Optimization of the long-distance bus Network at BlaBlaCar

Thomas Bernardi

Data Scientist, Data Pro Supply squad
thomas.bernardi@blablacar.com


BlaBlaCar

BlaBlaCar

"The go-to marketplace for shared road travel"

22 countries

100 million members

25 million travelers
per quarter

BlaBlaCar

BlaBlaCar Bus

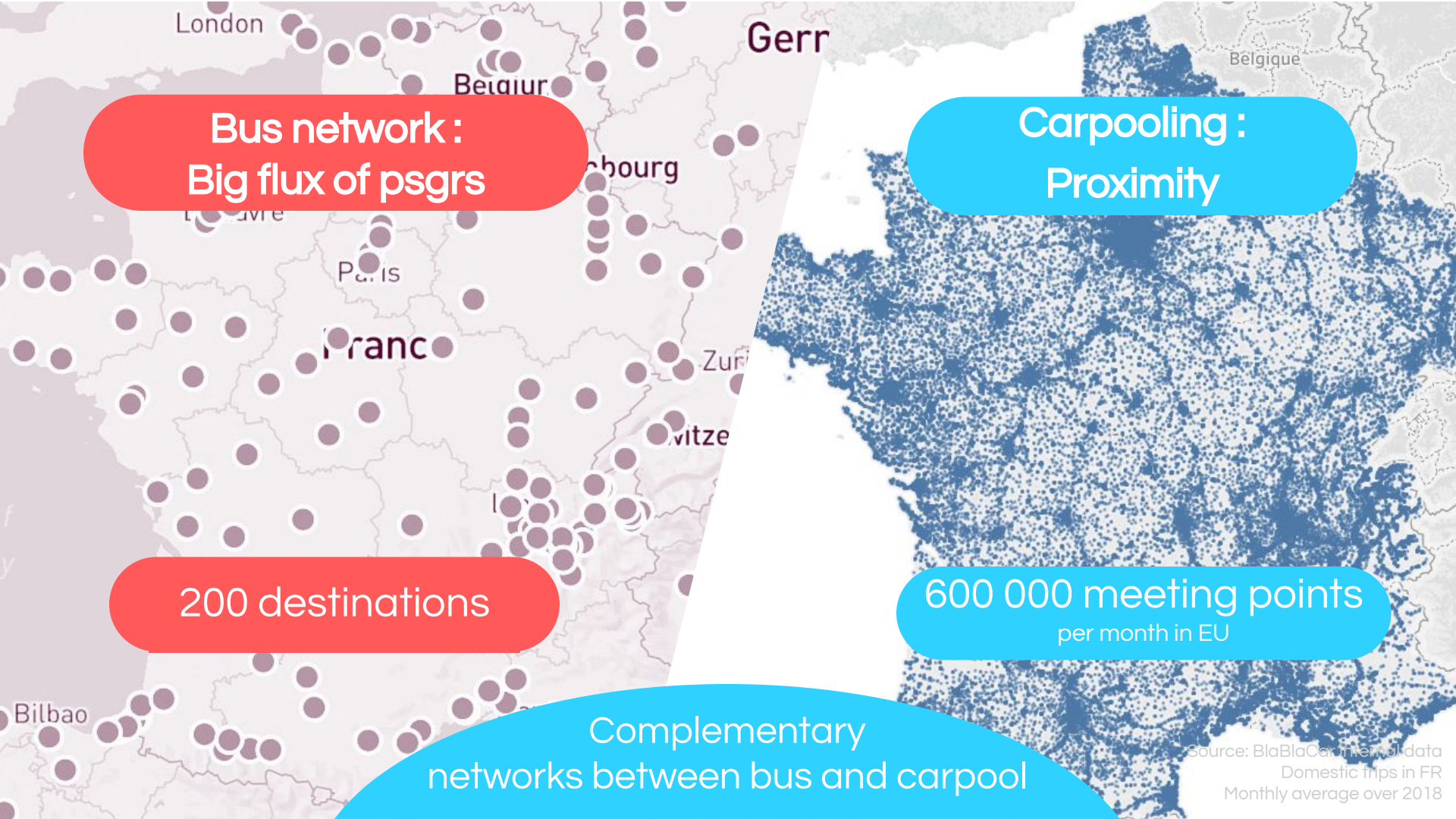
BlaBlaCar Daily

1,6 millions

tonnes of CO_2

saved per year

the equivalent of
Paris's annual
transport emissions


London
Gerr
Belgique
Bus network :
Big flux of psgrs
Carpooling :
Proximity
Paris
200 destinations
600 000 meeting points
per month in EU
Bilbao
Complementary
networks between bus and carpool
Source: BlaBlaCar internal data
Domestic trips in FR
Monthly average over 2018

