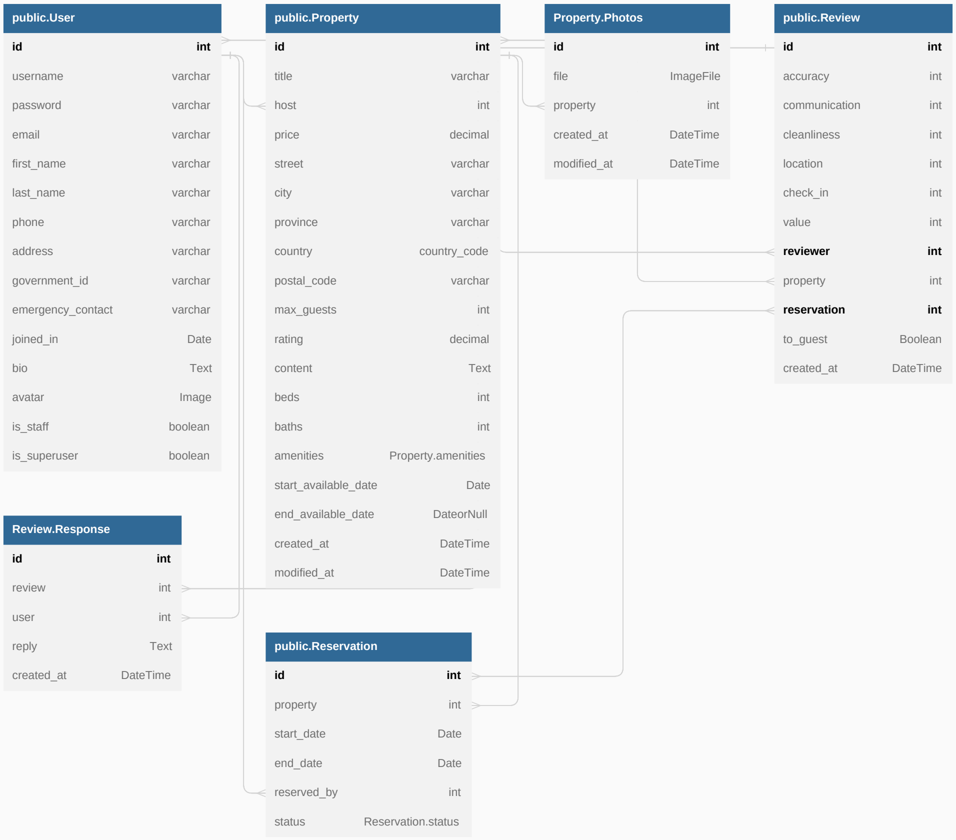
docs

Huipeng Lei, Arthur Li, Jamie Ren

March 2023

1 ER Diagram



public.User	
id	**int**
username	varchar
password	varchar
email	varchar
first_name	varchar
last_name	varchar
phone	varchar
address	varchar
government_id	varchar
emergency_contact	varchar
joined_in	Date
bio	Text
avatar	Image
is_staff	boolean
is_superuser	boolean

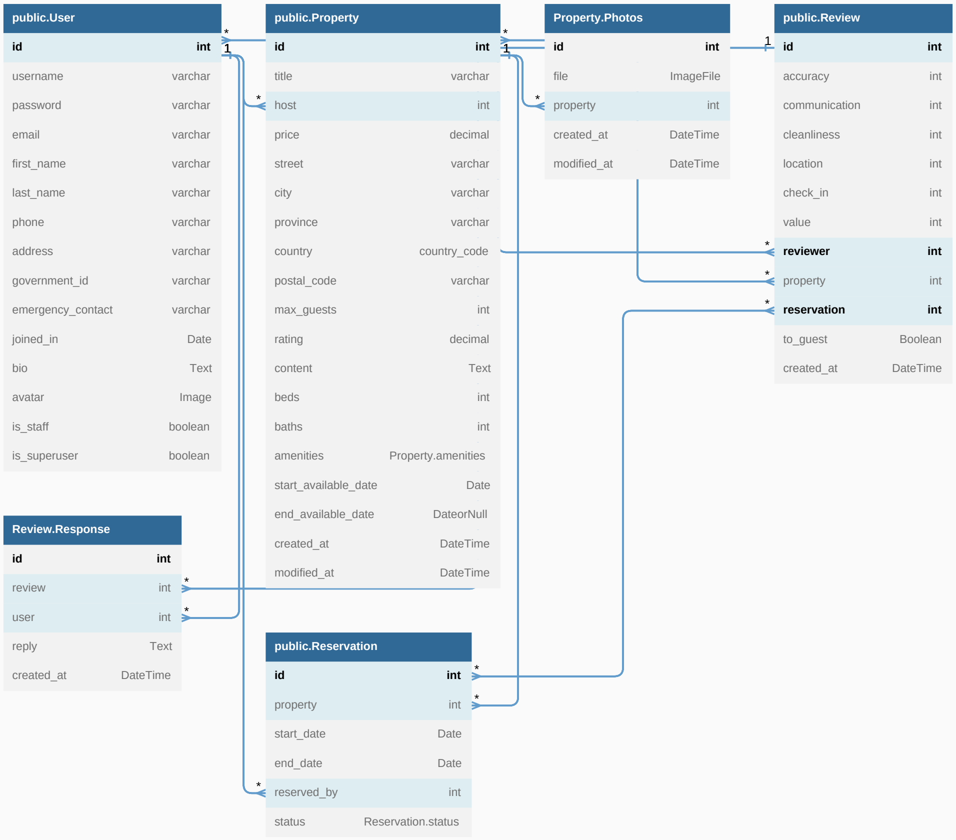
public.Property	
id	**int**
title	varchar
host	int
price	decimal
street	varchar
city	varchar
province	varchar
country	country_code
postal_code	varchar
max_guests	int
rating	decimal
content	Text
beds	int
baths	int
amenities	Property.amenities
start_available_date	Date
end_available_date	DateorNull
created_at	DateTime
modified_at	DateTime

