# BlablaMove - Architecture - Week 2

## >> Two use cases, Bob & Alice :

*> Use Case 1 - Bob :*
1. Bob is a lambda student who wants to move.
2. Bob need to transport his bed from is parents house (Nice) to is new student apartment (Sophia).
3. Bob is a smart guy so he decides to use BlablaMove.
4. He logs in on BlablaMove : he has the right amount of points.
5. He fill a form where he gives some information : the start point of the things he wants to move, the arrival point, the size of his bed, the weight of his bed, when he wants to move (range ?) and the maximum  number of points he wants to spend.
6. The system give him a list of results who answer his need. (Number of points/Date/Hours/...)
7. Bob chose a ride for his bed.
8. The system answer him with a recap.
9. Bob confirms.
10. He receives a confirmation mail from BlablaMove : Charlie can help him to move his things.
11. --- Ellipse ---
12. At the chosen date, Charlie goes to Bob house and take his bed.
13. Charlie goes to Sophia.
14. Bob receives a notification that confirm the delivery of his bed.
15. Bob can now confirm the transaction to BlaBlaMove.
16. After the confirmation from Bob, BlablaMove collects the points that were needed for this transaction.

*> Use Case 2 - Alice :*
1. Alice is a student who lives in Nice and goes to Sophia in car every day for her studies.
2. She decides to create a BlablaMove account to help others student to move their things.
3. She creates her account and specify her type of car (5 places, medium) and her disponibilities.
4. On BlaBlaMove she offers to transport things between Nice and Sophia every day, between 7:30 am (Nice) and 8:30 am (Sophia).
5. When she creates the offer, BlaBlaMove suggests an amount of point she should charge for the delivery. (Based on the number of points that are usually charged for this distance. )
6. She can choose to charge the amount BlaBlaMove suggests her, or she can make a new offer. The number of points she can charge for a delivery will be in a certain range proposed by the system, it can’t be to expensive compared to the average offers.
7. One day she receives a mail from BlaBlaMove : Dimitri wants her to transport a box from Nice to Sophia at a certain date.
8. She agrees to do it and confirm on BlablaMove.
9. --- Ellipse ---
10. At the chosen date, she goes to Dimitri's house and take his box in her car
11. She goes to Sophia and leave the box where Dimitri told her to.
12. She confirms on BlablaMove that she delivered the box.
13. She receives points for the delivery.

# BlablaMove - Architecture - Week 2

*> Use Cases variations to handle :*
1. Bob chooses Alice to move his bed but, in the end, Alice isn't available and refuse (She cancels on BlabalMove).
2. Bob chooses Alice but want to cancel the transfert after confirmation.
3. Bob or Alice aren't at the meeting point.
4. The package is not delivered or is broken during the transfert.
5. Bob want to move something but he don't have enough points.
6. One car isn't enough to get the box from A to B. (Connection necessary)

---

>> Focus on billing :

