Design Report for Web-Based System

Authors:Spyridon Doukeris(4871),Dimokratis Stathakis(5010)

Technologies:

Jdk 11.0.21 Tomcat 9.0.83 IDE:netbeans 20

(uml at the end)

Introduction:

The 'Pet Care' system aims to facilitate pet owners by providing a reliable platform where they can find hospitable care for their furry friends while they are away and pet keepers to keep the pets safe. This project serves as a practical ground to hone their skills in client and server technologies, including HTML, JavaScript, AJAX, JAVA Servlets, CSS, REST APIs, and JSP.

Spark/RestAPIs:

1.PetsRESTAPI class

Initialize API: sets up an EditPetsTable instance for database operations.

GET Requests:

i)/pets: Retrieves all pets from the database.

ii) /owner/:owner_id: Fetches pets belonging to a specified owner.

iii) /pets/:type/:breed: Gets pets filtered by type and breed, including additional filtering by weight range if specified.

POST Request:

i)/pet: Adds a new pet to the database using provided JSON data.

PUT Request:

i)/petWeight/:pet_id/:weight: Updates the weight of a specific pet identified by its ID.

DELETE Request:

i) /petDeletion/:pet_id: Deletes a pet from the database based on its ID.

2.AdminRESTAPI class

Initialize API: Initializes instances for EditPetOwnersTable, EditPetKeepersTable, EditBookingsTable, and EditPetsTable for various database operations.

GET Requests:

i)/PO: Retrieves all pet owners from the database.

ii)/PK: Fetches all full pet keepers' details.

iii)/profit: Calculates and returns the total profit and pet keepers' earnings from all finished bookings.

iv) /pets: Counts and returns the number of cats and dogs in the database.

DELETE Requests:

i) /PO/delete/:id: Deletes a pet owner from the database based on the provided ID and returns the updated list of pet owners.

ii)/PK/delete/:id: Removes a pet keeper from the database using the specified ID and returns the updated list of pet keepers.

3.PetKeeperRESTAPI class

Initialize API: Sets up the API on port 4568 and initializes instances for EditBookingsTable, EditReviewsTable, and EditMessagesTable for database operations.

GET Requests:

i)/api/petKeepers: Retrieves all pet keepers from the database.

ii)/keeperAPI/:id: Provides detailed information for a specific pet keeper, including total finished bookings, total days of bookings, average review score, and review texts.

iii)/keeperAPI/booking/:id: Retrieves booking details for a specific pet keeper, focusing on requested or accepted bookings.

iv)/keeperAPI/chatgtp/:text: Returns a response from ChatGPT based on the provided text.

v)/keeperAPI/messages/:id: Retrieves all messages associated with a specific booking ID.

PUT Request:

vi)/keeperAPI/booking/:id/:status: Updates the status of a specific booking identified by ID.

POST Request:

i)/keeperAPI/messages/:id/send: Sends a new message related to a specific booking ID, with the sender set as "keeper".

4.PetOwnerRESTAPI class

Initialize API: initializes instances for EditPetsTable, EditBookingsTable, EditPetKeepersTable, EditReviewsTable, EditPetOwnersTable, and EditMessagesTable for database operations.

GET Requests:

i)/api/petOwners/:ownerId/availablePet: Checks for an available pet for a specified pet owner.

ii)/api/petOwners/:ownerId/hasActiveOrPendingBooking: Determines if a pet owner has an active or pending booking.

iii)/api/availablePetKeepers/:petType: Retrieves a list of available pet keepers for a specified pet type.

iv)/api/petOwners/:ownerId/petType: Fetches the pet type owned by a specific pet owner.

v)/ownerAPI/booking/:ownerId: Retrieves all bookings associated with a particular pet owner.

vi)/ownerAPI/reviews/:ownerId: Gets all reviews written by a specific pet owner.

vii)/api/checkReview/:bookingId: Checks if a review exists for a specific booking.

viii)/api/petOwnerLocation/:ownerId: Fetches the location (latitude and longitude) of a pet owner.

ix)api/messages/:id: Retrieves all messages associated with a specific booking ID.

