']
                user.save()

                costumer = Costumer(user=user,

costumerFirstName=form.cleaned_data['first_name'],

costumerLastName=form.cleaned_data['last_name'],

costumerEmail=form.cleaned_data['email'],

costumerPhoneNumber=form.cleaned_data['phone_number'],

costumerAFM=form.cleaned_data['AFM'],

                                    )
                costumer.save()

                # Και πλέον κάνε εγγραφή τον χρήστη
                login(request, user)

                # μεταφορά στην αρχική σελίδα
                return HttpResponseRedirect('home')

    # Αν δεν υπάρχουν δεδομένα για POST απλά εμφάνισε την φόρμα
    else:
        form = RegisterForm()

    return render(request, 'register1.html', {'form': form})
```

createRental view

```
@login_required(login_url='home.html')
# Το view για τη σελίδα κράτησης του οχήματος
def createRental(request, pk):
    dataHolder = []
    dataClean = []
    current = request.user
    carData = Car.objects.get(id=pk)
    pickupPlace = PlaceToStart.objects.all()
    rawData = Rental.objects.filter(carId=pk)
    priceOfAddi = Extras.objects.last()

    # Δημιουργία μιας λίστας με τις ημερομηνίες που πρέπει να
αποκλειστούν απο το datepicker.
    # Αυτές που δεν είναι διαθέσιμο το όχημα
    for x in rawData:
        if x.finishDate > datetime.now().date():
            srtDate = x.startDate
            endDate = x.finishDate
```

```
            delta = endDate - srtDate
            dataHolder = [(srtDate + timedelta(days=i)) for i in
range(delta.days + 1)]

    dataClean = [x.strftime("%m/%d/%Y") for x in dataHolder]
    dataClean = dumps(dataClean)

    context = {
        'current': current,
        'carData': carData,
        'dataClean': dataClean,
        'pickupPlace': pickupPlace,
        'priceOfAddi': priceOfAddi
    }
    return render(request, 'rental.html', context, )
```

Αρχείο rental.html κώδικας

```
{% extends 'main.html' %}
{% load static  %}
{% block content %}

<form class="container">
    <div class="row">
        <div class="col-12">
            <div class="linkRent">
                <a class="backLinkCustomer" href="{% url 'home'  %}">
                    <i class="fas fa-arrow-circle-left
customerArrow"></i>
                    <span class="backTextCustomer">Back to
home</span>
                </a>
            </div>
        </div>
    </div>

    <form action="{% url 'order' pk=carData.pk  %}" method="POST"
id="formRental">
        {% csrf_token %}
        <div class="row">
            <div class="col-12 col-md-6">
                <div class="conUserData">
                    <div class="row">
                        <div class="col-12">
                            <div class="tileTitle">
                                Στοιχεία ενοικίασης
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Όνομα Χρήστη:
                            </div>
                        </div>
                        <div class="col-7">
                            <div class="UserData">
                                <label>
```

```
                                    <input class="readOnlyForm"
value=" {{ current.costumer.costumerFirstName}} {{
current.costumer.costumerLastName}} " readonly name='user'>
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Email Επικοινωνίας:
                            </div>
                        </div>
                        <div class="col-7">
                            <div class="UserData">

                                <label>
                                    <input class="readOnlyForm"
value="{{current.email}}" readonly name='mail'>
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Όχημα:
                            </div>
                        </div>
                        <div class="col-7">
                            <div class="UserData">
                                <label>
                                    <input class="readOnlyForm"
value="{{ carData.carModel}}" readonly name='carModel'>
                                </label>

                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Τιμή ανα ημέρα:
                            </div>
                        </div>
                        <div class="col-7">
                            <div class="UserData" id='priceField'>
                                {{ carData.price }}
                            </div>
                        </div>
                    </div>

                </div>
            </div>
            <div class="col-12 col-md-6">
                <div class="conDateData">
                    <div class="row">
                        <div class="col-12">
                            <div class="tileTitle">
                                Ημερομηνία Ενοικίασης
                            </div>
```

```
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Ημερομηνία Παραλαβής:
                            </div>
                        </div>
                        <div class="col-7">

                            <label>
                                <input type="text" class="dateSelect
startDateJq " required="required" name='startDate' />
                            </label>

                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Ημερομηνία παράδοσης:
                            </div>
                        </div>
                        <div class="col-7">

                            <label>
                                <input type="text" class="dateSelect
endDateJq" required="required" name='endDate' />
                            </label>