Property.Photos	
id	**int**
file	ImageFile
property	int
created_at	DateTime
modified_at	DateTime

public.Review	
id	**int**
accuracy	int
communication	int
cleanliness	int
location	int
check_in	int
value	int
reviewer	**int**
property	int
reservation	**int**
to_guest	Boolean
created_at	DateTime

Review.Response	
id	**int**
review	int
user	int
reply	Text
created_at	DateTime

public.Reservation	
id	**int**
property	int
start_date	Date
end_date	Date
reserved_by	int
status	Reservation.status



public.User	
id	**int**
username	varchar
password	varchar
email	varchar
first_name	varchar
last_name	varchar
phone	varchar
address	varchar
government_id	varchar
emergency_contact	varchar
joined_in	Date
bio	Text
avatar	Image
is_staff	boolean
is_superuser	boolean

public.Property	
id	**int**
title	varchar
host	int
price	decimal
street	varchar
city	varchar
province	varchar
country	country_code
postal_code	varchar
max_guests	int
rating	decimal
content	Text
beds	int
baths	int
amenities	Property.amenities
start_available_date	Date
end_available_date	DateorNull
created_at	DateTime
modified_at	DateTime

Property.Photos	
id	**int**
file	ImageFile
property	int
created_at	DateTime
modified_at	DateTime

public.Review	
id	**int**
accuracy	int
communication	int
cleanliness	int
location	int
check_in	int
value	int
reviewer	**int**
property	int
reservation	**int**
to_guest	Boolean
created_at	DateTime

Review.Response	
id	**int**
review	int
user	int
reply	Text
created_at	DateTime

public.Reservation	
id	**int**
property	int
start_date	Date
end_date	Date
reserved_by	int
status	Reservation.status

```
Enum Property.amenities {
    wifi
    tv
    kitchen
    workspace
    air_conditioning
    heating
    washer
    dryer
}


Enum Reservation.status {
    completed
    approved
    canceled
    pending
    denied
    expired
    terminated
    pending_cancel
}
```

2 Endpoints

Important Notes

- Used `django-multiselectfield` for the field `amenities` of model `Property`
- Used `CountryField` for the field `country` of model `Property`

APIs of our application

Authentication

Endpoint: `/accounts/register/`
Methods: `POST`
Fields/payload: `avatar` , `username` , `first_name` , `last_name` , `email` , `phone` , `password` , `num_reviews`
Description: Registers a new user with simpleJWT.

Endpoint: `accounts/login/`
Methods: `POST`
Fields/payload: `username` , `password`
Description: Authenticate the user with the give username and password using simpleJWT.