POST Requests:

i)/api/addBooking: Adds a new booking to the database.

ii)/api/submitReview: Submits a new review to the database.

iii)/api/messages/:id/send: Sends a new message related to a specific booking ID, with the sender set as "owner".

PUT Request:

i)/api/finishBooking/:bookingId: Marks a booking as finished in the database.

Main Methods And Classes:

1.Class EditBookingsTable

Purpose:

Manages the operations related to the 'bookings' table in the database, such as adding, retrieving, updating, and creating bookings.

Functions:

JSON conversion functions: Convert Bookings objects to and from JSON.

getAllBookings(): Retrieves a list of all bookings from the database.

takeAllPetKeepersBooking(String id): Fetches all bookings associated with a specific pet keeper.

databaseToBooking(int id): Retrieves a specific booking from the database using its ID.

updateBooking(int bookingID, String status): Updates the status of a specific booking.

createBookingTable(): Creates the 'bookings' table in the database.

createNewBooking(Booking bor): Adds a new booking record to the database. (using some extra functions to convert message to json)

hasOwnerMadeRequest(String ownerId): Checks if a pet owner has any active or pending booking requests.

getBookingsForOwner(String ownerId): Retrieves all bookings for a specific pet owner based on ownerid.

getBookingById(String bookingId): Fetch es a specific booking by its ID.

updateBookingStatus(String bookingId, String status): Updates the status of a booking based on its ID.

2.Class EditMessagesTable

Purpose:

Manages operations related to the 'messages' table in the database, including adding new messages, converting messages to/from JSON, and retrieving messages associated with bookings.

Functions:

JSON conversion functions: Convert Messages objects to and from JSON.

databaseToMessage(int booking_id): Retrieves a list of all messages associated with a specific booking ID from the database.

createMessageTable(): Creates the 'messages' table in the database.

createNewMessage(Message msg): Adds a new message record to the database.

3.Class EditPetKeepersTable

Purpose:

Manages operations related to the 'petkeepers' table in the database, including adding, deleting, updating, and retrieving pet keeper records.

Functions:

JSON conversion functions: Convert Petkeepers objects to and from JSON.

deletePetKeeper(String id): Deletes a pet keeper from the database using their ID.

getAllFullPetKeepers(): Retrieves a list of all pet keepers with full details from the database.

updatePetKeeper(String username, String field, String value): Updates a specific field of a pet keeper's record.

printPetKeeperDetails(String username, String password): Prints the details of a pet keeper based on username and password.

databaseToPetKeepers(String username, String password): Retrieves a pet keeper from the database based on username and password.

findUsernameToPetKeepers(String username): Finds a pet keeper based on their username.

findEmailToPetKeepers(String email): Finds a pet keeper based on their email.

getAvailableKeepers(String type) : Retrieves a list of available pet keepers based on a

specified type.

getKeepers(String type): Retrieves a list of pet keepers who are specialized in a specific type of pet care.

4.Class EditPetOwnersTable

Purpose: Manages operations related to the 'petowners' table in the database, including adding, deleting, updating, and retrieving pet owner records.

Functions:

JSON conversion functions: Convert Petowners objects to and from JSON.

deletePetOwner(String id): Deletes a pet owner from the database using their ID

getAllPetOwners (): Retrieves a list of all pet owners with full details from the database.

updatePetOwner(String username, String personalpage): Updates the personal page field of a pet owner's record based on their username.

databaseToPetOwners(String username, String password): Retrieves a pet owner from the database based on their username and password.

findUsernameToPetOwners(String username): Finds a pet owner based on their username.

findEmailToPetOwners(String email): Finds a pet owner based on their email.

databasePetOwnerToJSON(String username, String password): Converts a pet owner's data from the database to JSON format based on their username and password.

createPetOwnersTable(): Creates the 'pet owners' table in the database.

addNewPetOwner(PetOwner user): Adds a new pet owner record to the database.

getPetOwnerById(String ownerId): Retrieves the details of a pet owner from the database using their owner ID.