                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Σημείο Παραλαβής:
                            </div>
                        </div>
                        <div class="col-7">
                            <select name="pickUpPlace" id=""
class="pickUpPlace">
                                {% for i in pickupPlace  %}
                                <option value="{{  i.pk  }}">{{
i.placeToStart  }}</option>
                                {% endfor  %}
                            </select>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-12 col-md-6">
                <div class="conDateData">
                    <div class="row">
                        <div class="col-12">
                            <div class="tileTitle">
                                Επιπλέον Επιλογές
                            </div>
                        </div>
                    </div>
```

```
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">

                                Γεμάτο Ρεζερβουάρ:
                            </div>
                        </div>
                        <div class="col-3">
                            <div class="UserData">
                                <input class="tgl tgl-flat fuel"
id="cb1" type="checkbox" name="fuel" />
                                <label class="tgl-btn"
for="cb1"></label>
                            </div>
                        </div>
                        <div class="col-5">

                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Μικτή Ασφάλεια:
                            </div>
                        </div>
                        <div class="col-3">
                            <div class="UserData">
                                <input class="tgl tgl-flat insurance
" id="cb2" type="checkbox" name="insurance" />
                                <label class="tgl-btn"
for="cb2"></label>
                            </div>
                        </div>
                        <div class="col-4">
                            <div class="insuranceText">

                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-5">
                            <div class="textUserData">
                                Συνολικό κόστος:
                            </div>
                        </div>
                        <div class="col-3">
                            <div class="UserData">
                                <div class="priceTotal">

                                </div>
                            </div>
                        </div>
                        <div class="col-4">
                            <div class="UserData">

                            </div>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-12 col-md-6">
```

```
                <div class="conDateData">
                    <div class="row">
                        <div class="col-12">
                            <div class="tileTitle">
                                Επιβεβαίωση παραγγελίας
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-12">
                            <div class="infoRenting">
                               Ελέγξτε τα στοιχεία που δώσατε και
προχωρήστε στην επιβεβαίωση της κράτησης .
                            </div>
                        </div>
                    </div>
                    <div class="row ">
                        <div class="col-md-4 col-4"></div>
                            <div class="col-md-4  col-4" >
                                <div class="orderButton">
                                    <button type='submit'
formaction="{% url 'order' pk=carData.pk  %}" formmethod="post"
                                        value="Κράτηση"
class="raise">Κράτηση
                                    </button>
                                </div>
                            </div>
                        <div class="col-md-4 col-4"></div>
                    </div>
                </div>
            </div>
        </div>
    </form>
</form>

<script>
    var data            = JSON.parse("{{dataClean|escapejs}}");
    var currentDate     = new Date();
    var currentPrice    = $('#priceField').text();
      // we are taking this price in unsafe way because we don't use
this data to calculate the price in the backend
    var fuel            = {{priceOfAddi.fuel}};
    var insurance       = {{priceOfAddi.insurance}};
   // var place             = $('.pickUpPlace').text();


    $('.dateSelect').datepicker({
        format: 'mm/dd/yyyy',
        autoClose:true,
        datesDisabled:data,
        startDate: currentDate ,
    });
    $('.dateSelect').datepicker()
    .on('changeDate', function() {

        var startDate   = new Date($(".startDateJq").val());
        var endDate     = new Date($(".endDateJq").val());
        var diffDate    = (endDate  - startDate) / (1000 * 60 * 60 *
24);
        var days        = Math.round(diffDate);
        console.log(days)
```

```
        var price       = (currentPrice * days)
        $(".priceTotal").html(price);
        $(".insurance, .fuel").on('change',function(){
            if(  $(".insurance").is(':checked') &&
$(".fuel").is(':checked') ){
                $(".priceTotal").html(price+insurance*days+fuel)

            }else if(  $(".insurance").is(':checked') &&
!($(".fuel").is('unchecked')) ){
                $(".priceTotal").html(price+insurance*days)

            }else if(  !($(".insurance").is('unchecked')) &&
$(".fuel").is(':checked') ){
                $(".priceTotal").html(price+fuel)
                $(".insuranceText").toggle();
            }
            else if(  !($(".insurance").is('unchecked')) &&
!($(".fuel").is('unchecked')) ){
                $(".priceTotal").html(price)
            }
        });
    });
</script>
{% endblock %}
```

Αρχείο pdfViews.py κώδικας