Endpoint: `accounts/logout/`
Methods: `GET`
Authenticated: **`Yes`**
Description: Log the user's session out of Django. Blacklist the current token. If the user is unauthenticated, simply ignore the request

Endpoint: `accounts/profile/`
Methods: `GET` , `PUT`
Fields/payload: `avatar` , `username` , `first_name` , `last_name` , `email` , `phone` , `password` , `num_reviews`
Authenticated: **`Yes`**
Description: View the profile of the current logged in user. Also could update the profile of the current logged in user.
Example:

```
{
    "id": 1,
    "avatar": null,
    "username": "Arthur",
    "first_name": "Arthur",
    "last_name": "Li",
    "email": "ar.li@gmail.com",
    "phone": "",
    "num_reviews": 0,
    "reviews_from_guest": [],
    "reviews_from_host": []
}
```

Endpoint: `accounts/profile/<int:pk>`
Methods: `GET`
Authenticated: **`No`** (Since we don't have to be logged in to view other user's profile)
Description: View the profile of the user with the specified `pk` . Returns **HTTP404** if not found.

Example:

We are logged in as user 1, but we can view other user's profile, in this case, user 2's profile.

url: `accounts/profile/2/`

```
{
    "id": 2,
    "avatar": null,
    "username": "lucas",
    "first_name": "Lucas",
    "last_name": "Lei",
    "email": "lucas@123.com",
    "phone": "123",
    "num_reviews": 1,
    "reviews_from_guest": [
        {
            "id": 1,
            "reviewer": "bob",
            "responses": [],
            "review": "TEST",
            "accuracy": 5,
            "communication": 5,
            "cleanliness": 3,
            "location": 2,
            "check_in": 5,
            "value": 5,
            "to_guest": false,
            "created_at": "2023-03-14T23:02:59.982232Z",
            "property": 1,
            "reservation": 1
        }
    ],
    "reviews_from_host": []
}
```

Property

Endpoint: `/properties/`

Methods: `GET`

Fields/payload: `search` , `filters`

Search: `?search=<any>`

Filters: Begins with `?`

- *location*
  `city=<allvaluesinDB>&country=<allvaluesinDB>&street=<str>&province=<allvaluesinDB>`
- *date*
  `available_date_after=<YYYY-MM-DD>&available_date_before=<YYYY-MM-DD>`
- *price*
  `price_max=<float>&price_min=<float>`
- *number of guests*
  `max_guests_max=<int>&max_guests_min=<int>`
- *amenities* (can be empty or any number of choices)
  `amenities=wifi&amenities=tv&amenities=kitchen&amenities=workspace&amenities=air_conditioning&amenities=heating&amenities=washer&amenities=dryer`

Order by:

- Price `ordering=price` or `ordering=-price`
- Rating `ordering=rating` or `ordering=-rating`

Description:

This endpoint will list all the properties in the database associated with the specified filters and search results. The final result will be ordered by **price** or **rating**.

Endpoint: `/properties/<int:pk>/view/`

Methods: `GET`

Fields/payload: `id` , `host` , `photos` , `amenities` , `title` , `price` , `street` , `city` , `province` , `postal_code` , `max_guests` , `rating` , `content` , `available_date` , `created_at` , `modified_at`

Description:

A detail view of the property with the specified `pk` . Returns **HTTP404** if not found.

Example:

```
{
    "id": 6,
    "host": {
        "id": 3,
        "avatar": null,
        "username": "lucas",
        "first_name": "",
        "last_name": "",
        "email": "lucas@123.com",
        "phone": ""
    },
    "photos": [],
    "amenities": [
        "tv"
    ],
    "title": "house",
    "price": "250.00",
    "street": "12 bloor",
    "city": "calgary",
    "province": "Alberta",
    "country": "CA",
    "postal_code": "333",
    "max_guests": 5,
    "rating": "5.00",
    "content": "",
    "available_date": "2023-03-24T00:00:00Z",
    "created_at": "2023-03-11T09:21:04.692599Z",
    "modified_at": "2023-03-12T05:28:36.982044Z"
}
```

Endpoint: `/properties/host/listings/`

Methods: `GET` , `POST`

Fields/payload: `photos` , `amenities` , `title` , `price` , `street` , `city` , `province` , `postal_code` , `max_guests` , `content` , `available_date`

Authenticated: **`Yes`**

Description:

This endpoint is used to retrieve or create a new property listing for a host. The host is the current logged in user. **Also have searching, filtering and orderby functionality. Same as `/properties/`**

