Pieter-Jan Vermunicht

Technical Manual

Situation

We currently have 3 databases systems from our client: GLPI, Centreon and OCS. These are all SQL databases systems that contain a lot of information about the same pool of computers. Our task has two parts: part 1 is to unify this database information and make it available in a user-friendly way. The second part is the integration with an AI and to provide management platform for the tickets the AI generates.

Database

The choice of which database system to use was based on the client database. Because this one is a SQL database it which easiest to also use the same scheme to build the combined database. This will make it easier in the future to combine data.

MariaDB (https://mariadb.org/)

For the software itself there exist many SQL server applications but I decided to use MariaDB. This is an opensource interpretation of SQL that can run on almost all platforms. It is also performant and can handle the large databases we were handed.

At this moment we have all 3 client databases deployed on a MariaDB server together with the combined database. The combined database contains all relevant information for user-accounts and permissions and a lookup-list of all computers.

Combining databases

Our main goal is to link all 3 database systems we were given by our client. Based on my research one way to link all 3 together could be based on the name of each computer, this name is generally the same in GLPI, Centreon and Ocs. With this table we combine the identifier each computer has in their respective databases so we can easily fetch information about a computer from all three databases.

Afterwards we will focus on moving the data from GLPI, Centreon and OCS to the combined database so we have all information available in one place.

Back-End

Django Rest Framework (https://www.django-rest-framework.org/)

We decided to use Django Rest Framework as the basis for our database API. Versatile and fast back-end framework that can easily be adapter to our current database. Python programming is also widely used and can run on many platforms. This forms the interpretation layer between the database and the front-end.

The back end connects to the database and can access all information from the client databases. This data is formatted and paginated in API calls to aid the front-end.

Components:

- drf_yasg: This documentation tool automates generation of Swagger and OpenAPI. Saves us from the hassle of having to set-up OpenAPI and swagger manually.

- Pagination: easily divides queries into pages so the front-end can fetch data selectively.

Front-End

I mainly focused on the back-end database, so I did not have a big role in deciding which technologies to use here.

Angular (https://angular.io/)

The front-end is responsible for visualizing the database and formatting the data to be visually pleasing. We chose to continue using Angular.

Machines

This fetches all computers from the combined database and displays them as a list. Each of these entries has a details page which at this get's the information from GLPI, OCS and Centreon separately. This consists of the hardware specs of the machine, all tickets linked to the computer and a graph of the metric data.