Αναφορά Project 359 (Διαδικτυοκεντρικός προγραμματισμός)

1. Εξήγηση του Project
2. Οι κλάσεις και τις μέθοδοι που χρησιμοποιήσαμε
3. Η γενικότερη αρχιτεκτονική του συστήματος.
    a. ● Πόσα, ποια και με τι αρμοδιότητα servlets χρησιμοποιήσατε (η ότι άλλη τεχνολογία χρησιμοποιήσατε)
    b. ● Πόσα, ποια και με τι αρμοδιότητα JavaScript Libraries/html/jsp χρησιμοποιήσατε
    c. ● Πόσα, ποια και με τι αρμοδιότητα css χρησιμοποιήσατε
    d. ● Τι κάνατε στην πλευρά του client με javascript
    e. ● Ποια είναι τα Ajax τμήματα της εφαρμογής σας;
    f. ● Που χρησιμοποιήσετε REST requests?
    g. ● Ποια APIs χρησιμοποιήσατε;
    h. ● Αν χρειάστηκε να κάνετε αλλαγές/προσθήκες στο σχήμα της βάσης

Το project υλοποιήθηκε από
Θεολόγος Κοκκινέλλης
Φώτης Πελαντάκης

1.

Το project "Pet Care" αποτελεί σύστημα που σκοπεύει στην εξυπηρέτηση τις ανάγκες των pet owners και των pet keepers. Ο κεντρικός στόχος του συστήματος είναι να διευκολύνει τη διαδικασία φιλοξενίας κατοικιδίων από τους keepers για τα κατοικίδια των owners..

Στα πλαίσια του έργου, καθορίζονται πέντε βασικά σενάρια, καλύπτοντας τις ανάγκες τόσο των ιδιοκτητών όσο και των φροντιστών. Ανάμεσα σε αυτά, συμπεριλαμβάνονται η δυνατότητα επιλογής keeper, η καταχώρηση πληροφοριών για το κατοικίδιο, η διαχείριση κρατήσεων(bookings)), η επικοινωνία μεταξύ owner και keeper, και η δυνατότητα κριτικής μετά τη φιλοξενία(reviews).

Το σύστημα περιλαμβάνει τέσσερις κύριους τύπους χρηστών: τον κεντρικό διαχειριστή (admin), τον ιδιοκτήτη κατοικίδιου (owner), τον άνθρωπο που θα φιλοξενήσει το κατοικίδιο (keeper) και τους επισκέπτες. Κάθε τύπος χρήστη έχει συγκεκριμένες λειτουργίες που τον εξυπηρετούν, με τον κεντρικό διαχειριστή να έχει πλήρη έλεγχο του συστήματος, τη δυνατότητα διαγραφής χρηστών και την προβολή στατιστικών.

Συγκεκριμένα οι δυνατότητες - λειτουργίες του κάθε τύπου χρήστη είναι:

- **Λειτουργίες Administrator**
    - ● Ξεχωριστό Login
    - ● Προβολή Χρηστών με δυνατότητα διαγραφής
    - ● Προβολή στατιστικών
- **Λειτουργίες Pet Keeper**
    - ● 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
- **Λειτουργίες Pet Owner**
    - ● Ενημέρωση στοιχείων
    - ● Καταχώρηση κατοικιδίου
    - ● Εύρεση και προβολή pet keepers με βάση διάφορα κριτήρια
    - ● Επιλογή, Προβολή Φιλοξενία
    - ● Κριτική Pet Keeper (μετά το πέρας της φιλοξενίας, αν θέλει).
- **Λειτουργίες Επισκέπτη**
    - ● Εύρεση και προβολή pet keeper (χωρίς τη δυνατότητα κράτησης)
    - ● Αρχική σελίδα συστήματος

3 α)

