ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

"Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης"

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Επαγγελματικός Κατάλογος Πώλησης & Ενοικίασης Ακινήτων (Web Εφαρμογή - Real Estate)**
Ονοματεπώνυμο Φοιτητή	**Σιούλας Δημήτριος**
Πατρώνυμο	**Βασίλειος**
Αριθμός Μητρώου	**ΜΠΣΠ20048**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης Νοέμβριος 2023

Τριμελής Εξεταστική Επιτροπή		
Ευθύμιος Αλέπης Αναπληρωτής καθηγητής	**Ευάγγελος Σακκόπουλος** Αναπληρωτής καθηγητής	**Μαρία Βίρβου** Καθηγήτρια

ΒΕΒΑΙΩΣΗ ΕΚΠΟΝΗΣΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

«Δηλώνω υπεύθυνα ότι το έργο που εκπονήθηκε και παρουσιάζεται στην υποβαλλόμενη διπλωματική εργασία, έχει γραφτεί από εμένα αποκλειστικά στο σύνολό της. Δεν έχει υποβληθεί ούτε έχει εγκριθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού προγράμματος ή προπτυχιακού τίτλου σπουδών, ούτε είναι εργασία ή τμήμα εργασίας ακαδημαϊκού ή επαγγελματικού χαρακτήρα. Δηλώνω επίσης ότι αναφέρονται καταλλήλως στο σύνολό τους οι πηγές στις οποίες ανέτρεξα για την εκπόνηση της συγκεκριμένης εργασίας. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου.»

Τίτλος

Επαγγελματικός Κατάλογος Πώλησης & Ενοικίασης Ακινήτων (Web Εφαρμογή - Real Estate)

Σημαντικοί Όροι: Ακίνητα, Σύστημα Διαχείρισης Ακινήτων, Πλατφόρμα Real Estate, Διαδικτυακή Αγορά Ακινήτων, Ψηφιακή Μεσιτεία Ακινήτων, Πώληση και Ενοικίαση Ακινήτων, Ανάπτυξη Λογισμικού Real Estate, Ηλεκτρονική Πλατφόρμα Ακινήτων, Αναζήτηση Ακινήτων, Αγορά Ακινήτων, Πωλήσεις Ακινήτων, Διαδικτυακή Διαχείριση Ακινήτων, Χρήστες και Μεσίτες σε Συστήματα Real Estate, Διεπαφή Χρήστη για Συστήματα Ακινήτων, Τεχνολογίες Ανάπτυξης Ιστοσελίδων Ακινήτων, Σύστημα Πληροφοριών Για Ακίνητα.

Περίληψη

Στο πλαίσιο της παρούσας μεταπτυχιακής διατριβής, παρουσιάζεται η ανάπτυξη μιας πλήρους web εφαρμογής για τον τομέα του real estate, χρησιμοποιώντας το Laravel framework. Η εφαρμογή περιλαμβάνει τις βασικές λειτουργίες ενός συστήματος αναζήτησης ακινήτων για αγορά ή ενοικίαση, παρέχοντας εκτεταμένες πληροφορίες για τα ακίνητα και παρέχοντας έναν πλήρως προσαρμόσιμο πίνακα διαχείρισης για την ανάρτηση νέων ακινήτων και την προσθήκη των προδιαγραφών τους.

Στο πλαίσιο της διατριβής αναλύονται αναλυτικά τα πακέτα του Laravel framework που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, το σχήμα και η αρχιτεκτονική της βάσης δεδομένων. Επιπλέον, αναφέρονται οι τεχνολογίες που ενσωματώθηκαν στην εφαρμογή, προσδίδοντας την λειτουργικότητα διαχείρισης των ακινήτων και των χρηστών αλλά και των σχέσεων μεταξύ τους.

Η διατριβή περιλαμβάνει επίσης ανασκόπηση του πεδίου του real estate και της ευκολίας αναζήτησης ακινήτων που προσφέρει η εφαρμογή, προσφέροντας συγχρόνως μια σημαντική προστιθέμενη αξία στην εμπειρία των χρηστών.

Title

Real Estate Catalogue- Sell or Hire (Web Application - Real Estate)

Key words: Real Estate, Property Management System, Real Estate Platform, Online Real Estate Market, Digital Real Estate Brokerage, Real Estate Sales and Leasing, Real Estate Software Development, Electronic Real Estate Platform, Real Estate Search, Real Estate Purchase, Real Estate Sales, Online Real Estate Management, Users and Agents in Real Estate Systems, User Interface for Real Estate Systems, Web Development Technologies for Real Estate, Real Estate Information System.

Abstract

Within the scope of this postgraduate thesis, the development of a comprehensive web application for the real estate sector using the Laravel framework is presented. The application encompasses the fundamental functions of a real estate search system for buying or renting properties, providing extensive information about the properties, and offering a fully customizable dashboard for posting new properties and adding their specifications.

The thesis extensively covers the Laravel framework packages used for application development, the schema, and the database architecture. Furthermore, it discusses the technologies integrated into the application, enabling features such as hanling properties and users through the admin dashboard.

The thesis also includes an overview of the real estate field and the ease of property search provided by the application, simultaneously offering significant added value to the user experience.

Περιεχόμενα

Περιεχόμενα **1**
Εισαγωγή **2**
Σημασία της διατριβής **2**
Ανασκόπηση πεδίου **2**
Σύντομη περιγραφή συστήματος **3**
Χρήστες Αναζήτησης Ακινήτων **3**
Πράκτορες Ακινήτων **3**
Διαχειριστές **3**
Αρχιτεκτονική Βάσης Δεδομένων **5**
Αρχιτεκτονική Συστήματος **16**
Design Patterns **16**
MVC Design Pattern 16
Dependency Injection Design Pattern 17
Facade Pattern 18
Factory Pattern 18
Repository Pattern 19
Πακέτα Laravel **20**
Laravel Breeze **20**
Laravel Dompdf **20**
Laravel Telescope **20**
Περιγραφή Συστήματος **22**
Από την σκοπιά του απλού χρήστη **22**
Εγγεγραμμένος Χρήστης **28**
Πράκτορας/agent **32**
Διαχειριστής **36**
Τεχνολογίες που χρησιμοποιήθηκαν **40**
Γιατί επιλέχτηκε ως βασικό framework της εφαρμογής το laravel ... **40**
Ευκολία Ανάπτυξης **40**
Ενσωματωμένες Λειτουργίες Ασφαλείας **40**
Προσαρμοσμένη Διαχείριση Βάσεων Δεδομένων **40**
Ευελιξία και Επεκτασιμότητα **40**
Κοινότητα και Υποστήριξη **40**
Μελλοντικές Επεκτάσεις της Web Εφαρμογής **42**
1 Εικονικές Περιηγήσεις: **42**
2 Ενσωμάτωση Τεχνητής Νοημοσύνης: **42**
3 Χάρτης με Σημεία Ενδιαφέροντος: **42**
4 Φίλτρα Αναζήτησης Βάσει Περιοχής: **42**
5 Ενσωμάτωση AR (Augmented Reality): **42**
6 Αξιολογήσεις Και Σχόλια: **43**
7 Προσωπικό Dashboard για Πωλητές: **43**
8 Ενσωμάτωση Chatbot: **43**
Βιβλιογραφία **1**

Εισαγωγή

Στη σύγχρονη εποχή, όπου η τεχνολογία διεισδύει σε κάθε πτυχή της καθημερινότητάς μας, οι εφαρμογές real estate αποτελούν ένα από τα πλέον αναπόσπαστα εργαλεία τόσο για τους επαγγελματίες του κλάδου όσο και για τους καταναλωτές. Προσφέροντας μια γκάμα λειτουργιών και δυνατοτήτων, αυτές οι εφαρμογές έχουν επαναπροσδιορίσει τον τρόπο με τον οποίο αντιλαμβανόμαστε την αγορά ακινήτων.

Καταρχάς, οι εφαρμογές real estate προσφέρουν τη δυνατότητα στους χρήστες να περιηγούνται σε χιλιάδες ακίνητα με το πάτημα ενός κουμπιού. Αυτό εξοικονομεί χρόνο και προσπάθεια, επιτρέποντας στους υποψήφιους αγοραστές ή ενοικιαστές να διαλέξουν αυτό που ταιριάζει καλύτερα στις ανάγκες τους χωρίς την ανάγκη να επισκεφτούν φυσικά τον χώρο.

Επιπλέον, με τη βοήθεια προηγμένων αλγορίθμων και τεχνητής νοημοσύνης, οι εφαρμογές αυτές μπορούν να προβλέπουν τις προτιμήσεις των χρηστών, προσφέροντας προτάσεις που είναι περισσότερο σχετικές με τις ανάγκες και τα ενδιαφέροντά τους. Αυτό καθιστά τη διαδικασία αναζήτησης ακινήτων ακόμη πιο εξατομικευμένη και αποτελεσματική.

Επίσης, η δυνατότητα για άμεση επικοινωνία με τους πωλητές ή τους μεσίτες μέσω των εφαρμογών αυτών απλοποιεί τη διαδικασία και καθιστά την επικοινωνία πιο διαφανή.
Τέλος, οι εφαρμογές real estate είναι ένα αναπόσπαστο εργαλείο για τους επαγγελματίες του κλάδου, καθώς τους παρέχουν πρόσβαση σε στατιστικά στοιχεία, αναλύσεις της αγοράς και τάσεις, βοηθώντας τους να προσαρμόζονται καλύτερα στις ανάγκες της αγοράς.

Συνολικά, οι εφαρμογές real estate αποτελούν ένα απαραίτητο εργαλείο στον σύγχρονο κόσμο των ακινήτων, προσφέροντας τόσο στους επαγγελματίες όσο και στους χρήστες τη δυνατότητα για μια πιο εξατομικευμένη, αποτελεσματική και άνετη εμπειρία.

Σημασία της διατριβής

Στη σύγχρονη εποχή, η αγορά ακινήτων έχει γίνει πολύ ανταγωνιστική, και οι άνθρωποι ψάχνουν για εύκολους και αποτελεσματικούς τρόπους να βρουν ακίνητα που ταιριάζουν στις ανάγκες τους. Ένα web app για το real estate προσφέρει ευκολία πρόσβασης σε πληροφορίες και δυνατότητα αναζήτησης ακινήτων με βάση διάφορα κριτήρια βασιζόμενα στα χαρακτηριστικά των ακινήτων ώστε οι χρήστες μέσω της αναζήτησης να βρουν ευκολότερα τα επιθυμητά αποτελέσματα.

Η διαδικασία αναζήτησης ακινήτων μπορεί να είναι χρονοβόρα. Ένα εύχρηστο web app μπορεί να εξοικονομήσει χρόνο στους χρήστες, καθώς τους επιτρέπει να προβαίνουν σε αναζητήσεις και επικοινωνία με πράκτορες διαχειριστές ακινήτων από την ευκολία που τους προσφέρει η χρήση μιας web εφαρμογής.

Οι πράκτορες ακινήτων επίσης ωφελούνται από το σύστημά, καθώς τους παρέχει ένα χρήσιμο εργαλείο για τη διαχείριση των ακινήτων τους και την επικοινωνία με πιθανούς μελλοντικούς πελάτες. Αυτό τους επιτρέπει να αυξήσουν την προβολή των ακινήτων τους και να βρουν ευκολότερα πιθανούς αγοραστές.

Οι χρήστες μπορούν να εξατομικεύσουν τις αναζητήσεις τους και να αποθηκεύουν αγαπημένα ακίνητα, βοηθώντας τους να βρίσκουν τα ακίνητα που ανταποκρίνονται ακριβώς στις προτιμήσεις τους μέσω του συστήματος της web εφαρμογής.

Το σύστημά επιτρέπει στους διαχειριστές να παρακολουθούν στατιστικά στοιχεία χρήσης της εφαρμογής. Αυτή η πληροφορία μπορεί να χρησιμοποιηθεί για τη λήψη αποφάσεων και τη βελτίωση της εμπειρίας των χρηστών καθώς και για μελλοντικές επεκτάσεις για ενδεχόμενα χρήσιμα features της εφαρμογής.

Συνοψίζοντας, η συγκεκριμένη web εφαρμογή προσφέρει ένα εργαλείο που εξυπηρετεί τόσο τους αναζητητές ακινήτων όσο και τους πράκτορες, επιφέροντας ευκολία, εξοικονόμηση χρόνου και βελτίωση στην αγορά ή ενοικίαση ακινήτων.

Ανασκόπηση πεδίου

Στον τομέα του real estate αναδεικνύονται συνεχώς ψηφιακές πλατφόρμες, που επιτρέπουν ώστε οι διαδικασίες ανάρτησης αγγελιών, αναζήτησης ακινήτων και ολοκλήρωσης της αγοράς/ενοικίασης ακινήτου, να γίνονται σχεδόν εξ ολοκλήρου με ψηφιακό τρόπο.
Ορισμένα από τα λειτουργικά χαρακτηριστικά που θα πρέπει να πληρούν τέτοιες πλατφόρμες αναφέρονται στη συνέχεια:

- Layout και ξεκάθαρη πλοήγηση ιστοσελίδας
- Responsive ιστοσελίδα και Cross-Browser υποστήριξη
- Διασύνδεση ιστοσελίδας με Social Media
- Σύνθετη Λειτουργία Αναζήτησης Ακινήτων
- Λειτουργία Αποθήκευσης Αναζητήσεων
- Λειτουργία σύγκρισης ακινήτων και στατιστικών
- Σελίδα παρουσίασης ακινήτου
- Περιήγηση στο ακίνητο με τη χρήση εικονικής πραγματικότητας
- κλπ.

Ορισμένες από τις πλέον γνωστές τέτοιες πλατφόρμες είναι:
www.spitogatos.gr
www.prosperty.com
www.realestateonline.gr
κλπ.

Σύντομη περιγραφή συστήματος

Το πεδίο της παρούσας διπλωματικής εργασίας εστιάζει στην ανάπτυξη ενός web application για την αναζήτηση για αγορά και ενοικίαση ακινήτων. Το σύστημα αυτό προορίζεται για τρεις βασικές κατηγορίες χρηστών:

Χρήστες Αναζήτησης Ακινήτων

Οι χρήστες αυτοί μπορούν να δημιουργούν προσωπικούς λογαριασμούς, να αποθηκεύουν αγαπημένα τους ακίνητα, να επικοινωνούν με πράκτορες για ακίνητα που τους ενδιαφέρουν, και να ζητούν επισκέψεις σε ακίνητα μέσω της ειδικά σχεδιασμένης φόρμας επικοινωνίας, επιλέγοντας επιθυμητή ημερομηνία και ώρα. Οι χρήστες που ενδιαφέρονται για εύρεση του ακινήτου που εξυπηρετεί καλύτερα της ανάγκες τους έχουν τη δυνατότητα να χρησιμοποιούν φόρμες αναζήτησης που παρέχονται από το σύστημα, ώστε να φιλτράρουν τις επιλογές τους βάσει διάφορων κριτηρίων όπως η τοποθεσία του ακινήτου, η διαθεσιμότητα ενός ακινήτου για αγορά ή ενοικίαση, καθώς και άλλα χαρακτηριστικά όπως οι λεπτομερής παροχές που προσφέρει το ακίνητο.

Πράκτορες Ακινήτων

Οι πράκτορες/διαχειριστές των ακινήτων διαθέτουν δική τους διαχειριστική διεπαφή (πίνακα ελέγχου) για την προσθήκη και διαχείριση των ακινήτων που διαθέτουν προς ενοικίαση ή πώληση, με την δυνατότητα ανταπόκρισης σε αιτήματα και μηνύματα χρηστών. Επιπλέον, οι πράκτορες ακινήτων μπορούν να καταχωρούν τα ακίνητα που διαχειρίζονται μέσω ειδικών φορμών που παρέχονται από το σύστημα. Οι φόρμες αυτές επιτρέπουν την καταχώρηση πληροφοριών για κάθε ακίνητο με πολλές λεπτομέρειες για τα χαρακτηριστικά του, όπως η τοποθεσία, το μέγεθος, οι παροχές, φωτογραφίες, ο αριθμός των δωματίων και άλλα σχετικά στοιχεία, ώστε να παρέχεται μια πλήρης εικόνα του ακινήτου στους πιθανούς ενδιαφερόμενους επισκέπτες της εφαρμογής.

Διαχειριστές

Οι διαχειριστές έχουν την επιπλέον δυνατότητα μέσω του ειδικά διαμορφωμένου για διαχειριστές πίνακα ελέγχου της εφαρμογής να ενεργοποιούν ή απενεργοποιούν ακίνητα και πράκτορες, και να παρακολουθούν τα στατιστικά στοιχεία του συστήματος μέσω της δικής τους διαχειριστικής διεπαφής έτσι ώστε να έχουν ολοκληρωμένη εικόνα της χρήσης του συστήματος.

Το web app προσφέρει εξατομικευμένες λειτουργίες για κάθε κατηγορία χρηστών, διευκολύνοντας την αναζήτηση και διαχείριση ακινήτων, την επικοινωνία μεταξύ χρηστών και πρακτόρων, καθώς και την γενικότερη διαχείριση και παρακολούθηση της πλατφόρμας από τους διαχειριστές. Με αυτόν τον τρόπο, το σύστημα αποτελεί ένα ολοκληρωμένο ψηφιακό περιβάλλον για την

αναζήτηση και εύρεση ακινήτων για αγορά και ενοικίαση, προσφέροντας μια πλήρη και ευέλικτη λύση στους εμπλεκόμενους στην αγορά ακινήτων.

Αρχιτεκτονική Βάσης Δεδομένων

Για την υλοποίηση της βάσης δεδομένων που χρησιμοποιείται στο σύστημά μας, επιλέξαμε τη MariaDB, μια αξιόπιστη σχεσιακή βάση δεδομένων με την έκδοση 10.3.38. Η MariaDB αποτελεί μια ανοικτού κώδικα λύση που προσφέρει υψηλή απόδοση, αξιοπιστία και ευελιξία στην αποθήκευση και διαχείριση των δεδομένων μας.

Οι κύριοι λόγοι για την επιλογή της MariaDB περιλαμβάνουν:

Σχεσιακό Μοντέλο: Η MariaDB βασίζεται στο σχεσιακό μοντέλο των βάσεων δεδομένων, το οποίο είναι ιδανικό για την αποθήκευση συνδέσεων μεταξύ διαφορετικών τύπων δεδομένων, όπως οι πληροφορίες των χρηστών, τα ακίνητα και οι πράκτορες.

Απόδοση: Η MariaDB προσφέρει υψηλή απόδοση και δυνατότητες βελτιστοποίησης των αιτημάτων SQL, επιτρέποντάς μας να ανταποκριθούμε αποτελεσματικά στις απαιτήσεις του συστήματός μας.

Ασφάλεια: Η MariaDB προσφέρει προηγμένες λειτουργίες ασφαλείας όπως δικαιώματα χρηστών και κρυπτογράφηση, εξασφαλίζοντας την ασφάλεια των δεδομένων μας.