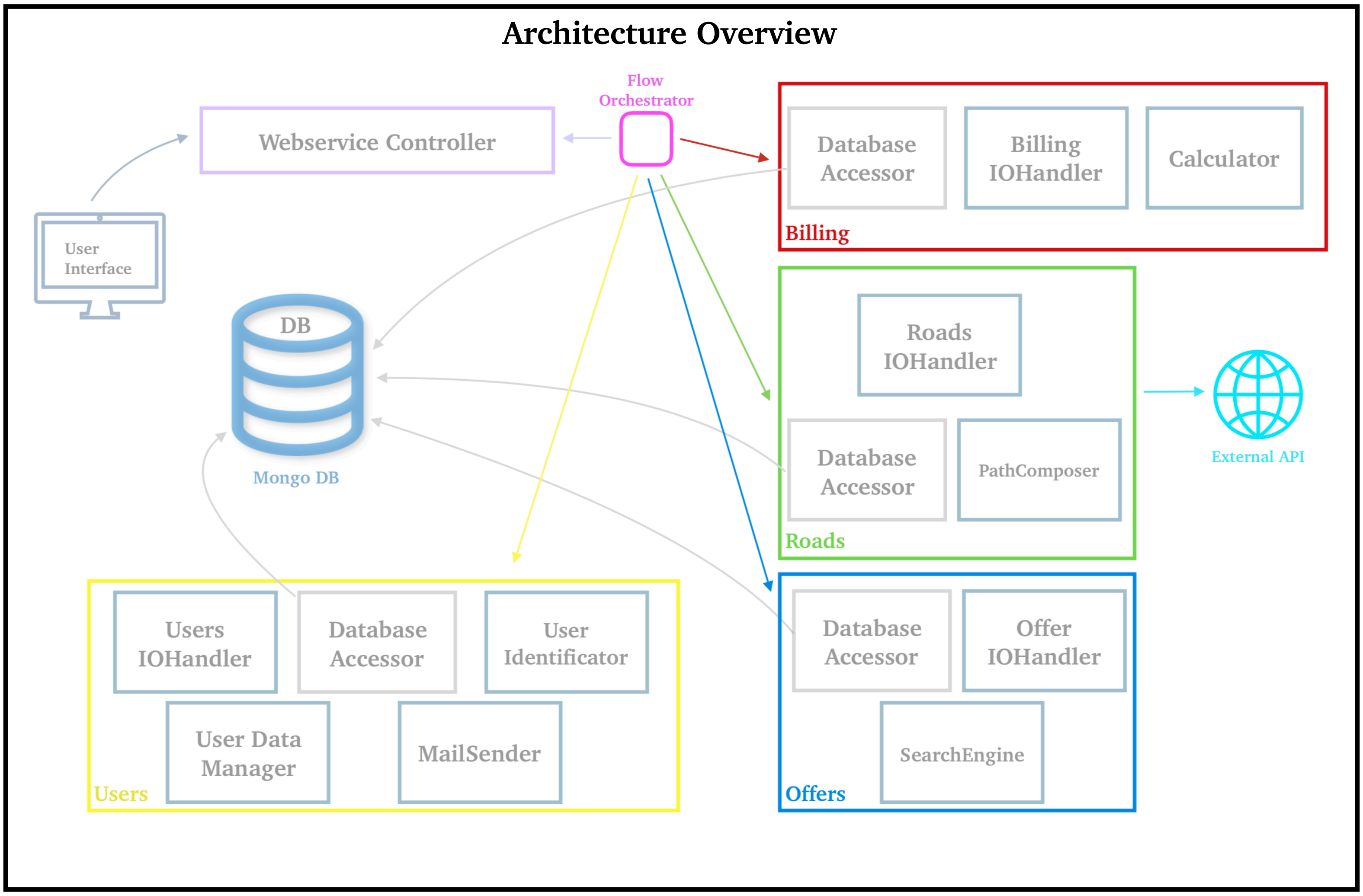
The calculation of our billing will be done according to the following parameters :

* Different prices depending on the distance to cover, the weight and size of the package(s).
* When a user wants to make a reservation, the more he will be close to the date of the move, the more he will spend points on the delivery. ( If he want to make a reservation for the next day, it will be more expensive than for the next week.)
* BlaBlaMove will take 10% of the points that Bob will pay for Alice to move his things. ( If Bob pay 100 points, Alice will receive 90 points and BlablaMove 10 points.)
* Optionally, when he will fill the form to do the research at the start of the process, Bob will be able to say how many points he want to spend on his move. (If he has 300 points, maybe he want to spend only 100. So if an offer cost more than 100 points, it won’t appear in his research.)
* At the end of the process, the points transactions will needs a double validation : first Alice will confirm that she has deliver the package of Bob, then Bob will confirm that is package was indeed safely delivered. If there’s a problem in the validation process, a rollback will be possible.
* When Alice make an offer, she can choose a minimal amount of points she wants to charge for the delivery. However, depending of the size/weight of the things Bob wants to move, it will cost more to Bob. So if he wants to move his bed, it will be more expensive for him than 2 light boxes. But the minimum price for the delivery will be the price that Alice selected when she put the offer online.

---

## Before the changes imposed by the client

---



---

>> Technological choices :

* **Java - Springboot :** Quick to deploy, strong code basis, everyone in the group already used Springboot in previous projects.
* **SQL Database :** Transactional management of the requests
* **Docker :** Independent deployment of BlaBlaMove

---

# BlablaMove - Architecture - Week 2

>> Quick overview of the work to do :

* **Week 42:**
  - Beginning of the code :
    - Start of database (Data model)
    - Listenning services
    - Components for Billing, Road and Users (Identification, Offer, Facturation…)

* **Week 43 (MVP):**
  - Work on the modules connection
  - Setting up the mocks for externals API
  - Properly finish the data model for the database
  - Billing :
    - Implementation of a first calculation and simple billing process

* **Week 44:**
  - Work on User
  - Finalization of Billing
  - Start of a demo preparation for the POC

* **Week 45:**
  - POC demo

---

>> A quick overview of what has been done in recent weeks, compared to what was planned (and listed above) :

* **Week 42:**
  - We have talked about improvements for our Billing system.
  - We have separated the tasks to be carried out during this week.
  - We have put in place a preliminary architecture for our project.
  - We also have begun our implementation of BlablaMove.

