Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών

ΗΥ359 – Διαδικτυακός Προγραμματισμός (Χειμερινό εξάμηνο 2023-2024)

Διδάσκων: Μ. Μουνταντωνάκης

Θέμα: Ομαδική Εργασία (project)

Pet-Care

Αλέξανδρος Φουρτούνης-5031
Γεωργία Μαρίνα Τσάντα-5020

Οδηγίες εγκατάστασης και εκτέλεσης

- Αρχικά απαραίτητη προϋπόθεση για τη δημιουργία αλλά και διαχείριση της βάσεως δεδομένων είναι η εγκατάσταση του προγράμματος XAMPP και η έναρξη των Modules: Apache, MySQL και Tomcat (σε Windows) ή Apache και MySQL (σε MacOs , Linux).
- Στο φάκελο που έχουμε παραδώσει περιέχεται το zip file με το project. Κάνουμε αποσυμπίεση και ανοίγουμε τον φάκελο στο Netbeans.
- Επιλέγουμε JDK (εμείς χρησιμοποιούμε JDK 11), εγκαθιστούμε τον Tomcat και ανοίγοντας το Netbeans επιλέγουμε Window->Services->Add server και επιλέγουμε το Apache Tomcat or TomEE, τοποθετώντας στο Server Location το path που το έχουμε εγκαταστήσει.
- Έπειτα μπορούμε να τρέχουμε το project πατώντας δεξί κλικ και run.

Γενικές πληροφορίες για την εργασία

Το Pet Care έιναι ένα Maven structured java WebApp. Σκοπός του είναι να επιτρέψει σε ιδιοκτήτες κατοικιδίων να αφήνουν τα κατοικίδια τους για φιλοξενία όσο λείπουν και να παρέχει το κατάλληλο σύστημα γι αυτή την υποδομή.

Οι τύποι χρηστών της εφαρμογής είναι:

1. Κεντρικός administrator
2. Pet Owner
3. Pet Keeper
4. Επισκέπτες

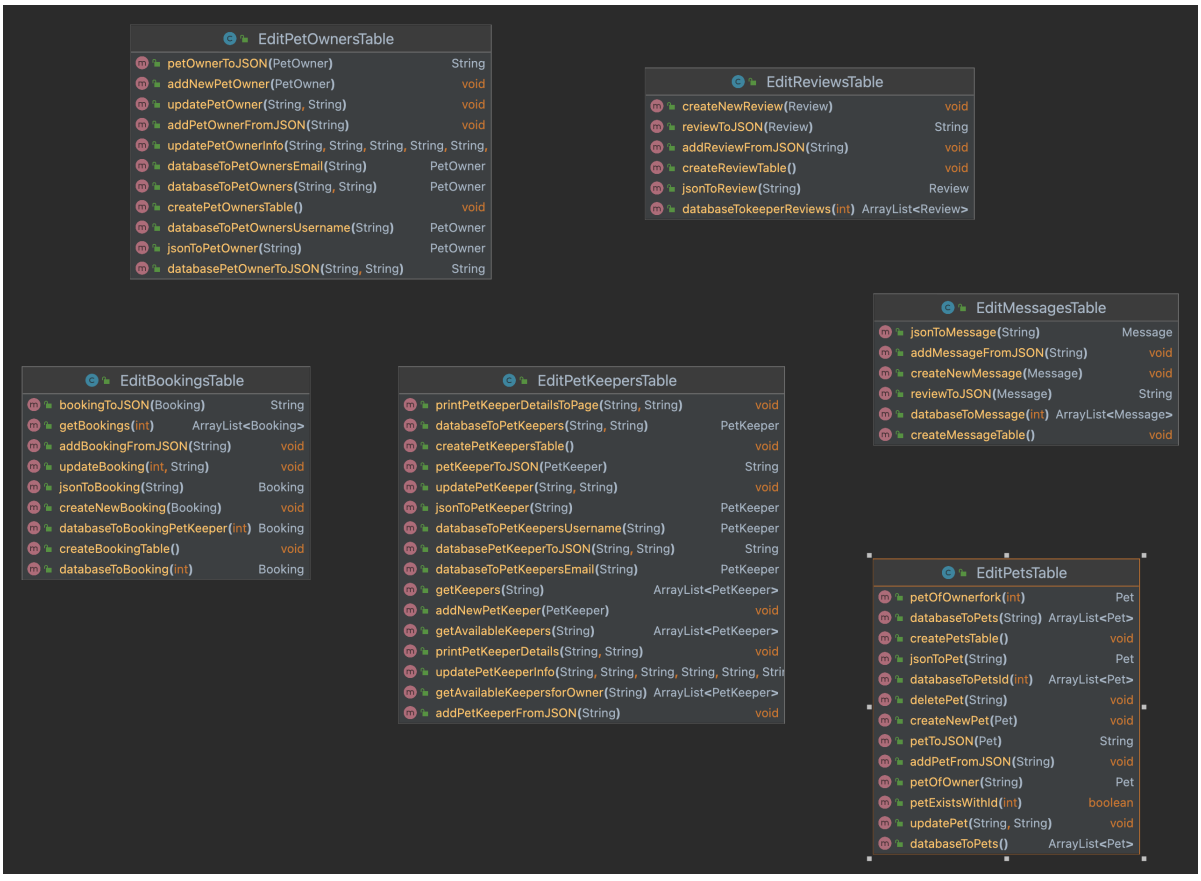
Θα χωρίσουμε την αναφορά για τη σχεδίαση στα μέρη Backend και Frontend.