5.Class EditPetsTable

Purpose: Handles operations for the 'pets' table in the database, including CRUD operations for pets, converting pets to/from JSON, and specific queries based on pet and owner data. Functions: JSON conversion functions: Convert Pet objects to and from JSON.

Functions:

JSON conversion functions: Convert Pets objects to and from JSON.

databaseToPets(): Retrieves all pets from the database.

databaseToPets(String type): Fetches pets of a specified type from the database.

petOfOwner(String id): Gets the pet(s) belonging to a specified owner.

addPetFromJSON(String json): Adds a pet to the database using JSON data.

createPetsTable(): Creates the 'pets' table in the database.

createNewPet(Pet bt): Adds a new pet record to the database.

updatePet(String id, String weight): Updates the weight of a specific pet.

deletePet(String id): Deletes a pet from the database using its ID.

getAvailablePetForOwner(String ownerId): Finds an available pet for a specific owner, not involved in active bookings.

getPetTypeByOwnerId(String ownerId): Retrieves the type of pet for a specific owner.

6.Class EditReviewsTable

Purpose: Manages operations related to the 'reviews' table in the database, including adding new reviews, converting reviews to/from JSON, and retrieving reviews associated with pet owners and keepers.

Functions:

JSON conversion functions: Convert Review objects to and from JSON.

databaseTokeeperReviews(String keeper_id): Retrieves a list of reviews for a specific pet keeper from the database.

createReviewTable(): Creates the 'reviews' table in the database.

createNewReview(Review rev): Adds a new review record to the database.

getReviewsForOwner(String ownerId): Retrieves all reviews associated with a specific pet owner.

reviewExistsForBooking(String ownerId, String keeperId): Checks if a review exists for a specific booking, based on owner and keeper IDs.

Servlets:

Register/login Servlets:

GetPetOwner Servlet:

GET Method: Handles retrieval of PetOwner details based on username, password, or email. It supports session-based access and returns pet owner information in JSON format.

POST Method: Manages two main operations: Logout functionality: Invalidates the session if the logout parameter is set. Adding a new PetOwner: Accepts pet owner details, creates a new PetOwner object, and adds it to the database.

GetPetKeeper Servlet:

GET Method: Retrieves PetKeeper details based on username, password, or email. Supports session validation and returns pet keeper information in JSON format.

POST Method: Handles two functionalities: Logout: Invalidates the session for a pet keeper. Adds or updates a PetKeeper: Based on the provided parameters, either a new PetKeeper is added or an existing one is updated in the database.

Ajax requests:

ajax.js

getPetKeepers() Function:

- **This function performs an Ajax GET request to "http://localhost:4568/api/petKeepers".**
- **It fetches data about pet keepers and uses it to create a table on the webpage.**

getPetKeeperUser() and getPetOwnerUser() Functions:

- **Both functions make Ajax GET requests to 'GetPetKeeper?' + data and 'GetPetOwner?' + data respectively.**
- **They are used to check for the existence of a pet keeper or pet owner based on the provided username or email.**

addPetOwner() and addPetKeeper() Functions:

- **These functions perform Ajax POST requests to 'GetPetOwner?' + data and 'GetPetKeeper?' + data.**
- **They are used to add a new pet owner or pet keeper to the system, including their geolocation data.**

getUser() Function:

- **It sends an Ajax GET request to 'GetPetKeeper?' and then, depending on the response, to 'GetPetOwner?'.**
- **The function is used for user authentication, checking if the user is a pet keeper or pet owner.**

loginUser() Function:

- **This function calls getUser() internally to perform the Ajax operation for user login.**
- **It handles the login process and navigates to different pages based on user type.**

updateUser() Function:

- **It makes an Ajax POST request to 'GetPetKeeper?' + data.**
- **The function is used for updating user information.**

logoutUser() Function:

- **This function sends an Ajax POST request to 'GetPetKeeper?' + data.**
- **It handles user logout, clearing user data from local storage.**

checkLoggedIn() Function:

- **Sends an Ajax GET request to either 'GetPetOwner' or 'GetPetKeeper?' based on the user type.**
- **This function is used to check if the user is logged in and retrieve their data.**

ajaxAdmin.js

getPO() Function:

- **Performs an Ajax GET request to "http://localhost:4560/adminAPI/PO".**
- **Retrieves data for pet owners and updates the HTML content.**

getPK() Function:

- **Sends an Ajax GET request to "http://localhost:4560/adminAPI/PK".**
- **Fetches data for pet keepers and updates the webpage.**

deletePO(id) Function:

- **Makes an Ajax DELETE request to "http://localhost:4560/adminAPI/PO/delete/"+id.**
- **Used for deleting a pet owner based on their ID.**

deletePK(id) Function:

- **Executes an Ajax DELETE request to "http://localhost:4560/adminAPI/PK/delete/"+id.**
- **Used for deleting a pet keeper based on their ID.**

getProfit() Function:

- **Sends an Ajax GET request to "http://localhost:4560/adminAPI/profit".**
- **Retrieves profit information.**

getPets() Function:

- **Performs an Ajax GET request to "http://localhost:4560/adminAPI/pets".**
- **Fetches the number of cats and dogs.**

ajaxPK.js

getPKInfo() Function:

- **Fetches pet keeper's information using a GET request to "http://localhost:4568/keeperAPI/" + user.**
- **Updates the status and review tables on the page with the data received.**

getBooking() Function:

- **Retrieves booking information with a GET request to "http://localhost:4568/keeperAPI/booking/" + user.**
- **Populates the jobs table based on the response.**

sendStatus(but_name) Function:

- **Sends a PUT request to "http://localhost:4568/keeperAPI/booking/" + book_id + "/" + but_name to update the booking status.**
- **Refreshes the jobs table after the request.**

askChatGPT() Function:

- **Sends a GET request to "http://localhost:4568/keeperAPI/chatgtp/" + text to interact with a chatbot service (presumably ChatGPT).**
- **Updates the chat display with the user's query and the chatbot's response.**

getMessages() Function:

- **Fetches messages related to a particular booking using a GET request to "http://localhost:4568/keeperAPI/messages/" + book_id.**
- **Displays these messages in the owner's chat section.**

sendMessages() Function:

- **Sends a POST request to "http://localhost:4568/keeperAPI/messages/" + book_id + "/send" to submit a new message.**
- **Updates the chat display with new messages.**

ajaxPO.js

getOwnerBookings():

- **Makes a GET request to "http://localhost:4562/ownerAPI/booking/"+ownerId.**
- **Retrieves the booking details for the pet owner identified by ownerId.**
- **Updates the DOM with the bookings data received.**

getOwnerReviews():

- **Sends a GET request to "http://localhost:4562/ownerAPI/reviews/"+ownerId.**
- **Fetches reviews associated with the pet owner.**
- **Populates the reviews table with the data received.**

getAvailablePetKeepersByType(petType):

- **Fetches available pet keepers based on the pet type (dog/cat) through a GET request to "http://localhost:4562/api/availablePetKeepers/"+encodeURIComponent(petType).**
- **Utilizes the pet owner's location to sort pet keepers by proximity.**
- **Displays available pet keepers in a table format.**

getPetOwnerLocation():

- **Obtains the location of the pet owner via a GET request to "http://localhost:4562/api/petOwnerLocation/"+encodeURIComponent(ownerId).**
- **Used for calculating distances to available pet keepers.**

sendRequest(keeperData):

- **Handles booking requests. The exact endpoint and method are not specified in the provided snippet, but it likely involves sending booking data to the server.**

hasNotRequested(callback):

- **Checks if the user has any active or pending booking requests.**
- **Makes a GET request to 'http://localhost:4562/api/petOwners/' + userId + '/hasActiveOrPendingBooking'.**

hasAvailablePet(callback):

- **Determines if the pet owner has pets available for booking.**
- **Sends a GET request to 'http://localhost:4562/api/petOwners/' + ownerId + '/availablePet'.**

addBookingRequest(bookingData):