* **Week 43:**
  - We have established entry points for our user.
  - We have established the internal routing for our system.
  - We have established data structures used inside our business logic.
  - And finally we have established a temporary no SQL Database.

* **Week 44:**
  - We added and improved user entry points.
  - We continued to establish the internal routing and data structure.

- We have containerized our application.

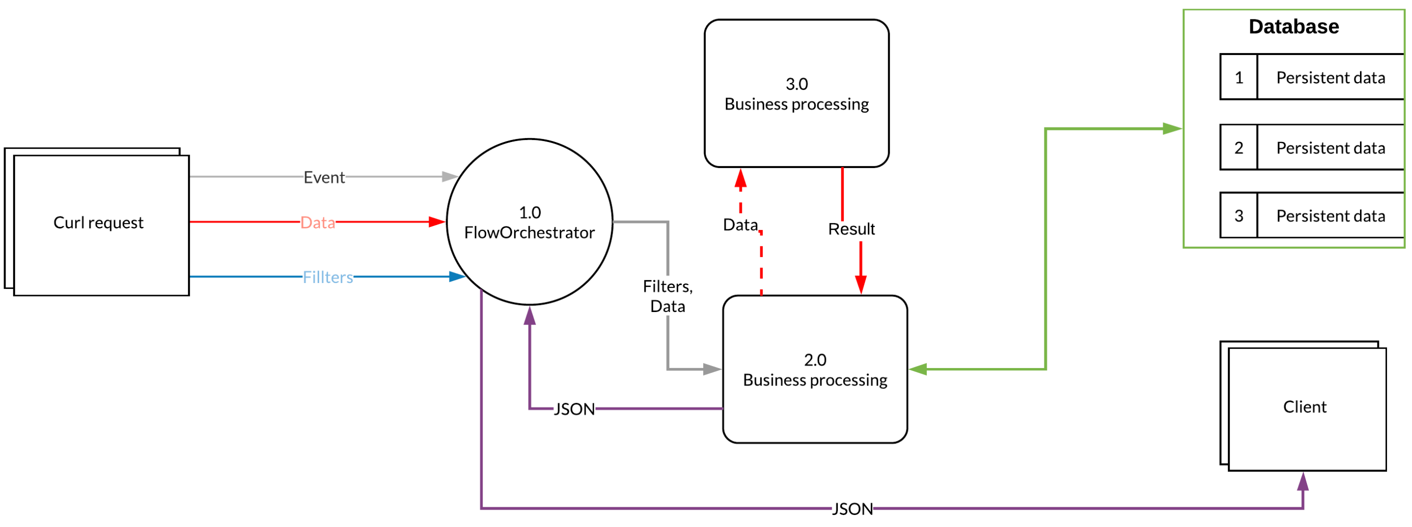
* Week 45:

- End-to-end testing.
- We have realized scripted scenario for Bob and Alice use cases.
- We have prepared a presentation and a demonstration of our POC.