Κλάσεις και μέθοδοι

Στις κλάσεις της βάσης δεδομένων που μας είχε δωθεί, τροποποιήσαμε/ διορθώσαμε κάποιες από τις υπάρχουσες μεθόδους και προσθέσαμε ορισμένες νέες που μας βοήθησαν στην υλοποίηση μας.

Στην επόμενη σελίδα μπορείτε να δείτε τα UML Diagrams των tables της βάσης δεδομένων και main classes.

Main Classes UML Diagram



Database Tables UML Diagram

Back end

Στο back end χρησιμοποίησαμε:

- Java
- MySQL

Για τη διαχείριση της βάσης δεδομένων χρησιμοποιούσαμε τις κλάσεις Edit<Name>Table, οι οποίες εκτελούσαν ερωτήματα SQL προς τη βάση δεδομένων, setters, getters και τα λοιπά.

Το project μας περιέχει 18 συνολικά Servlets, που χειρίζονται Get, Post requests. Ο κύριος στόχος τους είναι να δημιουργήσουν, να ενημερώσουν ή να επιστρέψουν δεδομένα από τη βάση δεδομένων στον πελάτη χρησιμοποιώντας κλάσεις της βάσης δεδομένων και μορφοποιώντας τα αποτελέσματα όπως απαιτείται. Καλούνται με AJAX και λαμβάνουν πάντα μορφοποιημένα δεδομένα Json στην περίπτωση Post. Πιο συγκεκριμένα έχουμε:

1. **ChatGPTAPI**: Χρησιμοποιείται το ChatGTP ώστε ο pet keeper να μπορεί να στείλει ερωτήσεις και να παίρνει απαντήσεις. Στέλνει ένα prompt ενός pet keeper στο OpenAI API χρησιμοποιώντας HTTP POST, λαμβάνει μια απάντηση και εξάγει το μήνυμα που δημιουργείται από την απάντηση JSON.
2. **GetAvailablePetKeepers**: Ανακτά τους διαθέσιμους pet keepers με βάση τον τύπο κατοικίδιου του συνδεδεμένου pet owner. Χειρίζεται τη μέθοδο HTTP GET και επιστρέφει έναν πίνακα JSON με σχετικές λεπτομέρειες.
3. **GetBookings**: Ανακτά πληροφορίες κράτησης για τον συνδεδεμένο pet keeper. Χειρίζεται τη μέθοδο HTTP GET, επιστρέφοντας έναν πίνακα JSON που περιέχει λεπτομέρειες κράτησης.
4. **GetBookingsOwner**: διαχειρίζεται τις κρατήσεις για τον συνδεδεμένο pet owner. Χειρίζεται και τις δύο μεθόδους HTTP GET και POST. Στη μέθοδο

GET, ανακτά τις πληροφορίες κράτησης που σχετίζονται με το αναγνωριστικό του ιδιοκτήτη του κατοικίδιου και απαντά με έναν πίνακα JSON. Στη μέθοδο POST, ενημερώνει την κατάσταση μιας κράτησης σε "finished" με βάση το παρεχόμενο αναγνωριστικό κράτησης.