```
from django.template.loader import get_template
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.views import View
from xhtml2pdf import pisa
from io import BytesIO
from carRental.models import RentalCompany, Car, Manufacturer,
Rental, Costumer, PlaceToStart, Extras, CanceledOrders
from itertools import chain
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User


def to_dict(instance):
    opts = instance._meta
    data = {}
    for f in chain(opts.concrete_fields, opts.private_fields):
        data[f.name] = f.value_from_object(instance)
    for f in opts.many_to_many:
        data[f.name] = [i.id for i in f.value_from_object(instance)]
    return data


def render_to_pdf(template_src, context_dict={}):
    template = get_template(template_src)
    html = template.render(context_dict)
    result = BytesIO()
    pdf = pisa.pisaDocument(BytesIO(html.encode("ISO-8859-1")),
result)
    if not pdf.err:
        return HttpResponse(result.getvalue(),
content_type='application/pdf')
    return None
```

```
# Opens up page as PDF
class ViewPDF(View):
    def get(self, request, pk, *args, **kwargs):
        rawData = Rental.objects.get(id=pk)
        cleanData = to_dict(rawData)

        context = {
            'cleanData': cleanData,
        }
        pdf = render_to_pdf('pdfInvoice.html', context)

        return HttpResponse(pdf, content_type='application/pdf')


# Automatically downloads to PDF file
class DownloadPDF(View):
    def get(self, request, pk, *args, **kwargs):
        current = request.user
        rawData = Rental.objects.get(id=pk)
        cleanData = to_dict(rawData)

        pdf = render_to_pdf('pdfInvoice.html', cleanData)

        response = HttpResponse(pdf, content_type='application/pdf')
        filename = "Invoice_%s.pdf" % ("12341231")
        content = "attachment; filename='%s'" % (filename)
        response['Content-Disposition'] = content
        return response
```

Αρχείο toggle.js

```
$('.clickFaq').click(function() {
    $('.displayFaq').toggle('1000');
    $("i", this).toggleClass("fa-arrow-circle-up fa-arrow-circle-down
");
});


$(".open").hide();
$('.questionUs').click(function(){
    $(this).next().slideToggle();
});


$(document).ready(function($) {
    $(".scroll").click(function(event){
        event.preventDefault();
        $('html,body').animate({scrollTop:$(this.hash).offset().top},
500);
    });
});
```

Αρχείο graphViews.py

```
from django.shortcuts import render, redirect
from .models import *
from django.http import HttpResponse, JsonResponse
from django.db.models import Count, Sum, Max
from collections import defaultdict, Counter
from datetime import datetime, timedelta, date
from django.contrib.auth.decorators import login_required


@login_required(login_url='home.html')
def graph(request):
    dateMoneyList, dataDateMoneyList, locationValues = [], [], []
    format = "%Y/%m/%d"
    current = request.user
    dataHolder = Rental.objects.all()
    location = PlaceToStart.objects.all().values_list('id',
'placeToStart')
    dataHolderPayed =
dataHolder.filter(payed=True).values('orderDate', 'price')
    favCar =
dataHolder.values('carModel').annotate(car_count=Count('carModel'))
    favPlace = dataHolder.values_list('placeToStart')
    dataCarList = [val for x in favCar for key, val in x.items()]
    placeData = dict(Counter([x for tup in favPlace for x in tup]))

    for x_id, place in location:
        locationValues.append((place, placeData[x_id]))

    for x in dataHolderPayed:
        for key, value in x.items():
            if key == 'orderDate':
                dataDateMoneyList.append(value.strftime(format))
            if key == 'price':
                dateMoneyList.append(value)

    CombinedData = [(i, j) for i, j in zip(dataDateMoneyList,
dateMoneyList)]
    sumCombindedData = defaultdict(int)
    for key, val in CombinedData:
        sumCombindedData[key] += val

    # dataCarDict={dataCarList[i]: dataCarList[i + 1] for i in
range(0, len(dataCarList), 2)}
    # I don't know why dict itteration doesn't work 9in graph.html
that's why we are doing this this way

    dailyDate = list(sumCombindedData.keys())
    dailyMoney = list(sumCombindedData.values())
    orderedCarsName = dataCarList[::2]
    orderedCarsQuantity = dataCarList[1::2]
    placeName = [x[0] for x in locationValues]
    placeQuantity = [x[1] for x in locationValues]

    context = {
        'current': current,
        'dailyDate': dailyDate,
        'dailyMoney': dailyMoney,
```