- **Adds a new booking request.**
- **Utilizes a POST request to 'http://localhost:4562/api/addBooking', sending the bookingData.**

checkPetTypeAndFetchKeepers(ownerId, enteredPetType, callback):

- **Verifies the pet type and fetches pet keepers accordingly.**
- **Makes a GET request to 'http://localhost:4562/api/petOwners/${ownerId}/petType'.**

getMessages():

- **Retrieves messages related to a booking.**
- **Uses a GET request to 'http://localhost:4562/api/messages/' + book_id.**

sandMessages():

- **Sends a new message related to a booking.**
- **Involves a POST request to 'http://localhost:4562/api/messages/' + book_id + "/send".**

finishBooking(bookingId):

- **Marks a booking as finished.**
- **Executes a PUT request to 'http://localhost:4562/api/finishBooking/' + bookingId.**

ajaxRest.js

addPet():

- **Sends a GET request to "http://localhost:4567/petsAPI/pets".**
- **Retrieves a list of pets and displays them using createTableFromJSON.**
- **On failure, it displays an error message in the 'msg' element.**

addPet():

- **Makes a POST request to "http://localhost:4567/petsAPI/pet".**
- **Submits data for adding a new pet, collected from a form.**
- **On success or failure, displays the response or error message in the 'msg' element.**

getPetsTBW():

- **Sends a GET request to "http://localhost:4567/petsAPI/pets/"+type+"/"+breed+"?fromWeight="+fromWeight+"&toWeight="+toWeight.**
- **Fetches pets based on specified criteria (type, breed, weight range).**
- **Displays the results using createTableFromJSON or shows an error message.**

updatePetWeight():

- **Executes a PUT request to "http://localhost:4567/petsAPI/petWeight/"+name+"/"+weight.**
- **Updates the weight of a specified pet.**
- **Shows the response or error message in the 'msg' element.**

deletePet():

- **Makes a DELETE request to "http://localhost:4567/petsAPI/petDeletion/"+name.**
- **Deletes a pet based on the given identifier.**
- **Displays the server's response or an error message.**

checkOwner(indexOwner, indexSubmit):

- **Sends a GET request to "http://localhost:4567/petsAPI/owner/" + o_v.**
- **Validates if an owner exists in the database.**

- **Based on the response, it enables or disables a submit button and shows/hides an error message.**

Apis:

Chatgpt and OpenStreetMaps APIs

JavaScript Libraries/HTML/JSP Used:

Bootstrap (JavaScript Library) (<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>)

Google Charts (JavaScript Library)

