Iteration 4: Design

Jeff Caldwell edited this page 7 minutes ago · 37 revisions

Barrios Technology - Forecasting and Prediction Modeling

Team Noname

Jonathan Morgan

Carlos Cardenas

Josue Lozano

Jeremy Miles

Jeff Caldwell

1. Introduction

i. Purpose of the system

The ISS analysis tool is a web application that provides analysts at Barrios, an aerospace company, with logistics optimization, usage forecasting, and prediction accuracy analysis surrounding consumable delivery to the International Space Station (ISS).

ii. Design goals

1. Provide a clean, usable, intuitive interface for Barrios employees to upload user-generated data and view analysis results.
2. Provide accurate and useful analyses, including supply-chain optimization, predictive modeling, and prediction accuracy analysis.

iii. Definitions, acronyms, and abbreviations

- **ISS**: International Space Station

- **IMS**: Inventory Management System - Barrios' ISS inventory management datastore
- **ORM**: Object Relational Mapper - A framework-specific set of classes and methods for interacting with a database
- **ASGI**: Asynchronous Server Gateway Interface - An asynchronous calling convention for Python web servers
- **DOM**: The *Document Object Model* - A data structure that represents an `HTML` document.

iv. References

- Emmett: A "full-stack" web framework written in Python
- HTMX: A client-side JavaScript library that enhances HTML for easy communication with the server
- Hypermedia APIs vs. Data APIs
- PostgreSQL: A relational SQL database
- Pandas: A Python library for exploring and manipulating tabular data
- DuckDB: An analytical database system that allows developers to run SQL queries directly on flat CSV files
- Prophet: A forecasting algorithm

v. Overview

The ISS analysis tool will give Barrios analysts the ability to quickly transform user-provided data into three types of analysis: prior prediction accuracy assessment, logistics optimization, and predictive modeling.

2. Current software architecture

The current architecture is based on a `Model-View-Controller` architecture. HTTP requests to the server will be handled by `Controllers`, which will coordinate with `Models` to generate or persist analyses based on how the user interacts with `Views`. In its current state, the system handles basic file uploads and the persistence of user-provided data to