1. **add_booking:** Αυτό το servlet χρησιμοποιείται για την προσθήκη νέας κράτησης στο σύστημα.
2. **add_pet_keeper:** Αυτό το servlet εκτελεί τη λειτουργία προσθήκης νέου φροντιστή κατοικιδίων στο σύστημα.
3. **add_pet_owner:** Αυτό το servlet υλοποιεί τη λειτουργία προσθήκης νέου ιδιοκτήτη κατοικιδίων. Αποθηκεύει τα προσωπικά στοιχεία του ιδιοκτήτη στη βάση δεδομένων.
4. **ChatGPTQueryServlet:** Αυτό το servlet υλοποιεί την επικοινωνία με το ChatGPT
5. **countCats:** Αυτό το servlet υπολογίζει τον συνολικό αριθμό των γάτων στο σύστημα.
6. **countDogs:** Αντίστοιχα με το προηγούμενο, αυτό το servlet υπολογίζει τον συνολικό αριθμό των σκύλων στο σύστημα.
7. **countKeepers:** Αυτό το servlet επιστρέφει τον συνολικό αριθμό των pet keepers στο σύστημα.
8. **countOwners:** Αντίστοιχα, αυτό το servlet επιστρέφει τον συνολικό αριθμό των pet owners.
9. **Delete_pet:** Αυτό το servlet διαγράφει ένα κατοικίδιο από τη βάση δεδομένων, με βάση το ID του.
10. **DeleteDB:** Αυτό το servlet εκτελεί τη λειτουργία διαγραφής όλων των δεδομένων από τη βάση δεδομένων.
11. **deleteKeeper:** Αυτό το servlet διαγράφει έναν φροντιστή από τη βάση δεδομένων, με βάση το ID του.
12. **deleteOwner:** Αντίστοιχα, αυτό το servlet διαγράφει έναν ιδιοκτήτη από τη βάση δεδομένων, με βάση το ID του.
13. **earnings:** Αυτό το servlet υπολογίζει τα κέρδη των keepers κατοικιδίων, παρέχοντας πληροφορίες σχετικά με τα έσοδα που προκύπτουν από τις φιλοξενίες.
14. **EditKeeper:** Αυτό το servlet επιτρέπει την επεξεργασία πληροφοριών του keeper, όπως το όνομα, η διεύθυνση κ.λπ.
15. **EditOwner:** Αντίστοιχα, αυτό το servlet επιτρέπει την επεξεργασία πληροφοριών του owner, όπως το όνομα, η διεύθυνση κ.λπ.
16. **emailExistance:** Αυτό το servlet ελέγχει την ύπαρξη ενός συγκεκριμένου email στο σύστημα κατά την εγγραφή του keeper ή owner.
17. **find_booking_id:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με μια κράτηση βάσει του ID της.
18. **find_owner_lat_lon:** Αυτό το servlet επιτρέπει την εύρεση ιδιοκτήτη βάσει των γεωγραφικών του συντεταγμένων.

19. **find_ownerID:** Αυτό το servlet επιστρέφει πληροφορίες για έναν ιδιοκτήτη βάσει του ID του.
20. **find_pet_info:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με ένα κατοικίδιο βάσει του ID του.
21. **Get_pet:** Αυτό το servlet επιστρέφει πληροφορίες για ένα κατοικίδιο βάσει του ID του.
22. **GetKeeper:** Αυτό το servlet επιστρέφει πληροφορίες για ένα φροντιστή βάσει του ID του.
23. **GetKeeperInfo:** Αυτό το servlet επιστρέφει λεπτομερείς πληροφορίες για ένα φροντιστή βάσει του ID του.
24. **GetMessagesServlet:** Αυτό το servlet χρησιμοποιείται για τη λήψη μηνυμάτων μεταξύ ιδιοκτήτη και keeper.
25. **GetOwner:** Αυτό το servlet επιστρέφει πληροφορίες για έναν ιδιοκτήτη βάσει του ID του.
26. **GetOwnerId:** Αυτό το servlet επιστρέφει το ID ενός ιδιοκτήτη βάσει των παρεχόμενων πληροφοριών.
27. **GetPetKeeper:** Αυτό το servlet παρέχει περισσότερες λεπτομέρειες για έναν φροντιστή, συμπεριλαμβανομένων των κατοικιδίων που φροντίζει και των αξιολογήσεών του.
28. **Id_existance:** Αυτό το servlet ελέγχει την ύπαρξη ενός συγκεκριμένου ID στο σύστημα.
29. **InitDB:** Αυτό το servlet χρησιμοποιείται για την αρχικοποίηση της βάσης δεδομένων με αρχικά δεδομένα.
30. **New_pet:** Αυτό το servlet προσθέτει ένα νέο κατοικίδιο στο σύστημα.
31. **Put_pet:** Αυτό το servlet εκτελεί τη λειτουργία ενημέρωσης πληροφοριών για ένα κατοικίδιο.
32. **SendMessagesServlet:** Αυτό το servlet χρησιμοποιείται για την αποστολή μηνυμάτων μεταξύ ιδιοκτήτη και φροντιστή.
33. **SendReviewsServlet:** Αυτό το servlet επιτρέπει στον ιδιοκτήτη να στείλει κριτική στον φροντιστή μετά το πέρας της φιλοξενίας.
34. **show_keepers:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με τους διαθέσιμους φροντιστές.
35. **show_keepers_with_owner_filters:** Αυτό το servlet επιστρέφει φροντιστές με βάση τα φίλτρα που έχει ορίσει ο ιδιοκτήτης.
36. **show_owners:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με τους εγγεγραμμένους ιδιοκτήτες.
37. **ShowBookingServlet:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με τις υπάρχουσες κρατήσεις στο σύστημα.
38. **StatusOfkeeperId:** Αυτό το servlet επιστρέφει την κατάσταση ενός φροντιστή βάσει του ID του.
39. **StatusOfkeeperIdServlet:** Αντίστοιχο με το προηγούμενο, αυτό το servlet υπολογίζει την κατάσταση ενός φροντιστή.

