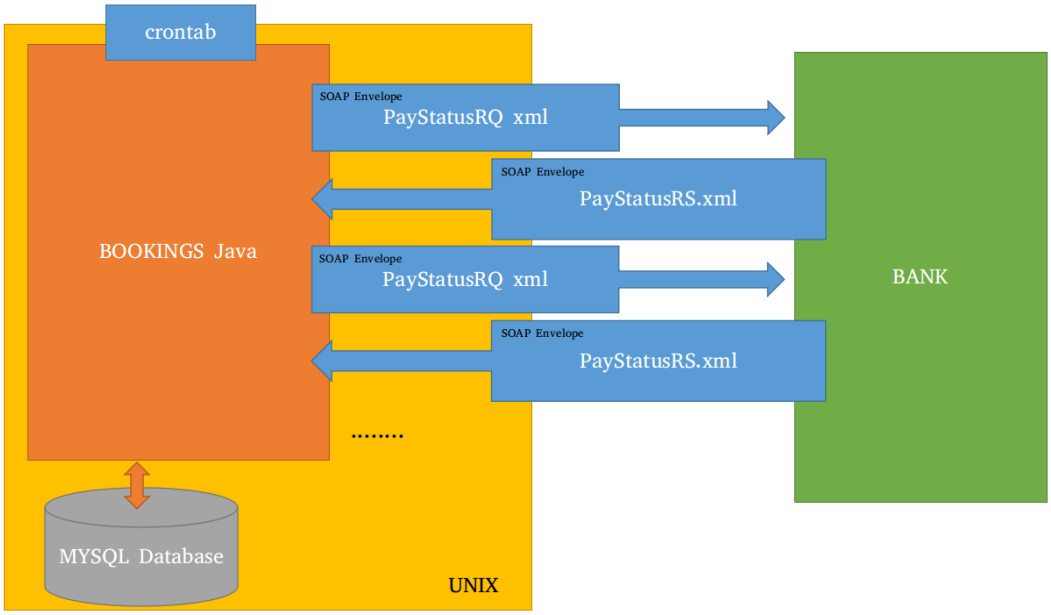
Our system have a crontab file which launch the java BOOKINGS process "*Check pending status Bookings*". This task, read from the database the bookings that have ***PayStatus***="***pending***", and send a request to the external system BANK, to update the payment of the booking if proceed.

This crontab, launch the task each 5 minutes, and this task launch a message for each booking that found in the database as PayStatus="pending" called PayStatusRQ (see DATA for the exercise for further details). Usually it sends about 200 request and BANK takes about 1 second to reply, but the number of request could vary about +100 and response time from BANK could vary to 1 to 2 seconds.

The BOOKINGS system reads from the database the bookings in Paystatus pending and creates a request to send it to BANK using JAVA code.



There is a problem in production: Sometimes the task, takes more than 5 minutes to send all PayStatus messages to BANK, so the following execution overlaps the preceding execution, and finally it causes a block in the process and the system BOOKINGS collapses. The client finally have to call to us alerting that the payments are not being updated so we discover it after some hours have passed. When we relaunch the process a lot of bookings are pending so the problem is worse, so manually we update some bookings in the database that we know that are paid (checking and external system manually), and after that, we can restore the service. The client requirement is that the booking pay status should be updated in no more than 10 minutes.

DATA for the exercise

Request example RQ_Paystatus:

Message 1 example

```
<PayStatusRQ>
    <Booking ID=”111” PayStatus=”pending” Start=”01-01-2018” End=”07-01-2018”>
        <Flight Number=”12334” Origin=”MAD” Destination=”BCN” Company=”IB”>
            <Price Amount=”210” Currency=”Euro”>
        </Flight>
        <Hotel Name=”HotelTest” Code=”HTS1”>
            <Price Amount=”180” Currency=”Euro”>
        </Hotel>
        <TotalPrice Amount=”390” Currency=”Euro”>
    </Booking>
</PayStatusRQ>
```

Message 2 example

```
<PayStatusRQ>
    <Booking ID=”222” PayStatus=”pending” Start=”01-01-2018” End=”07-01-2018”>
        <Flight Number=”123125” Origin=”MAD” Destination=”BCN” Company=”IB”>
            <Price Amount=”220” Currency=”Euro”>
        </Flight>
        <Hotel Name=”HotelTest” Code=”HTS1”>
            <Price Amount=”180” Currency=”Euro”>
        </Hotel>
        <TotalPrice Amount=”400” Currency=”Euro”>
    </Booking>
</PayStatusRQ>
```

Message 3 example

```
<PayStatusRQ>
    <Booking ID=”333” PayStatus=”pending” Start=”01-01-2018” End=”07-01-2018”>
        <Flight Number=”123123” Origin=”MAD” Destination=”BCN” Company=”IB”>
            <Price Amount=”200” Currency=”Euro”>
        </Flight>
        <Hotel Name=”HotelTest” Code=”HTS1”>
            <Price Amount=”180” Currency=”Euro”>
        </Hotel>
        <TotalPrice Amount=”380” Currency=”Euro”>
    </Booking>
</PayStatusRQ>
```

Response example RS_Paystatus

Message 1 example

```
<PayStatusRS>
    <Booking ID=”111” PayStatus=”paid”/>
</PayStatusRS>
```

Message 2 example

```
<PayStatusRS>
    <Booking ID=”222” PayStatus=”paid”/>
</PayStatusRS>
```

Message 3 example

```
<PayStatusRS>
    <Booking ID=”333” PayStatus=”pending”/>
```

</PayStatusRS>

Database

The database is a MySQL database, accessed from the application using

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

Each node/attribute of the Booking xml are stored in a table as a column

Booking ID	PayStatus	Start	End	FlightXML	HotelXML	TotalPrice	TotalPriceCurrency
111	pending	*01-01-2018*	*07-01-2018*	<Flight ..... >	<Hotel ....>	390	Euro
222	pending	*01-01-2018*	*07-01-2018*	<Flight ..... >	<Hotel ....>	400	Euro
333	pending	*01-01-2018*	*07-01-2018*	<Flight ..... >	<Hotel ....>	380	Euro

.....

Using the query to get the pending bookings info

SELECT * FROM Bookings WHERE PayStatus = “pending”

Bookings JAVA process

The process receive the activation from the crontab and the main class begin the query to the database, and then build a SOAP request that contains the XML and send it to a endpoint in BANK system.

It waits until BANK replies, and then reading the response updates the database.

QUESTIONS:

1- Which solutions can you design to avoid the problem in production of overlaping requests?

2 - Which improvements could you make in the systems and database (unless in BANK) and which new systems you could include to improve this task?

3 - Is your solution a 100% reliable system to do this task?