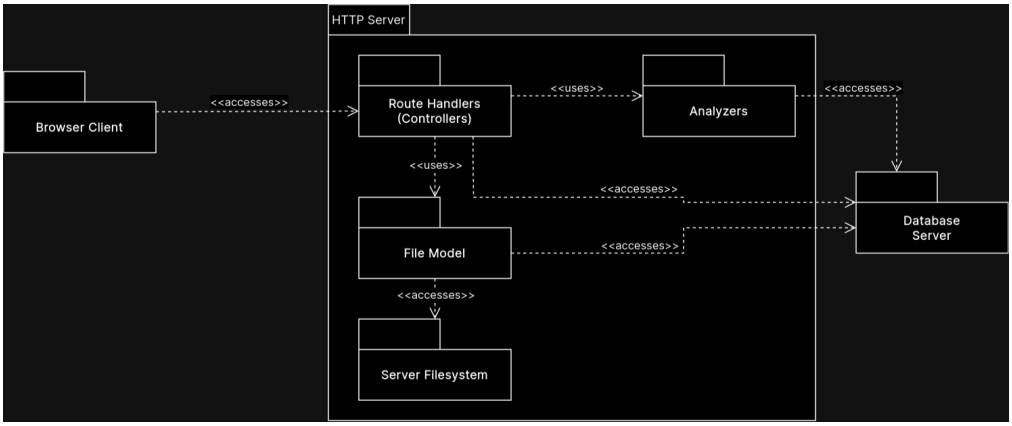
Clone this wiki locally

3. Proposed software architecture

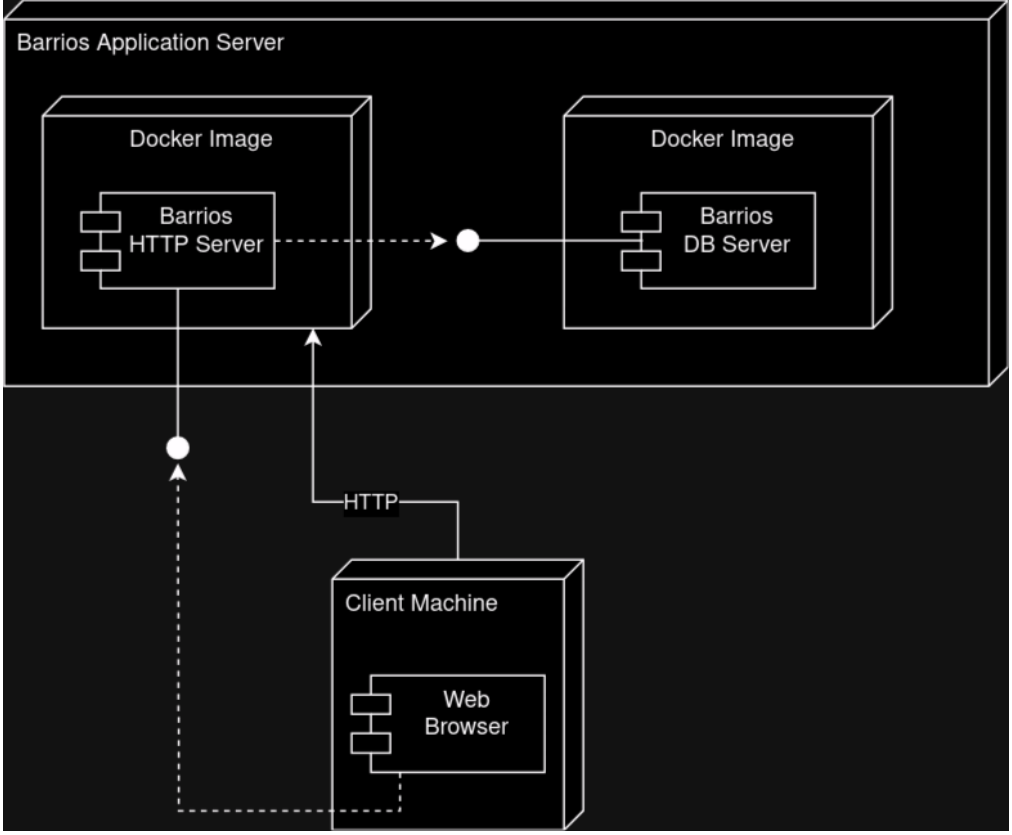
The proposed system has not changed since it was proposed in Iteration 3, with the exception that the proposed system might use DuckDB for performant, on-the-fly analysis of user-provided csv files. The system will also use an HTTP server framework named Emmett to facilitate a basic Model-View-Controller application structure. With this structure, controllers will respond to HTTP requests sent from a browser-based HTML interface augmented with the HTMX library and coordinate the creation of optimization analysis and predictive modeling using either the Pandas data library or DuckDB and the Prophet algorithm. The system will also facilitate uploading and storing client-generated data using PostgreSQL.

