APERTIF Task Database

A microservices architecture with Django and Python

Nico Vermaas, Vanessa Moss, Roy de Goei.

www.astron.nl

Workflow

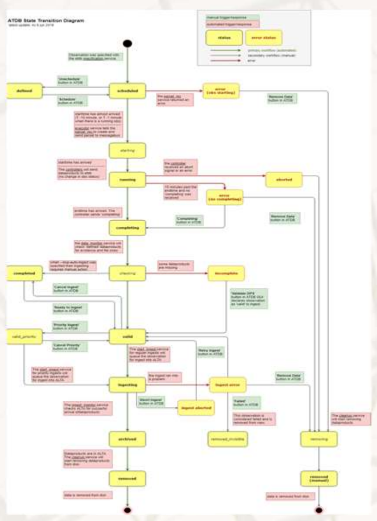
STATE MACHINE

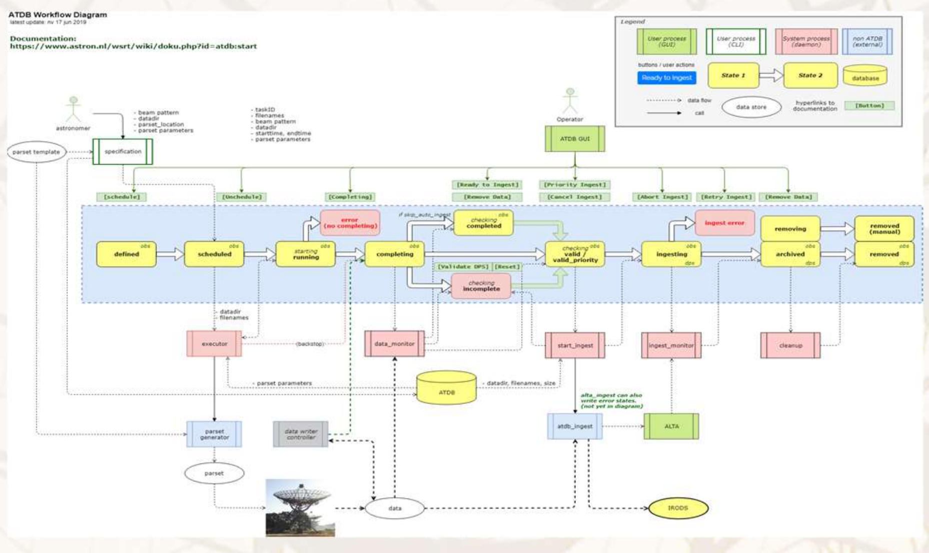
Every observation (task) always has a specific status, which is kept in the central ATDB database on the 'server' side.

A cluster of specialized services on the 'client' side 'listen' only for the status that they are interested in, execute their task accordingly, and set the observation to another status.

Ideally the work flows automatically from initial 'specification' by astronomers all the way to removing the data after successful ingest in ALTA. (bottom diagram).

The (right) state transition diagram shows all possible states, causes, effects and paths through ATDB.

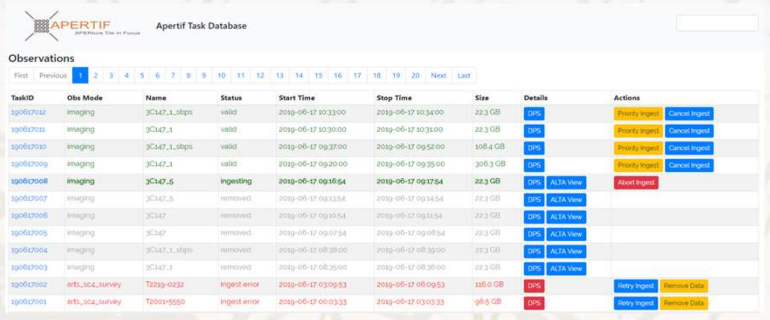




User Experience

ATDB GUI

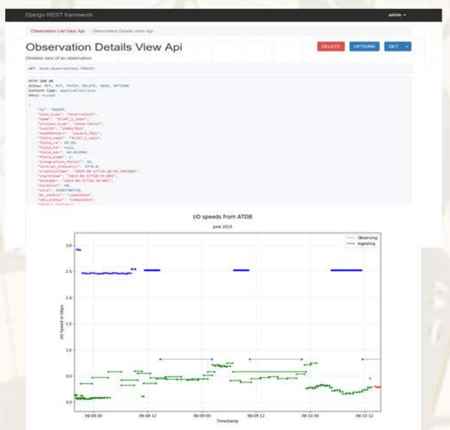
The main GUI is created with **Django's** html templates, served by the **Nginx** & **Gunicorn** backend webserver and styled with **bootstrap**.It gives feedback over the (mosty automated) workflow and offers some basic control.