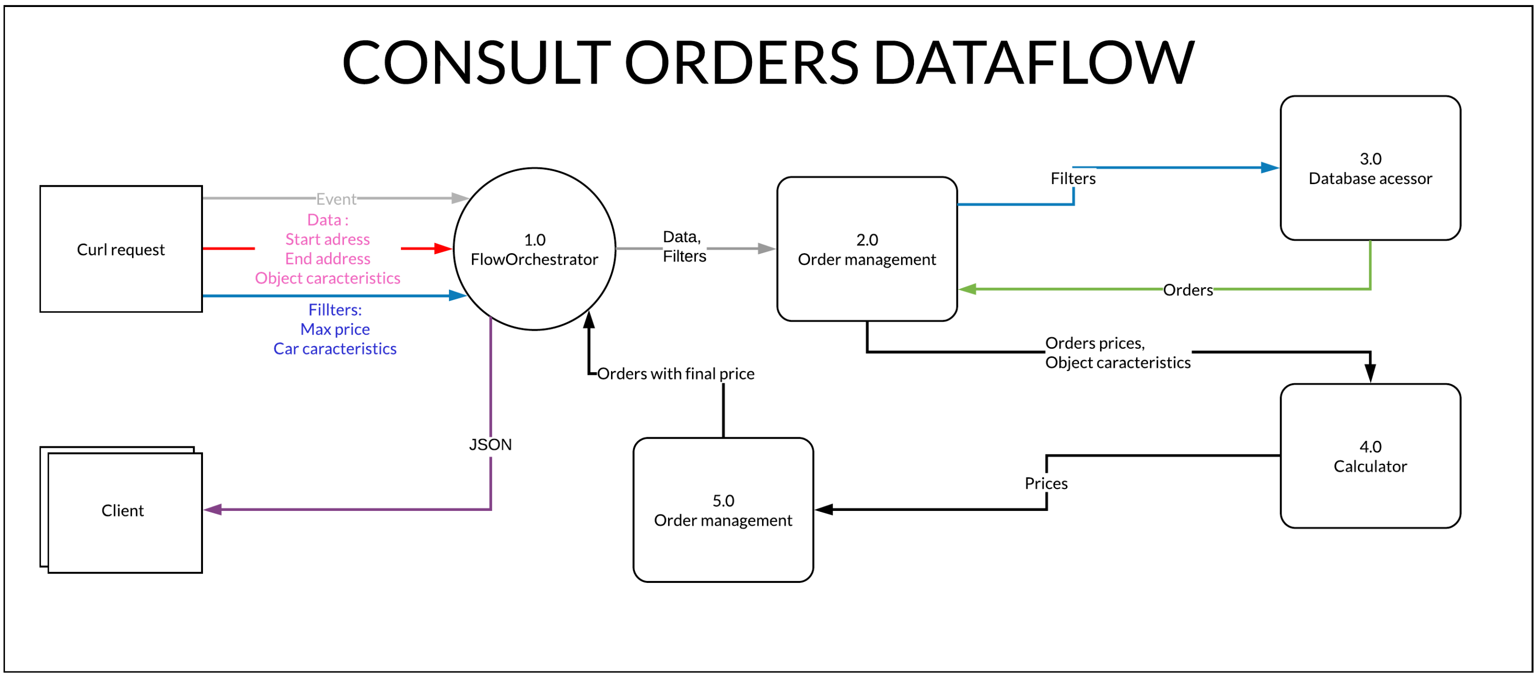
---

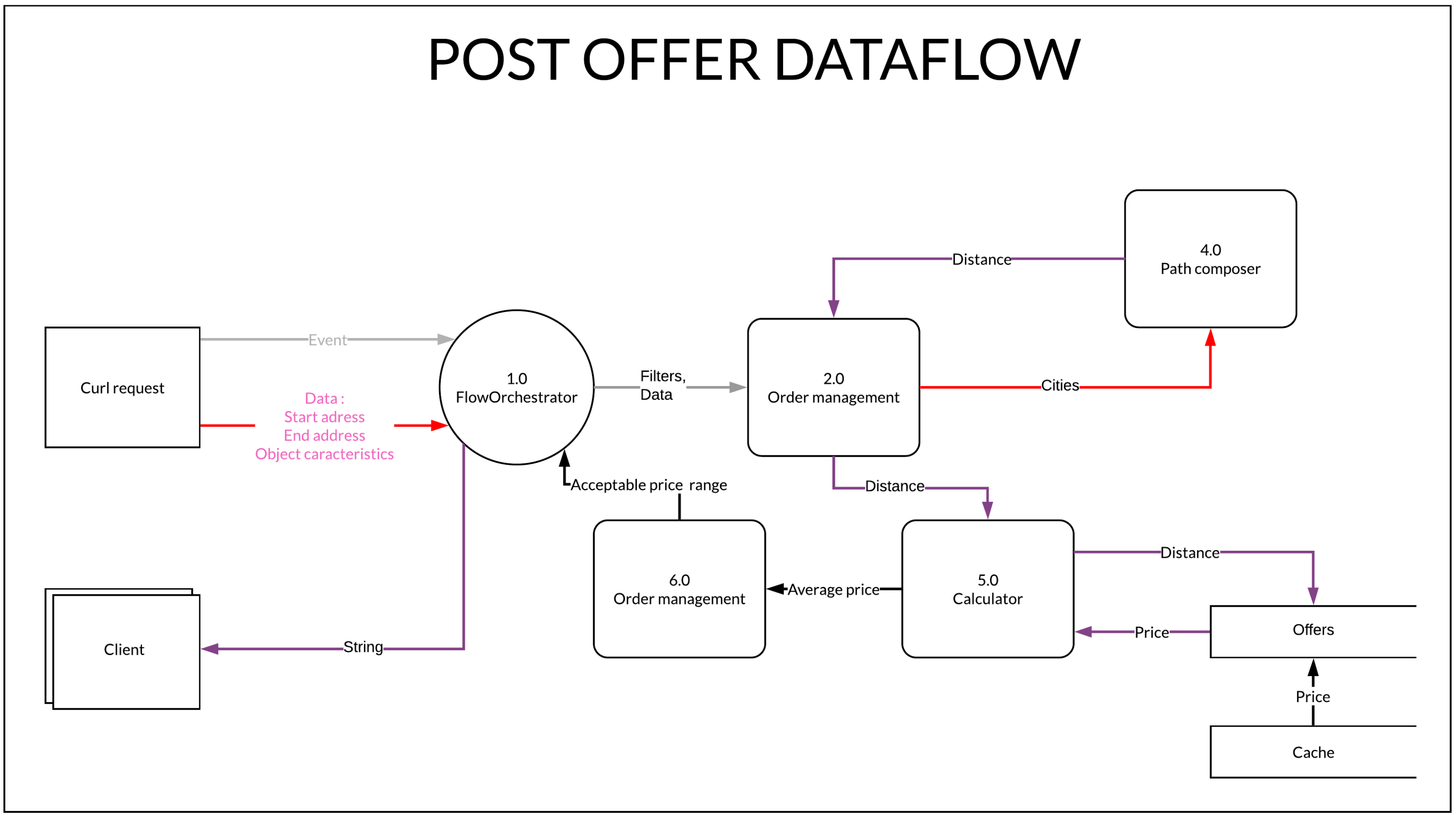
You will find below some diagrams representing different data flows in our POC.