Today's agenda

01 Problem scoping

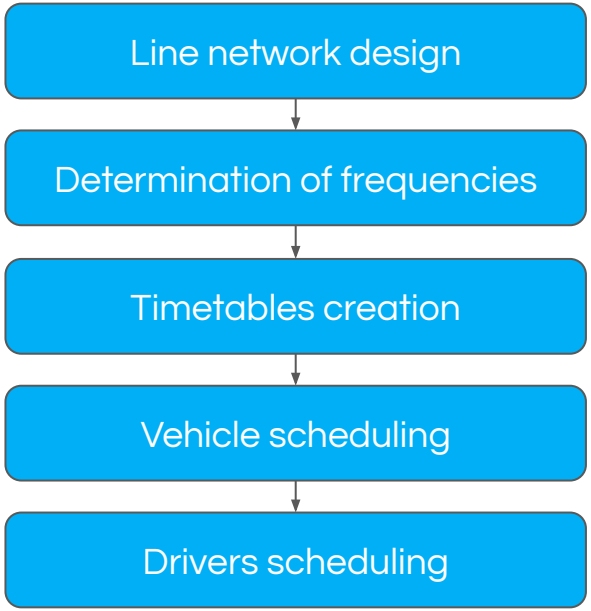
02 Insights on the modeling approach

03 How we make it real for end users

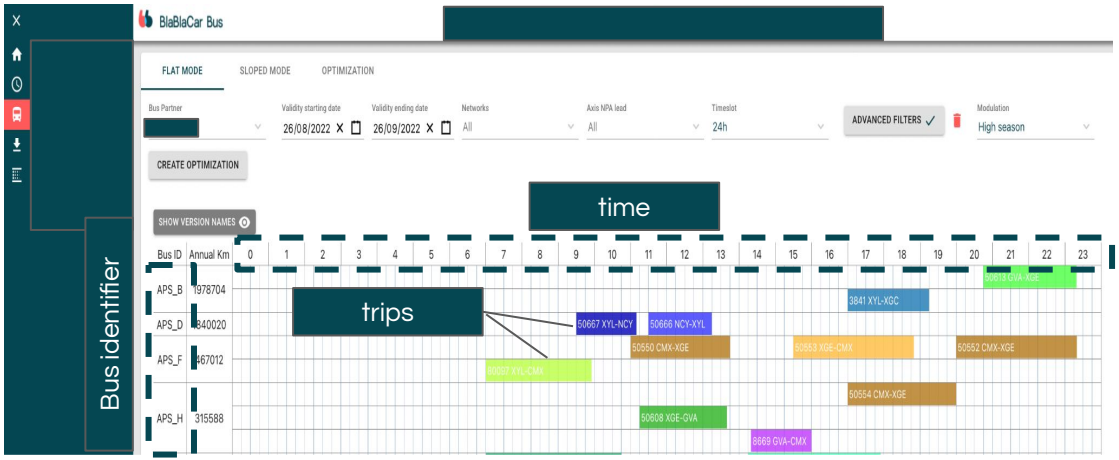
01

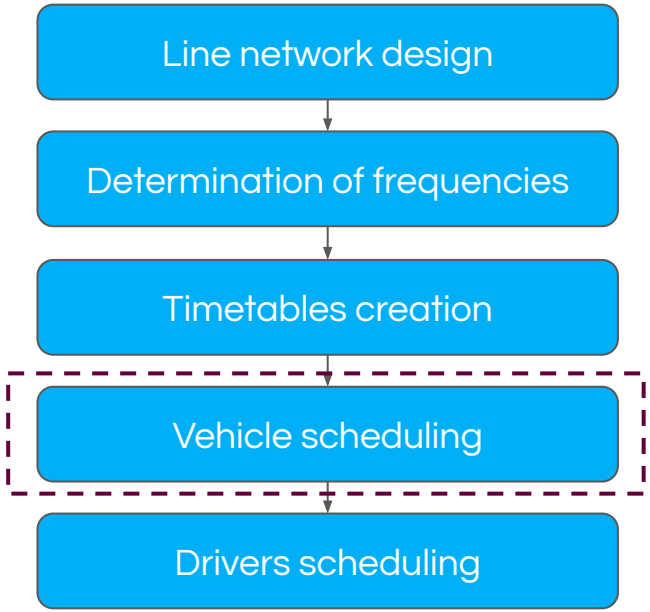
Problem scoping

Bus transportation planning is usually splitted in 5 steps



We focused on Vehicle scheduling





Objective:
Find a matching between trips and vehicles, minimizing the resources needed

Two criteria made Vehicle scheduling an ideal candidate project

Limited optim universe