i. Overview

ii. Subsystem decomposition



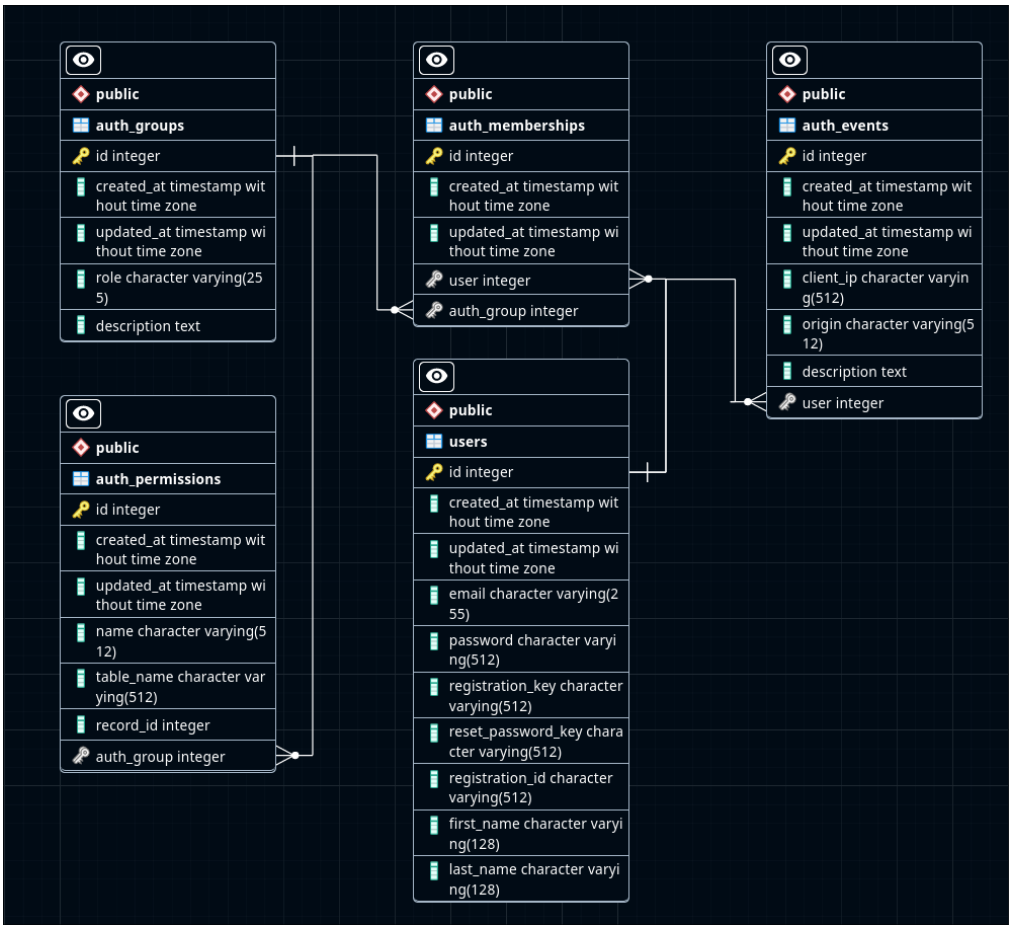
iii. Hardware/software mapping



iv. Persistent data management

