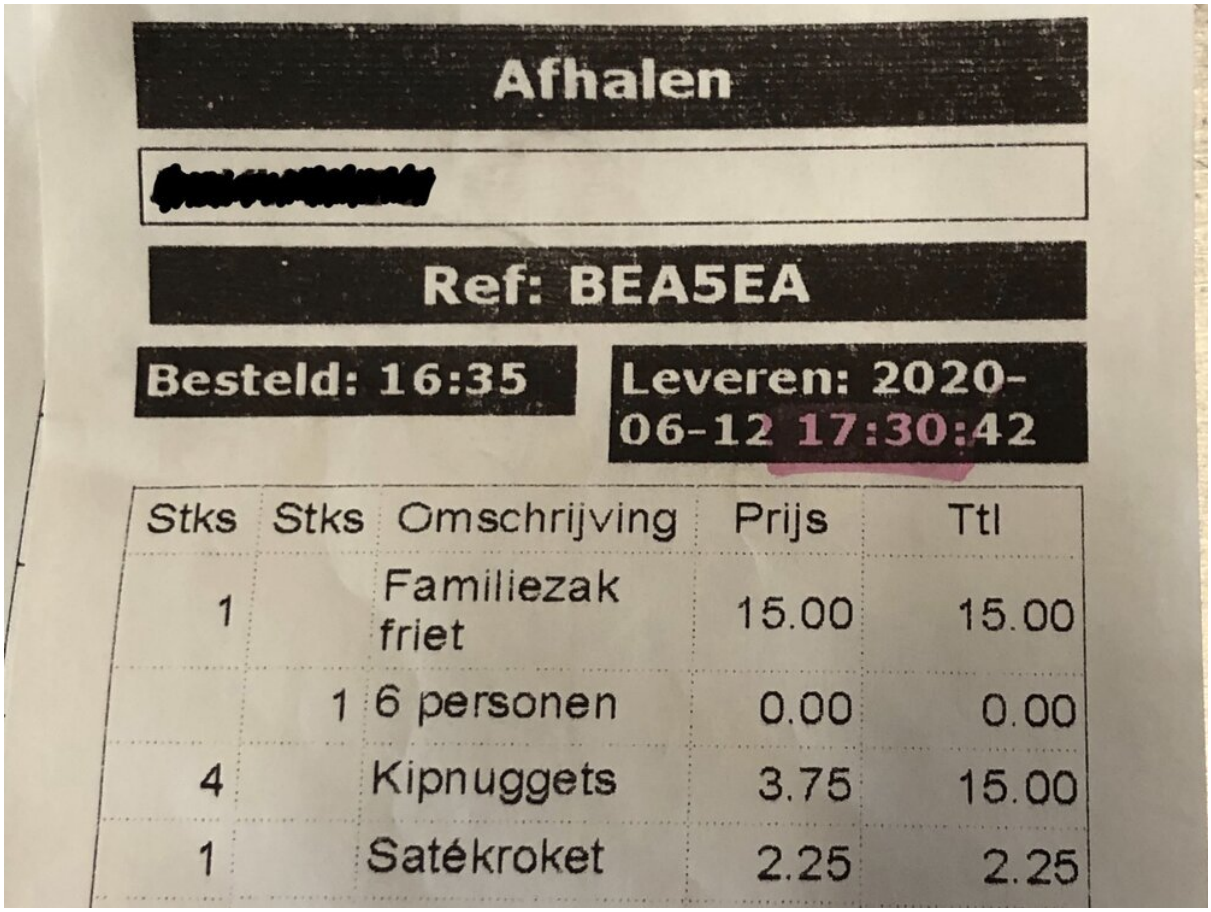
Linking Lemmid Store with kitchens

Integrating with external systems that you don't control

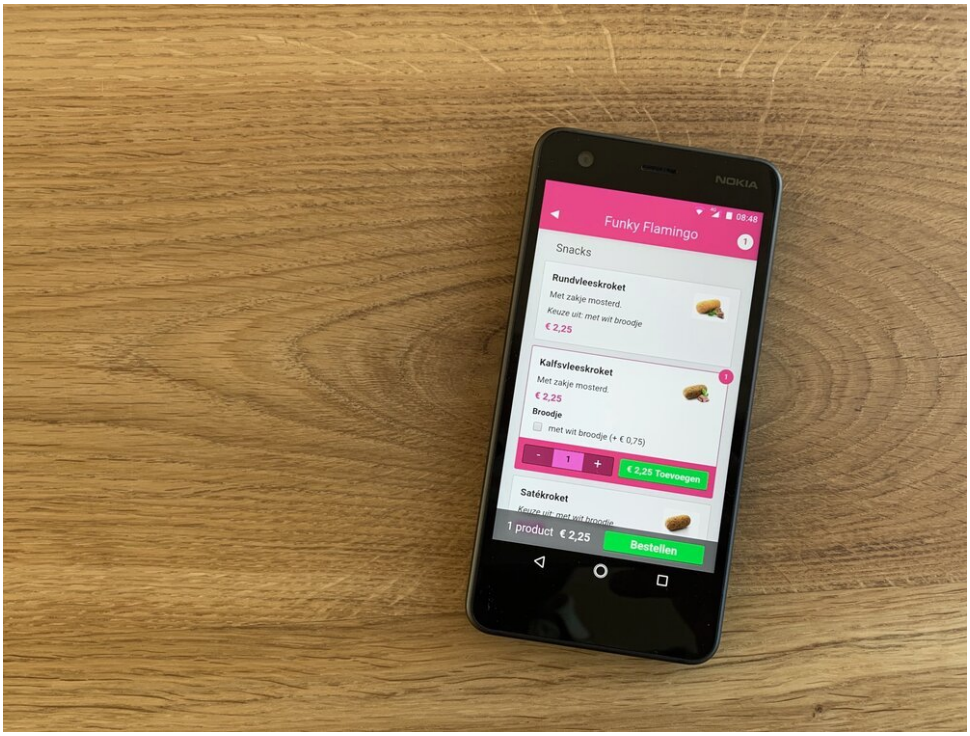
Willem L. Middelkoop
June 12, 2020



As part of the food ordering app I am building, I needed to design a reliable way to link the app to external systems. These external systems are beyond my direct control and include different checkout registers, kitchen management systems and ticket printers. Read along for more on designing for the unknown and unreliable.

Food ordering app

The app I am building is Lemmid Store, it is a user friendly app that allows people to place (and pay for) takeaway food orders. You can find all posts on Lemmid Store here.