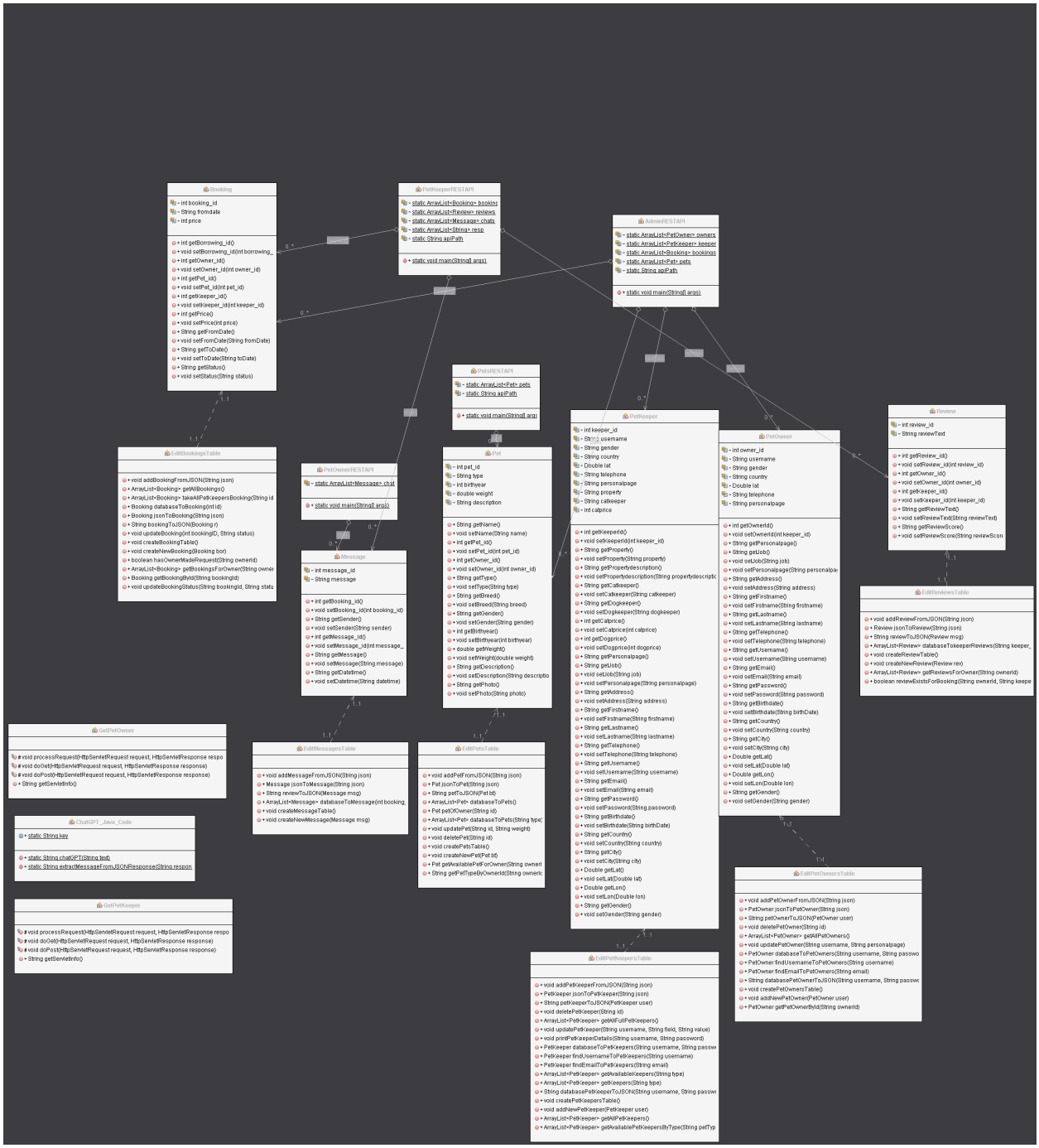
Custom JavaScript Files

CSS:

Bootstrap (CSS Framework)