40. **UpdateBookingStatus:** Αυτό το servlet επιτρέπει την ενημέρωση της κατάστασης μιας κράτησης.
41. **UserInfo:** Αυτό το servlet επιστρέφει τις βασικές πληροφορίες ενός χρήστη.
42. **UserInfoOwner:** Αυτό το servlet επιστρέφει πληροφορίες για έναν ιδιοκτήτη.
43. **usernameExistance:** Αυτό το servlet ελέγχει την ύπαρξη ενός συγκεκριμένου ονόματος χρήστη στο σύστημα.
44. **viewKeepers:** Αυτό το servlet επιστρέφει πληροφορίες σχετικά με τους φροντιστές.

Μέθοδοι στην ajax.js:

sendReview(): Υποβάλλει μια κριτική για έναν χρήστη. Συλλέγει το κείμενο της κριτικής και τη βαθμολογία από την ιστοσελίδα, στέλνει αυτά τα δεδομένα στο 'SendReviewsServlet' χρησιμοποιώντας ένα POST request και ενημερώνει την ιστοσελίδα με βάση το response του server.

finished(): Μια απλή συνάρτηση που καλεί την updateBooking_status με την κατάσταση "finished".

updateBooking_status(status): Ενημερώνει την κατάσταση της κράτησης στέλνοντας ένα GET request στην 'UpdateBookingStatus' με το αναγνωριστικό της κράτησης και τη νέα κατάσταση. Στη συνέχεια ενημερώνει την ιστοσελίδα με το αποτέλεσμα.

find_owner_id(callback): Λαμβάνει το αναγνωριστικό owner για έναν συνδεδεμένο χρήστη στέλνοντας ένα GET request στο 'find_ownerID'. Σε περίπτωση επιτυχίας, αποθηκεύει το αναγνωριστικό ιδιοκτήτη και εκτελεί την συνάρτηση callback.

show_keepers_for_owner(): Ανακτά και εμφανίζει τους κηδεμόνες κατοικίδιων ζώων που είναι κατάλληλοι για έναν ιδιοκτήτη. Λαμβάνει το αναγνωριστικό του ιδιοκτήτη και τον τύπο του κατοικίδιου ζώου, και στη συνέχεια φορτώνει μια λίστα με κηδεμόνες κατοικίδιων ζώων από τον διακομιστή, ελέγχοντας τη διαθεσιμότητά τους.