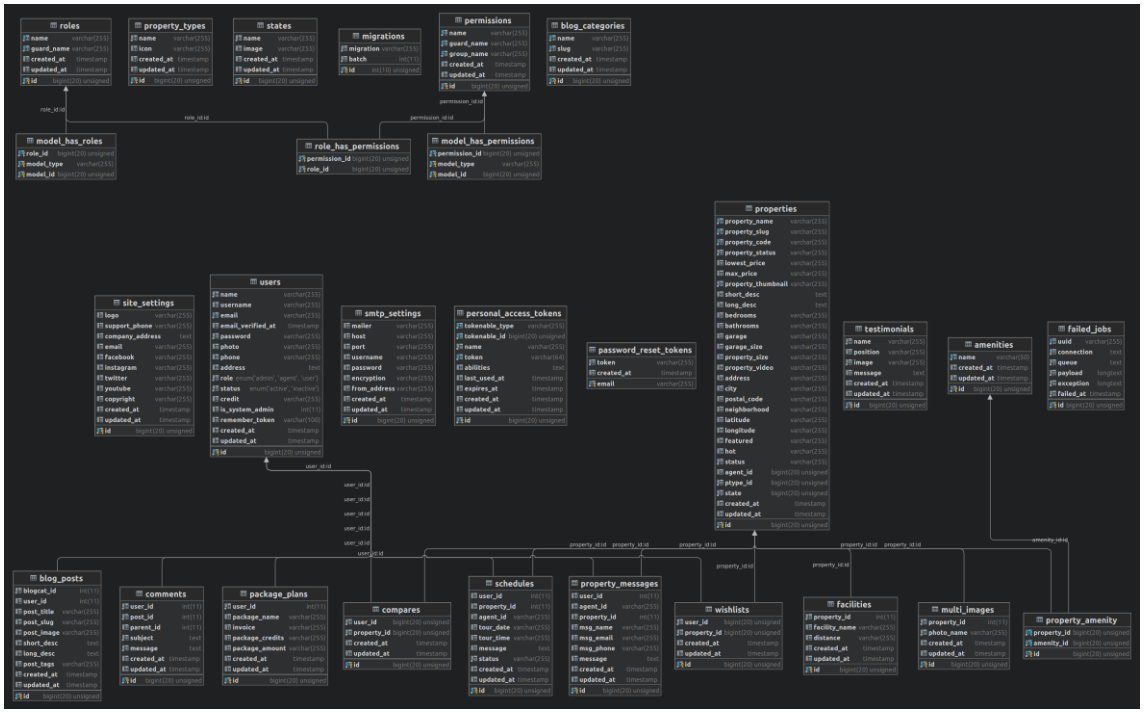
Ανοικτός Κώδικας: Ως ανοικτός κώδικας λογισμικό, η MariaDB είναι ελεύθερα διαθέσιμη και υποστηρίζεται από μια ενεργή κοινότητα προγραμματιστών.

Η MariaDB συνέβαλε καθοριστικά στην αξιοπιστία και απόδοση του συστήματός μας. Η συγκεκριμένη έκδοση 10.3.38 παρέχει τις τελευταίες βελτιώσεις και δυνατότητες, εξασφαλίζοντας ότι η βάση δεδομένων λειτουργεί απροβλημάτιστα και ανταποκρίνεται στις ανάγκες του συστήματος μας.

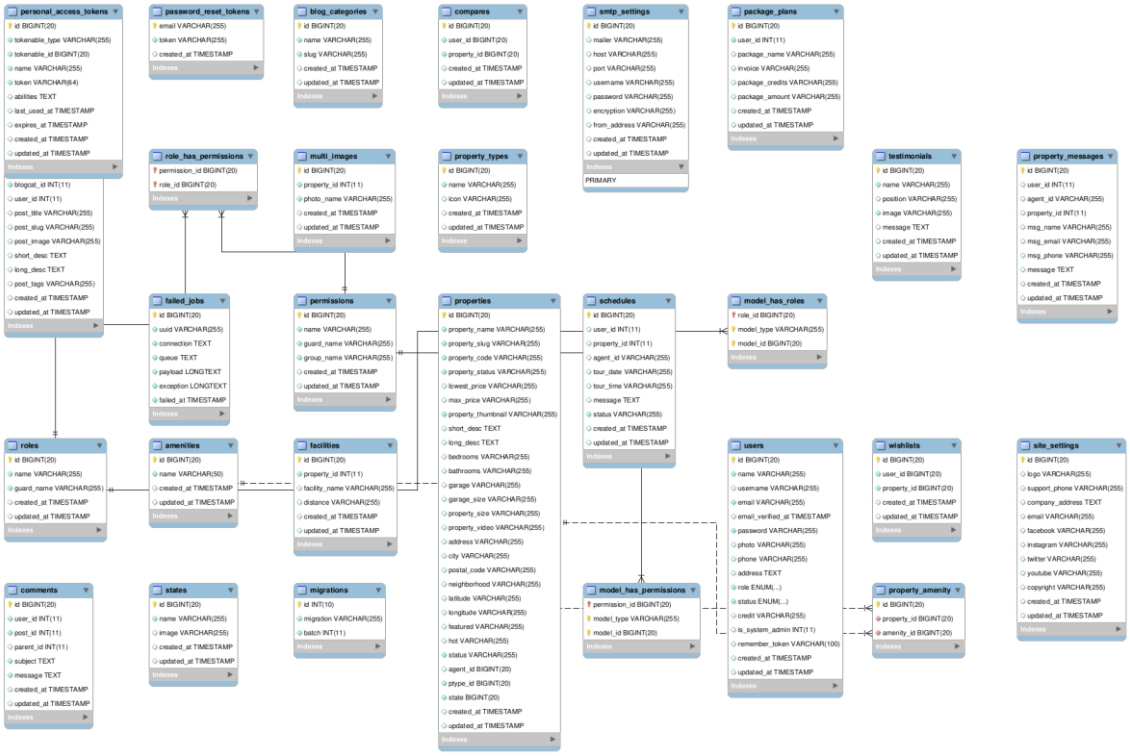
Συνολικά, η MariaDB αποτελεί μια ιδανική επιλογή για τη βάση δεδομένων μας και συνέβαλε σημαντικά στην επιτυχία και τη λειτουργία του συστήματός μας.

Στο σύστημα μας, εφαρμόσαμε το Eloquent ORM του Laravel για τη διαχείριση της βάσης δεδομένων. Το Object-Relational Mapping (ORM) είναι μια τεχνολογία που μας επέτρεψε να δημιουργήσουμε ένα πιο αντικειμενοστραφές και δομημένο σύστημα, επιτρέποντάς μας να χρησιμοποιήσουμε αντικείμενα και μεθόδους για τη διαχείριση και αλληλεπίδραση με τη βάση δεδομένων, αντί να αναγκαστούμε να γράψουμε SQL ερωτήματα χειροκίνητα.

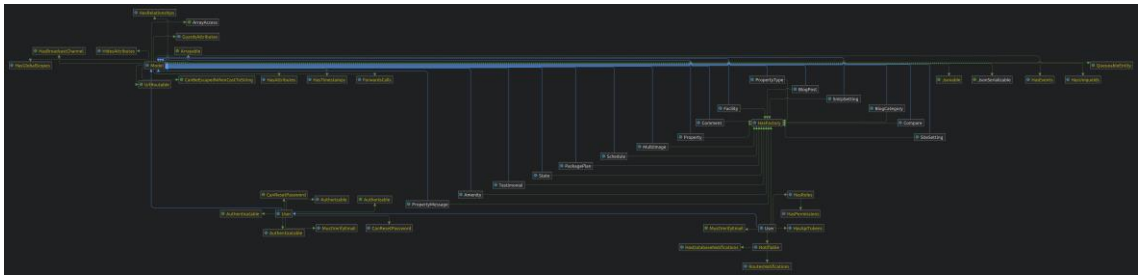
Χρησιμοποιώντας το Eloquent ORM, μπορέσαμε να αναπτύξουμε μια πιο δομημένη και αντικειμενοστραφή προσέγγιση για τη διαχείριση των δεδομένων μας. Τα Eloquent models αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων και παρέχουν έναν βολικό τρόπο για τη διαχείριση, ανάκτηση, ενημέρωση και διαγραφή δεδομένων. Αυτό το ORM μας βοήθησε να επικεντρωθούμε στην ανάπτυξη της λογικής της εφαρμογής μας αντί να ασχοληθούμε με τις λεπτομέρειες της διαχείρισης της βάσης δεδομένων.



Εικόνα 1:Σχήμα Οντοτήτων



Εικόνα 2:Σχήμα σχεσιακής βάσης



Εικόνα 3: Σχήμα Laravel

```
create table amenities
(
    id         bigint unsigned auto_increment
        primary key,
    name       varchar(50) not null,
    created_at timestamp   null,
    updated_at timestamp   null,
    constraint amenities_name_unique
        unique (name)
)
    collate = utf8mb4_unicode_ci;

create table blog_categories
(
    id         bigint unsigned auto_increment
        primary key,
    name       varchar(255) not null,
    slug       varchar(255) not null,
    created_at timestamp    null,
    updated_at timestamp    null,
    constraint blog_categories_name_unique
        unique (name),
    constraint blog_categories_slug_unique
        unique (slug)
)
    collate = utf8mb4_unicode_ci;

create table blog_posts
(
    id         bigint unsigned auto_increment
        primary key,
    blogcat_id int          not null,
    user_id    int          null,
    post_title varchar(255) null,
    post_slug  varchar(255) null,
    post_image varchar(255) null,
    short_desc text         null,
    long_desc  text         null,
    post_tags  varchar(255) null,
    created_at timestamp    null,
    updated_at timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table comments
(
    id         bigint unsigned auto_increment
        primary key,
```

```
    user_id    int       not null,
    post_id    int       not null,
    parent_id  int       null,
    subject    text      not null,
    message    text      not null,
    created_at timestamp null,
    updated_at timestamp null
)
    collate = utf8mb4_unicode_ci;

create table compares
(
    id         bigint unsigned auto_increment
        primary key,
    user_id    bigint unsigned not null,
    property_id bigint unsigned not null,
    created_at  timestamp      null,
    updated_at  timestamp      null
)
    collate = utf8mb4_unicode_ci;

create table facilities
(
    id           bigint unsigned auto_increment
        primary key,
    property_id  int          not null,
    facility_name varchar(255) null,
    distance     varchar(255) null,
    created_at   timestamp    null,
    updated_at   timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table failed_jobs
(
    id         bigint unsigned auto_increment
        primary key,
    uuid       varchar(255)                        not null,
    connection text                                not null,
    queue      text                                not null,
    payload    longtext                            not null,
    exception  longtext                            not null,
    failed_at  timestamp default current_timestamp() not null,
    constraint failed_jobs_uuid_unique
        unique (uuid)
)
    collate = utf8mb4_unicode_ci;

create table migrations
(
    id        int unsigned auto_increment
        primary key,
    migration varchar(255) not null,
    batch     int          not null
)
    collate = utf8mb4_unicode_ci;

create table multi_images
(
    id         bigint unsigned auto_increment
```

```
        primary key,
    property_id int          not null,
    photo_name  varchar(255) not null,
    created_at  timestamp    null,
    updated_at  timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table package_plans
(
    id             bigint unsigned auto_increment
        primary key,
    user_id        int          not null,
    package_name   varchar(255) null,
    invoice        varchar(255) null,
    package_credits varchar(255) null,
    package_amount  varchar(255) null,
    created_at     timestamp    null,
    updated_at     timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table password_reset_tokens
(
    email      varchar(255) not null
        primary key,
    token      varchar(255) not null,
    created_at timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table permissions
(
    id         bigint unsigned auto_increment
        primary key,
    name       varchar(255) not null,
    guard_name varchar(255) not null,
    group_name varchar(255) not null,
    created_at timestamp    null,
    updated_at timestamp    null,
    constraint permissions_name_guard_name_group_name_unique
        unique (name, guard_name, group_name)
)
    collate = utf8mb4_unicode_ci;

create table model_has_permissions
(
    permission_id bigint unsigned not null,
    model_type    varchar(255)    not null,
    model_id      bigint unsigned not null,
    primary key (permission_id, model_id, model_type),
    constraint model_has_permissions_permission_id_foreign
        foreign key (permission_id) references permissions (id)
            on delete cascade
)
    collate = utf8mb4_unicode_ci;

create index model_has_permissions_model_id_model_type_index
    on model_has_permissions (model_id, model_type);
```

```
create table personal_access_tokens
(
    id             bigint unsigned auto_increment
        primary key,
    tokenable_type varchar(255)    not null,
    tokenable_id   bigint unsigned not null,
    name           varchar(255)    not null,
    token          varchar(64)     not null,
    abilities      text            null,
    last_used_at   timestamp       null,
    expires_at     timestamp       null,
    created_at     timestamp       null,
    updated_at     timestamp       null,
    constraint personal_access_tokens_token_unique
        unique (token)
)
    collate = utf8mb4_unicode_ci;

create index personal_access_tokens_tokenable_type_tokenable_id_index
    on personal_access_tokens (tokenable_type, tokenable_id);

create table properties
(
    id                bigint unsigned auto_increment
        primary key,
    property_name     varchar(255)            not null,
    property_slug     varchar(255)            not null,
    property_code     varchar(255)            not null,
    property_status   varchar(255)            not null,
    lowest_price      varchar(255)            null,
    max_price         varchar(255)            null,
    property_thumbnail varchar(255)           not null,
    short_desc        text                    null,
    long_desc         text                    null,
    bedrooms          varchar(255)            null,
    bathrooms         varchar(255)            null,
    garage            varchar(255)            null,
    garage_size       varchar(255)            null,
    property_size     varchar(255)            null,
    property_video    varchar(255)            null,
    address           varchar(255)            null,
    city              varchar(255)            null,
    postal_code       varchar(255)            null,
    neighborhood      varchar(255)            null,
    latitude          varchar(255)            null,
    longitude         varchar(255)            null,
    featured          varchar(255)            null,
    hot               varchar(255)            null,
    status            varchar(255) default '0' not null,
    agent_id          bigint unsigned         null,
    ptype_id          bigint unsigned         not null,
    state             bigint unsigned         not null,
    created_at        timestamp               null,
    updated_at        timestamp               null,
    constraint properties_property_code_unique
        unique (property_code),
    constraint properties_property_name_unique
        unique (property_name),
    constraint properties_property_slug_unique
        unique (property_slug)
```

```
)
   collate = utf8mb4_unicode_ci;

create table property_amenity
(
   id          bigint unsigned auto_increment
      primary key,
   property_id bigint unsigned not null,
   amenity_id  bigint unsigned not null,
   constraint property_amenity_property_id_amenity_id_unique
      unique (property_id, amenity_id),
   constraint property_amenity_amenity_id_foreign
      foreign key (amenity_id) references amenities (id)
         on delete cascade,
   constraint property_amenity_property_id_foreign
      foreign key (property_id) references properties (id)
         on delete cascade
)
   collate = utf8mb4_unicode_ci;

create table property_messages
(
   id          bigint unsigned auto_increment
      primary key,
   user_id     int          null,
   agent_id    varchar(255) null,
   property_id int          null,
   msg_name    varchar(255) null,
   msg_email   varchar(255) null,
   msg_phone   varchar(255) null,
   message     text         null,
   created_at  timestamp    null,
   updated_at  timestamp    null
)
   collate = utf8mb4_unicode_ci;

create table property_types
(
   id         bigint unsigned auto_increment
      primary key,
   name       varchar(255) not null,
   icon       varchar(255) null,
   created_at timestamp    null,
   updated_at timestamp    null
)
   collate = utf8mb4_unicode_ci;

create table roles
(
   id         bigint unsigned auto_increment
      primary key,
   name       varchar(255) not null,
   guard_name varchar(255) not null,
   created_at timestamp    null,
   updated_at timestamp    null,
   constraint roles_name_guard_name_unique
      unique (name, guard_name)
)
   collate = utf8mb4_unicode_ci;
```

```
create table model_has_roles
(
   role_id    bigint unsigned not null,
   model_type varchar(255)    not null,
   model_id   bigint unsigned not null,
   primary key (role_id, model_id, model_type),
   constraint model_has_roles_role_id_foreign
      foreign key (role_id) references roles (id)
         on delete cascade
)
   collate = utf8mb4_unicode_ci;

create index model_has_roles_model_id_model_type_index
   on model_has_roles (model_id, model_type);

create table role_has_permissions
(
   permission_id bigint unsigned not null,
   role_id       bigint unsigned not null,
   primary key (permission_id, role_id),
   constraint role_has_permissions_permission_id_foreign
      foreign key (permission_id) references permissions (id)
         on delete cascade,
   constraint role_has_permissions_role_id_foreign
      foreign key (role_id) references roles (id)
         on delete cascade
)
   collate = utf8mb4_unicode_ci;

create table schedules
(
   id          bigint unsigned auto_increment
      primary key,
   user_id     int                    null,
   property_id int                    null,
   agent_id    varchar(255)           null,
   tour_date   varchar(255)           null,
   tour_time   varchar(255)           null,
   message     text                   null,
   status      varchar(255) default '0' not null,
   created_at  timestamp              null,
   updated_at  timestamp              null
)
   collate = utf8mb4_unicode_ci;

create table site_settings
(
   id              bigint unsigned auto_increment
      primary key,
   logo            varchar(255) null,
   support_phone   varchar(255) null,
   company_address text         null,
   email           varchar(255) null,
   facebook        varchar(255) null,
   instagram       varchar(255) null,
   twitter         varchar(255) null,
   youtube         varchar(255) null,
   copyright       varchar(255) null,
   created_at      timestamp    null,
   updated_at      timestamp    null
```

```
)
   collate = utf8mb4_unicode_ci;

create table smtp_settings
(
   id          bigint unsigned auto_increment
      primary key,
   mailer      varchar(255) null,
   host        varchar(255) null,
   port        varchar(255) null,
   username    varchar(255) null,
   password    varchar(255) null,
   encryption  varchar(255) null,
   from_address varchar(255) null,
   created_at  timestamp    null,
   updated_at  timestamp    null
)
   collate = utf8mb4_unicode_ci;

create table states
(
   id        bigint unsigned auto_increment
      primary key,
   name      varchar(255) not null,
   image     varchar(255) null,
   created_at timestamp    null,
   updated_at timestamp    null
)
   collate = utf8mb4_unicode_ci;

create table telescope_entries
(
   sequence              bigint unsigned auto_increment
      primary key,
   uuid                  char(36)            not null,
   batch_id              char(36)            not null,
   family_hash           varchar(255)        null,
   should_display_on_index tinyint(1) default 1 not null,
   type                  varchar(20)         not null,
   content               longtext            not null,
   created_at            datetime            null,
   constraint telescope_entries_uuid_unique
      unique (uuid)
)
   collate = utf8mb4_unicode_ci;

create index telescope_entries_batch_id_index
   on telescope_entries (batch_id);

create index telescope_entries_created_at_index
   on telescope_entries (created_at);

create index telescope_entries_family_hash_index
   on telescope_entries (family_hash);

create index telescope_entries_type_should_display_on_index_index
   on telescope_entries (type, should_display_on_index);

create table telescope_entries_tags
(
```

```
    entry_uuid char(36)     not null,
    tag        varchar(255) not null,
    constraint telescope_entries_tags_entry_uuid_foreign
        foreign key (entry_uuid) references telescope_entries (uuid)
            on delete cascade
)
    collate = utf8mb4_unicode_ci;

create index telescope_entries_tags_entry_uuid_tag_index
    on telescope_entries_tags (entry_uuid, tag);

create index telescope_entries_tags_tag_index
    on telescope_entries_tags (tag);

create table telescope_monitoring
(
    tag varchar(255) not null
)
    collate = utf8mb4_unicode_ci;

create table testimonials
(
    id         bigint unsigned auto_increment
        primary key,
    name       varchar(255) not null,
    position   varchar(255) null,
    image      varchar(255) not null,
    message    text         null,
    created_at timestamp    null,
    updated_at timestamp    null
)
    collate = utf8mb4_unicode_ci;

create table users
(
    id               bigint unsigned auto_increment
        primary key,
    name             varchar(255)                                  not null,
    username         varchar(255)                                  null,
    email            varchar(255)                                  not null,
    email_verified_at timestamp                                    null,
    password         varchar(255)                                  not null,
    photo            varchar(255)                                  null,
    phone            varchar(255)                                  null,
    address          text                                          null,
    role             enum ('admin', 'agent', 'user') default 'user'     not null,
    status           enum ('active', 'inactive')     default 'inactive' not null,
    credit           varchar(255)                    default '0'        null,
    is_system_admin  int                             default 0          null,
    remember_token   varchar(100)                                  null,
    created_at       timestamp                                     null,
    updated_at       timestamp                                     null,
    constraint users_email_unique
        unique (email)
)
    collate = utf8mb4_unicode_ci;

create table wishlists
(
    id         bigint unsigned auto_increment
```

```
        primary key,
    user_id     bigint unsigned not null,
    property_id bigint unsigned not null,
    created_at  timestamp       null,
    updated_at  timestamp       null
)
    collate = utf8mb4_unicode_ci;
```

Αρχιτεκτονική Συστήματος

Design Patterns

MVC Design Pattern

Το σύστημα βασίζεται στη δομή του Laravel MVC (Model-View-Controller). Αυτή η αρχιτεκτονική επιτρέπει τον διαχωρισμό του συστήματος σε τρία βασικά επίπεδα: το μοντέλο (Model), τη θέα (View) και τον ελεγκτή (Controller).