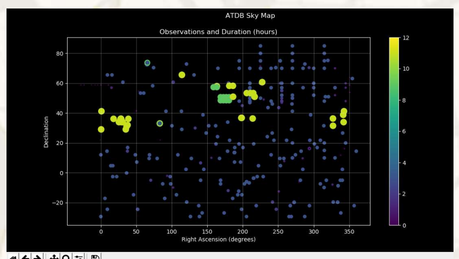


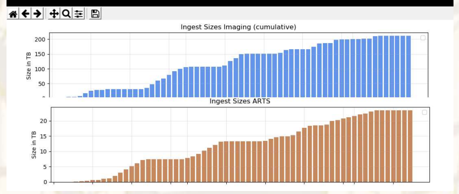
REST API

Generated by **Django Rest Framework**, which together with **Django-filters**, offers a fully queryable REST API. It can be accessed from a browser, but also with regular http requests like GET, POST, PUT, DELETE from the commandline or programs.

All the **atdb services** access this REST API from within Python, just like several spin off monitoring and reporting tools.





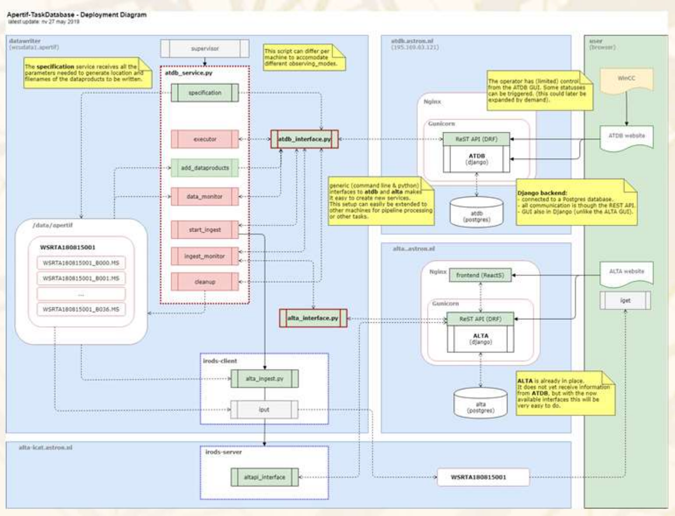


Infrastructure

Clients & Servers

The central column shows the servers for **ATDB** (Apertif Task Database) and its sister project **ALTA** (Apertif Long Term Archive). Both these systems live on separated (virtual) machines that do nothing more than hosting the **Postgress** database and running the backend applications in their **Gunicorn** and **Nginx** webservers.

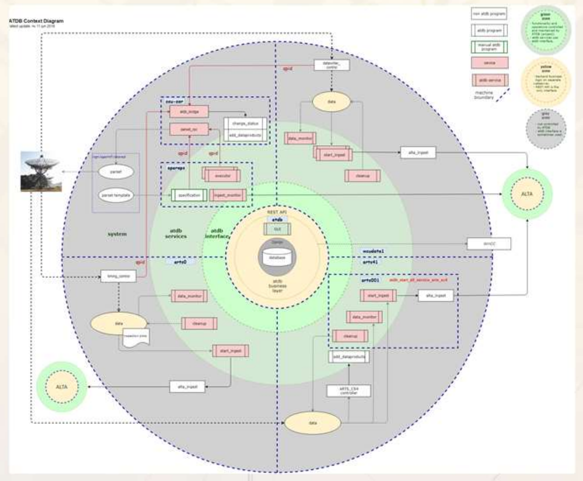
All communication with the backends is done through an 'interface layer' that wraps the http calls to the REST API into higher level Python functions that the **atdb services** (or any other Python program) can use.



Context

In practice the **atdb_services** are spread out over different machines depending on their specific tasks. Some are centralized and centrally maintained. Others are installed near their specific data and near the grey horizon there are user maintained scripts.