At this stage, many parameters are fixed (lines, frequencies), limiting free parameters

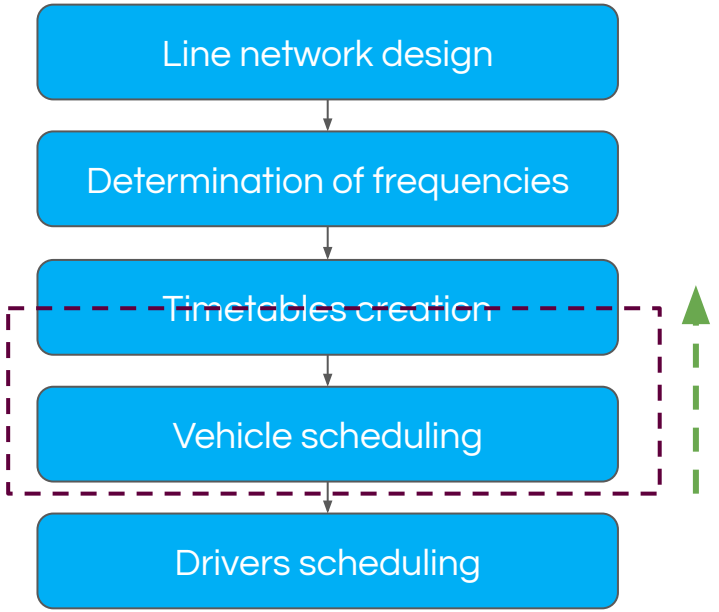
Easily quantifiable gains

We minimize the number of vehicles needed to operate a set of trips. Gains are “number of buses saved”.

Observation of business experts' optimizations led us to extend the initial scope of the algorithm

Network planners are open to modify the departure time of a trip

Hence we decided to develop an optimization algorithm that **both attributes trips to vehicles and modify departure times** to unlock higher savings potential.



