Models

User -> Inherit from AbstractUser

Field	Type
phonenumber	PositiveInteger
profile	image
comments	GenericRelation(‘properties.Comment’)

Property

Field	Type
name	char
address	char
city	char
country	char
postalcode	char
owner	User Model
images	image
description	char
number_of_guests	int
number_of_beds	int
number_of_baths	int
wifi	bool
petfriendly	bool
tv	bool
pillows	bool
reviews	GenericRelation(‘Comment’)

Comment

Field	Type
user	User Model
date	dateTimeField
body	char
rating	PositiveInteger
replyingTo	PositiveInteger
commentID	PositiveInteger
endOfCommentChain	bool

PropertyAvailabilities

Field	Type
Property	Property Model
start_time	DateField
end_time	DateField
price	decimal

Reservation

Field	Type
status	Char
property	Property Model
sender	User Model
res_start_time	DateField
res_end_time	DateField
price	Decimal

Notification

Field	Type
user	User Model
message	char
date	DateTimeField
content_type	ContentType
object_id	PositiveInteger
content_object	GenericForeignKey

API Calls

Login (**api/**)

Url and Function	Payload

token/
Obtain the token pairs.
	POST
	Body:
	`"refresh": string -> the refresh token`
	`"access": string -> access token`

token/refresh/ Refresh the access token.	**POST** **Body:** `"access": string -> access token`

accounts/

Url and Function	Payload

create-user/
Creates an user.
	POST
	Body:
	`"username": string -> username of user`
	`"first_name": string -> first name of user`
	`"last_name": string -> last name of user`
	`"password": string -> password of user`
	`"phoneNumber": positive integer -> phone number of user`
	`"avatars": image -> profile picture of user (can be null)`

update-user/
Updates the profile of the user currently logged in.
	POST
	Body:
	`"username": string -> username of user`
	`"first_name": string -> first name of user`
	`"last_name": string -> last name of user`
	`"password": string -> password of user`
	`"phoneNumber": positive int -> phone number of user`

	"avatars": image -> profile picture of user (can be null)
create-comment-user/<int:pk>/ Create a user comment on a specified user.	**POST** **URL:** "<pk:int>" : id of the user to be commented on **Body:** "user": User -> user (a host) making the comment "body": string -> contents of the comment "rating": int -> rating from 1 to 5 "content_type": User "replyingTo": null
create-comment-property/<int:pk>/ Create a property review/reply on a specified property.	**POST** **URL:** "<pk:int>" : id of the property to be commented on **Body:** "user": User -> user (user of the property host) making the comment "body": string -> contents of the comment "rating": int -> rating from 1 to 5 "content_type": Property "replyingTo": 1) null if it is a review and not a reply 2) positive int -> commentID of the comment it is replying to
view-comment-user/<int:pk>/ View comments made for a specified user. Pagination support	**GET** **URL:** "<pk:int>" : id of the user to be commented on **Response Body:** "user": User -> user (a host) making the comment "body": string -> contents of the comment

	"rating": int -> rating from 1 to 5 "content_type": User "replyingTo": null
view-comment-property/<int:pk>/ View comments made for a specified property. Pagination support	**GET** **URL:** "<pk:int>" : id of the property to be commented on **Response Body:** "user": User -> user (user of the property host) making the comment "body": string -> contents of the comment "rating": int -> rating from 1 to 5 "content_type": Property "replyingTo": 1) null if it is a review and not a reply 2) positive int -> commentID of the comment it is replying to

properties/

Url and Function	Payload
property-create/ Create a new property.	**POST** **Body:** "name": string -> name of the property "address": string -> address of the property "city": string -> city name the property resides in "country": string -> country name the property resides in "postal_code": string -> postal code for the property "images": image -> the images the property has(in links) "description": string -> description of the property "number_of_guests": int -> number of guests for the property "number_of_beds": int -> number of beds for the property "number_of_baths": int -> number of baths for the property "wifi": if the property has wifi or not

	"petfriendly": bool -> if the property is pet friendly or not "tv": bool -> if the property has tv or not "pillows": bool -> if the property has pillows or not
property-update/<int:pk>/ Update a property with specific primary key pk.	**PUT** **URL:** "<pk:int>" : id of the property to update **Body:** "name": string -> name of the property "address": string -> address of the property "city": string -> city name the property resides in "country": string -> country name the property resides in "postal_code": string -> postal code for the property "images": image -> the images the property has(in links) "description": string -> description of the property "number_of_guests": int -> number of guests for the property "number_of_beds": int -> number of beds for the property "number_of_baths": int -> number of baths for the property "wifi": if the property has wifi or not "petfriendly": bool -> if the property is pet friendly or not "tv": bool -> if the property has tv or not "pillows": bool -> if the property has pillows or not
property-view/<int:pk>/ Retrieve all information of a particular property.	**GET** **URL:** "<pk:int>" : id of the property to view **Response Body:** "id": int -> id of the property "name": string -> name of the property "address": string -> address of the property "city": string -> city name the property resides in

	"country": string -> country name the property resides in "postal_code": string -> postal code for the property "images": image -> the images the property has(in links) "description": string -> description of the property "number_of_guests": int -> number of guests for the property "number_of_beds": int -> number of beds for the property "number_of_baths": int -> number of baths for the property "wifi": if the property has wifi or not "petfriendly": bool -> if the property is pet friendly or not "tv": bool -> if the property has tv or not "pillows": bool -> if the property has pillows or not
property-search/ List properties based on filters and sort. Pagination support	**GET** **Query Parameter:** "wifi" : string -> query parameter Enum : “True” "petfriendly" : string -> query parameter Enum : “True” "tv" : string -> query parameter Enum : “True” "pillows" : string -> query parameter Enum : “True” "sortby" : string -> query parameter Enum : “guests”, “beds” **Response Body:** "id": int -> id of the property "name": string -> name of the property "address": string -> address of the property "city": string -> city name the property resides in "country": string -> country name the property resides in "postal_code": string -> postal code for the property "images": image -> the images the property has(in links)