5. **GetPetKeeper**: διεκπεραιώνει αιτήματα HTTP GET για την ανάκτηση πληροφοριών pet keeper. Αναμένει παραμέτρους για το όνομα χρήστη και τον κωδικό πρόσβασης, επικυρώνει τα διαπιστευτήρια και αποκρίνεται με δεδομένα JSON που αντιπροσωπεύουν τον pet keeper.
6. **GetProfileInfo**: Ανακτά τις προσωπικές πληροφορίες προφίλ ενός pet keeper, χειρίζεται το HTTP GET. Ελέγχει εάν ένας pet keeper είναι συνδεδεμένος, ανακτά τα στοιχεία του από τη βάση δεδομένων με βάση το session attribute, και αποκρίνεται με τις πληροφορίες προφίλ σε μορφή JSON.
7. **GetReviews**: Ανακτά κριτικές για έναν συνδεδεμένο pet keeper, χειρίζεται το HTTP GET. Παίρνει το αναγνωριστικό του pet keeper με βάση το session attribute, ανακτά τις κριτικές που σχετίζονται με αυτόν από τη βάση δεδομένων και απαντά με τις κριτικές σε μορφή JSON.
8. **HandleGPT**: Χειρίζεται αιτήματα HTTP POST, εξάγοντας το περιεχόμενο JSON από το αίτημα. Στη συνέχεια αναλύει το JSON για να λάβει ένα ερώτημα χρήστη και χρησιμοποιεί το ChatGPTAPI για να δημιουργήσει μια απάντηση χρησιμοποιώντας το μοντέλο ChatGPT. Τέλος, στέλνει την απόκριση που δημιουργήθηκε πίσω στην απόκριση HTTP.
9. **KeepersList**: Ανταποκρίνεται σε αιτήματα HTTP GET ανακτώντας μια λίστα pet keeper από τη βάση δεδομένων χρησιμοποιώντας το EditPetKeepersTable. Χρησιμοποιείται για τη λειτουργία του απλού χρήστη.
10. **Login**: Χειρίζεται τόσο αιτήματα GET όσο και POST για έναν pet keeper. Στη μέθοδο POST, επεξεργάζεται τις προσπάθειες σύνδεσης ελέγχοντας το παρεχόμενο όνομα χρήστη και κωδικό πρόσβασης στη βάση δεδομένων. Εάν είναι επιτυχής, ορίζει τον χρήστη ως συνδεδεμένο και επιστρέφει τις

πληροφορίες χρήστη σε μορφή JSON. Στη μέθοδο GET, επαληθεύει εάν ένας χρήστης είναι συνδεδεμένος και επιστρέφει το όνομα χρήστη.

11. **LoginOwner**: Όπως η Login αλλά για έναν pet owner.
12. **Logout**: Χειρίζεται αιτήματα POST για αποσύνδεση χρηστών. Εάν ένας χρήστης είναι συνδεδεμένος, ακυρώνει τη συνεδρία, αποσυνδέοντας ουσιαστικά τον χρήστη.
13. **MakeBooking**: Χειρίζεται αιτήματα POST για τη δημιουργία νέων εγγραφών κράτησης από έναν pet owner. Αναλύει τα δεδομένα JSON από το αίτημα, δημιουργεί μια νέα κράτηση, ορίζει την προεπιλεγμένη κατάστασή της σε "requested" και την προσθέτει στη βάση δεδομένων.
14. **MakeReview**: Χειρίζεται αιτήματα POST για τη δημιουργία νέων αξιολογήσεων από έναν pet owner. Αναλύει τα δεδομένα JSON από το αίτημα, τα μετατρέπει σε αντικείμενο Review και προσθέτει την κριτική στη βάση δεδομένων χρησιμοποιώντας το EditReviewsTable.
15. **OnwersProfile**: Χειρίζεται αιτήματα GET και POST που σχετίζονται με το προφίλ του pet owner. Για ένα αίτημα GET, ανακτά τις πληροφορίες προφίλ του με βάση τον συνδεδεμένο χρήστη και απαντά με την αναπαράσταση JSON. Για ένα αίτημα POST, ενημερώνει τις πληροφορίες προφίλ του στη βάση δεδομένων χρησιμοποιώντας τα λαμβανόμενα δεδομένα JSON και απαντά με τις ενημερωμένες πληροφορίες.
16. **Register**: Διεκπεραιώνει την εγγραφή χρηστών τόσο για τους pet keepers όσο και για τους pet owners. Επεξεργάζεται αιτήματα POST που περιέχουν δεδομένα JSON που αντιπροσωπεύουν είτε έναν owner είτε έναν keeper. Ελέγχει ποιον αφορά η εγγραφή και την εκτελεί ανάλογα. Επαληθεύει τη μοναδικότητα του ονόματος χρήστη και του email πριν προσθέσει τον νέο χρήστη στη βάση δεδομένων.
17. **RegisterPet**: Χειρίζεται την εγγραφή κατοικίδιων ζώων. Επεξεργάζεται αιτήματα POST που περιέχουν δεδομένα JSON που αντιπροσωπεύουν ένα

κατοικίδιο. Ελέγχει εάν ένα κατοικίδιο με το παρεχόμενο αναγνωριστικό υπάρχει ήδη στη βάση δεδομένων πριν προσθέσει το νέο κατοικίδιο.