02

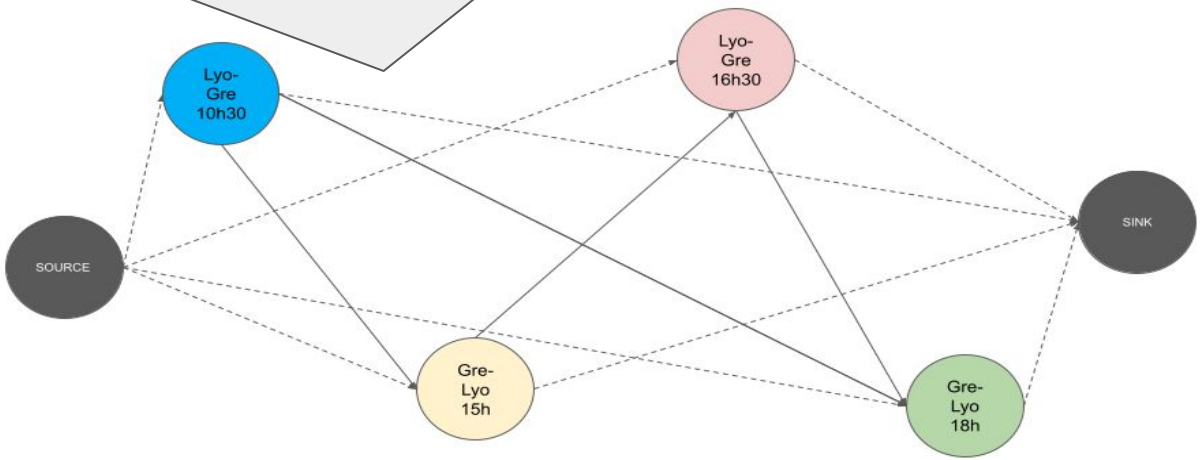
Insights on the modeling approach

We modeled the problem with a graph

Let's take as an example a typical day on the axis Lyon-Grenoble.

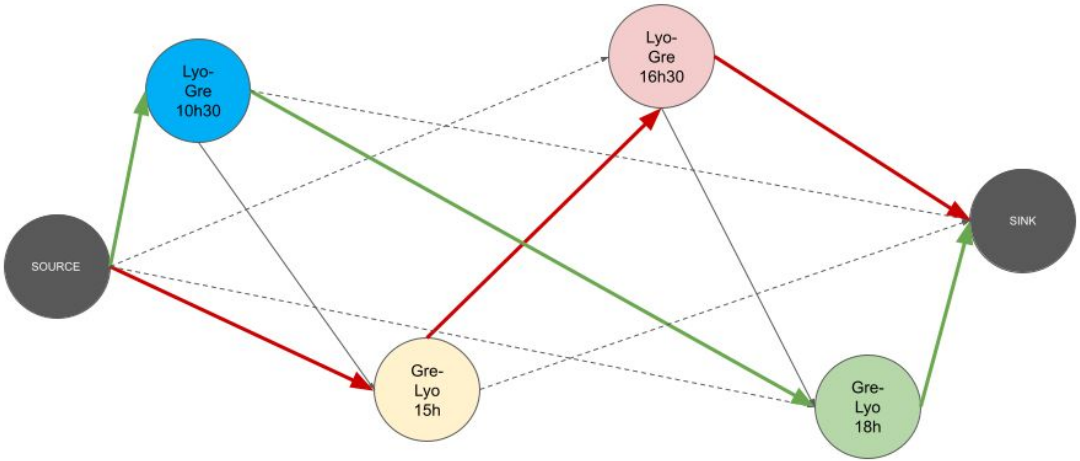
- **Vertices** are the **trips** to be done
- There is a **directed edge** between trip A and trip B if **a vehicle can do trip B immediately after trip A**
- The **decision variables** in the MIP formulation **are the edges**

Although it is feasible for a bus to do Lyo-Gre at 10h30 and then Lyo-Gre at 16h30**, this requires the bus to run empty from Grenoble to Lyon between the two trips, which is money loss and **so prohibited from a business point of view. There is therefore no edges between these two services.