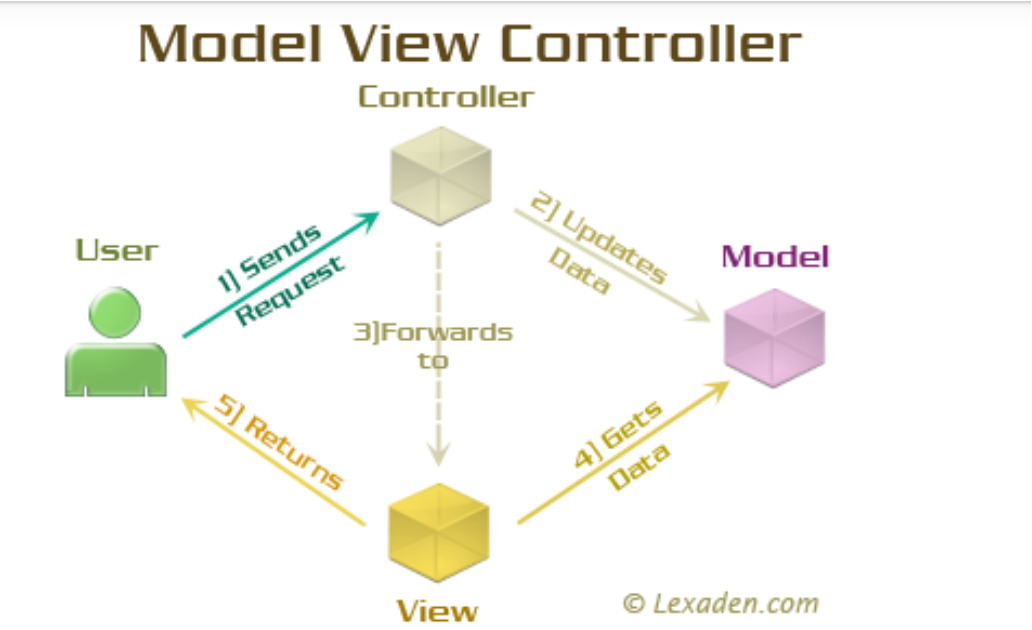
Ας αναλύσουμε την αρχιτεκτονική του συστήματος:

Μοντέλο (Model): Το μοντέλο αντιπροσωπεύει τη δομή των δεδομένων και τη λογική της εφαρμογής. Στην περίπτωσή μας, το μοντέλο απεικονίζει τα δεδομένα των ακινήτων και τους χρήστες αλλά και τις σχέσεις μεταξύ τους, και προσφέρει λειτουργίες για την ανάκτηση, ενημέρωση και διαγραφή δεδομένων από τη βάση δεδομένων MariaDB.

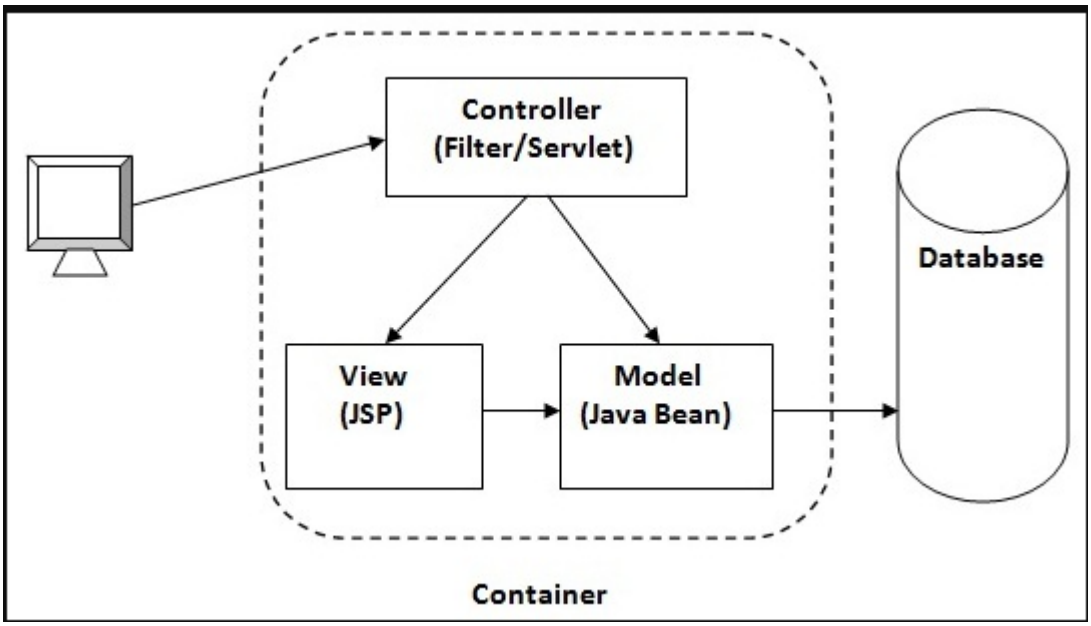
Ελεγκτής (Controller): Ο controller αναλαμβάνει τον ρόλο του ενδιάμεσου στοιχείου μεταξύ του μοντέλου και του view. Στο Laravel, οι ελεγκτές διαχειρίζονται τις διάφορες αιτήσεις από τους χρήστες. Για παράδειγμα, όταν ένας χρήστης κάνει αναζήτηση για ακίνητα, ο ελεγκτής δημιουργεί την κατάλληλη ερώτηση προς το μοντέλο, λαμβάνει τα αποτελέσματα και τα μεταφέρει στο view.

Θέα (View): Το view αφορά την παρουσίαση των δεδομένων στους χρήστες και τη διαδραστική αλληλεπίδραση μαζί τους. Στο συγκεκριμένο σύστημα, η θέα αναλαμβάνει τον ρόλο της προβολής των ακινήτων και των διάφορων σελίδων στους χρήστες. Συχνά χρησιμοποιούμε το Blade Template Engine του Laravel για τη δημιουργία δυναμικών σελίδων.

Το MVC επιτρέπει τον διαχωρισμό της λογικής της εφαρμογής από την παρουσίαση των δεδομένων. Αυτό έχει πολλά πλεονεκτήματα, όπως την ευκολία στη συντήρηση, την επαναχρησιμοποίηση του κώδικα και τη διαχείριση των διάφορων στοιχείων της εφαρμογής μεμονωμένα. Αυτή η δομή επιτρέπει επίσης στην εφαρμογή να είναι ευέλικτη και επεκτάσιμη, καθώς μπορεί να προστεθούν νέα μοντέλα, ελεγκτές και θέες με σχετική ευκολία.



Εικόνα 4: Απεικόνιση MVC



Εικόνα 5: Υλοποίηση MVC

Dependency Injection Design Pattern

Η χρήση της ενσωμάτωσης εξαρτήσεων (Dependency Injection - DI) στο Laravel αποτελεί κομβικό στοιχείο της αρχιτεκτονικής του και παίζει κρίσιμο ρόλο στην ορθή λειτουργία και την διαχείριση των εξαρτήσεων της εφαρμογής. Το Dependency Injection αντιπροσωπεύει έναν από τους σημαντικότερους σχεδιασμούς που υιοθετεί το Laravel, παρέχοντας τα ακόλουθα πλεονεκτήματα:

Διαχωρισμός των αναθέσεων (Separation of Concerns): Χάρη στο DI, ο κώδικας χωρίζεται σε μικρότερα, αυτόνομα τμήματα. Αυτό επιτρέπει στους προγραμματιστές να

αναπτύσσουν, να διαχειρίζονται και να δοκιμάζουν κάθε μέρος της εφαρμογής χωριστά, καθιστώντας τον κώδικα πιο διαχειρίσιμο.

Επαναχρησιμοποίηση του Κώδικα: Η χρήση του DI συμβάλλει στη δημιουργία επαναχρησιμοποιήσιμου κώδικα. Έτσι παρέχεται η δυνατότητα να ενσωματωθούν εξαρτήσεις σε διάφορα μέρη της εφαρμογής και να τις ανακαλύπτετε εύκολα, αντικαθιστώντας τις όταν χρειαστεί.

Ευκολία Μονάδων Δοκιμών (Unit Testing): Ο DI επιτρέπει την εισαγωγή ψευδών αντικειμένων κατά τη διάρκεια των δοκιμών, επιτρέποντας τη δοκιμή κάθε μονάδας κώδικα χωρίς την ανάγκη να αλλάζετε τον πραγματικό κώδικα της εφαρμογής.

Επιδόσεις: Η χρήση DI μπορεί να βελτιώσει τις επιδόσεις, αφού επιτρέπει τη φόρτωση εξαρτήσεων μόνο όταν χρειάζονται, μειώνοντας τον πόρο που καταναλώνεται.

Απλότητα: Ο DI βοηθά στην απλοποίηση του κώδικα, αφού επιτρέπει την ενσωμάτωση των εξαρτήσεων χωρίς να ανησυχεί

Facade Pattern

Στο πλαίσιο της ανάπτυξης του συστήματος, χρησιμοποιήθηκε το Facade Pattern για τη δημιουργία μιας απλής και κατανοητής διεπαφής που παρέχει πρόσβαση σε πολυπλοκότερες λειτουργίες του συστήματος. Αναλύουμε τον τρόπο χρήσης και τα οφέλη του Facade Pattern:

Διεπαφή Προς Το Σύστημα: Το Facade Pattern δημιουργεί μια διεπαφή (Facade) που εκθέτει μόνο τις σημαντικές λειτουργίες που απαιτούνται από τον χρήστη. Αυτό κρύβει την πολυπλοκότητα των εσωτερικών συστατικών του συστήματος, καθιστώντας τη χρήση του πιο εύκολη και προσβάσιμη.

Επαναχρησιμοποίηση Κώδικα: Με το Facade Pattern, η λειτουργία του συστήματος είναι οργανωμένη σε διάφορες κλάσεις και υπηρεσίες. Η διεπαφή (Facade) συγκεντρώνει και εκθέτει αυτές τις λειτουργίες, επιτρέποντας την εύκολη επαναχρησιμοποίηση τους χωρίς την ανάγκη για αναδιάρθρωση του κώδικα.

Προστασία από Πολυπλοκότητα: Όταν ένα σύστημα γίνεται πολύπλοκο με πολλές κλάσεις και υπηρεσίες, το Facade Pattern βοηθά στην δημιουργία μιας απλής επιφάνειας για τον χρήστη, περιορίζοντας την πολυπλοκότητα της αλληλεπίδρασης με το σύστημα.

Επιδόσεις και Συντήρηση: Με το Facade Pattern, είναι εφικτό να βελτιώσετε τις επιδόσεις, καθώς μπορείτε να ελαχιστοποιήσετε τις απαιτήσεις για εξαρτήσεις που δεν χρειάζεστε. Επίσης, αν χρειαστεί να γίνει συντήρηση ή αλλαγή, μπορείτε να το κάνετε σε ένα μόνο σημείο, τη διεπαφή Facade, αντί να ψάχνετε σε όλο το σύστημα.

Συνοψίζοντας, το Facade Pattern απλοποιεί τη χρήση πολύπλοκων συστημάτων, κάνοντάς το πιο εύκολο στην χρήση, επαναχρησιμοποιήσιμο και ελαχιστοποιώντας την πολυπλοκότητα. Αυτό εξυπηρετεί την διατήρηση του κώδικα, την επιδόσεις και τη γενική κατανόηση του συστήματος από τους προγραμματιστές.

Factory Pattern

Στο πλαίσιο της ανάπτυξης του συστήματος, χρησιμοποιήθηκε το Factory Pattern, το οποίο αποτελεί ένα δημοφιλές design pattern στην αντικειμενοστραφή προγραμματισμό. Το Factory Pattern χρησιμοποιήθηκε για τη δημιουργία αντικειμένων και για τη διαχείριση της δημιουργίας τους. Ας αναλύσουμε το Factory Pattern και τον τρόπο χρήσης του:

Κατανοητή Δημιουργία Αντικειμένων: Το Factory Pattern παρέχει μια διεπαφή για την δημιουργία αντικειμένων, εξαιρώντας τον κώδικα δημιουργίας από τον πελάτη. Αυτό σημαίνει ότι ο προγραμματιστής δεν χρειάζεται να ξέρει τις λεπτομέρειες της δημιουργίας αντικειμένων, αλλά μπορεί απλά να ζητήσει ένα αντικείμενο από το Factory.

Ευελιξία: Ένα από τα κύρια οφέλη του Factory Pattern είναι η ευελιξία. Μπορείτε να αλλάξετε τον τρόπο δημιουργίας αντικειμένων χωρίς να αλλάξετε τον κώδικα του πελάτη. Αυτό επιτρέπει την εύκολη αντικατάσταση και διαχείριση των αντικειμένων.

Εφαρμογή: Στο σύστημα, το Factory Pattern χρησιμοποιήθηκε για τη δημιουργία αντικειμένων τύπου "Property," όπως ακίνητα και πράκτορες. Η διαχείριση της δημιουργίας αυτών των αντικειμένων αφαιρέθηκε από τον πελάτη, και ανατέθηκε σε ένα Factory που υλοποιεί τις σχετικές λειτουργίες.

Παράδειγμα Χρήσης: Για παράδειγμα, στη δημιουργία ενός νέου ακινήτου, χρησιμοποιούμε ένα Property Factory που αναλαμβάνει την κατάλληλη δημιουργία του αντικειμένου ανάλογα με τις παρεχόμενες παραμέτρους.

Συμπέρασμα: Το Factory Pattern βοηθά στην διατήρηση της αρχιτεκτονικής του κώδικα, την αποσαφήνιση της δημιουργίας αντικειμένων και την επίτευξη της ευελιξίας και επαναχρησιμοποίησης του κώδικα. Στο συγκεκριμένο σύστημα, βοήθησε στην αποδοτική δημιουργία και διαχείριση ακινήτων και πρακτόρων.

Repository Pattern

Στο σύστημά μας, χρησιμοποιήσαμε το Repository Pattern, ένα σημαντικό design pattern που βοηθά στη διαχείριση της επικοινωνίας με τη βάση δεδομένων. Ας αναλύσουμε περαιτέρω το Repository Pattern και τον τρόπο χρήσης του:

Διαχωρισμός των Επιπέδων: Το Repository Pattern βοηθά στο να διαχωριστούν τα επίπεδα της εφαρμογής. Στην περίπτωσή μας, το Repository είναι το ενδιάμεσο επίπεδο μεταξύ της επιχειρησιακής λογικής της εφαρμογής και της βάσης δεδομένων.

Καλύτερη Διαχείριση της Βάσης Δεδομένων: Το Repository Pattern αφαιρεί τη λογική ανάκτησης και αποθήκευσης δεδομένων από την επιχειρησιακή λογική της εφαρμογής και την αναλαμβάνει αποκλειστικά. Αυτό βοηθά στην καλύτερη οργάνωση του κώδικα και στην επιτάχυνση της ανάπτυξης.

Επαναχρησιμοποίηση Κώδικα: Ένα Repository μπορεί να χρησιμοποιηθεί για τη διαχείριση δεδομένων από πολλά μέρη της εφαρμογής χωρίς την ανάγκη επανάληψης του ίδιου κώδικα. Αυτό βελτιώνει τη συντήρηση του κώδικα.

Στην Πράξη: Στο σύστημα μας, το Repository Pattern χρησιμοποιήθηκε για τη διαχείριση των δεδομένων των ακινήτων και των πρακτόρων. Παρέχει μια συγκεκριμένη διεπαφή (interface) για την ανάκτηση, αποθήκευση, ενημέρωση και διαγραφή δεδομένων από τη βάση δεδομένων, χωρίς να απαιτείται γνώση της δομής της βάσης από τον πελάτη.

Παράδειγμα Χρήσης: Για παράδειγμα, για την ανάκτηση ακινήτων από τη βάση δεδομένων, αντί να έχουμε συχνή πρόσβαση στη βάση από τον πελάτη, χρησιμοποιούμε το PropertyRepository που παρέχει μια καθαρή διεπαφή για αυτό τον σκοπό.

Συμπέρασμα: Το Repository Pattern είναι ένα ισχυρό εργαλείο στην αρχιτεκτονική της εφαρμογής, βοηθά στη διαχείριση της βάσης δεδομένων και τη βελτιστοποίηση της ανάπτυξης του κώδικα.

Πακέτα Laravel

Laravel Breeze

Το Laravel Breeze είναι ένα εξαιρετικά χρήσιμο πακέτο που χρησιμοποιήθηκε στο σύστημα μας για την ανάπτυξη και την επιτάχυνση της διαδικασίας της ανάπτυξης της εφαρμογής σε Laravel. Το Laravel Breeze προσφέρει μια αρκετά ολοκληρωμένη αρχιτεκτονική για την προετοιμασία και την δημιουργία ενός λειτουργικού συστήματος αυθεντικοποίησης (authentication) χρηστών και αρχικών σελίδων.

Αναλυτικά, τα κύρια χαρακτηριστικά του Laravel Breeze που χρησιμοποιήσαμε είναι τα εξής:

Εγκατάσταση και Διαχείριση Χρηστών: Το Breeze διευκολύνει την εγκατάσταση και τη διαχείριση των χρηστών μέσω της αυθεντικοποίησης, της εγγραφής και της επαναφοράς κωδικού πρόσβασης.

Επαναφορά Κωδικού Πρόσβασης: Το Breeze προσφέρει λειτουργίες για την επαναφορά του κωδικού πρόσβασης μέσω email, επιτρέποντας στους χρήστες να ανακτήσουν την πρόσβαση στον λογαριασμό τους.

Συνοδευτικά Χαρακτηριστικά: Το Breeze παρέχει επίσης συνοδευτικές λειτουργίες όπως αποσύνδεση χρήστη, διαχείριση προφίλ χρήστη και επανεγγραφή.

Το λόγο για τον οποίο επιλέξαμε να χρησιμοποιήσουμε το Laravel Breeze είναι η ταχύτητα και η αποτελεσματικότητα που προσφέρει στη δημιουργία του συστήματος αυθεντικοποίησης και στις αρχικές σελίδες. Αντί να χρησιμοποιήσουμε χρόνο και πόρους για τη δημιουργία αυτών των βασικών λειτουργιών από το μηδέν, το Laravel Breeze προσφέρει μια έτοιμη λύση που είναι εύκολο να προσαρμοστεί στις ανάγκες μας.

Συνοψίζοντας, το Laravel Breeze αποτελεί έναν πολύτιμο σύμμαχο στη διαδικασία ανάπτυξης εφαρμογών Laravel, καθώς παρέχει ένα πλήρες και έτοιμο σετ εργαλείων για τη διαχείριση των χρηστών και την αυθεντικοποίηση, επιταχύνοντας σημαντικά τη διαδικασία ανάπτυξης.

Laravel Dompdf

Το Laravel Dompdf ενσωματώθηκε στο σύστημά μας για να παρέχει τη δυνατότητα στους χρήστες με ρόλο πράκτορα/agent να δημιουργούν και να εξάγουν τα τιμολόγια των πακέτων που χρησιμοποίησαν σε μορφή PDF. Αυτή η δυνατότητα είναι πολύ χρήσιμη για τους πράκτορες, καθώς τους επιτρέπει να δημιουργούν επαγγελματικά τιμολόγια για τις συναλλαγές που πραγματοποιούν με τους χρήστες του συστήματός μας. Αυτό διευκολύνει τη διαχείριση των οικονομικών διαδικασιών και παρέχει μια επαγγελματική εικόνα στην επικοινωνία με τους πελάτες. Επιπλέον, η δυνατότητα εξαγωγής σε PDF εξασφαλίζει ότι τα τιμολόγια είναι εύκολα προσβάσιμα και διακριτικά για τους χρήστες του συστήματός μας.