OwnerBooking(): Χειρίζεται τη διαδικασία κράτησης για έναν ιδιοκτήτη κατοικίδιου ζώου. Συγκεντρώνει τα στοιχεία της κράτησης και τα αποστέλλει στο τελικό σημείο 'add_booking' χρησιμοποιώντας ένα POST request, και στη συνέχεια εμφανίζει το αποτέλεσμα στην ιστοσελίδα.

initDB(): Ξεκινά την εγκατάσταση της βάσης δεδομένων στέλνοντας ένα GET request στην 'InitDB' και ενημερώνει την ιστοσελίδα ανάλογα με το αν το initialization ήταν επιτυχέςς ή όχι.

find_pet(): Ανακτά τα στοιχεία κατοικίδιων ζώων για μια κράτηση, στέλνοντας ένα GET request σε ένα συγκεκριμένο σημείο και ενημερώνοντας την ιστοσελίδα με πληροφορίες για τα κατοικίδια ζώα.

find_status(): Λαμβάνει την κατάσταση διαθεσιμότητας ενός pet keeper.

SaveKeeperId(): Αποθηκεύει το αναγνωριστικό του επιλεγμένου pet keeper, διευκολύνοντας τη διαδικασία κράτησης με τη σύνδεση ενός συγκεκριμένου keeper με μια κράτηση.

add_petkeeper(): Προσθέτει έναν νέο pet keeper στο σύστημα, συλλέγοντας τα στοιχεία του και στέλνοντάς τα στον διακομιστή.

createJson_for_pet_keeper(): Διαμορφώνει τα δεδομένα του pet keeper σε αντικείμενο JSON, προετοιμάζοντάς τα για μετάδοση ή αποθήκευση.

add_petowner(): Καταχωρεί έναν νέο pet owner, αποστέλλοντας τα στοιχεία του ιδιοκτήτη στον διακομιστή.

loadBookingData(): Φορτώνει δεδομένα που σχετίζονται με κρατήσεις, για αναθεώρηση ή τροποποίηση.

DisplayPetKeepers(): Παρουσιάζει μια λίστα με τους Pet Keepers χρήσιμη για σκοπούς επιλογής ή ενημέρωσης.

fetchPetOwners(): Ανακτά πληροφορίες σχετικά με τους pet owners για σκοπούς επιλογής ή ενημέρωσης.

sendMessage(): Ενεργοποιεί τη λειτουργία ανταλλαγής μηνυμάτων, χειριζόμενη την αποστολή μηνυμάτων μεταξύ pet keepers-owners.

sendQuestion(): Αποστέλλει ερωτήσεις που δημιουργούνται από τον χρήστη στο chatGPT.

sendPredefinedQuestion(): Χειρίζεται την αποστολή προκαθορισμένων ερωτήσεων (πχ Τι ρατσα ειναι ο σκυλος κλπ).

find_booking_id(): Αναζητά ένα συγκεκριμένο αναγνωριστικό κράτησης για την τροποποίηση κρατήσεων.

sendMessage_to_keeper(): Στένει μηνύματα προς τους pet keepers.

showKeeperInfo(): Εμφανίζει λεπτομερείς πληροφορίες σχετικά με έναν pet keeper.

saveAllBookings(): Διατηρεί τις ενημερώσεις όλων των κρατήσεων, διασφαλίζοντας την ακεραιότητα των δεδομένων και τα ενημερωμένα αρχεία.

Save_new_detailsOwners(): Ενημερώνει και αποθηκεύει νέα στοιχεία για τους pet owners, αντικατοπτρίζοντας τις αλλαγές στις πληροφορίες του owner.

Save_new_detailsKeepers(): Ενημερώνει και αποθηκεύει νέες πληροφορίες για τους pet keepers, διατηρώντας τα τρέχοντα προφίλ κατόχων.

isLoggedInKeeper(): Ελέγχει αν ένας pet keeper είναι συνδεδεμένος, σημαντικό για τη διαχείριση των sessions.

isLoggedInKeeper(): Ελέγχει αν ένας pet owner είναι συνδεδεμένος, σημαντικό για τη διαχείριση των sessions.

logout(): Διαχειρίζεται τη διαδικασία αποσύνδεσης.

getUser(): Ανακτά τα στοιχεία του τρέχοντος χρήστη.

getUserOwner(): Συγκεκριμένα αντλεί λεπτομέρειες για τους pet owners.

verifyLocation(): Επιβεβαιώνει την ακρίβεια μιας τοποθεσίας.

show_map(lat, lon): Εμφανίζει έναν χάρτη χρησιμοποιώντας το OpenLayers, εμφανίζοντας έναν δείκτη στο καθορισμένο γεωγραφικό πλάτος και μήκος. Περιλαμβάνει ένα αναδυόμενο παράθυρο με τις λεπτομέρειες της τοποθεσίας κατά το κλικ του δείκτη.

