Final project

Auctions API using Django and Django REST Framework

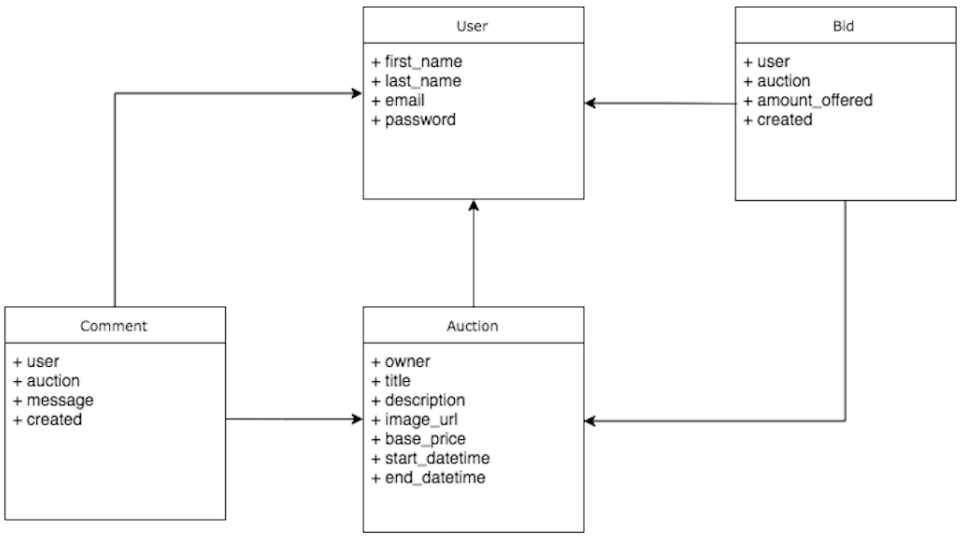
In order to put into practice all the concepts we've covered during the course, you'll be in charge of implementing a fully working REST API from scratch, using Django, DRF and other libraries you might need/know.

The general idea of this final project is to combine everything you know so far, to make a real world implementation of an API. This won't only include the Python solution that must return the expected results, but you'll also need to take care of the design of the Django project, writing test cases, deployment to a production environment, etc.

👉 Hands on!

This time you'll implement an API related to an Auctions system, where users can create their own publications and others can bid a certain amount of money for the product on sale or write a message asking for something.

This is the data model that will be used.



Note: feel free to add/remove fields or make any modifications that you need in the models

👉 Your tasks

1) API Endpoints

Implement the following endpoints in your API. All the data returned from any of those endpoints must in JSON format, and paginated.

- **GET /api/auctions**
- **GET /api/auctions/:auction_id**
- **GET /api/auctions/:auction_id/bids**
- **GET /api/auctions/:auction_id/comments**
- **POST /api/auctions**
- **PUT /api/auctions/:auction_id**
- **PATCH /api/auctions/:auction_id**
- **DELETE /api/auctions/:auction_id**
- **POST /api/bids**
- **DELETE /api/bids/:bid_id**
- **POST /api/comments**
- **DELETE /api/comments/:comment_id**
- **POST /api/signup**
- **POST /api/login**

2) Authentication

Choose one (or more) of the following authentication methods and implement it as part of the API.

- Basic authentication
- Session authentication
- Token authentication
- JSONWebToken authentication
- Custom authentication

3) Permissions

Make usage of DRF permission features to determine what an user is able or unable to do. For example, editing an auction should only be able for the owner and not for an external user.

4) Django custom command

Implement a Django custom command that gets all the finished auctions and prints the user with the highest offer for each auction.

5) Testing

Having all the functionalities (or most of them) well tested is one of the most important things in a professional project. Make sure to implement test cases for all the endpoints in your API, considering both successful and failure scenarios.

6) Deployment

In order to show the final result of this project, it must be deployed in a production environment where everyone can access. For this task we recommend you to use Heroku (https://www.heroku.com/), but if you know or want to use any other service feel free to do it.

7) Prepare for Demo Day

One of the best ways to show in action all the endpoints that you've implemented in your API is using a service called Postman (https://www.getpostman.com/).

There you have the chance to perform every request to your API already deployed in Heroku, and save all the customization for each request (URL, authentication, payloads, etc) separately in what they call a **collection**.

Make sure to have your Postman Collection saved and ready with all your requests configured for the Demo Day 😉

🤓 Optional Topics

If you're done with the project and you still have time (and energy), you can take on any of these optional topics:

a) Implement frontend

You can design a website that acts as frontend for your API. You could also include a frontend using pure Django and templates, but be aware that by not using the API you might be duplicating a lot of code.

b) Integrate a payment processor

We recommend to use Stripe (https://stripe.com/). It has a "sandbox" mode with Credit Card numbers for testing. This will need a minimal frontend implemented to work.

c) Include a mobile app

This is way out of the scope of the course and project; but if you know how to build simple mobile apps (maybe with React Native or Phonegap), you could use your API to power it.