Example:

```
{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "host": {
                "id": 2,
                "avatar": null,
                "username": "bob",
                "first_name": "bob",
                "last_name": "wong",
                "email": "bob@123.com",
                "phone": "123"
            },
            "photos": [
                {
                    "id": 1,
                    "file": "http://127.0.0.1:8000/property_photos/QQ%E5%9B%BE%E7%89%8720161217210517_Z7cXJrA.jpg",
                    "created_at": "2023-03-13T01:46:51.060080Z",
                    "modified_at": "2023-03-13T01:46:51.060080Z",
                    "property": 1
                },
                {
                    "id": 2,
                    "file": "http://127.0.0.1:8000/property_photos/QQ%E5%9B%BE%E7%89%8720161217210522_CTa56D0.jpg",
                    "created_at": "2023-03-13T01:46:51.360533Z",
                    "modified_at": "2023-03-13T01:46:51.360533Z",
                    "property": 1
                },
                {
                    "id": 3,
                    "file": "http://127.0.0.1:8000/property_photos/QQ%E5%9B%BE%E7%89%8720161217210525_K4IKltW.jpg",
                    "created_at": "2023-03-13T01:46:51.591787Z",
                    "modified_at": "2023-03-13T01:46:51.591787Z",
                    "property": 1
                }
            ],
            "amenities": [
                "wifi",
                "tv"
            ],
            "num_reviews": 1,
            "title": "house",
            "price": "250.00",
            "street": "12 bloor",
            "city": "calgary",
            "province": "Alberta",
            "country": "CA",
            "postal_code": "333",
            "max_guests": 5,
            "rating": "5.00",
            "content": "",
            "available_date": "2023-05-01T00:00:00Z",
            "created_at": "2023-03-13T01:46:50.790245Z",
            "modified_at": "2023-03-13T05:22:09.193780Z"
        },
        {
            "id": 2,
            "host": {
                "id": 4,
                "avatar": null,
                "username": "lucas",
                "first_name": "Lucas",
                "last_name": "Lei",
                "email": "lucas@123.com",
                "phone": "123"
            },
            "photos": [
                {
                    "id": 4,
                    "file": "http://127.0.0.1:8000/property_photos/35d5e52fc9d560a89a9fc59fb327e50.jpg",
                    "created_at": "2023-03-13T03:10:48.032492Z",
                    "modified_at": "2023-03-13T03:10:48.032492Z",
                    "property": 2
                }
            ],
            "amenities": [
                "air_conditioning",
                "wifi"
            ],
            "num_reviews": 1,
            "title": "Good camp",
            "price": "1000.00",
            "street": "outside town",
            "city": "toronto",
            "province": "Ontario",
            "country": "CA",
            "postal_code": "333",
            "max_guests": 5,
            "rating": "4.50",
            "content": "",
            "available_date": "2023-03-12T00:00:00Z",
            "created_at": "2023-03-13T03:10:47.806491Z",
            "modified_at": "2023-03-13T05:22:09.586882Z"
        }
    ]
}
```

Endpoint: `/properties/host/listings/<int:pk>/edit/`
Methods: `GET, PUT, PATCH, DELETE`
Fields/payload: `photos` , `amenities` , `title` , `price` , `street` , `city` , `province` , `postal_code` , `max_guests` , `content` , `available_date` (These fields are optional)
Authenticated: **`Yes`**
Description:
This endpoint is used to retrieve, update or delete an existing property listing for a host. **Requires the user to be Authenticated**

`HTTP GET Request` :
This method retrieves the details of a specific property listing identified by its primary key `pk` .

`HTTP PUT Request` :
This method updates all fields of an existing property listing identified by its primary key `pk` . All fields are required.

`HTTP PATCH Request` :
This method updates one or more fields of an existing property listing identified by its primary key `pk` . All fields are optional.

`HTTP DELETE Request` :
This method deletes an existing property listing identified by its primary key `pk` .

Reviews

Endpoint: `/properties/<int:pk>/reviews/`
Methods: `GET`
Fields/payload: `review` , `accuracy` , `communication` , `cleanliness` , `location` , `check_in` , `value`
Authenticated: **`No`**
Description:
This endpoint allows clients to retrieve reviews for a particular property identified by its primary key `pk` .

Endpoint: `/properties/<int:pk>/reviews/<int:reservation_id>/addreview/`
Methods: `POST`
Fields/payload: `review` , `accuracy` , `communication` , `cleanliness` , `location` , `check_in` , `value`
Authenticated: **`Yes`**

Description:

This endpoint allows clients to add a review for a particular property identified by its primary key `pk` for a reservation identified by its primary key `reservation_id`. The client needs to be authenticated and the client has either completed the reservation or been terminated with this particular property.

```
Fields:
    review (TextField): Review text
    accuracy (PositiveIntegerField): Accuracy rating
    communication (PositiveIntegerField): Communication rating
    cleanliness (PositiveIntegerField): Cleanliness rating
    location (PositiveIntegerField): Location rating
    check_in (PositiveIntegerField): Check-in rating
    value (PositiveIntegerField): Value rating
    reviewer (ForeignKey): Reviewer
```

Endpoint: `/properties/<int:pk>/reviews/<int:reservation_id>/reviewguest/`

Methods: `POST`

Fields/payload: `review`

Authenticated: `Yes`

Description:

This endpoint allows clients to add a review for the guest that completed a reservation at a particular property they hosted identified by its primary key `pk` for a reservation identified by its primary key `reservation_id`. The client needs to be authenticated and the client has either completed the reservation or been terminated with this particular property.

