Challenges encountered.

front-end:

Construction of the front-end framework:

Public Components Development:

Page resolution:

Back-end:

Back-end to front-end presentation of api.

Once the back-end developer has developed the interface, they need to let the front-end developer know the name of the interface. At first we tried to get the front-end developer to read the beginning of the back-end code to understand what parameters the back-end needed to fetch, but this was not efficient because the format of the fetching section of the back-end code was not entirely consistent due to the different development habits of different developers. These formatting inconsistencies can take some time for the developer to explain.

This problem has been solved by using the team feature of postman, where we create a postman team workspace and when a back-end developer finishes a feature, he first tests it using postman and the saved test results can be seen by the team members in the same group. So, by using this method, the team solves the job of testing the api and presenting the interface at the same time.

Databases:

Synchronising databases.

In the early stages of development, the structure of the database is frequently changed. This makes it necessary to synchronise everyone's database structure and data. The first thing we thought of was whether we could synchronise the database via git in the same way as the code, but we soon realised that this would not work, firstly because it was difficult to change the location of the data, which was stored in a single folder. Secondly, the files are not split by table or by library, so synchronising all the database files is not a good option.

To solve this problem we went on to look at a number of sources. We found that we could use a unified database server, but this seemed to cost some money and setting up such a database would take some time.

In the end we opted to solve this problem by manually exporting and importing sql files, and we used a uniform naming convention to sort the exported database versions (date plus version). Developers can check if the latest database version is available after syncing the code repository with git, and if so, allow navicat to import it. At the moment this method still requires some extra effort and can be fixed later with some automatic detection scripts.