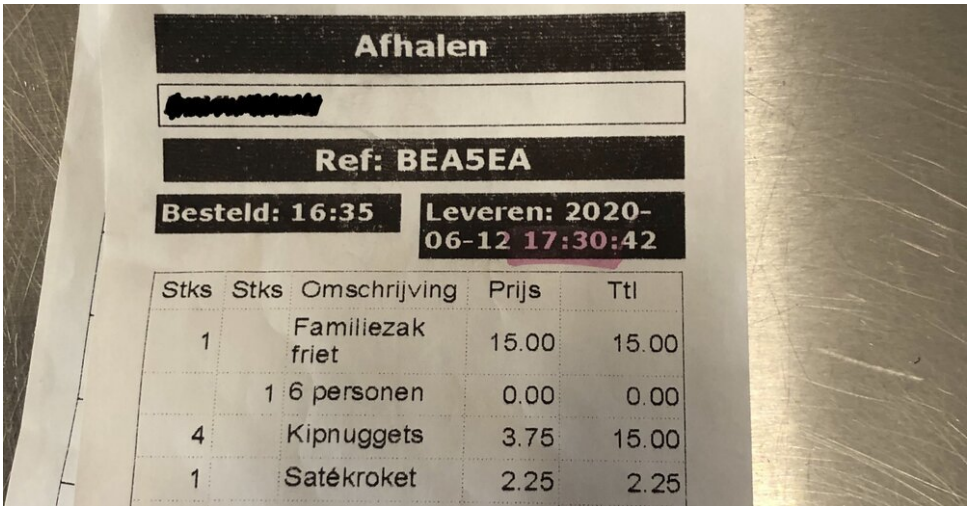


Lemmid Store - an app to order takeaway food

External systems

The need to connect to external systems is driven by the desire to integrate the food ordering app into the business processes related to the food delivery service. Think of things like:

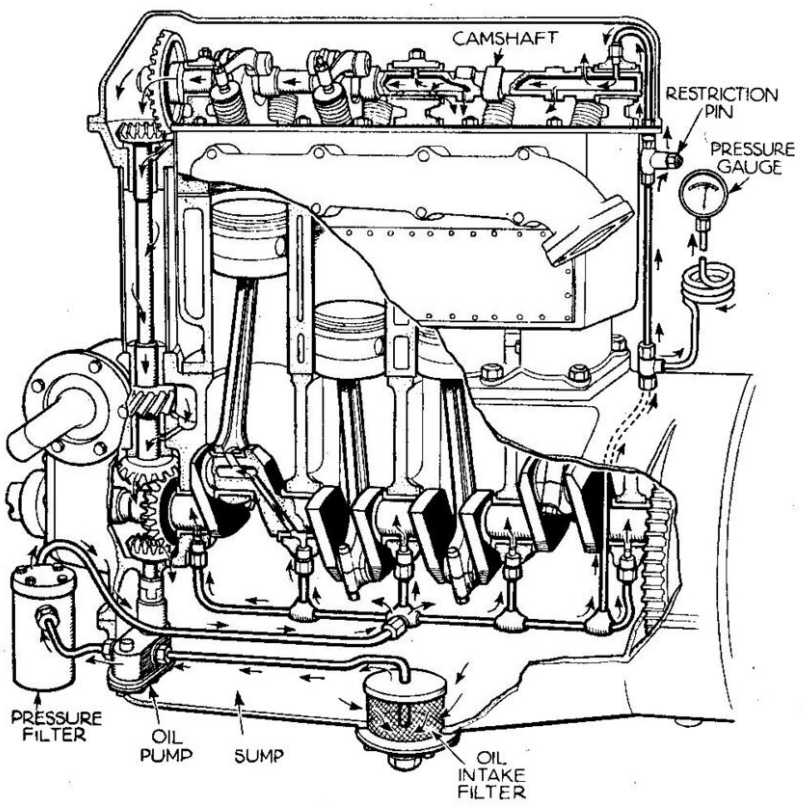
- collecting order data into administrative records (e.g. bookkeeping)
- forwarding orders to the kitchens, where the meals are prepared and packed
- providing easy access to routing information, showing where the food should be delivered
- scheduling food orders that are meant for later deliveries