Laravel Telescope

Για την διαδικασία της ανάπτυξης και του development του συστήματός μας, χρησιμοποιήσαμε το εργαλείο Laravel Telescope. Το Laravel Telescope είναι ένα αποκλειστικό εργαλείο ανάπτυξης που προσφέρει ενδιαφέρουσες δυνατότητες και χρήσιμες πληροφορίες για την παρακολούθηση των αιτημάτων, και των αιτημάτων της βάσης δεδομένων, των εργασιών και πολλών άλλων πτυχών της εφαρμογής μας.

Χρησιμοποιώντας το Laravel Telescope, ήταν εφικτή η παρακολούθηση και η ανάλυση της απόδοσης της εφαρμογής και ο εντοπισμός προβλημάτων. Επιπλέον, το Laravel Telescope μας επέτρεψε να παρακολουθούμε την δραστηριότητα των χρηστών, την απόδοση των αιτημάτων API, και να εντοπίσουμε επιπρόσθετες πληροφορίες για τον τρόπο με τον οποίο οι χρήστες αλληλοεπιδρούν με την πλατφόρμα μας.

Με το Laravel Telescope, είχαμε τη δυνατότητα να αναπτύξουμε το σύστημα μας με περισσότερη ασφάλεια, αποτελεσματικότητα και ευκολία. Το συγκριμένο εργαλείο ανάπτυξης

συνέβαλε στη βελτιστοποίηση της διαδικασίας ανάπτυξης της εφαρμογής και της συνολικής ποιότητας του κώδικά μας.

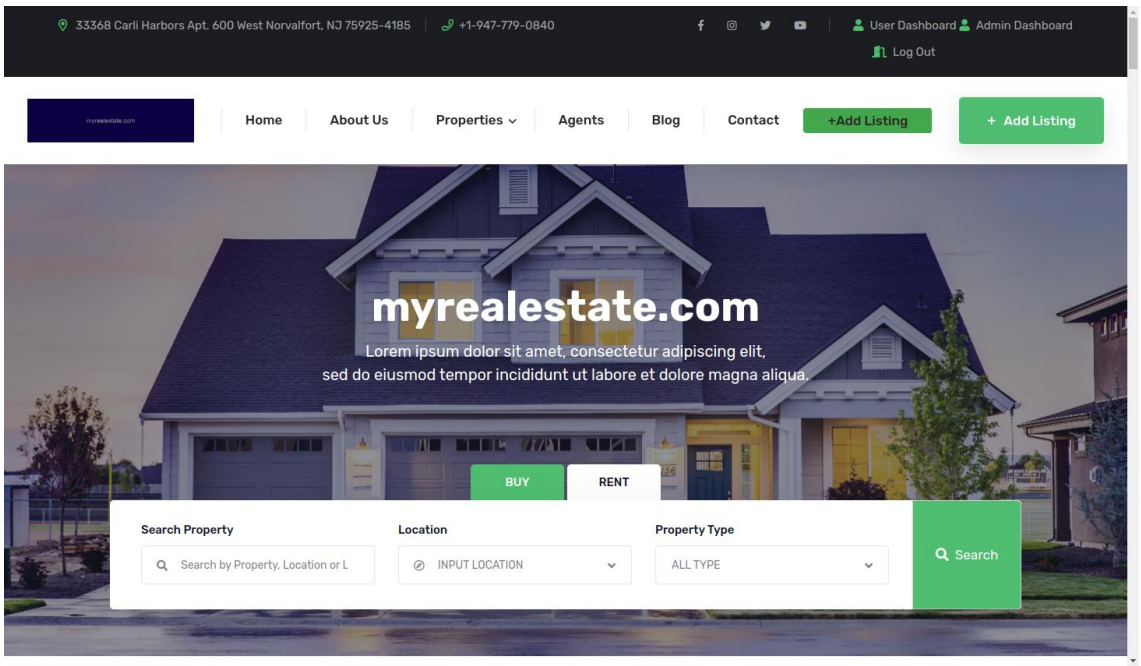
Περιγραφή Συστήματος

Από την σκοπιά του απλού χρήστη

Το σύστημα προσφέρει πλήρη υποστήριξη για τη σύνδεση και την εγγραφή νέων χρηστών. Επιπρόσθετα, προσφέρει δυνατότητες για την αλλαγή κωδικών και την επιβεβαίωση του email τους. Αυτές οι λειτουργίες αυξάνουν την ασφάλεια των χρηστών και εξασφαλίζουν ελεγχόμενη πρόσβαση στο σύστημα, διασφαλίζοντας την ιδιωτικότητα και την ακεραιότητα των δεδομένων τους.

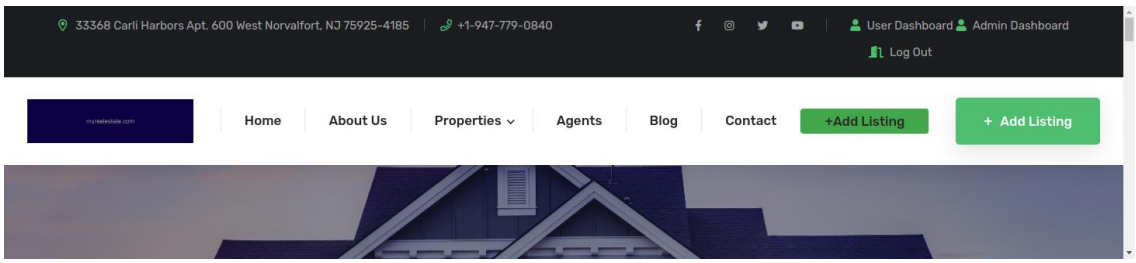
Ο απλός χρήστης (user) έχει πρόσβαση σε μια πληθώρα λειτουργιών στο σύστημά.

Περιήγηση: Ο χρήστης μπορεί να εξερευνήσει την ιστοσελίδα και να δει όλα τα διαθέσιμα ακίνητα. Αυτή η επιλογή του επιτρέπει να αποκτήσει μια γενική εικόνα των ακινήτων που υπάρχουν διαθέσιμα για αγορά ή ενοικίαση στην ιστοσελίδα ώστε να επιλέξει εκείνα που εξυπηρετούν καλύτερα τις ανάγκες του.



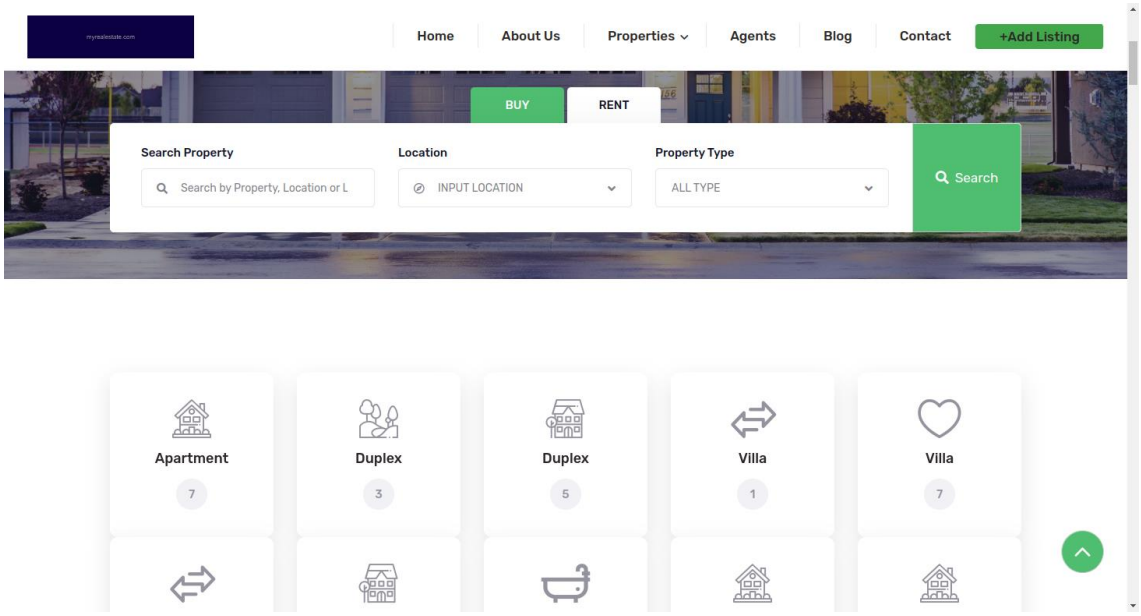
Εικόνα 6:Βασική σελίδα/οθόνη

Μενού Εφαρμογής: Ο χρήστης μπορεί να χρησιμοποιήσει το μενού της εφαρμογής για να ανακαλύψει όλες τις διαθέσιμες επιλογές, συμπεριλαμβανομένων των αναζητήσεων ακινήτων και των επιλογών κατηγορίας ώστε να βρει ευκολότερα εκείνα που τον ικανοποιούν.

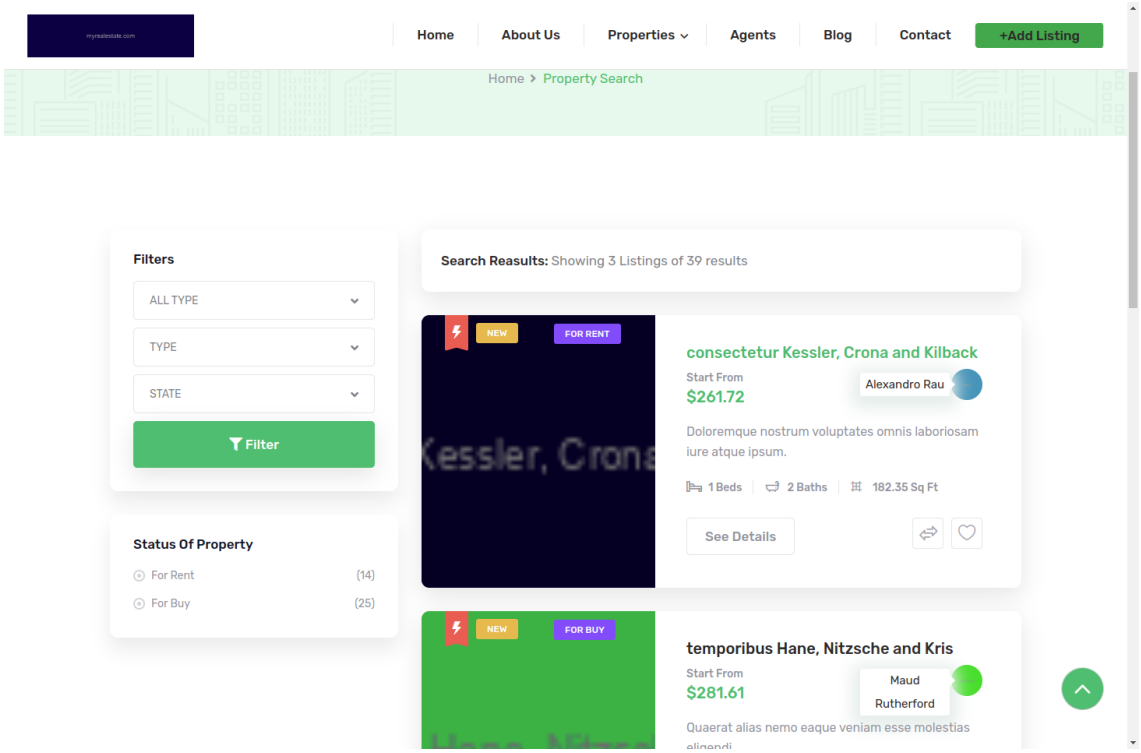


Εικόνα 7: Βασικό μενού

Φόρμες Αναζήτησης: Οι φόρμες αναζήτησης προσφέρουν στον χρήστη τη δυνατότητα να προσδιορίσει τα κριτήρια της αναζήτησής του. Μπορεί να αναζητήσει ακίνητα βάσει διάφορων παραμέτρων, όπως η διαθεσιμότητα για αγορά ή ενοικίαση, η κατηγορία του ακινήτου (πχ. διαμέρισμα, βίλα), και η περιοχή που ενδιαφέρεται να ψάξει.



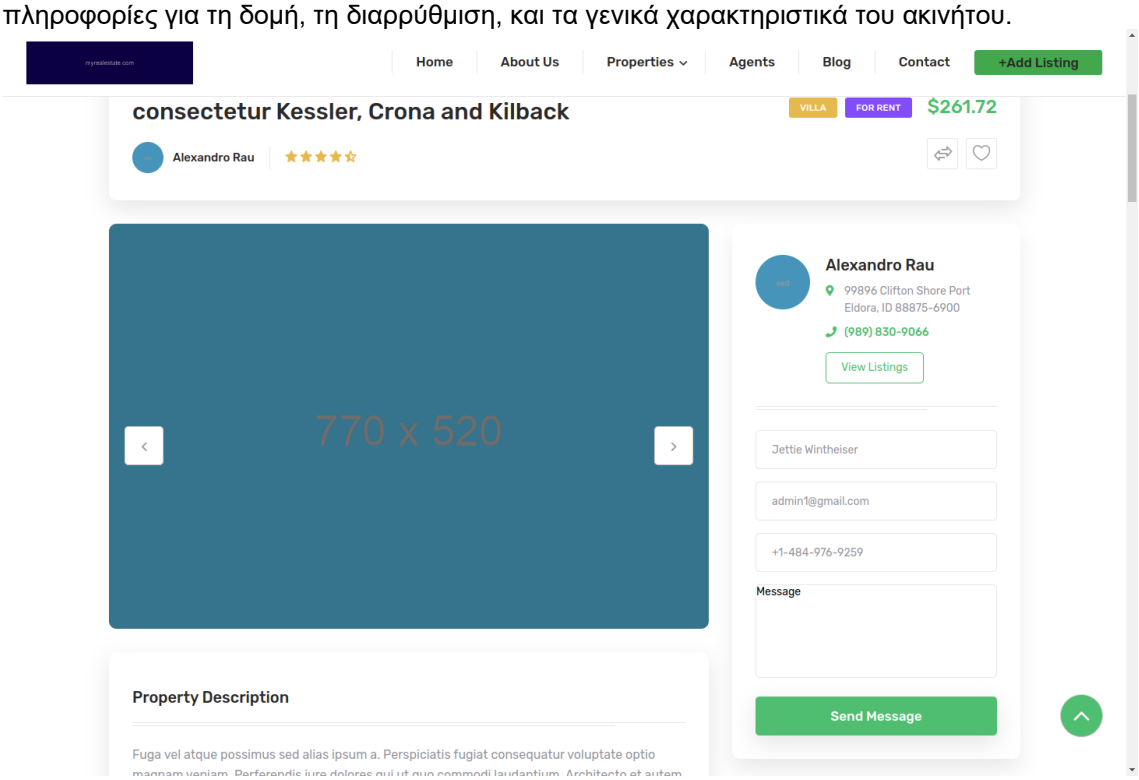
Εικόνα 8: Εμφάνιση υπομενού



Εικόνα 9: Σελίδα/οθόνη ακινήτου

Επίσης, ο χρήστης μπορεί να επισκεφτεί την ιδικά διαμορφωμένη σελίδα του κάθε ακινήτου για να ανακαλύψει ακόμα περισσότερες λεπτομέρειες. Στη σελίδα αυτή, θα βρει:

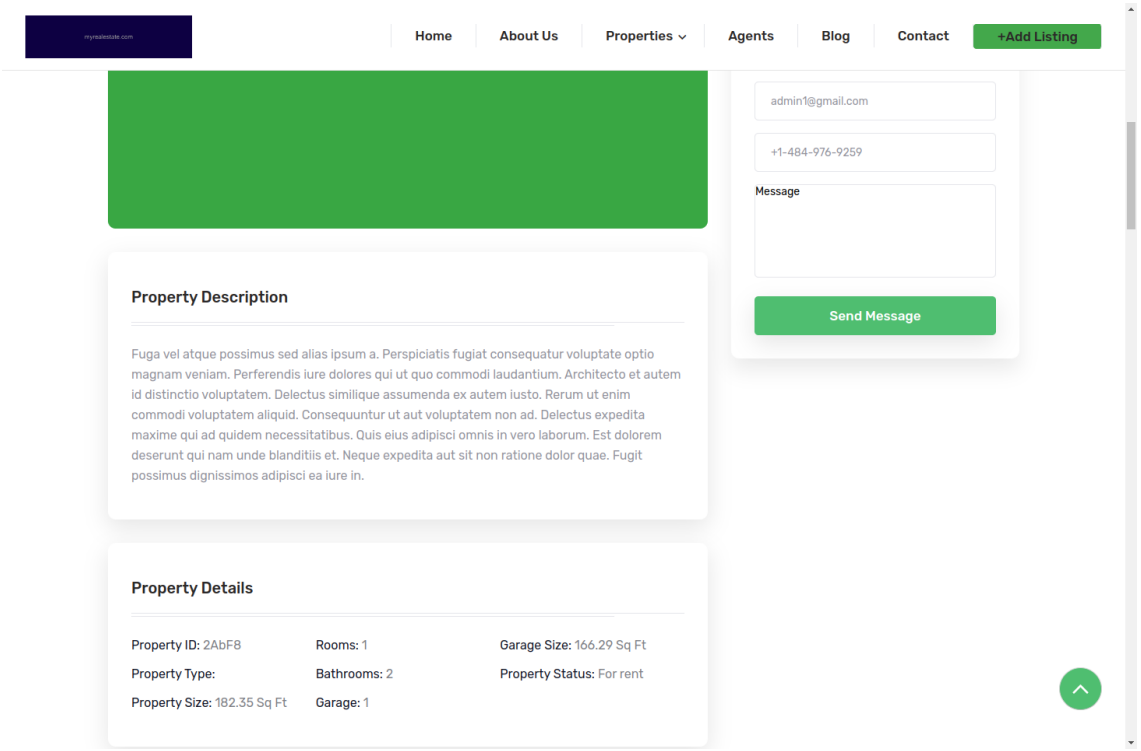
Περιγραφή του ακινήτου: Μια λεπτομερής περιγραφή του ακινήτου που περιλαμβάνει πληροφορίες για τη δομή, τη διαρρύθμιση, και τα γενικά χαρακτηριστικά του ακινήτου.



Εικόνα 10:Περιγραφή ακινήτου

Αναλυτικές πληροφορίες: Ο αριθμός των δωματίων, το μέγεθος σε τετραγωνικά μέτρα, ο αριθμός των μπάνιων, και η ύπαρξη γκαράζ.