A vehicle schedule is a subset of edges

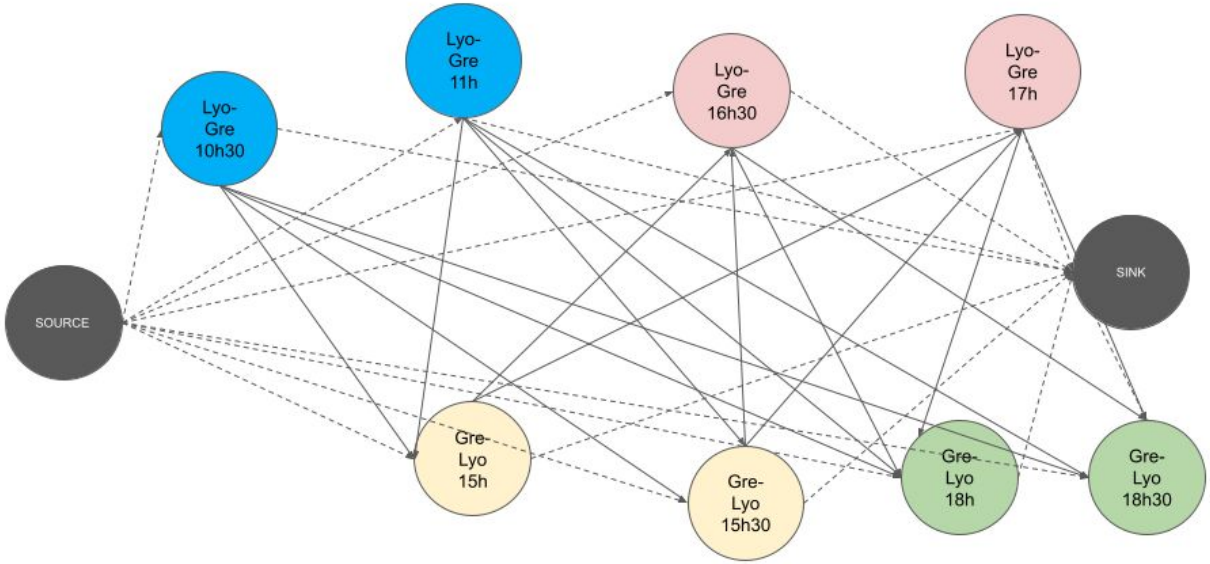
We see that a **vehicle schedule is a set of edges** such that **each vertex except the source and the sink have exactly one entering edge and one leaving edge (condition 1)**



- **The set of selected colored edges on the graph verify the above condition**
- **It defines a vehicle schedule**: for example vehicle red will do Gre-Lyo at 15h and Lyo-Gre at 16h30
- The schedule that minimizes the number of vehicles needed can be represented by **the set of edges with the minimal number of edges leaving the SOURCE** among all set verifying condition 1

Zoom on trips departure times flexibility feature

We **duplicate trips nodes** to model the different possible departure times, and add the MIP constraint “exactly one option is visited in the selected set of edges”



This feature **increases greatly the size of the MIP formulation**: the number of edges is multiplied by ~ $\mathbf{n}^2$ where **n** is the number of options for the departure time.
This was **adapted given our business constraints** as we want to limit options for departures times (typically +/- 30 mins and +/- 1 hour)

Takeaways on the graph modeling approach

\+ Many **business constraints can be translated into preprocessing on the graph edges,** which is often faster than using a MIP constraint.

\+ A nice consequence of the above is that it **allowed DS with no previous optimization experience and SWE to quickly contribute** to the core logic of the algo

\- Few constraints are harder to express in the framework compared to having decision variables **x_{bt}** for bus **b** doing trip **t**.

03

How we make it real for end users

2 North stars since the beginning of the project

1. Make a tool that is trusted and used by Ops teams

2. Being able to move smoothly from the POC to industrialization phase

2 North stars since the beginning of the project

1 Make a tool that is trusted and used by Ops teams

Do everything to allow your future users to be really involved in the development

- **Don't neglect visualization.** In our case, Network planners already had a tool to visualize vehicle schedules. We replicated the main features of the visualization with Plotly to facilitate discussions. It even helped to identify improvements on the original visualization.