```
        'placeName': placeName,
        'placeQuantity': placeQuantity,

        'orderedCarsName': orderedCarsName,
        'orderedCarsQuantity': orderedCarsQuantity,
    }

    return render(request, 'graph.html', context)
```

Αρχείο graph.html

```
{% extends 'main.html' %}
{% load static  %}
{% block content %}
<script>
    $(document).ready(function(){
    var ctx = document.getElementById('myChart').getContext('2d');
    var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: [
            {% for x in orderedCarsName  %}
                '{{ x  }}',
            {% endfor  %}

        ],
        datasets: [{
            label: 'Cars',
            data: [
            {% for x in orderedCarsQuantity  %}
                '{{ x  }}',
            {% endfor  %}

            ],
            backgroundColor: [
                'rgba(255, 99, 132, 0.2)',
                'rgba(54, 162, 235, 0.2)',
                'rgba(255, 206, 86, 0.2)',
                'rgba(75, 192, 192, 0.2)',
                'rgba(153, 102, 255, 0.2)',
                'rgba(255, 159, 64, 0.2)'
            ],
            borderColor: [
                'rgba(255, 99, 132, 1)',
                'rgba(54, 162, 235, 1)',
                'rgba(255, 206, 86, 1)',
                'rgba(75, 192, 192, 1)',
                'rgba(153, 102, 255, 1)',
                'rgba(255, 159, 64, 1)'
            ],
            borderWidth: 1
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
```

```
                    beginAtZero: true
                }
            }]
        }
    }
});


    var income =
document.getElementById('dailyIncome').getContext('2d');
    var dailyIncome = new Chart(income, {
        type: 'line',
        data: {
            labels: [
                {% for x in dailyDate  %}
                    '{{ x  }}',
                {% endfor  %}

            ],
            datasets: [{
                label: 'Daily income',
                data: [
                {% for x in dailyMoney  %}
                    '{{ x  }}',
                {% endfor  %}

                ],
                backgroundColor: [
                    'rgba(255, 99, 132, 0.2)',
                    'rgba(54, 162, 235, 0.2)',
                    'rgba(255, 206, 86, 0.2)',
                    'rgba(75, 192, 192, 0.2)',
                    'rgba(153, 102, 255, 0.2)',
                    'rgba(255, 159, 64, 0.2)'
                ],
                borderColor: [
                    'rgba(255, 99, 132, 1)',
                    'rgba(54, 162, 235, 1)',
                    'rgba(255, 206, 86, 1)',
                    'rgba(75, 192, 192, 1)',
                    'rgba(153, 102, 255, 1)',
                    'rgba(255, 159, 64, 1)'
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }]
            }
        }
    });
```

```
    var location =
document.getElementById('locationData').getContext('2d');
    var locationData = new Chart(location, {
        type: 'polarArea',
        data: {
            labels: [
                {% for x in placeName  %}
                    '{{ x  }}',
                {% endfor  %}

            ],
            datasets: [{
                label: 'Busiest pickup places',
                data: [
                {% for x in placeQuantity  %}
                    '{{ x  }}',
                {% endfor  %}

                ],
                backgroundColor: [
                    'rgba(255, 99, 132, 0.2)',
                    'rgba(54, 162, 235, 0.2)',
                    'rgba(255, 206, 86, 0.2)',
                    'rgba(75, 192, 192, 0.2)',
                    'rgba(153, 102, 255, 0.2)',
                    'rgba(255, 159, 64, 0.2)'
                ],
                borderColor: [
                    'rgba(255, 99, 132, 1)',
                    'rgba(54, 162, 235, 1)',
                    'rgba(255, 206, 86, 1)',
                    'rgba(75, 192, 192, 1)',
                    'rgba(153, 102, 255, 1)',
                    'rgba(255, 159, 64, 1)'
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }]
            }
        }
    });




    });
</script>


<div class="container-fluid over">
    <div class="row">
```

```
        <div class="col-md-4 col-12">
            <canvas id="myChart" width="100%" height="50%"></canvas>
        </div>
        <div class="col-md-4">
            <div class="dashboardTitle">
                Σέρρες Rent A Car - Στατιστικά
            </div>
            <div class="dashboardPic">
                <img class="dashPic" src="{% static 'media/six-rent-
a-car.jpg'  %}">
            </div>
        </div>
        <div class="col-md-4 col-12">
            <canvas id="dailyIncome" width="100%"
height="50%"></canvas>
        </div>
    </div>
     <div class="row">
        <div class="col-md-4 col-12">

        </div>
        <div class="col-md-4 col-12">
            <canvas id="locationData" width="100%"
height="50%"></canvas>
        </div>
        <div class="col-md-4 col-12">

        </div>
    </div>
</div>
{% endblock  %}
```