Part 1. RESTful API of FinInv's invoice financing platform

The FinInv platform is an invoice financing platform where sellers create invoices and sell these invoices to funders in order to get advance payment and thus better manage their cash flow. The users of the platform are sellers (who issue invoices and can then sell them), funders (who buy invoices and later get paid back by the buyers), and buyers (who receive invoices, approve them and eventually pay them).

The main process to be supported by the FinInv platform is the invoice financing process, which proceeds as follows. A seller creates an invoice addressed to a given buyer via the FinInv platform's API. The buyer can approve, reject or dispute the invoice. If they dispute the invoice, the seller can create a debit note or a credit note to correct the invoice. Note that the original invoice is never modified. Instead, a credit note or a debit note is added to it. When a credit or debit note is added to an invoice, the buyer can again accept, reject, or dispute the invoice. In the latter case, the seller might issue another credit or debit note. Sellers may also withdraw an unapproved invoice from the platform anytime. This is typically done in order to resolve invoice disputes outside the FinInv platform.

An invoice consists of a seller, a buyer, an issue date (assigned automatically when the invoice is submitted), a payment due date, a total amount and an invoice body. The invoice body has a relatively complex structure (consisting of line items) but we will treat it as a black-box in this exercise (e.g. you can treat it as a string here). Since detailed data about all buyers and sellers is already registered in FinInv, the buyer and the seller of an invoice can be referred to using their unique identifier in the FinInv platform.

A credit-debit note consists of a reference to an invoice, and an amount. The meaning is that the amount of the credit-debit note is added to the total amount of the invoice to which it is attached (i.e. if the invoice is for 1000 euros its has an attached credit-debit note with an amount of -200, then the effective amount of the invoice is 1000 + (-200) = 800.
Once an invoice is approved, it is offered to funders. Funders can view the invoices that are on offer and bid for an invoice. Each bid includes a "markup" (e.g. 2%, 3%). The markup is a percentage deducted from the invoice amount in order to compensate the funder. In addition, FinInv applies a 1% fee for each invoice sold via its platform. If an invoice is for 1000 euros, the markup is 3% and the fee is 1%, then the seller receives 960 euros for the invoice sale.

Bids for an invoice are collected during a period of 24 hours after an invoice is approved. At the end of this bidding period, the invoice is sold to the funder who offered the lowest markup. The funds (minus markup) are then debited from the account of the winning funder (i.e. the account of this funder held at FinInv). FinInv retains its 1% fee and transfers the remaining amount to the seller. Sometimes an invoice is not sold (if it receives no bid).

The buyer of an approved invoice eventually pays the amount of the invoice (plus late fees if applicable) to FinInv. FinInv then transfers the proceeds to the funder who bought the invoice (if the invoice has been sold) or to the seller (if the invoice was not sold).

**Task 1 [5 points].** Design a domain model for the FinInv platform based on the above description.

**Task 2 [5 points].** Based on the domain model, design a resource model using the RESTful service design method presented in the lectures.

**Task 3 [5 points].** Specify a state machine capturing the lifecycle of an invoice.

**Task 4 [15 points]** Specify an apiary blueprint of the RESTful interface that the FinInv platform will provide to its users. For each operation on a resource, you should specify an example of a request/response.

---

**Part 2. Booking System**

Given the experience you gained while developing RentIt’s sales and inventory management systems, you have been hired to implement a Hotel booking system. The Hotel booking system is similar to RentIt’s reservation subsystem, such that you will be able to port several parts of your code to this system. The Hotel booking management system provides a set of operations via a REST API.

1. The system must provide an operation to let customers query a list of rooms according to their requirements. The query takes as input the check-in/check-out dates and returns the list of available rooms, grouped by room type (e.g. single, double, twin, studio), number of each type and price per night (for simplicity, we will assume that the price is per room without taking into account the number of guests).
2. Based on the result of a previous query, the customer submits via the REST API a booking order which includes the information of the customer (e.g. name, address, email), the list of rooms (e.g. list of identifiers or reference to the rooms) and check-in/check-out dates. The booking system must either confirm the booking order with a 201 or reject the booking order if one of the request rooms is no longer available.
3. The customer can cancel a booking without any fee up to 10 days before the check-in date. If the cancellation request happens after this time limit, the system must notify the rejection of the request and notify the customer that he/she must pay the full booking amount.
4. For travel agencies, the REST API must also include the possibility of submitting a booking request. This kind of booking will block a room for up to 48 hours, meaning that the room will not be included as part of an availability query. However, once the period of 48 hours is finished, if the booking request is not confirmed, the room will be unblocked.

**Task 5 [10 points].** Design and implement the domain model for the booking system. Include a seed database for checking the coverage of your work.

**Task 6.** Based on the domain model above, implement a REST adapter including the following:

1. [5 points] Resource and Assemblers
2. [5 points] Rest controller
3. [5 points] Applications service
4. [5 points] Support for hypermedia