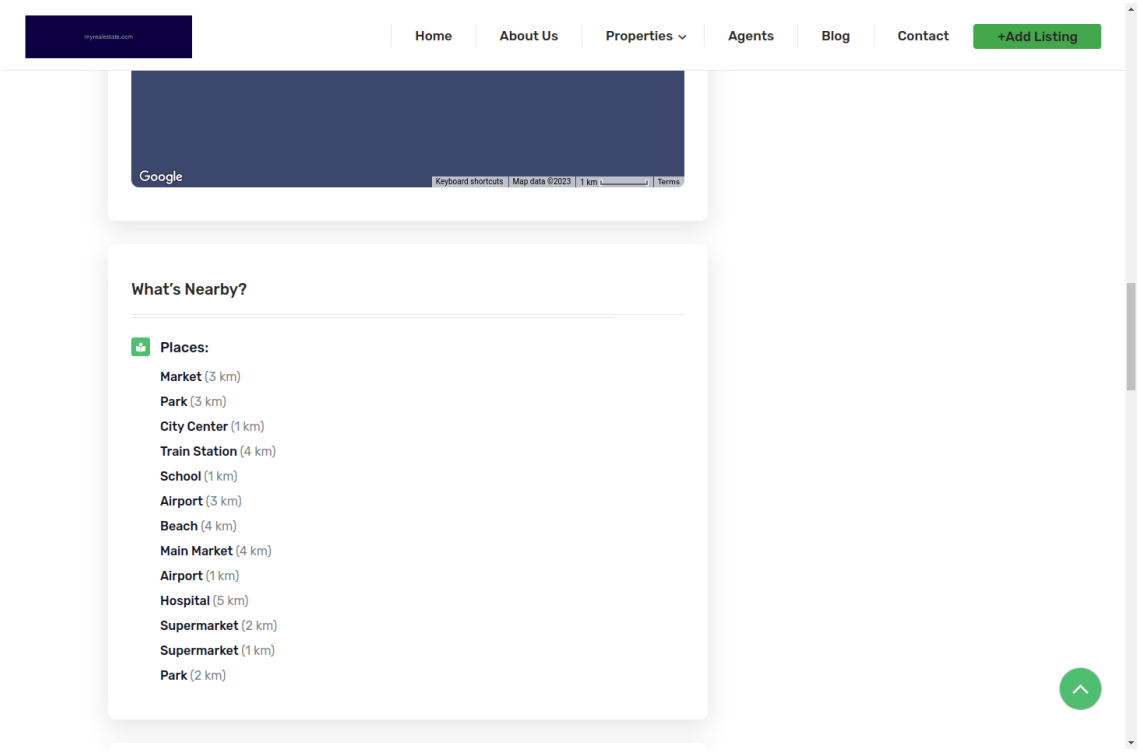
Παροχές του ακινήτου: Πληροφορίες σχετικά με τις παροχές που προσφέρει το ακίνητο, όπως κεντρική θέρμανση, κλιματισμός, πισίνα, κήπος, κλπ.



Εικόνα 11:Επιπλέον στοιχεία περιγραφής ακινήτου

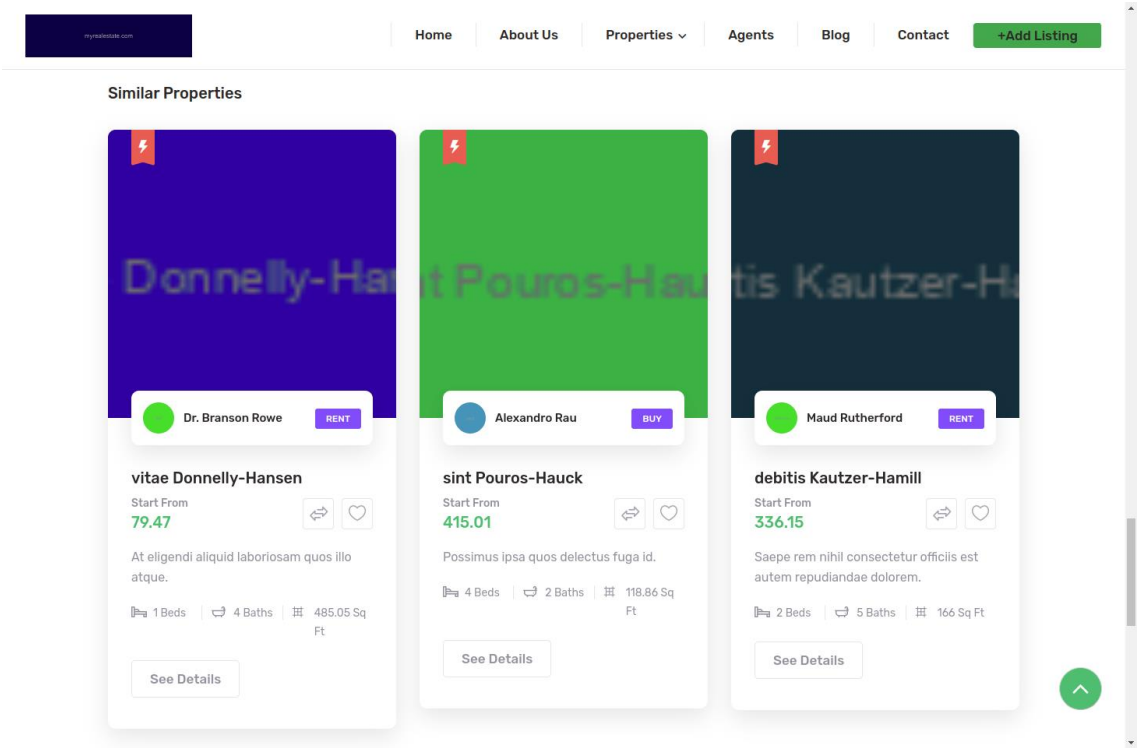
Τοποθεσία με χρήση Google Maps: Ο χρήστης μπορεί να δει την ακριβή τοποθεσία του ακινήτου στο χάρτη του Google Maps, καθώς και τα σημεία αναφοράς στην κοντινή περιοχή. Αυτό επιτυγχάνεται καθώς στην φόρμα καταχώρησης των ακινήτων υπάρχει ειδικό πεδίο για την εισαγωγή των συντεταγμένων.

Επιπλέον, ο απλός χρήστης μπορεί να δει βίντεο του ακινήτου, προσφέροντας μια ακόμα πιο ρεαλιστική εικόνα του ακινήτου. Αυτό επιτρέπει στον χρήστη να περιηγηθεί εικονικά στον χώρο και να αξιολογήσει εάν το συγκεκριμένο ακίνητο τον ικανοποιεί χωρίς να χρειάζεται να το επισκεφτεί φυσικά. Το βίντεο παρέχει μια πραγματική αίσθηση της διάταξης και της ατμόσφαιρας του ακινήτου, βοηθώντας τον χρήστη να πάρει πιο ουσιαστικές αποφάσεις σχετικά με την αγορά ή την ενοικίαση του ακινήτου που αναζητά.



Εικόνα 12: Προβολή γειτονικών περιοχών

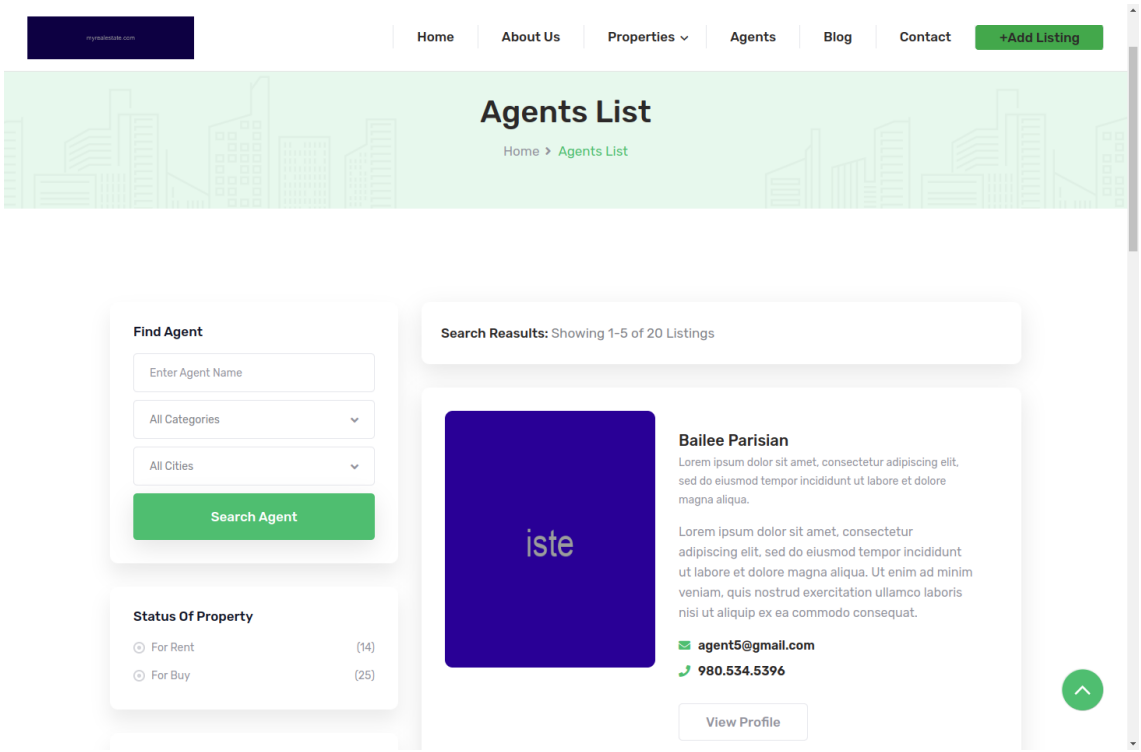
Προτάσεις: Το σύστημα προτείνει στον χρήστη μια λίστα με παρόμοια ακίνητα που ενδέχεται να τον ενδιαφέρουν, βασιζόμενο στις προηγούμενες αναζητήσεις και τις προτιμήσεις του καθώς και την κατηγορία που ανήκει το ακίνητο που έχει επισκεφτεί.



Εικόνα 13:Προβολή παρόμοιων ακινήτων

Αυτές οι πρόσθετες λεπτομέρειες προσφέρουν στον χρήστη μια ολοκληρωμένη εικόνα του ακινήτου και του περιβάλλοντός του, καθιστώντας την αναζήτηση ακινήτων πιο εύκολη και ενημερωμένη, βοηθώντας τον στην λήψη πιο ενημερωμένων αποφάσεων σχετικά με την αγορά ή την ενοικίαση ακινήτου.

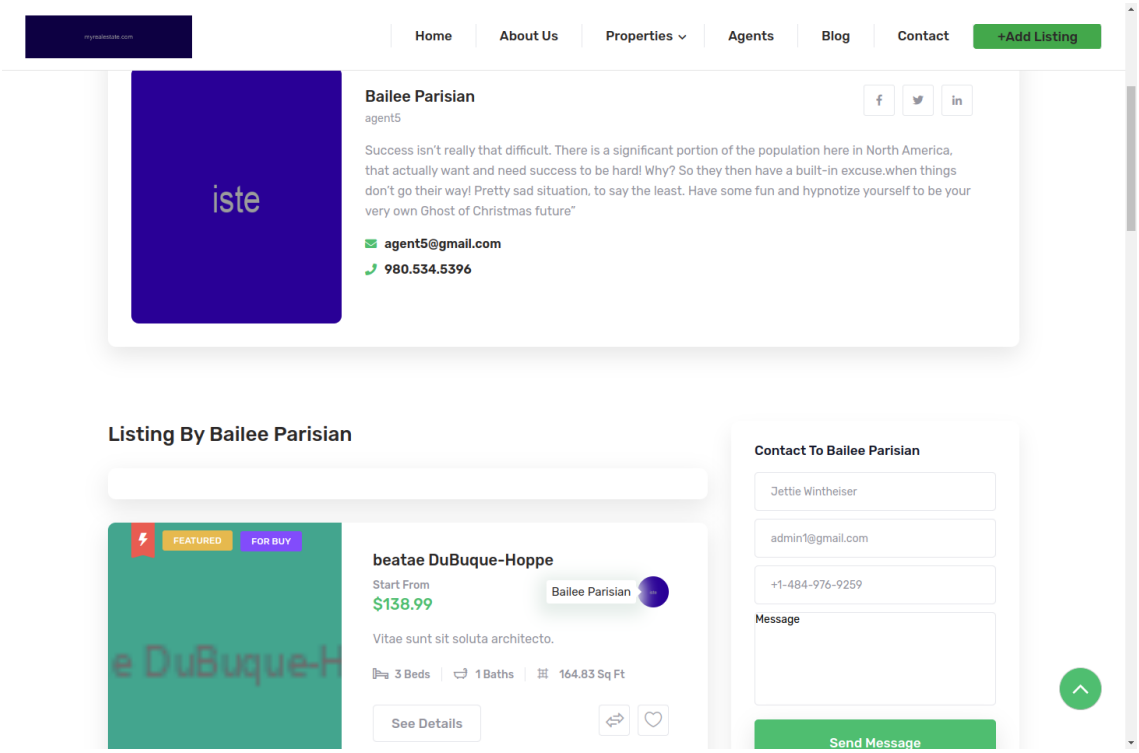
Επιπλέον, ο χρήστης έχει τη δυνατότητα να εξερευνήσει όλους τους εγγεγραμμένους πράκτορες/agents στην εφαρμογή. Αφού επιλέξει έναν συγκεκριμένο πράκτορα με τον οποίο θέλει να επικοινωνήσει, μπορεί να δει μια λίστα με όλα τα ακίνητα που διαχειρίζεται ο συγκεκριμένος πράκτορας.



[image209]

Εικόνα 14: Προβολή Μεσιτών

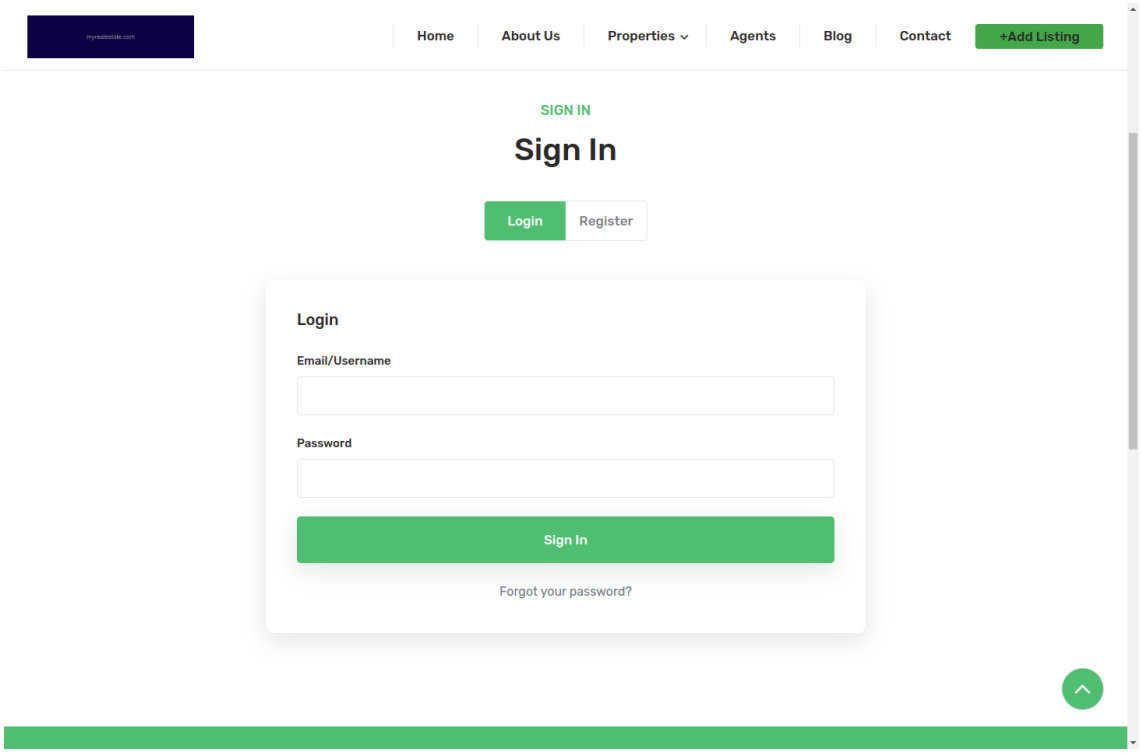
Αυτή η λειτουργία δίνει στον χρήστη τη δυνατότητα να εξετάσει τις καταχωρήσεις και τις προσφορές των διαφόρων πρακτόρων, ενισχύοντας την επιλογή του για το ποιον πράκτορα επιθυμεί να επικοινωνήσει. Με αυτόν τον τρόπο, ο χρήστης έχει τον πλήρη έλεγχο των επιλογών του και τη δυνατότητα να βρει ακίνητα που ταιριάζουν στις προτιμήσεις του.



Εικόνα 15: Προβολή λεπτομερειών μεσιτών

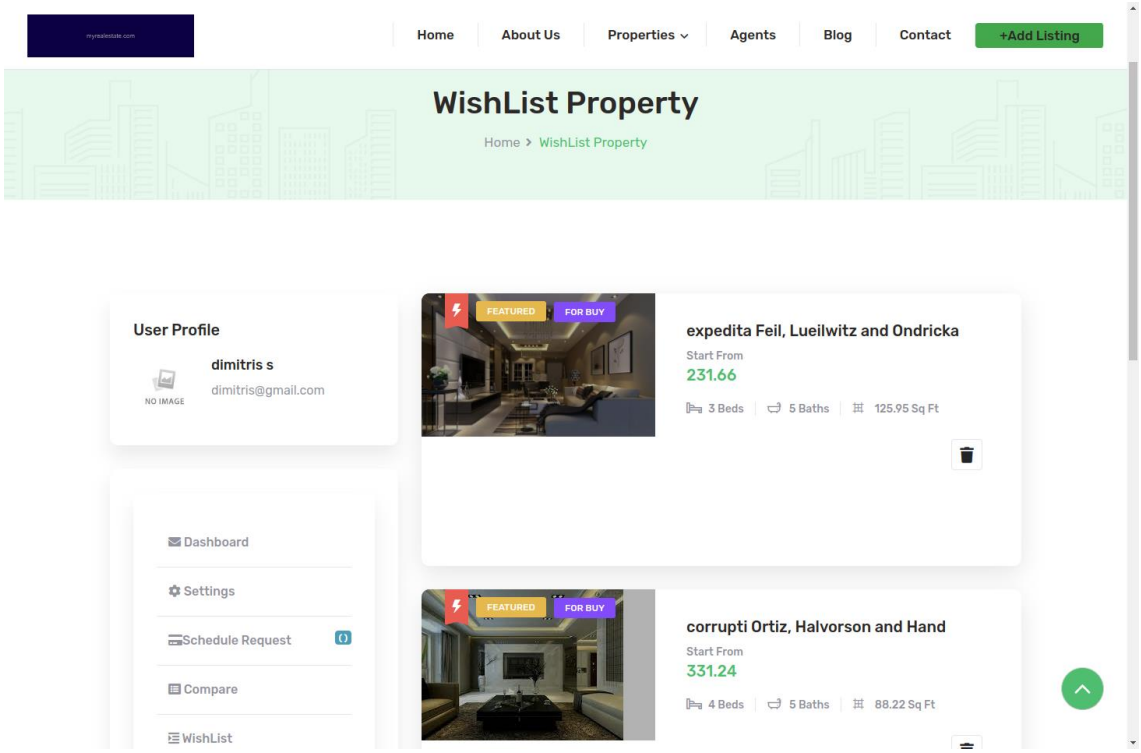
Εγγεγραμμένος Χρήστης

Επιπλέον, ο συνδεδεμένος εγγεγραμμένος χρήστης έχει τη δυνατότητα να ενημερώσει το προφίλ του στο σύστημα, προσθέτοντας προσωπικές πληροφορίες και προτιμήσεις. Αυτό το κάνει πιο εξατομικευμένο και βοηθά το σύστημα να παρέχει πιο εξατομικευμένες προτάσεις ακινήτων που ταιριάζουν στις ανάγκες του.