Inline CSS

Custom CSS Files


Booking
- int booking_id
- String fromdate
- int price
+ int getBorrowing_id()
+ void setBorrowing_id(int borrowing_
+ int getOwner_id()
+ void setOwner_id(int owner_id)
+ int getPet_id()
+ void setPet_id(int pet_id)
+ int getKeeper_id()
+ void setKeeper_id(int keeper_id)
+ int getPrice()
+ void setPrice(int price)
+ String getFromDate()
+ void setFromDate(String fromDate)
+ String getToDate()
+ void setToDate(String toDate)
+ String getStatus()
+ void setStatus(String status)
PetKeeperRESTAPI
- static ArrayList<Booking> booking
- static ArrayList<Review> reviews
- static ArrayList<Message> chats
- static ArrayList<String> resp
- static String apiPath
+ static void main(String[] args)
AdminRESTAPI
- static ArrayList<PetOwner> owners
- static ArrayList<PetKeeper> keeper
- static ArrayList<Booking> bookings
- static ArrayList<Pet> pets
- static String apiPath
+ static void main(String[] args)
PetsRESTAPI
- static ArrayList<Pet> pets
- static String apiPath
+ static void main(String[] args)
PetOwnerRESTAPI
- static ArrayList<Message> chat
+ static void main(String[] args)
Pet
- int pet_id
- String type
- int birthyear
- double weight
- String description
+ String getName()
+ void setName(String name)
+ int getPet_id()
+ void setPet_id(int pet_id)
+ int getOwner_id()
+ void setOwner_id(int owner_id)
+ String getType()
+ void setType(String type)
+ String getBreed()
+ void setBreed(String breed)
+ String getGender()
+ void setGender(String gender)
+ int getBirthyear()
+ void setBirthyear(int birthyear)
+ double getWeight()
+ void setWeight(double weight)
+ String getDescription()
+ void setDescription(String descriptio
+ String getPhoto()
+ void setPhoto(String photo)
PetKeeper
- int keeper_id
- String username
- String gender
- String country
- Double lat
- String telephone
- String personalpage
- String property
- String catkeeper
- int catprice
+ int getKeeperId()
+ void setKeeperId(int keeper_id)
+ String getProperty()
+ void setProperty(String property)
+ String getPropertydescription()
+ void setPropertydescription(String propertydescriptic
+ String getCatkeeper()
+ void setCatkeeper(String catkeeper)
+ String getDogkeeper()
+ void setDogkeeper(String dogkeeper)
+ int getCatprice()
+ void setCatprice(int catprice)
+ int getDogprice()
+ void setDogprice(int dogprice)
+ String getPersonalpage()
+ String getJob()
+ void setJob(String job)
+ void setPersonalpage(String personalpage)
+ String getAddress()
+ void setAddress(String address)
+ String getFirstname()
+ void setFirstname(String firstname)
+ String getLastname()
+ void setLastname(String lastname)
+ String getTelephone()
+ void setTelephone(String telephone)
+ String getUsername()
+ void setUsername(String username)
+ String getEmail()
+ void setEmail(String email)
+ String getPassword()
+ void setPassword(String password)
+ String getBirthdate()
+ void setBirthdate(String birthDate)
+ String getCountry()
+ void setCountry(String country)
+ String getCity()
+ void setCity(String city)
+ Double getLat()
+ void setLat(Double lat)
+ Double getLon()
+ void setLon(Double lon)
+ String getGender()
+ void setGender(String gender)
PetOwner
- int owner_id
- String username
- String gender
- String country
- Double lat
- String telephone
- String personalpage
+ int getOwnerId()
+ void setOwnerId(int keeper_id)
+ String getPersonalpage()
+ String getJob()
+ void setJob(String job)
+ void setPersonalpage(String personalpa
+ String getAddress()
+ void setAddress(String address)
+ String getFirstname()
+ void setFirstname(String firstname)
+ String getLastname()
+ void setLastname(String lastname)
+ String getTelephone()
+ void setTelephone(String telephone)
+ String getUsername()
+ void setUsername(String username)
+ String getEmail()
+ void setEmail(String email)
+ String getPassword()
+ void setPassword(String password)
+ String getBirthdate()
+ void setBirthdate(String birthDate)
+ String getCountry()
+ void setCountry(String country)
+ String getCity()
+ void setCity(String city)
+ Double getLat()
+ void setLat(Double lat)
+ Double getLon()
+ void setLon(Double lon)
+ String getGender()
+ void setGender(String gender)
Review
- int review_id
- String reviewText
+ int getReview_id()
+ void setReview_id(int review_id)
+ int getOwner_id()