	"description": string -> description of the property "number_of_guests": int -> number of guests for the property "number_of_beds": int -> number of beds for the property "number_of_baths": int -> number of baths for the property "wifi": if the property has wifi or not "petfriendly": bool -> if the property is pet friendly or not "tv": bool -> if the property has tv or not "pillows": bool -> if the property has pillows or not
property-owned/ List all properties of a particular user.	**GET** **Response Body:** "id": int -> id of the property "name": string -> name of the property "address": string -> address of the property "city": string -> city name the property resides in "country": string -> country name the property resides in "postal_code": string -> postal code for the property "images": image -> the images the property has(in links) "description": string -> description of the property "number_of_guests": int -> number of guests for the property "number_of_beds": int -> number of beds for the property "number_of_baths": int -> number of baths for the property "wifi": if the property has wifi or not "petfriendly": bool -> if the property is pet friendly or not "tv": bool -> if the property has tv or not "pillows": bool -> if the property has pillows or not
property-delete/<int:pk>/ Delete a property. Cancel all upcoming reservations and reject all requests	**DELETE** **URL:** "<pk:int>" : id of the property

reservation-create/ Create a new reservation.	**POST** **Body:** "property": Property->the property for the reservation "res_start_time": YYYY/MM/DD -> date which the reservation starts "res_end_time": YYYY/MM/DD -> date which the reservation ends "price": decimal -> the price per day of the stay
reservation-view/<int:pk>/ Retrieve information about a reservation.	**GET** **URL:** "<pk:int>" : id of the reservation **Response Body:** "id": int -> id of the reservation "status": string -> the status of the reservation "res_start_time": YYYY/MM/DD -> date which the reservation starts "res_end_time": YYYY/MM/DD -> date which the reservation ends "price": decimal -> the price per day of the stay "property": Property->the property for the reservation "requester": User -> The user who requested the stay
reservation-user/ Filter only the user reservation for current logged in user, ie. the reservations the current user made. Pagination support	**GET** **Response Body:** "id": int -> id of the reservation "status": string -> the status of the reservation "res_start_time": YYYY/MM/DD -> date which the reservation starts "res_end_time": YYYY/MM/DD -> date which the reservation ends "price": decimal -> the price per day of the stay "property": Property->the property for the reservation "requester": User -> The user who requested the stay

reservation-host/ Filter only the host reservation for current logged in user, ie. the reservations the current user received from others on their property. Pagination support	**GET** **Response Body:** "id": int -> id of the reservation "status": string -> the status of the reservation "res_start_time": YYYY/MM/DD -> date which the reservation starts "res_end_time": YYYY/MM/DD -> date which the reservation ends "price": decimal -> the price per day of the stay "property": Property->the property for the reservation "requester": User -> The user who requested the stay
reservation-status/?status=x Fill in x to one of the 8 status (Pending, Denied, Expired, Approved, Canceled, Terminated, PendingCancel, Completed)to filter the reservations with only that status for current logged in user. Pagination support	**GET** **Query Parameter:** "status" : string -> status to filter on Enum : “Pending”, “Denied”, “Expired”, “Approved”, “Canceled”, “Terminated”, “PendingCancel”, “Completed” **Response Body:** "id": int -> id of the reservation "status": string -> the status of the reservation "res_start_time": YYYY/MM/DD -> date which the reservation starts "res_end_time": YYYY/MM/DD -> date which the reservation ends "price": decimal -> the price per day of the stay "property": Property->the property for the reservation "requester": User -> The user who requested the stay
reservation-cancel/<int:pk>/ User cancels a reservation or sends a request if approved.	**POST** **URL:** "<pk:int>" : id of the reservation
reservation-approve-pending/<int:pk>/	**POST**

Host approves a pending reservation.	**URL:** "<pk:int>" : id of the reservation
reservation-deny-pending/<int:pk>/ Host denies a pending reservation.	**POST** **URL:** "<pk:int>" : id of the reservation
reservation-approve-cancel/<int:pk>/ Host approves a pending cancellation request.	**POST** **URL:** "<pk:int>" : id of the reservation
reservation-deny-cancel/<int:pk>/ Host denies a pending cancellation request.	**POST** **URL:** "<pk:int>" : id of the reservation
reservation-terminate/<int:pk>/ Host terminates an approved request.	**POST** **URL:** "<pk:int>" : id of the reservation
notification/list/ Shows all the notifications of the current logged in user. Pagination support	**GET** **Response Body:** "id": int -> id of the notification "date": date -> date of the notification "message": string -> the message of the notification "object_id": int -> ID of the object the notification is referring to "user": User -> The user the notification shows "content_type": ContentType -> type of the notification
notification/<int:pk>/ Reads a specific notification.	**GET** **URL:** "<pk:int>" : id of the notification

	Response Body:
	`"id": int -> id of the notification`
	`"date": date -> date of the notification`
	`"message": string -> the message of the notification`
	`"object_id": int -> ID of the object the notification is referring to`
	`"user": User -> The user the notification shows`
	`"content_type": ContentType -> type of the notification`

notification/delete/<int:pk>/ Deletes a specific notification.	**DELETE** **URL:** `"<pk:int>" : id of the notification`