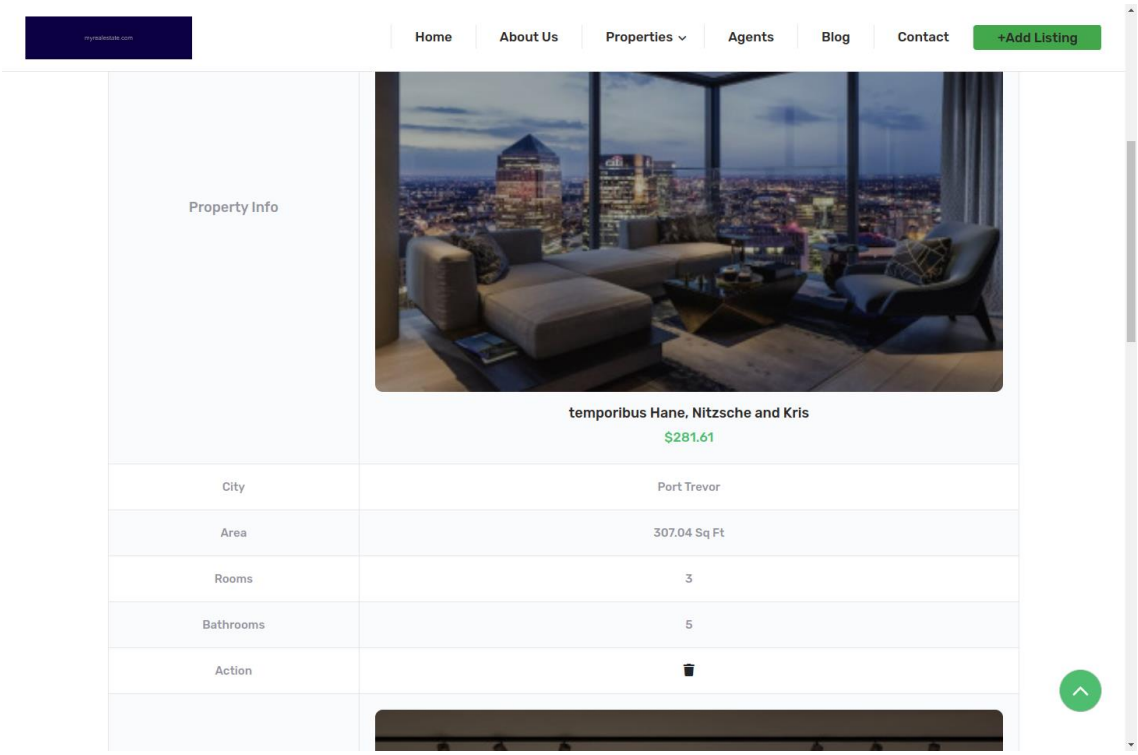
Εικόνα 16: Login φόρμα

Παράλληλα, μπορεί να προσθέσει ακίνητα που τον ενδιαφέρουν στη λίστα των αγαπημένων του. Αυτά τα ακίνητα θα είναι εύκολα προσβάσιμα στο μέλλον, επιτρέποντάς του να τα ανατρέξει και να τα συγκρίνει ευκολότερα. Αυτή η λειτουργία βοηθά τον χρήστη να διατηρήσει μια οργανωμένη λίστα με τα αγαπημένα του ακίνητα και να παρακολουθεί τις επιλογές του με άνεση.



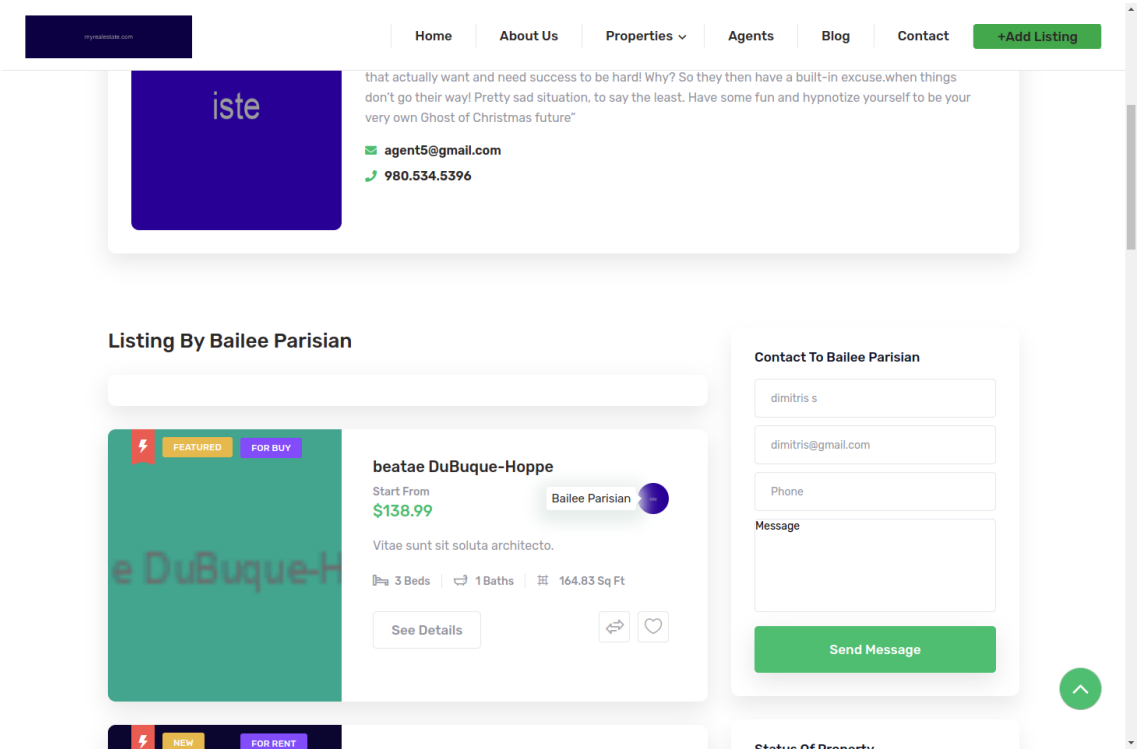
Εικόνα 17: Προβολή προτιμώμενων ακινήτων (εγγεγραμμένος χρήστης)

Επιπλέον, για τους εγγεγραμμένους χρήστες, υπάρχει η δυνατότητα προσθήκης ακινήτων σε μια επιπλέον λειτουργία που παρέχει το σύστημα για συγκρίσεις ακινήτων. Έτσι, ο χρήστης μπορεί να δει σε μια σελίδα τα ακίνητα που τον ενδιαφέρουν και να τα συγκρίνει βάσει των χαρακτηριστικών και των παροχών που προσφέρει καθένα από αυτά. Αυτή η λειτουργία επιτρέπει στον χρήστη να έχει μια καλύτερη και πιο πλήρη εικόνα των χαρακτηριστικών των ακινήτων, διευκολύνοντας τον στην λήψη αποφάσεων και την επιλογή του κατάλληλου ακινήτου που εξυπηρετεί καλύτερα τις ανάγκες του.



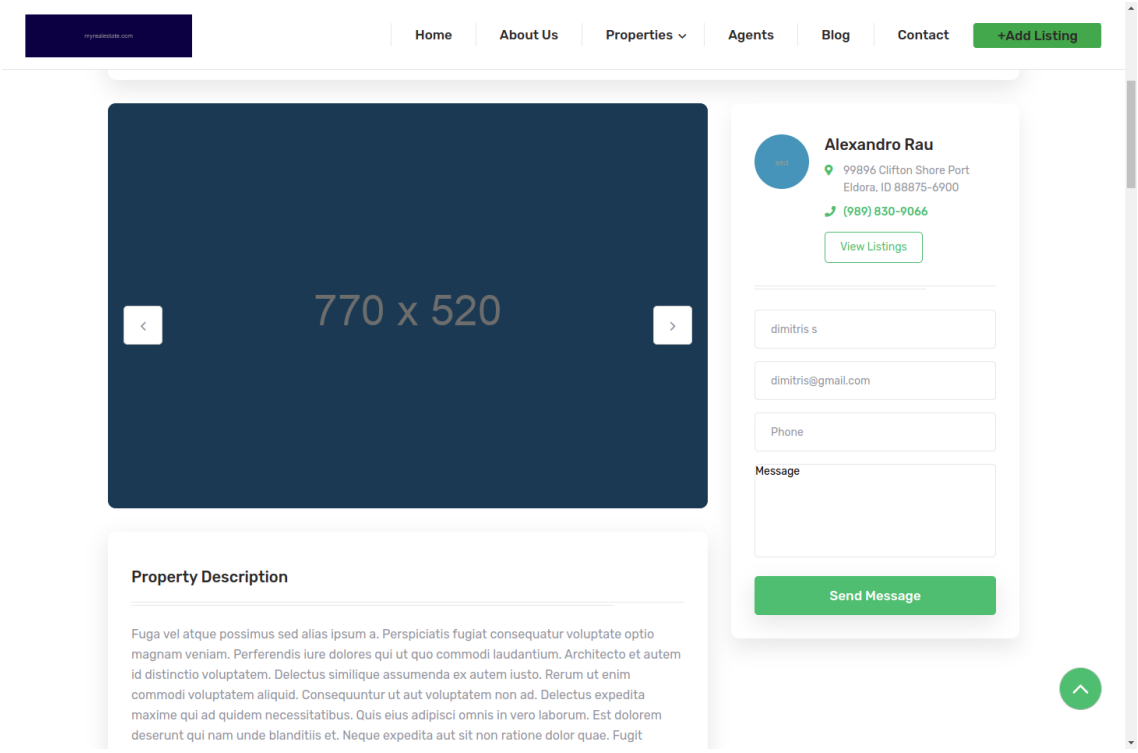
Εικόνα 18: Λεπτομέρειες ακινήτου (εγγεγραμμένος χρήστης)

Επιπλέον, για τους εγγεγραμμένους στο σύστημα χρήστες, υπάρχει η σημαντική λειτουργία της επικοινωνίας με τους πράκτορες που διαχειρίζονται τα ακίνητα που τους ενδιαφέρουν. Όταν ένας χρήστης μεταβαίνει στην καρτέλα ενός συγκεκριμένου πράκτορα, μπορεί να δει τη λίστα των ακινήτων που διαχειρίζεται αυτός ο πράκτορας. Στη συνέχεια, μπορεί να επικοινωνήσει μαζί του μέσω της πλατφόρμας, στέλνοντας του μηνύματα. Αυτή η λειτουργία διευκολύνει την επικοινωνία των χρηστών με τους πράκτορες και τη διαδικασία ενημέρωσής τους σχετικά με τα ακίνητα που τους ενδιαφέρουν.



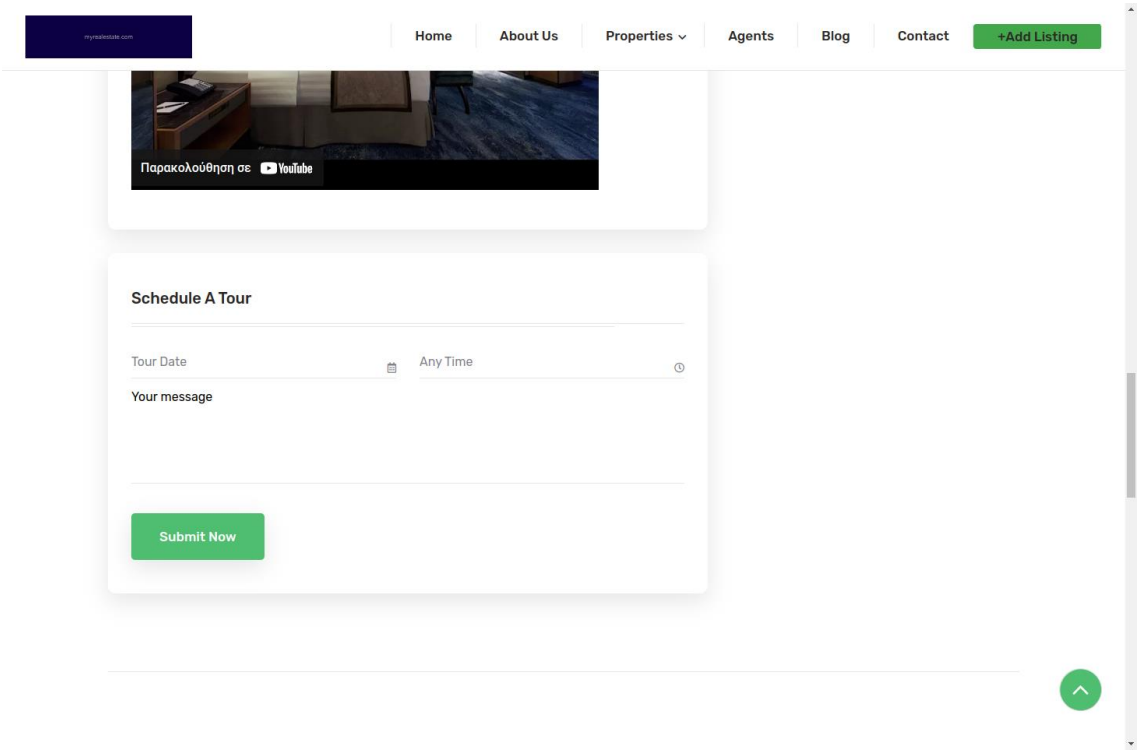
Εικόνα 19: Προβολή μεσιτών (εγγεγραμμένος χρήστης

Επίσης, όταν ένας εγγεγραμμένος χρήστης μεταβαίνει στη σελίδα ενός συγκεκριμένου ακινήτου, μπορεί να δει ποιος πράκτορας διαχειρίζεται το συγκεκριμένο ακίνητο. Από εκεί, μπορεί να επικοινωνήσει αμέσως με τον πράκτορα, αποστέλλοντας του άμεσο μήνυμα μέσω μιας κατάλληλης φόρμας επικοινωνίας. Αυτή η λειτουργία διευκολύνει την άμεση επικοινωνία με τον πράκτορα και τη δυνατότητα τελικών χρηστών να θέτουν ερωτήσεις ή να ζητούν περισσότερες πληροφορίες σχετικά με το ακίνητο που τους ενδιαφέρει.



Εικόνα 20: Λεπτομέρειες ακινήτου (εγγεγραμμένος χρήστης)

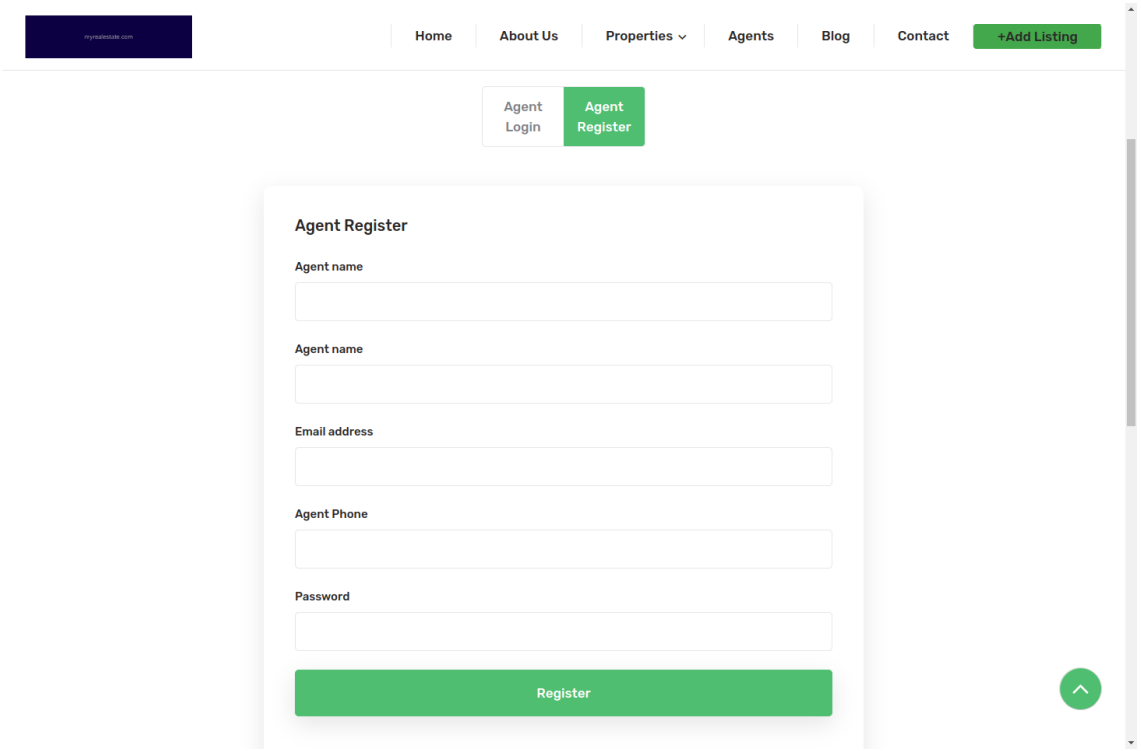
Επίσης, από την ίδια σελίδα, ο χρήστης μπορεί μέσω μιας ειδικής φόρμας να εκδηλώσει ενδιαφέρον για το συγκεκριμένο ακίνητο και να αιτηθεί μια επίσκεψη στον χώρο του ακινήτου. Το αίτημα αυτό θα διαβιβαστεί στον πράκτορα μέσω της εφαρμογής, και ο χρήστης θα λάβει απάντηση σχετικά με την διεκπεραίωση του συγκεκριμένου αιτήματος. Αυτή η λειτουργία επιτρέπει στους χρήστες να ζητούν επισκέψεις στα ακίνητα που τους ενδιαφέρουν, και στους πράκτορες να διαχειρίζονται αυτά τα αιτήματα με αποτελεσματικό τρόπο.



Εικόνα 21: Σελίδα/οθόνη εκδήλωσης ενδιαφέροντος για ακίνητο (εγγεγραμμένος χρήστης)

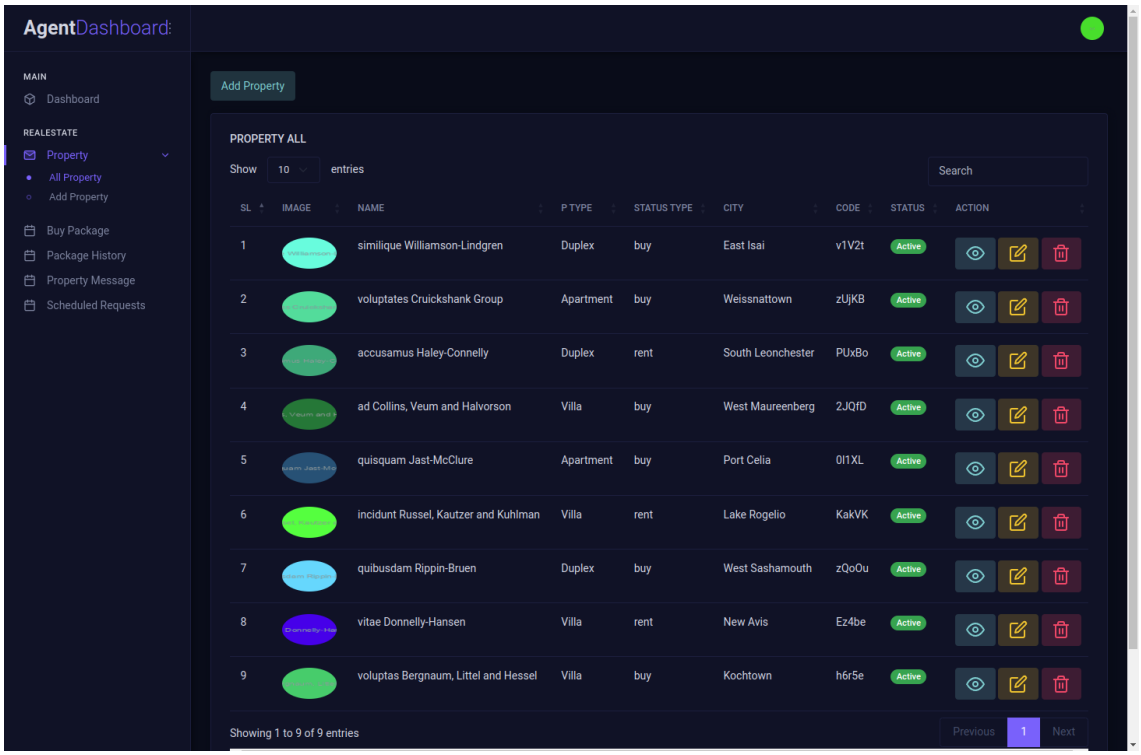
Πράκτορας/agent

Επιπλέον, το σύστημα προσφέρει τη δυνατότητα εγγραφής ενός χρήστη ως πράκτορας. Για να γίνει αυτό, ο χρήστης πρέπει να εγγραφεί στο σύστημα ως πράκτορας και να συνδεθεί στο διαχειριστικό περιβάλλον των πρακτόρων. Στη συνέχεια, μπορεί να προσθέσει τα ακίνητα που διαχειρίζεται, τα οποία στη συνέχεια θα είναι ορατά στους τελικούς χρήστες. Αυτή η λειτουργία επιτρέπει στους χρήστες του συστήματος που έχουν τον ρόλο του πράκτορα/agent να διαχειρίζονται τα ακίνητά τους με αποτελεσματικότητα και να τα προβάλλουν στους ενδιαφερόμενους χρήστες.