+ void setOwner_id(int owner_id)
+ int getKeeper_id()
+ void setKeeper_id(int keeper_id)
+ String getReviewText()
+ void setReviewText(String reviewText)
+ String getReviewScore()
+ void setReviewScore(String reviewScor
Message
- int message_id
- String message
+ int getBooking_id()
+ void setBooking_id(int booking_id)
+ String getSender()
+ void setSender(String sender)
+ int getMessage_id()
+ void setMessage_id(int message_
+ String getMessage()
+ void setMessage(String message)
+ String getDatetime()
+ void setDatetime(String datetime)
EditBookingsTable
+ void addBookingFromJSON(String json)
+ ArrayList<Booking> getAllBookings()
+ ArrayList<Booking> takeAllPetKeepersBooking(String id
+ Booking databaseToBooking(int id)
+ Booking jsonToBooking(String json)
+ String bookingToJSON(Booking r)
+ void updateBooking(int bookingID, String status)
+ void createBookingTable()
+ void createNewBooking(Booking bor)
+ boolean hasOwnerMadeRequest(String ownerId)
+ ArrayList<Booking> getBookingsForOwner(String owner
+ Booking getBookingById(String bookingId)
+ void updateBookingStatus(String bookingId, String statu
EditMessagesTable
+ void addMessageFromJSON(String json)
+ Message jsonToMessage(String json)
+ String reviewToJSON(Message msg)
+ ArrayList<Message> databaseToMessage(int booking_
+ void createMessageTable()
+ void createNewMessage(Message msg)
EditPetsTable
+ void addPetFromJSON(String json)
+ Pet jsonToPet(String json)
+ String petToJSON(Pet bf)
+ ArrayList<Pet> databaseToPets()
+ Pet petOfOwner(String id)
+ ArrayList<Pet> databaseToPets(String type)
+ void updatePet(String id, String weight)
+ void deletePet(String id)
+ void createPetsTable()
+ void createNewPet(Pet bf)
+ Pet getAvailablePetForOwner(String owneri
+ String getPetTypeByOwnerId(String ownerId
EditPetKeepersTable
+ void addPetKeeperFromJSON(String json)
+ PetKeeper jsonToPetKeeper(String json)
+ String petKeeperToJSON(PetKeeper user)
+ void deletePetKeeper(String id)
+ ArrayList<PetKeeper> getAllFullPetKeepers()
+ void updatePetKeeper(String username, String field, String value)
+ void printPetKeeperDetails(String username, String password)
+ PetKeeper databaseToPetKeepers(String username, String passwo
+ PetKeeper findUsernameToPetKeepers(String username)
+ PetKeeper findEmailToPetKeepers(String email)
+ ArrayList<PetKeeper> getAvailableKeepers(String type)
+ ArrayList<PetKeeper> getKeepers(String type)
+ String databasePetKeeperToJSON(String username, String passwo
+ void createPetKeepersTable()
+ void addNewPetKeeper(PetKeeper user)
+ ArrayList<PetKeeper> getAllPetKeepers()
+ ArrayList<PetKeeper> getAvailablePetKeepersByType(String petTyp
EditPetOwnersTable
+ void addPetOwnerFromJSON(String json)
+ PetOwner jsonToPetOwner(String json)
+ String petOwnerToJSON(PetOwner user)
+ void deletePetOwner(String id)
+ ArrayList<PetOwner> getAllPetOwners()
+ void updatePetOwner(String username, String personalpage)
+ PetOwner databaseToPetOwners(String username, String passwo
+ PetOwner findUsernameToPetOwners(String username)
+ PetOwner findEmailToPetOwners(String email)
+ String databasePetOwnerToJSON(String username, String passwo
+ void createPetOwnersTable()
+ void addNewPetOwner(PetOwner user)
+ PetOwner getPetOwnerById(String ownerId)
EditReviewsTable
+ void addReviewFromJSON(String json)
+ Review jsonToReview(String json)
+ String reviewToJSON(Review msg)
+ ArrayList<Review> databaseToKeeperReviews(String keeper_
+ void createReviewTable()
+ void createNewReview(Review rev)
+ ArrayList<Review> getReviewsForOwner(String ownerId)
+ boolean reviewExistsForBooking(String ownerId, String keepe
GetPetOwner
void processRequest()-HttpServletRequest request, HttpServletResponse respo
void doGet()-HttpServletRequest request, HttpServletResponse response)
void doPost()-HttpServletRequest request, HttpServletResponse response)
+ String getServletInfo()
ChatGPT _Java_Code
+ static String key
+ static String chatGPT(String text)
+ static String extractMessageFromJSONResponse(String respon
GetPetKeeper
void processRequest()-HttpServletRequest request, HttpServletResponse respo
void doGet()-HttpServletRequest request, HttpServletResponse response)
void doPost()-HttpServletRequest request, HttpServletResponse response)
+ String getServletInfo()
1..1
0..*