- **Be flexible**. During the course of the project, stakeholders might come with use cases for which they want to run the algo. Even if they do not correspond to the exact definition of the problem, it might be worth adapting the algo to address them and create value. That's what helped us unlock the budget for industrialization.

2 North stars since the beginning of the project

2

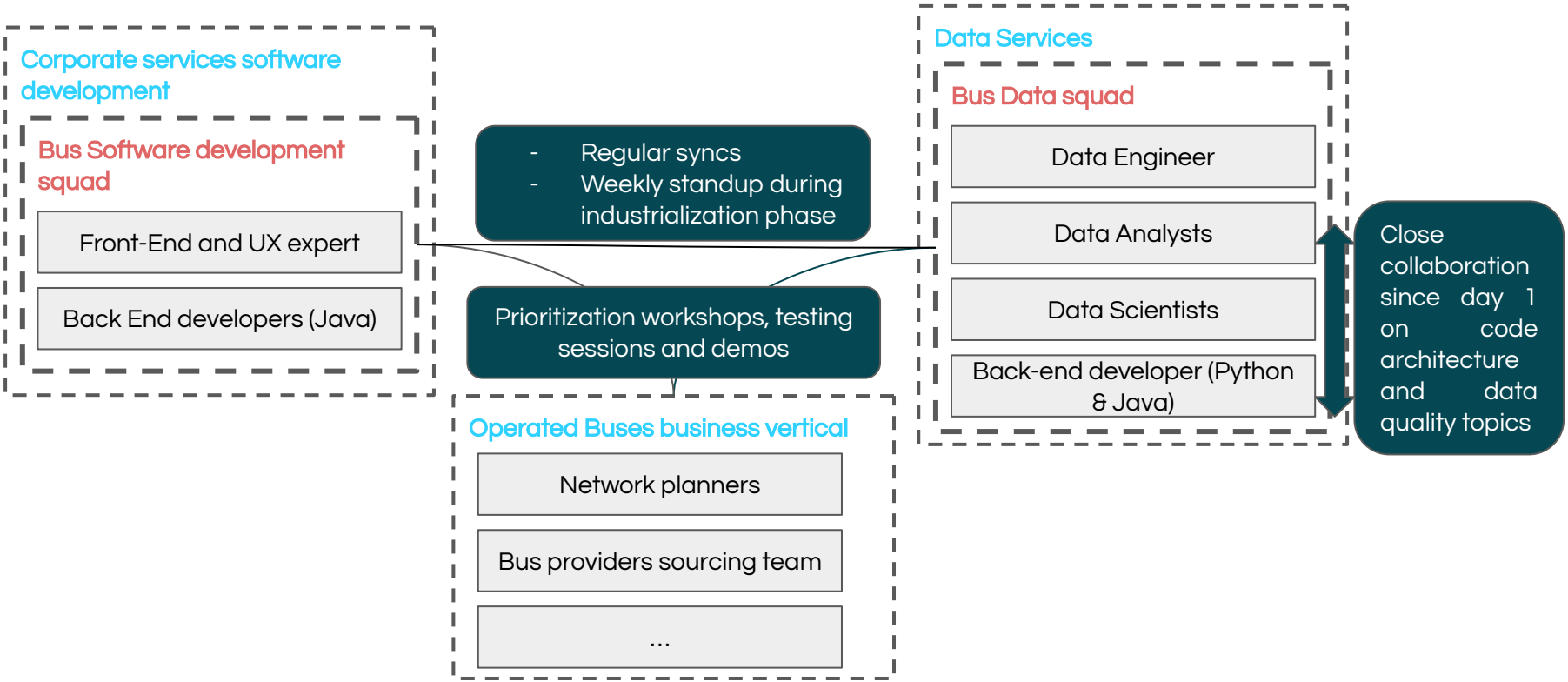
Being able to move smoothly from the POC to industrialization phase