18. **UpdateProfileInfo**: Επεξεργάζεται αιτήματα POST για να ενημερώσει τις πληροφορίες προφίλ ενός συνδεδεμένου pet keeper. Ανακτά τα δεδομένα JSON από το αίτημα, ενημερώνει τις πληροφορίες του στη βάση δεδομένων και απαντά με τις ενημερωμένες πληροφορίες.

Front end

Στο front end χρησιμοποίησαμε:

- HTML
- CSS
- Javascript
- Bootstrap
- JQuery
- Ajax

Έχουμε δύο φακέλους τους PetKeeper και PetOwner και κάποια αρχεία έξω από αυτά.

HTML

Έχουμε το index.html για την αρχική σελίδα του Webapp μας, που περιέχει τα sections About us, Login , Register και Our Pet Keepers. Από εκέι και πέρα έχουμε δύο φακέλους τους PetKeeper και PetOwner, οι οποίοι περιέχουν τα αντίστοιχα html για έναν keeper και owner. Γενικότερα για την εισαγωγή στοιχείων χρησιμοποιήσαμε forms.

Στον PetKeeper περιέχονται τα html petkeeperslogin, petkeeper, help, profile, με κύρια αρχική σελίδα το petkeeper.html και τα υπόλοιπα για τις λειτουργίες του.

Στον PetOwner περιέχονται τα html pologin, petowner, messages, profile, petkeepers, petregister, ownersbooking, ownersreview, με κύρια αρχική σελίδα το petowner.html και τα υπόλοιπα για τις λειτουργίες του.

CSS

Έχουμε το style.css για την αρχική μας σελίδα και το register.css για τη φόρμα εγγραφής. Στον PetKeeper, το petkeepers_styles.css για όλα τα html του keeper και

το styles.css για την φόρμα login. Στον PetOwner, το petowner.css για τια όλα τα html εκτός από τις φόρμες που χρησιμοποείται το register.css .

Javascript

Στην αρχική μας σελίδα χρησιμοποιούμε το ajax.js και το register.js για την εγγραφή χρήστη.

Στον PetKeeper έχουμε τα script.js για το login, το petkeeperlogin.js για την αρχική σελίδα του και τη διαχείριση του ChatGPT, ενώ τα profile.js, bookings.js για λειτουργίες του όπως η ενημέρωση στοιχείων του, η προβολή κρατήσεων κτλ.

Στον PetOwner έχουμε το loginj για το login και το petowner.js που περιέχει όλες τισ συναρτήσης για τις λειτουργίες του όπως καταχώρηση κατοικιδίων, κριτικής, κράτησης , προβολή στοιχείων του και ενημέρωση του κτλ.

Ajax

Χρησιμοποιείται σχεδόν σε όλες τις συναρτήσεις μας, ώστε να στέλνουμε requests στον server χωρίς να ανανεώνεται η σελίδα.

Γενικότερα

Στην υλοποίηση μας δεν έχουμε χρησιμοποιήσει REST μόνο Servlets.

Επίσης δεν χρειάστηκε να κάνουμε αλλαγές στο σχήμα της βάσης.

Λειτουργίες που υλοποιήθηκαν

User:

✓ Εύρεση και προβολή pet keeper (χωρίς τη δυνατότητα κράτησης)

✓ Αρχική σελίδα συστήματος

PetOwner:

✓ Ενημέρωση και προβολή στοιχείων

✓ Καταχώρηση κατοικιδίου

✓ Εύρεση και προβολή pet keepers (χωρίς ωστόσο με βάση τα κριτήρια

✓ Επιλογή, Προβολή Φιλοξενίας

✓ Κριτική Pet Keeper (μετά το πέρας της φιλοξενίας).

PetKeeper:

✓ Ενημέρωση στοιχείων/Καταχώρηση πληροφοριών για το χώρο φιλοξενίας

✓ Διαχείριση κρατήσεων

✓ Ερωτήσεις προς chatgpt για βοήθεια στα κατοικίδια που θα φιλοξενήσουν/ φιλοξενούν

✓ Μήνυμα προς ιδιοκτήτη

✓ Προβολή στατιστικών

Administrator:

✓ Ξεχωριστό Login

✓ Προβολή Χρηστών με δυνατότητα διαγραφής

✓ Προβολή στατιστικών