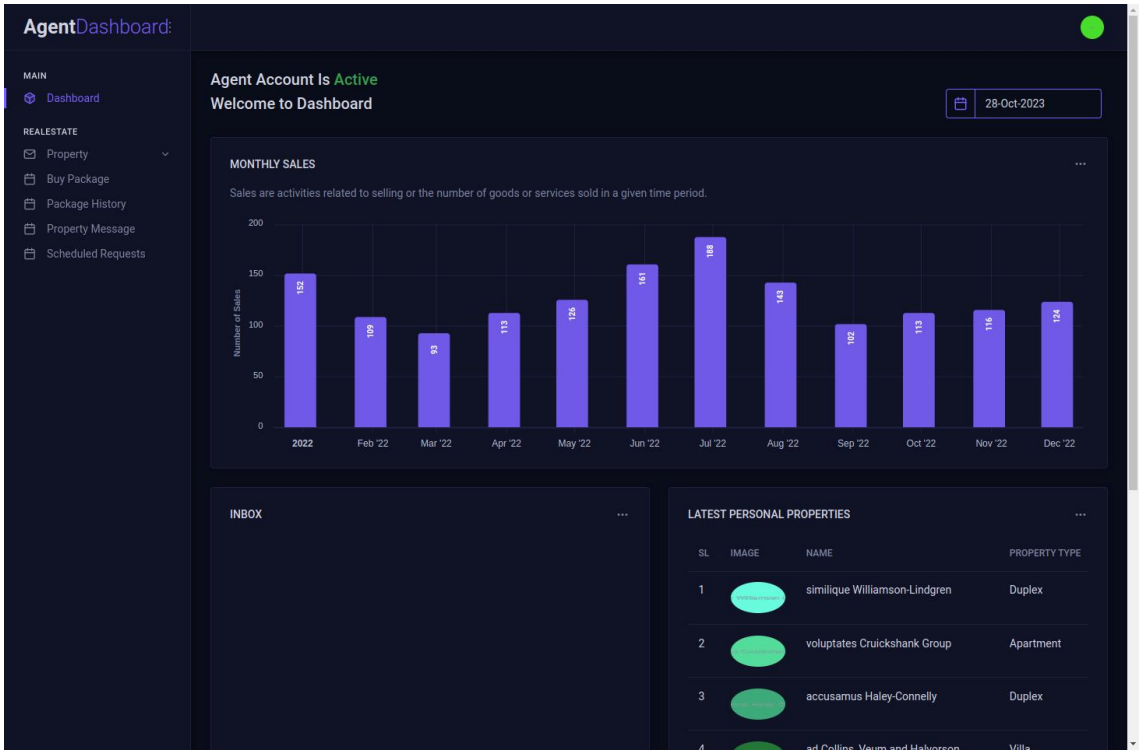


Εικόνα 22:Σελίδα / οθόνη εγγραφής πράκτορα/μεσίτη

Το σύστημα προσφέρει διάφορα πακέτα σε χρήστες που έχουν τον ρόλο του agent/πράκτορα, κάθε ένα από τα οποία καθορίζει τον μέγιστο αριθμό ακινήτων που μπορεί να προσθέσει ο πράκτορας στο σύστημα. Μόλις ο πράκτορας υπερβεί αυτό τον μέγιστο αριθμό ακινήτων, θα πρέπει να αγοράσει ένα επιπλέον πακέτο για να μπορεί να προσθέσει περισσότερα ακίνητα στον λογαριασμό του. Επιπρόσθετα, το σύστημα προσφέρει την δυνατότητα προβολής του ιστορικού των πακέτων που έχει χρησιμοποιήσει στο παρελθόν από το διαχειριστικό περιβάλλον του πράκτορα είναι ένα χρήσιμο χαρακτηριστικό. Αυτό του επιτρέπει να παρακολουθεί τον αριθμό των ακινήτων που έχει προσθέσει και να διαχειρίζεται τη χρήση των πακέτων του με βάση τις ανάγκες του και τον αριθμό των ακινήτων θέλει να διαχειρίζεται. Αυτό είναι σημαντικό για την αποτελεσματική διαχείριση των ακινήτων και την οικονομία στη χρήση των πακέτων καθώς ο αριθμός των καταχωρήσεων που μπορεί να περάσει στο σύστημα δεν είναι απεριόριστος.

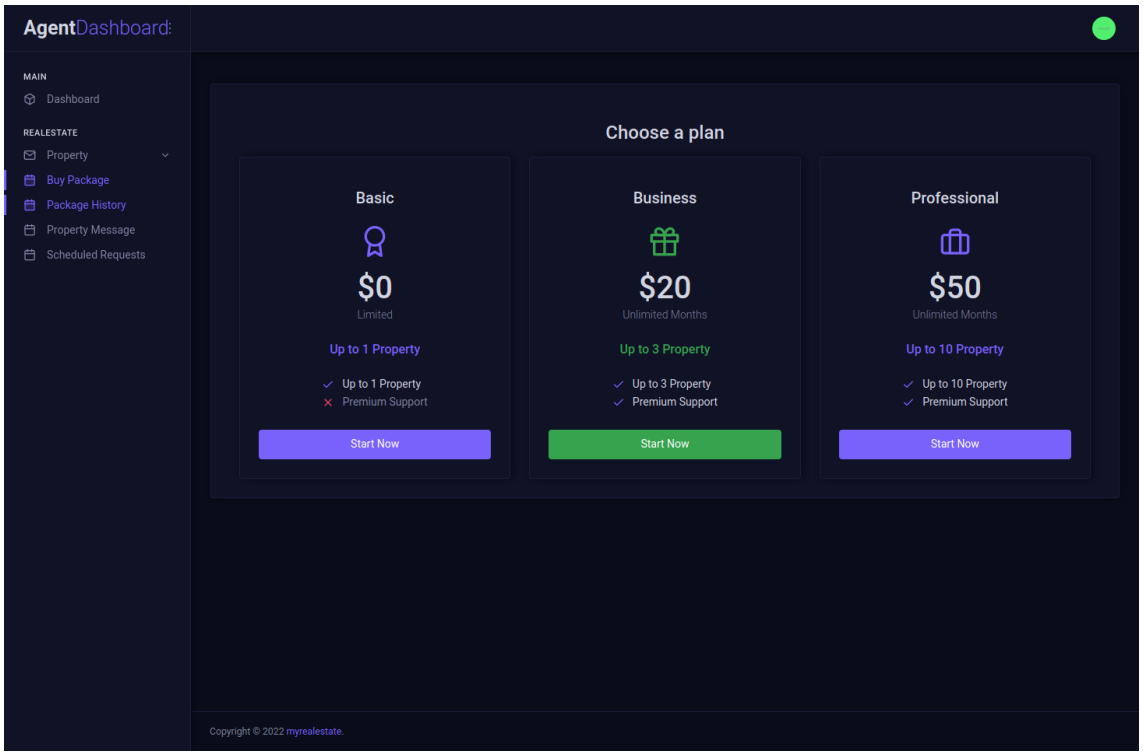


Εικόνα 23: Σελίδα/ οθόνη διαχείρισης ακινήτων[image220],[image221]



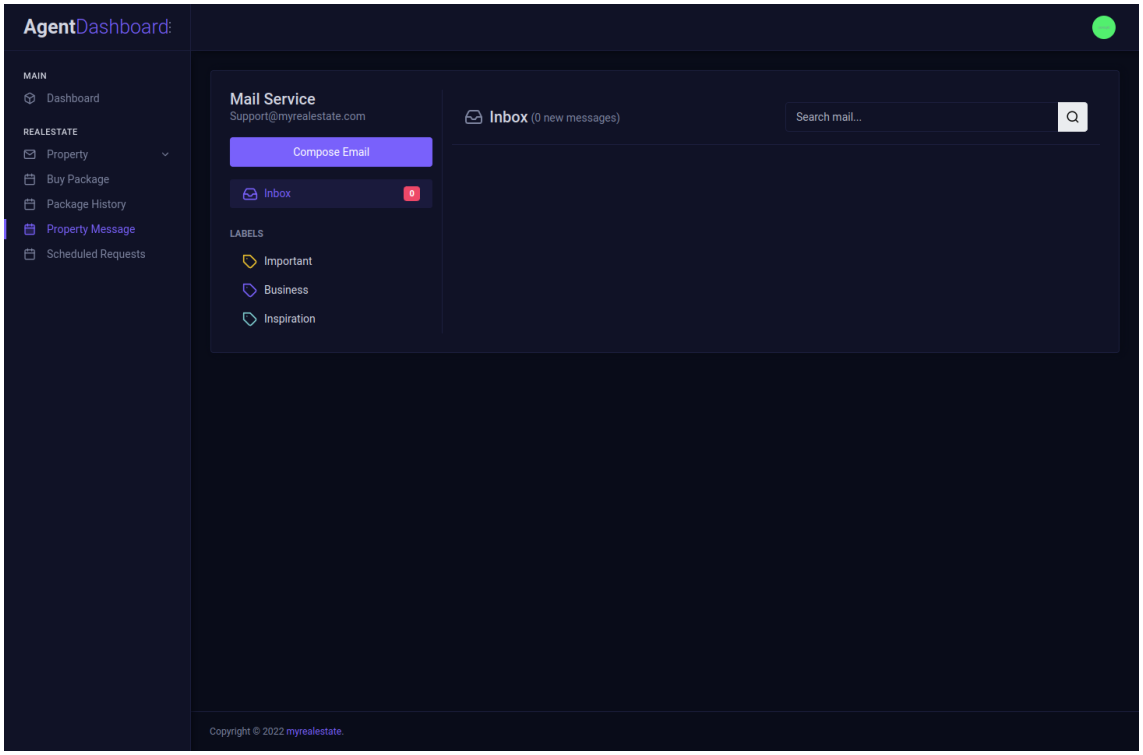
Εικόνα 24: Στατιστικά διαχείρισης ακινήτων

Αυτή η λειτουργία είναι σημαντική για τη διαχείριση των πρακτόρων και εξασφαλίζει ότι ο καθένας από αυτούς χρησιμοποιεί τα πακέτα που του προσφέρουν τον απαιτούμενο αριθμό ακινήτων.



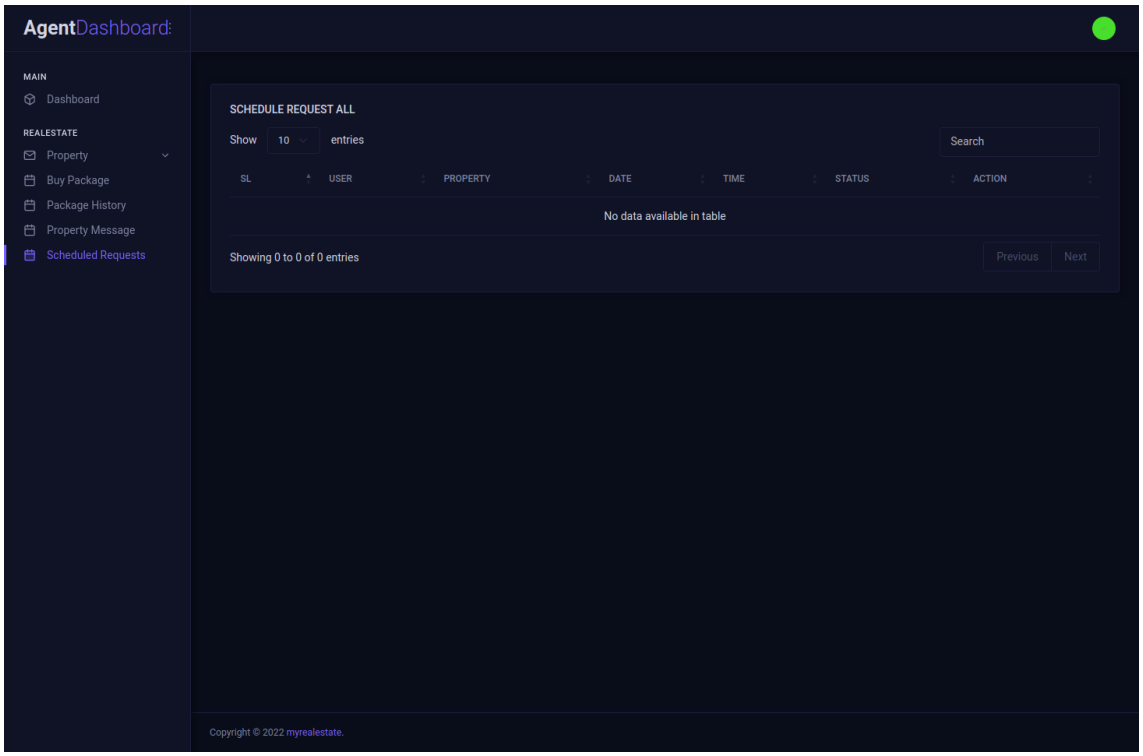
Εικόνα 25: Σελίδα / οθόνη επιλογής πακέτου

Μέσω του προσωπικού του διαχειριστικού περιβάλλοντος, ο πράκτορας μπορεί να ελέγξει τα μηνύματα που έχει λάβει από χρήστες που ενδιαφέρονται για κάποιο από τα ακίνητα που διαχειρίζεται. Αυτό του επιτρέπει να διαχειρίζεται αποτελεσματικά τα αιτήματα και τις επικοινωνίες με τους χρήστες που ενδιαφέρονται για τα ακίνητα που προσφέρει, και να ανταποκρίνεται στα ερωτήματα τους και στις αιτήσεις για προβολή ακινήτων.



Εικόνα 26: Οθόνη διαχείρισης μηνυμάτων

Οι πράκτορες μπορούν επίσης μέσω του διαχειριστικού περιβάλλοντος να δουν τα αιτήματα που έχουν από χρήστες για την μελλοντική επίσκεψη που θέλουν οι χρήστες να πραγματοποιήσουν σε κάποιο από τα ακίνητα που οι πράκτορες διαχειρίζονται και μέσω του διαχειριστικού περιβάλλοντος μπορούν να απαντήσουν σε αυτό το αίτημα. Στη συνέχεια, ένα email θα πάει στον χρήστη που πραγματοποίησε το αίτημα και έτσι θα ενημερωθεί ο χρήστης για την εξέλιξη του αιτήματος.

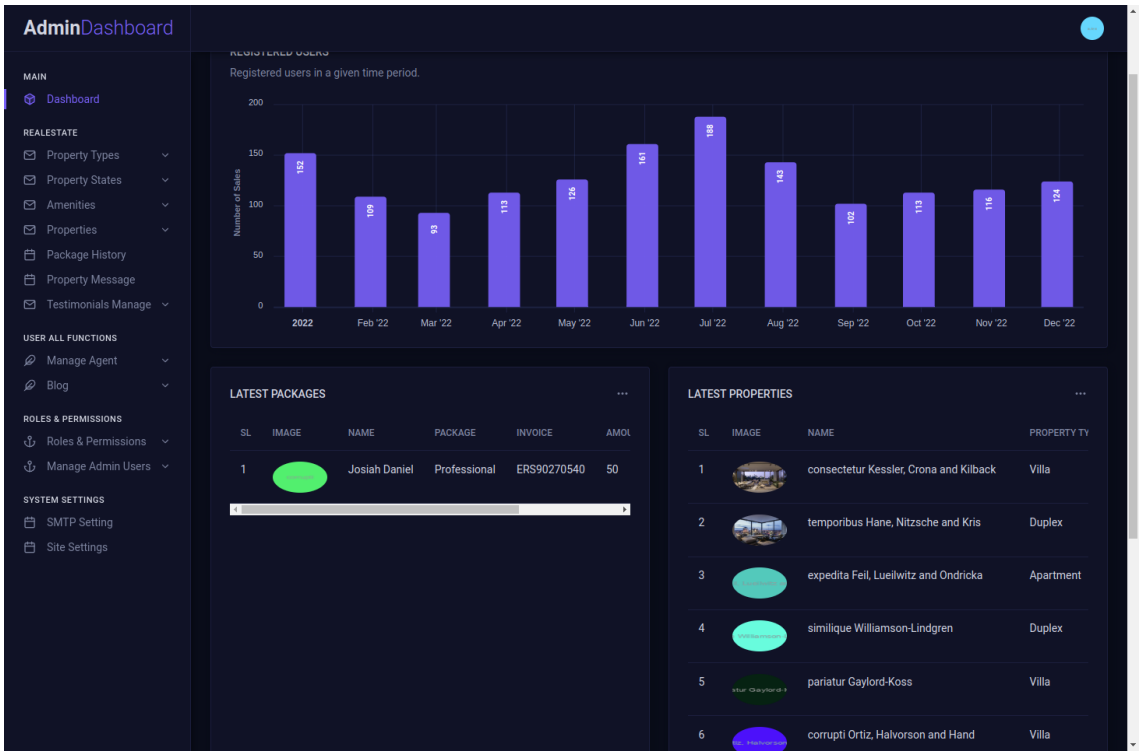


Εικόνα 27: Οθόνη διαχείρισης αιτημάτων

Διαχειριστής

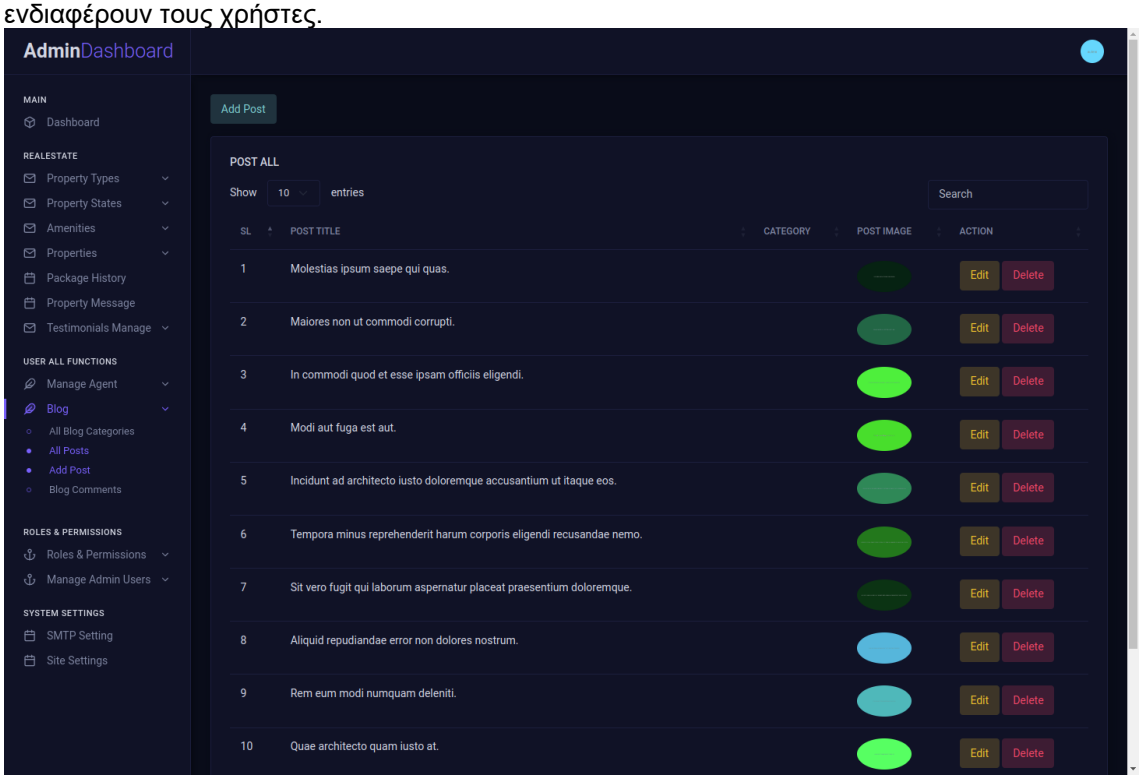
Ο ρόλος του διαχειριστή (admin) διαθέτει απεριόριστη πρόσβαση και δυνατότητες διαχείρισης όλων των καταχωρήσεων του συστήματος.