PetRegisterPage(): Φορτώνει τη σελίδα εγγραφής κατοικίδιου ζώου.

put_function(): Ενημερώνει τις πληροφορίες κατοικίδιου ζώου, συγκεκριμένα το βάρος, στον διακομιστή χρησιμοποιώντας ένα αίτημα PUT και εμφανίζει την απόκριση.

delete_function(): Διαγράφει τις πληροφορίες ενός κατοικίδιου ζώου με βάση το αναγνωριστικό του χρησιμοποιώντας ένα αίτημα DELETE και εμφανίζει την απάντηση του διακομιστή.

get_function(): Αντλεί κατοικίδια με βάση τα φίλτρα τύπου και φυλής χρησιμοποιώντας ένα αίτημα GET και εμφανίζει τις ανακτηθείσες πληροφορίες.

post_function(): Υποβάλλει νέες πληροφορίες κατοικίδιων ζώων στον διακομιστή χρησιμοποιώντας ένα αίτημα POST και χειρίζεται την απόκριση, συμπεριλαμβανομένης της επικύρωσης δεδομένων φόρμας.

validateFormData(): Επικυρώνει τα δεδομένα της φόρμας για την εγγραφή κατοικίδιων ζώων, ελέγχοντας τις σωστές μορφές και εμφανίζοντας μηνύματα για εσφαλμένες εισόδους.

displayMessage(message): Εμφανίζει ένα καθορισμένο μήνυμα.

3β)

- **Φόρτωση της Βιβλιοθήκης Google Charts (<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>):**
    - Φορτώνει τη βιβλιοθήκη Google Charts, η οποία επιτρέπει τη δημιουργία γραφημάτων και διαγραμμάτων στην ιστοσελίδα

- **Σύνδεση με τη βιβλιοθήκη jQuery (<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>):**
    - Συνδέει τη σελίδα με τη βιβλιοθήκη jQuery, η οποία παρέχει επεξεργασία DOM, AJAX και άλλες λειτουργίες JavaScript για τη διευκόλυνση της ανάπτυξης ιστοσελίδων.

3γ)

Χρησιμοποιήσαμε απλή css για να δώσουμε μια καλύτερη εμφάνιση στην σελίδα. Στην κεντρική σελίδα (για το css του index.html χρησιμοποιήθηκε flexbox)

3δ)

Ο client θα μπορεί να κάνει τις λειτουργίες που περιγράφηκαν στην περιγραφή του project στο (1)

3ε)

Τα Ajax τμήματα του project είναι όλα τα τμήματα της js (στο ajax.js), που καλούν το αντίστοιχο servlet μέσω του **var xhr = new XMLHttpRequest();** με τις open, send κτλ, ώστε να μεταφερθούν τιμές από/προς τον server.

3στ) Δεν χρησιμοποιήθηκαν Rest requests

3ζ)

- Χρησιμοποιήθηκαν εξής API τα
  - **OpenStreetMaps** (για την εμφάνιση του χάρτη)
  - **CHATGPT API** για πληροφορίες από το gpt
  - **Google Charts** για αναπαράσταση στατιστικών στοιχείων

3η) Δεν χρειάστηκαν να γίνουν τροποποιήσεις στην βάση

/teamInfo membersInfo
Θεολόγος Κοκκινέλλης, csd5082, csd5082@csd.uoc.gr

Φώτης Πελαντάκης, csd4988, csd4988@csd.uoc.gr

/documents
    /installationInstructions
        Για να τρέξουν τα αρχεία πρέπει να πατήσουμε Run (κατά προτίμηση μέσω του Netbeans)
    Τα έχουμε κάνει σχεδόν όλα εκτός από ένα bonus ερώτημα.
/software
    /webproject