Printed ticket - through an external connection - including detailed order data like product modifiers (customer's choices per product)

Challenges with connecting to external systems

You may not realise it, but connecting to external systems can be tricky. If you think of the ordering app as a kind of ”engine” (like in your car), you essentially want to integrate this engine with external parts. As you can imagine, not all parts are equally suitable to fit under the figurative hood. To make matters even more challenging: you do not know for sure how reliable the external parts are. You must design your integration in such a way that different external parts can be connected easily and that performance is independent on the reliability of these external parts.

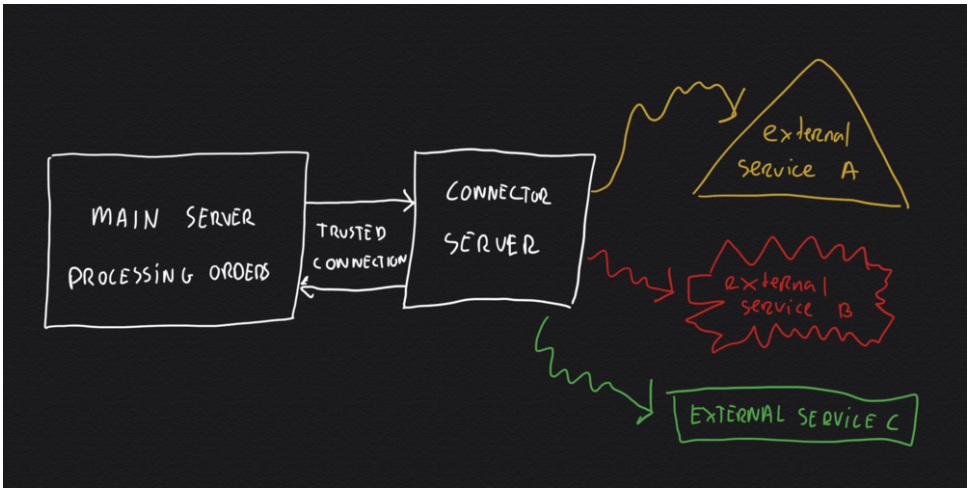


Integrating external systems into your app is like fitting third-party parts to an engine: you must carefully consider their fit and reliability (image of a combustion engine, public domain)

Like with a combustion engine, you must consider fit and reliability when connecting integrating your software with external systems. **You cannot assume that external software:**

- is a perfect fit
- is always consistent
- is always available / present
- is trustworthy

To design for this rather grim view on external connections, I thought it was a good idea to let the main backend server of my food ordering app offload the responsibility for connecting with external systems to a purpose built 'connector server'. The connector server would then independently connect to the external system. This way the main server only communicates with 'first party' parts and systems, minimising risks of misfits and possible downtimes.



Separating the main backend server from the external systems by introducing a purpose built 'connector server'

In practice this means that the 'connector server' is actually a different machine, running its own software stack. There are a few important advantages to this segmentation:

- **security:** the connector server does not keep a complete record of all orders, only those orders that need to be forwarded to an external system are kept in memory temporarily. A possible security problem in an external system will only expose a minimum amount of data.
- **fits anything:** the connector server can accommodate many different protocols and ways to connect to external systems. Connections to external systems can be added much like the server 'learning a new language to speak'. As it's already a 'multi language affluent', adding a new dialect or language is often easy. Connecting to different external API's is easier when you have all the building blocks in place to (re)structure data.
- **accommodate unreliability:** A simple queueing mechanism in the connector server allows it to elegantly handle possible downtime and problems with external connections. By logging and queueing per external service it becomes possible to minimise impact on other parts and connections when one services goes down.

Conclusion

However tempting it is to simply bolt on new and fancy parts to your app, you must carefully consider your setup to make sure it will perform reliable.

Designing backend server software to connect with external API's is challenging. You must hope for the best, but design for the worst!