Endpoint: `/properties/<int:pk>/reviews/<int:review_id>/reply/`

Methods: `GET, POST`

Fields/payload: `reply`

Authenticated: `No`

Description:

This endpoint allows users to retrieve and add replies to a specific review for a property. The `pk` parameter in the URL should be replaced with the ID of the property, while the `review_id` parameter should be replaced with the ID of the review.

Note: Only the authenticated client and either the client is the user that reviewed or the property owner can reply to the review with `id = pk`.

Endpoint: `/properties/<int:pk>/reviews/<int:review_id>/edit/`

Methods: `GET, POST`

Fields/payload: `reply`

Authenticated: `No`

Description:

This endpoint allows users to edit their own review of a property.

Endpoint: `/properties/<int:pk>/reviews/<int:review_id>/<int:response_id>/edit/`

Methods: `GET, POST`

Fields/payload: `reply`

Authenticated: `Yes`

Description:

This endpoint allows users to edit their own reply with the associated `response_id` to a specific review `review_id` of a property `pk`.

Endpoint: `/<int:pk>/view/reserve/`

Methods: `POST`

Fields/payload: `start_date` `end_date` `number_guest`

Authenticated: `Yes`

Description:

This endpoint will create a new reservations for user.

Reservations

Endpoint: `/reservations/`
Methods: `GET`
Fields/payload: `search`
Search: `?=<any>`
Authenticated: `Yes`
Description:
This endpoint will list all the reservations for user in the database associated with the specified search results.

Endpoint: `/reservations/host/`
Methods: `GET`
Fields/payload: `search`
Search: `?=<any>`
Authenticated: `Yes`
Description:
This endpoint will list all the reservations for host in the database associated with the specified search results.

Endpoint: `/reservations/<int:pk>/view/`
Methods: `GET`
Fields/payload: `start_date` `property` `end_date` `status` `reserved_by`
Authenticated: `Yes`
Description:
A detail view of the reservation for user with the specified `pk` .

Endpoint: `/reservations/host/<int:pk>/view/`
Methods: `GET`
Fields/payload: `start_date` `property` `end_date` `status` `reserved_by`
Authenticated: `Yes`
Description:
A detail view of the reservation for host with the specified `pk` .

Endpoint: `/reservations/<int:pk>/cancel/`
Methods: `PATCH`
Fields/payload: `status`
Authenticated: `Yes`
Description:
This endpoint is used to cancel an existing reservation for user.

Endpoint: `/reservations/host/<int:pk>/approved/`
Methods: `PATCH`
Fields/payload: `status`

Authenticated: `Yes`

Description:

This endpoint is used to approved an existing reservation of host.

Endpoint: `/reservations/host/<int:pk>/approvedcancel/`

Methods: `PATCH`

Fields/payload: `status`

Authenticated: `Yes`

Description:

This endpoint is used to approve an existing reservation which is canceled by user for host.

Endpoint: `/reservations/host/<int:pk>/deniedcancel/`

Methods: `PATCH`

Fields/payload: `status`

Authenticated: `Yes`

Description:

This endpoint is used to deny an existing reservation which is canceled by user for host.

Endpoint: `/reservations/host/<int:pk>/denied/`

Methods: `PATCH`

Fields/payload: `status`

Authenticated: `Yes`

Description:

This endpoint is used to deny an existing reservation which is requested by user for host.

Endpoint: `/reservations/host/<int:pk>/terminated/`

Methods: `PATCH`

Fields/payload: `status`

Authenticated: `Yes`

Description:

This endpoint is used to terminate an existing reservation which is canceled by user for host.

Notifications

Endpoint: `notifications/`

Methods: `GET`

Fields/payload: `message`

Authenticated: `Yes`

Description:

This endpoint shows a list of **unread** notifications for the current authenticated user to view.

Endpoint: `notifications/<int:pk>/`

Methods: `GET`

Fields/payload: `message`

Authenticated: `Yes`

Description:

This endpoint allows the current authenticated user to view a specific notification identified by `pk`. This will set the notification to be read, and it will then get deleted after.