The ATDB backend is the spider in the web of information, safely separated from a more dynamic client environment.



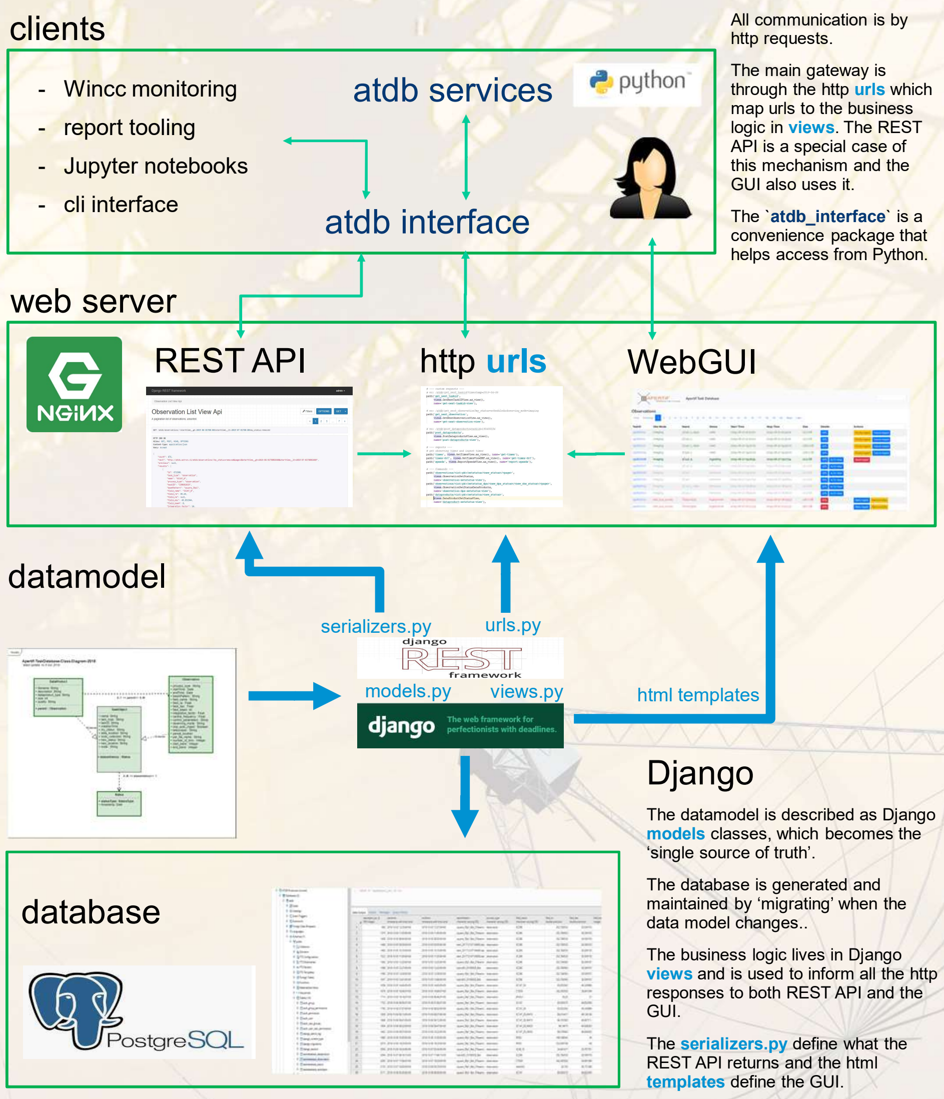
Architecture

Interfaces

All communication is by http requests.

The main gateway is through the http **urls** which map urls to the business logic in **views**. The REST API is a special case of this mechanism and the GUI also uses it.

The `**atdb_interface**` is a convenience package that helps access from Python.



Django

The datamodel is described as Django **models** classes, which becomes the 'single source of truth'.

The database is generated and maintained by 'migrating' when the data model changes..

The business logic lives in Django **views** and is used to inform all the http responses to both REST API and the GUI.

The **serializers.py** define what the REST API returns and the html **templates** define the GUI.

ASTRON is part of the Netherlands Organisation for Scientific Research

ASTRON Netherlands Institute for Radio Astronomy

NWO Netherlands Organisation for Scientific Research