## GENERAL APPLICATION DATAFLOW



# BlablaMove - Architecture - Week 2





## After the changes imposed by the client

As our client is worried about hackers who want to damage BlablaMove by denial of service, he wants the application to be more safe. Thus, some changes will be needed in our application.

Right now, if a financial transaction (debit - credit) is launched and the database crashes during the process, debit can be successful while credit can fail. In this case, the database state would be inconsistent.

To avoid this behavior we noted different ideas.
One of them is to create a satefy dedicated module. The Integrity Auditor (temporary title) would be hosted on a different thread that would check in background the database state. For that, we, for instance, could establish a journaling system that would check the state of the transactions, rollback or finish them if necessary.

The NoSQL -> SQL upgrade purpose is to ensure data consistency. An idea would be to merge our financial operations (bank account credit/debit) within a single SQL request for a simplified rollback.

We will also establish a data duplication (Master/Slave) system to secure and keep a backup of our data if the database goes down.

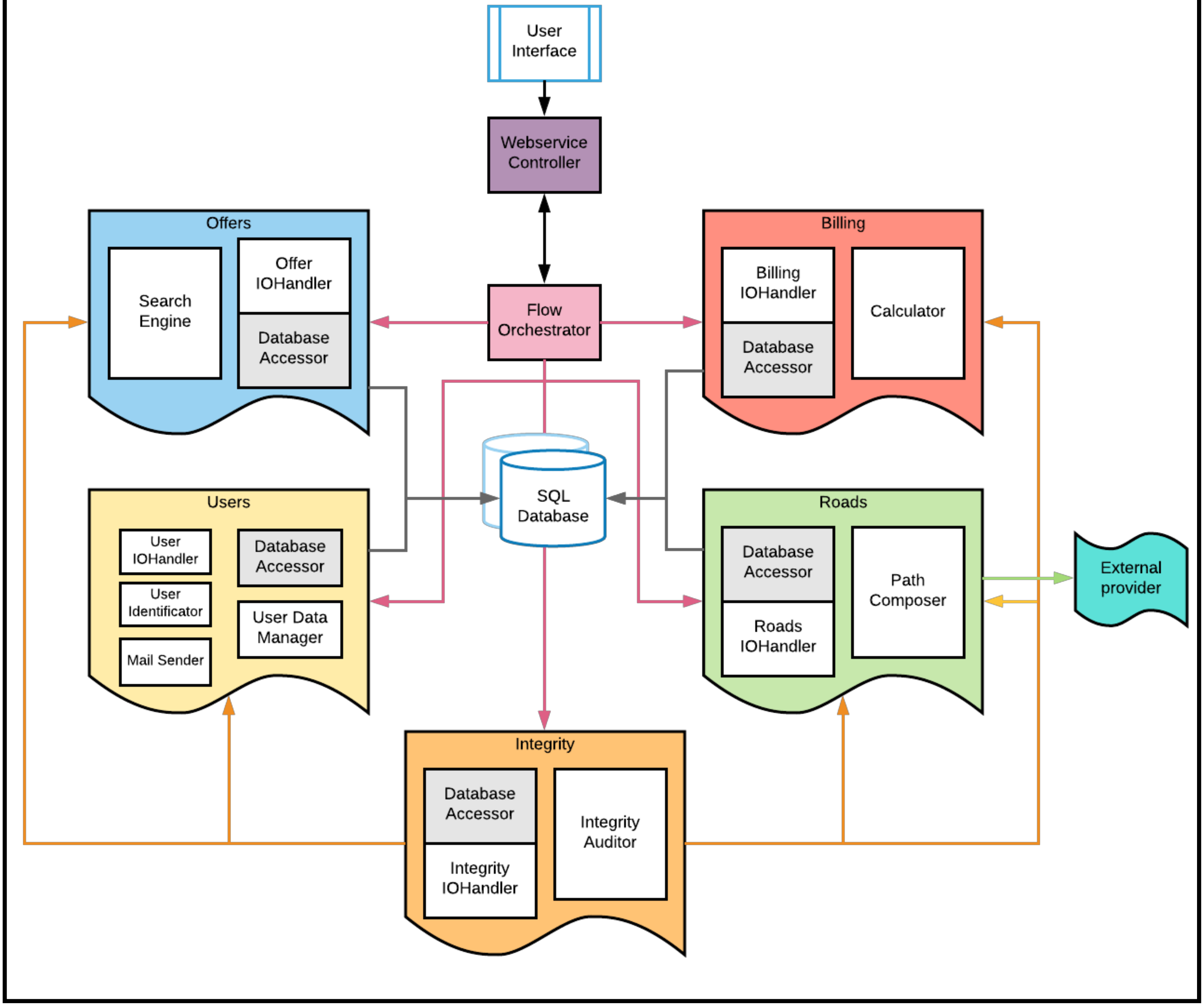
As we want to add a new component («Integrity») to check the data consistency, we have to do some choices.
Instead of being connected with the other components through the Flow Orchestrator, this new service will be directly linked to Billing, Road, Users and Offer. With this approach, we will avoid the overloading of the Flow Orchestrator.

Moreover, for the data duplication system, we will establish several instances of the same database and link them to ensure data replication and consistency.

Finally, with this changes in mind, we can assume that most of our code will hold up well. We will, of course, need to adjust the parts that were related to the database itself. So we will essentially modify each database accessor and our repositories in order to move from MongoDB to MySQL. Some dependencies and Java code has to be changed but we do not feel the need of refactoring the existing use-case flows.

# BlablaMove - Architecture - Week 2

## New Architecture Overview



## >> Our new use case, Roger:

*> Roger story :*
**1.** Like Bob, Roger is a student who wants to move with the help of BlaBlaMove. He has an account and a good amount of points.
**2.** Roger fills a form where he gives the informations needed for the moving of his things.
**3.** After few more clicks, he finally picks a ride.
**4.** On the other side, Alice confirms the ride and Roger receives a mail from BlaBlaMove.
**5.** --- Ellipse ---
**6.** The transportation takes place at the chosen date.
**7.** Roger receives a notification from BlaBlaMove that confirms the deliver of his things.

> Path 1 :
**8.** Roger wants to confirm that his package as indeed well arrived at the intended location.
**9.** When he confirms, he receives an error message from BlaBlaMove : the confirmation has not been saved and Alice will not be payed.
**10.** Roger needs to confirm again on BlaBlaMove so Alice can receives the points.
**11.** He does so and the rest of the transaction goes as planned.

> Path 2 :
**8.** Roger wants to confirm that his package as indeed well arrived at the intended location.
**9.** He confirms and returns to his life.
**10.** A little while later, he receives an email from the application. A problem occurred with his confirmation and he need to do it again.
**11.** He does so and the rest of the transaction goes as planned.

> Path 3 : For improvement.
**8.**
**9.**
**10.**
**11.**