API Documentation

1. User endpoints:
2. GET /api/v1/users:

* Using no parameters:

Returns a list of all the users (deleted and active, all roles included) in the database in the following format:

A computer screen shot of white text

Description automatically generated

* Available parameters:
* ?role=role&deleted=boolean
* **role** can take any of the following values: “SITTER”, “OWNER”, or “ADMIN”.
* **deleted** can either be set to ‘true’ or to ‘false’.
* These parameters are not mandatory. They also don’t have to (but can be) be used together.
* If role is specified, a list of users with that specific role will be returned.
* If deleted is specified and is set to true, only the deleted users will be returned and vice versa.
* Only the relevant relations are loaded. The three types of users share ‘role’ and ‘address’ so these will be loaded for all three. Owners and sitters share the relations ‘reviews\_received’ and ‘reviews\_given’, so these will be loaded for both user types. The relations ‘pets’ and ‘bookings’ will be loaded for owners, and ‘certifications’ and ‘sitting’ for sitters (along with all relations mentioned before).
* Example:

A computer screen shot of white text

Description automatically generated

* ?start\_date=date&end\_date=date&country=country&city=city
* start\_date and end\_date must be valid dates in the format YYYY-MM-DD and the chosen interval must be at least one day long with start date on the day following the current date of the request.
* country can either be ‘MOROCCO’ or ‘GHANA’
* city must be a valid string with a city name.
* These parameters are not mandatory but MUST be used together. The purpose of these parameters is to get all available sitters in a given country/city in a given date interval.

1. GET /api/v1/users/<user\_id>:

This route takes no parameters. Based on the role of the user being requested, the proper relationships will be loaded:

* Address and role for all users.
* Reviews\_received and reviews\_given for both owners and sitters.
* Pets and bookings for owners
* Certifications and sittings for sitters.

1. POST /api/v1/users

This route requires the use of one of the following parameters:

* ?login:

the body must be in the following format:

A screenshot of a computer code

Description automatically generated

This will return the user with the matching email and password

* ?register:

this route creates a new user and saves it to the database. The fields required to create a user are:

* fname: the user’s first name. Must be a valid string.
* lname: the user’s first name. Must be a valid string.
* email: must be a valid and unique email
* password: must be a valid string, at least 8 characters long, containing at least one special character, one lowercase letter, one upper case letter, and one digit. gender: must be either ‘f’ (for female) or ‘m’ for male.
* birthday: must be a valid date string. The user must be at least 18 years old.
* role: must be one of the following values: ‘owner’, ‘sitter’, ‘admin’.

Some optional fields include:

* phone: must be a string representing a valid phone number. This field must be unique and must be in the format +212 X XX XX XX XX (for phone numbers in MA region) or +233 XX XXX XXXX (for phone numbers in GH region)
* bank\_account\_number: a string representing a bank account number (at least 10 characters long).
* fee: a number representing the fee to pay a sitter for their services. Only users with the role sitter can set this field.

1. PUT /api/v1/users/<user\_id>:

* The only fields that can be updated for a user are:
* "email", "phone", "bank\_account\_number", "password" and "fee".

1. DELETE /api/v1/users/<user\_id>:

* A user is only deleted if they’ve never had a booking or given a review. The data of a user is set to null and the id and all important data (required to keep the database in balance) is kept. The ‘account\_stat’ field is set to ‘DELETED’ if the user can’t be permanently deleted.
* If these constraints aren’t being broken, then the user will be removed from the database.

1. Address endpoints:

A user is allowed to have only one address at a time. For that reason, all endpoints for ‘address’ are the same except for the request method.

1. GET /api/v1/users/<user\_id>/address

Gets the address of a user in the following format:

A computer screen with white text

Description automatically generated

1. POST /api/v1/users/<user\_id>/address:

Creates a new address instance in the database. A user can only have one address at a time, meaning that if a user already has an address, this request will fail.

Once the address is created, a json representation (the same as the one in the image above) is returned.

The only fields required to create an address are:

* Street: a string representing a street name.
* City: a string representing a city name
* Country: the value of country can either be ‘morocco’ or ‘ghana’. Any other value will result in an error.
* postal\_code: a string representing the zip code of the address.

The following fields can be set but are not required:

* building\_num: a number representing the number of the building the user resides in.
* apartment\_num: a number representing the apartment number of the address.
* Floor: the floor of the user’s residence.

1. PUT /api/v1/users/<user\_id>/address:

Updates the user’s address. All fields mentioned above are updatable. After the instance is updated, a json is returned containing the updated address.

1. DELETE /api/v1/users/<user\_id>/address:

Deletes the address associated with the user whose id matches ‘user\_id’.

1. Certification endpoints:

Certifications are only available for users with the role ‘SITTER’. If the user id specified in any of the certification endpoints

1. Pet endpoints:
2. GET /api/v1/users/<user\_id>/pets

Gets a list of all the pets associated with a user. This route only works if the user id is of a user with the role ‘owner’. The json returned takes the following format:

A computer screen shot of white text

Description automatically generated

1. GET /api/v1/users/<user\_id>/pets/<pet\_id>

Gets the pet with ‘pet\_id’ that is associated with the user with ‘user\_id’. The json returned is in the same format as the example above.

1. POST /api/v1/users/<user\_id>/pets

Creates a new pet instance. In order to create a pet instance, the following fields are required:

* Name: the name of the pet. A constraint to keep in mind when choosing a pet name is that a user cannot have two pets with the same name.
* Size: the pet size MUST be one of the following values:
* S for small.
* M for medium.
* L for large.
* Description: a string (at least 30 characters long) that describes the pet.

Birthdate: a valid date string representing the date of birth of the pet.

Gender: the gender of the pet. Must be either f (for female) or m (for male).

Breed: the breed of the pet. It must be one of the breeds available in the database (see breed endpoints).

Another field can (but isn’t required to) be set for a pet. The field is:

* Temperament: the default value of this field is ‘friendly’, but it can also take the value ‘aggressive’. This field isn’t required since it has a default value.

1. PUT /api/v1/users/<user\_id>/pets/<pet\_id>:

This endpoint allows you to update a pet that’s associated with a given user. Not all fields can be updated. The only updatable fields are:

* Name, size, description, and temperament.

Make sure to follow the constraints for the values of size and temperament.