Επιπλέον, οι διαχειριστές έχουν τη δυνατότητα να απορρίψουν ή να εγκρίνουν αιτήσεις εγγραφής χρηστών, να απενεργοποιήσουν ή να ενεργοποιήσουν ακίνητα και να επεξεργαστούν τις τοποθεσίες, τις παροχές και τις κατηγορίες των ακινήτων.



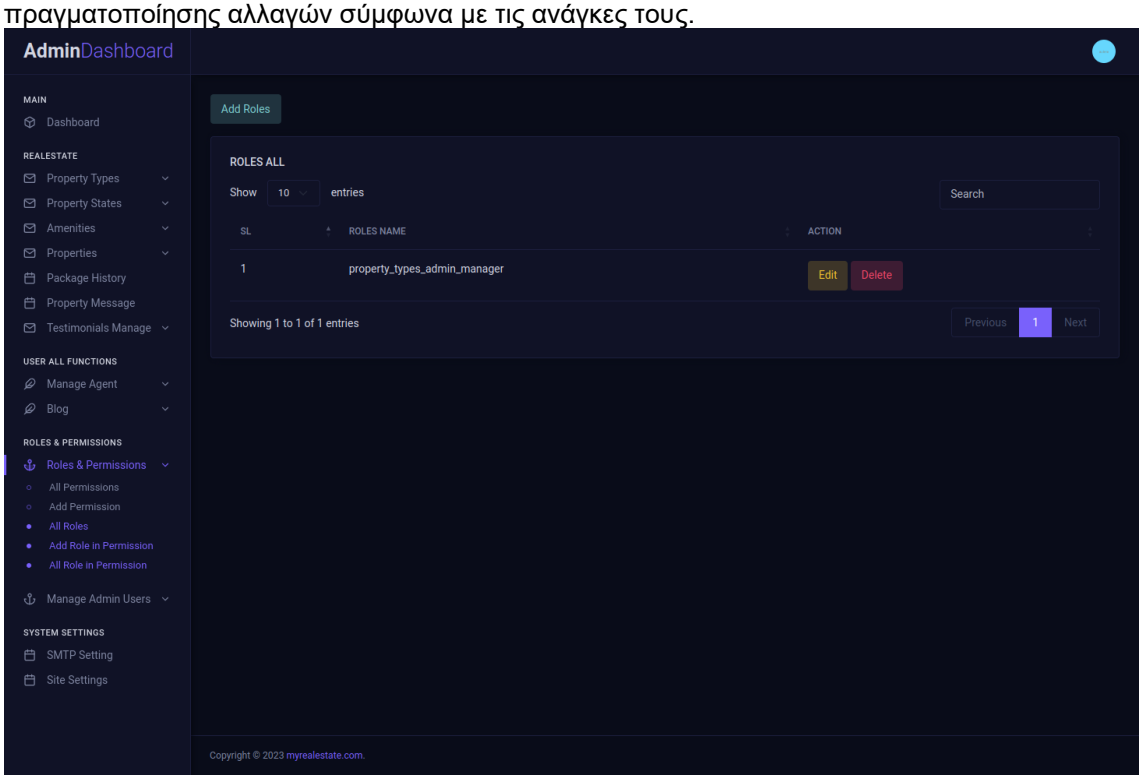
Εικόνα 28: Οθόνη διαχείρισης ακινήτων (διαχειριστής)

Επιπλέον, οι διαχειριστές διαθέτουν τη δυνατότητα blogging, όπου μπορούν να δημοσιεύουν νέα άρθρα και να διαχειρίζονται κατηγορίες άρθρων. Αυτά τα άρθρα θα είναι ορατά στην ιστοσελίδα, παρέχοντας πρόσθετο περιεχόμενο και πληροφορίες που μπορούν να ενδιαφέρουν τους χρήστες.



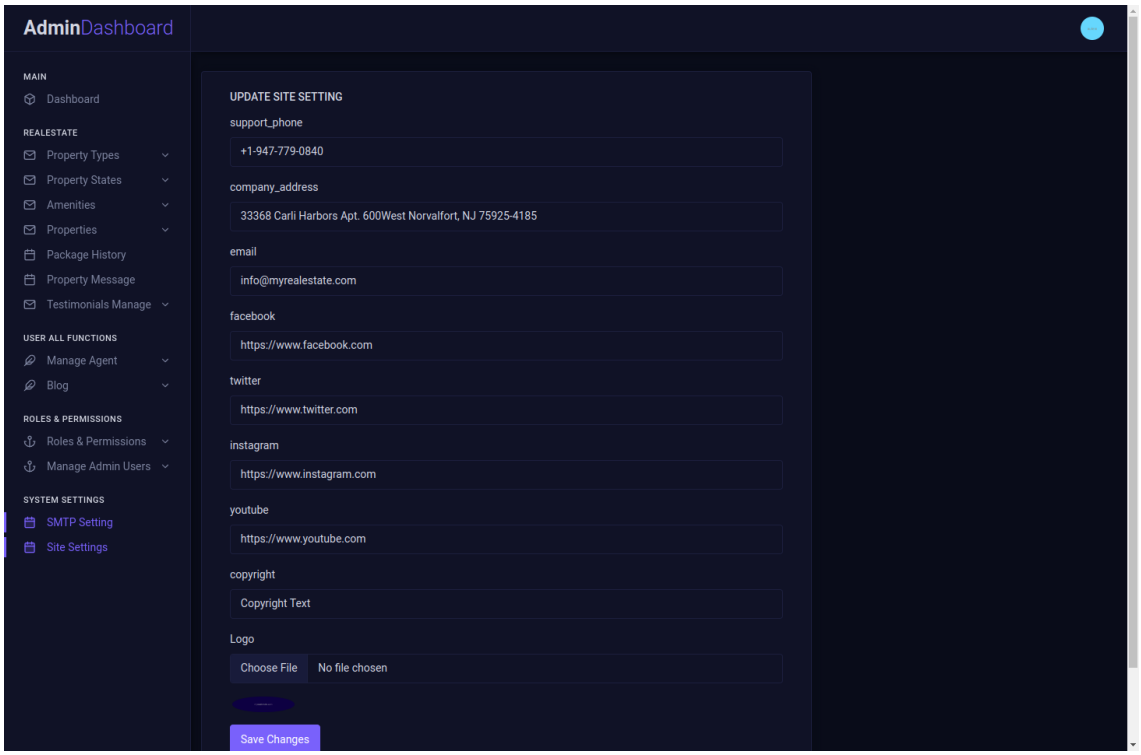
Εικόνα 29: Σελίδα / οθόνη διαχείρισης blogs (διαχειριστής)

Ακόμα, ο βασικός διαχειριστής του συστήματος (System Admin) μπορεί να δημιουργήσει νέους ρόλους διαχειριστών μέσω μιας ειδικής φόρμας και να προσθέσει εξατομικευμένα δικαιώματα και άδειες για κάθε ρόλο. Στη συνέχεια, μπορεί να αναθέσει αυτούς τους ρόλους σε άλλους διαχειριστές του συστήματος, δημιουργώντας έτσι ένα περιβάλλον πολλαπλών διαχειριστών με ελεγχόμενη πρόσβαση σε διάφορες λειτουργίες και δυνατότητα πραγματοποίησης αλλαγών σύμφωνα με τις ανάγκες τους.



Εικόνα 30: Σελίδα / οθόνη διαχείρισης χρηστών (διαχειριστής)

Τέλος, οι διαχειριστές έχουν πρόσβαση στις ρυθμίσεις του συστήματος, όπου μπορούν να ορίσουν βασικά χαρακτηριστικά, όπως τις ρυθμίσεις SMTP για την αποστολή email ή να προσαρμόσουν τις ρυθμίσεις της ιστοσελίδας για βέλτιστη λειτουργία. Αυτό επιτρέπει στους διαχειριστές να προσαρμόσουν το σύστημα σύμφωνα με τις ανάγκες τους και τις προτιμήσεις τους.



Εικόνα 31: Οθόνη ρυθμίσεων (διαχειριστής)

Τεχνολογίες που χρησιμοποιήθηκαν

Στο σύστημα μας, χρησιμοποιήσαμε σημαντικές τεχνολογίες που συνέβαλαν στη δημιουργία και λειτουργία της πλατφόρμας. Παρακάτω αναλύουμε τις τεχνολογίες που χρησιμοποιήσαμε:

Laravel 10.20.0: Η πλατφόρμα μας βασίζεται στο Laravel, ένα δημοφιλές PHP framework. Η έκδοση 10.20.0 του Laravel προσφέρει τελευταίες βελτιώσεις, ασφάλεια και επίδοση. Ο χρήστης του συστήματος επωφελείται από την ευκολία ανάπτυξης, τον έλεγχο και τη διαχείριση των εφαρμογών του.

PHP Version 8.1: Η έκδοση 8.1 της PHP αποτελεί τη γλώσσα προγραμματισμού πάνω στην οποία βασίζεται το Laravel. Η PHP 8.1 προσφέρει νέα χαρακτηριστικά, βελτιώσεις στην απόδοση και ασφάλεια, καθιστώντας την ιδανική για την ανάπτυξη του συστήματός μας.

MySQL Ver 15.1 Distrib 10.3.38-MariaDB: Χρησιμοποιήσαμε τη βάση δεδομένων MariaDB με την έκδοση 10.3.38. Η MariaDB είναι μια ανοικτού κώδικα σύστημα διαχείρισης βάσεων δεδομένων που βασίζεται στο MySQL, προσφέροντας σταθερότητα, απόδοση και αξιόπιστη αποθήκευση δεδομένων για το σύστημά μας.

Οι αναφερθείσες τεχνολογίες επιλέχθηκαν για να εξασφαλίσουν την αξιοπιστία, την ασφάλεια και την επίδοση του συστήματος μας, παρέχοντας την ιδανική υποδομή για τις λειτουργίες και τις απαιτήσεις των χρηστών μας.

Γιατί επιλέχτηκε ως βασικό framework της εφαρμογής το laravel

Στη δημιουργία μιας διαδικτυακής εφαρμογής για real estate (ακίνητα), η επιλογή του κατάλληλου εργαλείου ή τεχνολογίας είναι κρίσιμη για την επιτυχία του έργου. Εδώ είναι μερικοί λόγοι για τους οποίους επέλεξα το Laravel για την ανάπτυξη της διαδικτυακής εφαρμογής μου για ακίνητα:

Ευκολία Ανάπτυξης

Το Laravel είναι γνωστό για την ευκολία του στη χρήση και την ανάπτυξη. Επιτρέπει τη γρήγορη ανάπτυξη εφαρμογών, κάτι που είναι ιδιαίτερα επωφελές σε έναν τομέα όπως οι αγορές ακινήτων, όπου η ανταπόκριση στις αλλαγές της αγοράς είναι κρίσιμη.

Ενσωματωμένες Λειτουργίες Ασφαλείας

Η ασφάλεια είναι κρίσιμη στον τομέα των ακινήτων, όπου οι πληροφορίες των πελατών και άλλα ευαίσθητα δεδομένα πρέπει να προστατεύονται. Το Laravel παρέχει ενσωματωμένες λειτουργίες ασφαλείας που επιτρέπουν την αποτελεσματική προστασία των δεδομένων.

Προσαρμοσμένη Διαχείριση Βάσεων Δεδομένων

Ο τομέας των ακινήτων περιλαμβάνει τη διαχείριση πολύπλοκων δεδομένων. Το Laravel με τον πλούσιο ORM του, επιτρέπει την ευέλικτη διαχείριση των βάσεων δεδομένων, απλοποιώντας τη διαχείριση και την ανάλυση των δεδομένων των ακινήτων.

Ευελιξία και Επεκτασιμότητα

Το Laravel είναι ένα ευέλικτο και επεκτάσιμο framework που μπορεί να προσαρμοστεί για να ανταποκριθεί στις απαιτήσεις μιας ποικίλης γκάμας εφαρμογών, συμπεριλαμβανομένων των διαδικτυακών εφαρμογών ακινήτων.

Κοινότητα και Υποστήριξη

Υπάρχει μια μεγάλη κοινότητα προγραμματιστών Laravel που προσφέρει υποστήριξη, εκπαίδευση και πολλά πακέτα και επεκτάσεις που μπορούν να βοηθήσουν στην ανάπτυξη και τη βελτίωση της εφαρμογής μου.

Με βάση τα παραπάνω, το Laravel αποτέλεσε μια φυσική επιλογή για την ανάπτυξη της διαδικτυακής εφαρμογής μου στον τομέα των ακινήτων, προσφέροντας μια ισχυρή, ασφαλή και ευέλικτη λειτουργία στο τελικό αποτέλεσμα.

Μελλοντικές Επεκτάσεις της Web Εφαρμογής

Στον σύγχρονο κόσμο της τεχνολογίας, οι web εφαρμογές στον τομέα του real estate έχουν παρουσιάσει μια σημαντική εξέλιξη, αλλάζοντας τον τρόπο με τον οποίο οι άνθρωποι αγοράζουν, πωλούν και ενοικιάζουν ακίνητα. Ωστόσο, όπως με κάθε τεχνολογική εφαρμογή, υπάρχει πάντα χώρος για βελτίωση και επέκταση. Στην παρακάτω ενότητα, θα εξετάσουμε μερικές προτεινόμενες επεκτάσεις για την web εφαρμογή real estate, οι οποίες αναμένεται να προσφέρουν περισσότερη λειτουργικότητα, ευκολία στους χρήστες και να αναβαθμίσουν την εμπειρία τους.

1 Εικονικές Περιηγήσεις:

Προσθήκη λειτουργίας για εικονικές περιηγήσεις στα ακίνητα, ώστε οι χρήστες να μπορούν να δουν το εσωτερικό του ακινήτου μέσω της οθόνης τους.

Στην ψηφιακή εποχή μας, οι εικονικές περιηγήσεις έχουν γίνει ένα απαραίτητο εργαλείο για τον τομέα του real estate. Προσφέρουν στους πελάτες τη δυνατότητα να "επισκεφτούν" ένα ακίνητο από την άνεση του σπιτιού τους, χωρίς την ανάγκη για φυσική παρουσία. Αυτό είναι ιδιαίτερα χρήσιμο για πελάτες που βρίσκονται σε μεγάλες αποστάσεις ή για ακίνητα που δεν είναι εύκολα προσβάσιμα. Οι εικονικές περιηγήσεις αυξάνουν την εμπιστοσύνη των πελατών, καθώς τους δίνουν μια πιο ολοκληρωμένη εικόνα του ακινήτου και μπορούν να αυξήσουν την πιθανότητα μιας συναλλαγής.

2 Ενσωμάτωση Τεχνητής Νοημοσύνης:

Προτάσεις ακινήτων βάσει των προτιμήσεων και του ιστορικού αναζήτησης του χρήστη.

Η τεχνητή νοημοσύνη έχει τη δυνατότητα να αναλύει τις προτιμήσεις και το ιστορικό αναζήτησης των χρηστών, προσφέροντας προτάσεις που είναι πιο σχετικές με τις ανάγκες τους. Αυτό μπορεί να βελτιώσει σημαντικά την εμπειρία του χρήστη, καθώς οι πελάτες λαμβάνουν προτάσεις που είναι πιο πιθανό να τους ενδιαφέρουν, εξοικονομώντας χρόνο και προσπάθεια.

3 Χάρτης με Σημεία Ενδιαφέροντος:

Εμφάνιση κοντινών σημείων ενδιαφέροντος όπως σχολεία, σούπερ μάρκετ, νοσοκομεία κ.λπ. σε σχέση με το ακίνητο.

Η γεωγραφική τοποθεσία ενός ακινήτου είναι συχνά ένας από τους πιο σημαντικούς παράγοντες στην απόφαση αγοράς ή ενοικίασης. Ενσωματώνοντας έναν χάρτη που εμφανίζει κοντινά σημεία ενδιαφέροντος, όπως σχολεία, σούπερ μάρκετ και νοσοκομεία, οι πελάτες μπορούν να έχουν μια καλύτερη εικόνα της περιοχής και των παροχών που προσφέρει.

4 Φίλτρα Αναζήτησης Βάσει Περιοχής:

Δυνατότητα αναζήτησης ακινήτων με βάση συγκεκριμένες περιοχές ή γειτονιές.

Η δυνατότητα για εξατομικευμένη αναζήτηση βάσει περιοχής είναι ζωτικής σημασίας για τους χρήστες που έχουν συγκεκριμένες προτιμήσεις σχετικά με τον τόπο διαμονής τους. Είτε αναζητούν ένα ακίνητο κοντά στον χώρο εργασίας τους, είτε θέλουν να βρίσκονται σε μια συγκεκριμένη γειτονιά με συγκεκριμένες ανέσεις, τα φίλτρα αναζήτησης βάσει περιοχής παρέχουν την απαραίτητη ευελιξία.

5 Ενσωμάτωση AR (Augmented Reality):

Χρήση της τεχνολογίας AR για να βοηθήσει τους χρήστες να "βλέπουν" πώς θα φαινόταν το εσωτερικό ενός ακινήτου με δικά τους έπιπλα και διακοσμητικά.

Η τεχνολογία AR προσφέρει μια εντυπωσιακή νέα διάσταση στην εμπειρία των χρηστών. Επιτρέποντάς τους να "τοποθετούν" έπιπλα και διακοσμητικά στον χώρο ενός ακινήτου μέσω της οθόνης τους, οι πελάτες μπορούν να πάρουν μια πραγματική αίσθηση του πώς θα ήταν η ζωή τους σε αυτό το σπίτι.

6 Αξιολογήσεις Και Σχόλια:

Δυνατότητα για τους χρήστες να αφήνουν αξιολογήσεις και σχόλια για τα ακίνητα που έχουν επισκεφτεί.

Τα σχόλια και οι αξιολογήσεις από προηγούμενους πελάτες είναι ένας από τους καλύτερους τρόπους για να κερδίσετε την εμπιστοσύνη των μελλοντικών πελατών. Παρέχουν ειλικρινή ανατροφοδότηση για το ακίνητο και μπορούν να βοηθήσουν τους χρήστες να καταλάβουν τα θετικά και τα αρνητικά στοιχεία του κάθε σπιτιού.

7 Προσωπικό Dashboard για Πωλητές:

Ένα προσωπικό dashboard για τους πωλητές, όπου μπορούν να παρακολουθούν την πρόοδο των αγγελιών τους, τις προβολές, τα σχόλια και τις επαφές από ενδιαφερόμενους.

Ένα προσωπικό πιό ενημερομένο και λεπτομερές dashboard είναι ένα ισχυρό εργαλείο για τους πωλητές ακινήτων. Μπορούν να παρακολουθούν την απόδοση των αγγελιών τους, να βλέπουν πόσες φορές έχουν προβληθεί, ποιοι είναι ενδιαφερόμενοι και πότε πρέπει να ανανεώσουν τις αγγελίες τους.

8 Ενσωμάτωση Chatbot:

Ένας chatbot που θα βοηθά τους χρήστες στις απορίες τους και στην αναζήτηση ακινήτων.

Οι chatbots μπορούν να βελτιώσουν σημαντικά την εμπειρία του χρήστη, απαντώντας σε συχνές ερωτήσεις, βοηθώντας στην αναζήτηση ακινήτων και παρέχοντας άμεση υποστήριξη όταν οι χρήστες το χρειάζονται.

Βιβλιογραφία

https://laravel.com/. (2023). Ανάκτηση από https://laravel.com/
https://laravel.gr/. (2023). Ανάκτηση από https://laravel.gr/
https://www.spitogatos.gr/. (2023). Ανάκτηση από https://www.spitogatos.gr/
Learning MySQL and MariaDB. (2019). O'Reilly Media.
SCHOLTENS. (2023). *Mastering Laravel.* Sas155.
Stauffer, M. (2019). *Laravel: Up and Running.* O'Reilly Media, Incorporated.