Build a fruitful DS <> SWE collaboration from the dev day 1

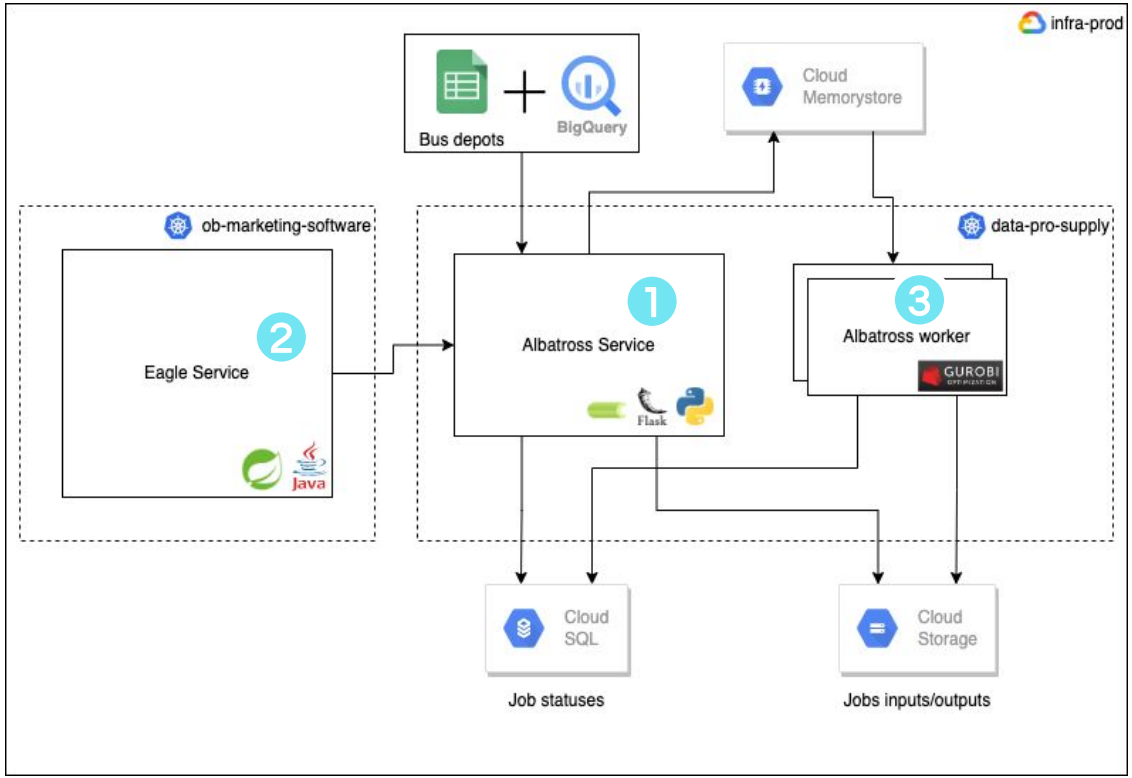
- **Make the life of the SWE easy** by producing structured, versioned code
- Use Pull Requests and informal discussions to **ask for feedback** on code quality and architecture. Industrialization objective is not to rewrite a clean codebase
- Inversely, **identify core features of the model that SWE can take** (such as the graph construction in our case). It will accelerate the dev of the POC and SWE will gain a deeper understanding of the algo, that will later accelerate the industrialization.

To conclude: what does the industrialized architecture of the tool look like?

Organisation of tech resources by business vertical enabled industrialization



Tool architecture follows SOA paradigm



1. The optimization algo (Albatross) has been packaged in a **Flask** app. Given that we have a single-use license and that some runs may take hours, we use **Celery** together with **Redis** to implement a queuing system.

2. The optimization runs can be launched through a **simple button integrating on one page the main Network planning platform** . The platform communicates with Albatross through a **REST API.**

3. BlaBlaCar infra being fully containerized, we use **Gurobi Web License Service** to tackle the optimization jobs.

Questions

“Thanks!