Task

At ZidShip, we work with a lot of couriers (more than 35 of them). It is not sustainable to custom implement courier integrations independently. We need to design a framework that defines a unified interface for the rest of the system to communicate through, and more importantly, allow us to integrate more couriers without making interface decisions all over again.

In this assignment, your task is to design such a framework with the following constrains:

- Have one unified interface for all couriers.
- The minimal feature set of every courier should be: creating a waybill, print waybill label, tracking shipment status, and mapping statuses to equivalent statuses in the system.
- Optional cancellation functionality for some couriers.

Bonus points:

- Implement one real courier as a proof of concept (SMSA, Aramex, Shipbox, etc.).
- Expose this API as a web service.
- Add mechanisms for doing HTTP calls and doing HTTP retries.

Notes:

- The focus of this task not only the code but also the architectural thought process. We would like to see a short writeup describing the tradeoffs you took and how you arrived at this specific design.

- We believe that API design is a craft, so we expect that you would follow ***best practices*** in API design. Including:
    - Well-thought input values and return values of each call.
    - Proper naming.
    - Documentation.
    - Tests.

You **must** use Python and Django for the application but can use whatever (Data-store/s, and Message Broker & Workers) you want.

- You should follow the best practices of that specific framework.
- You should be able to justify your choices, especially the data-stores.

If you have any questions concerning the project or the implementation, do not hesitate to contact us.