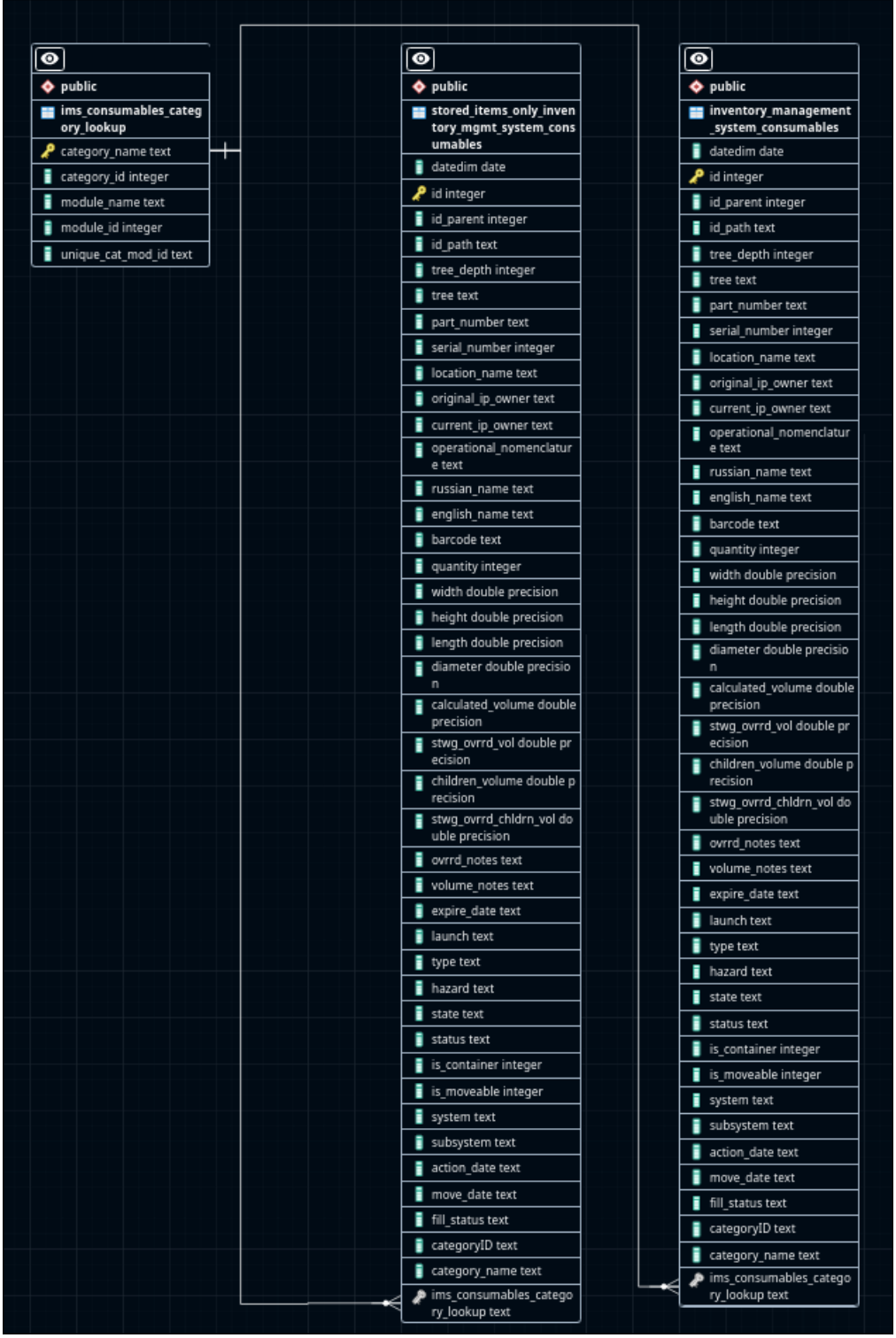
Entity Relation Diagrams

Auth & User

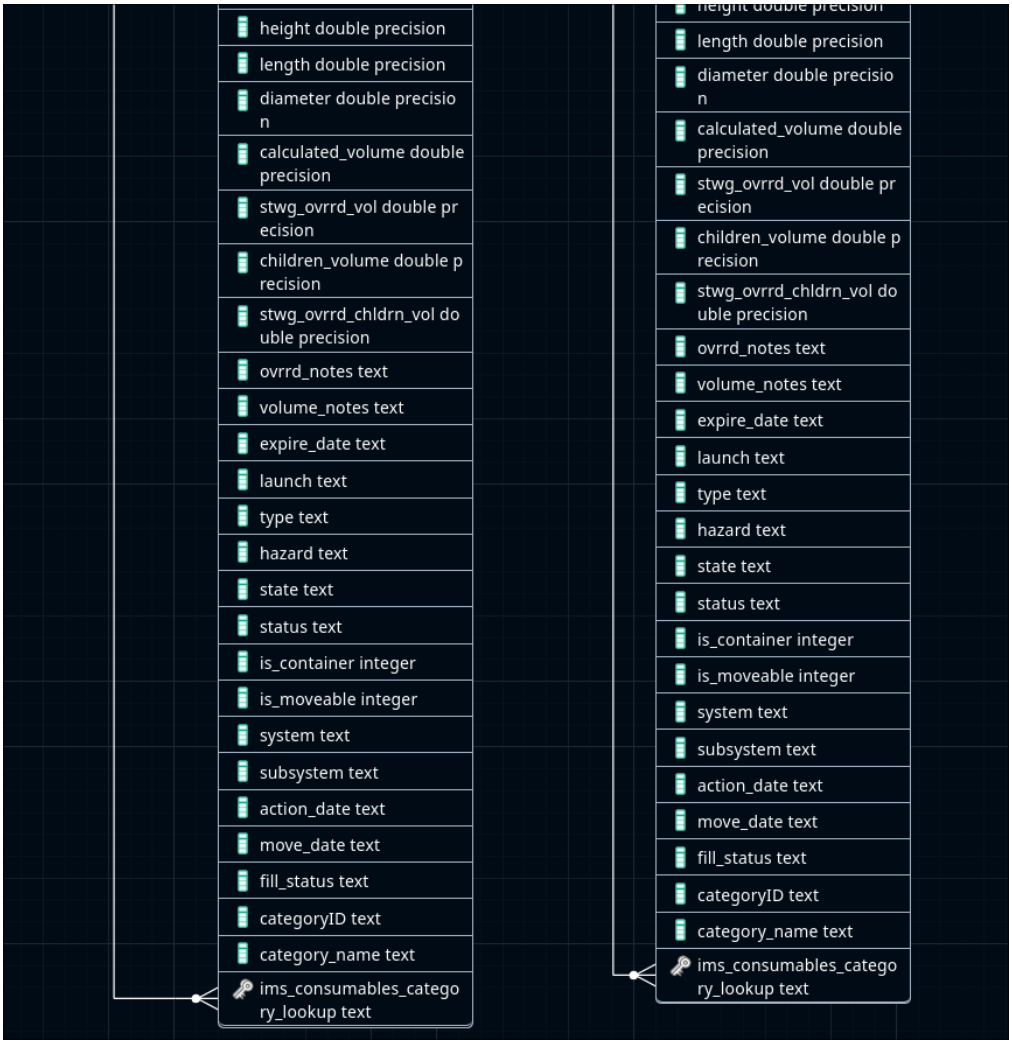


Flight Plans

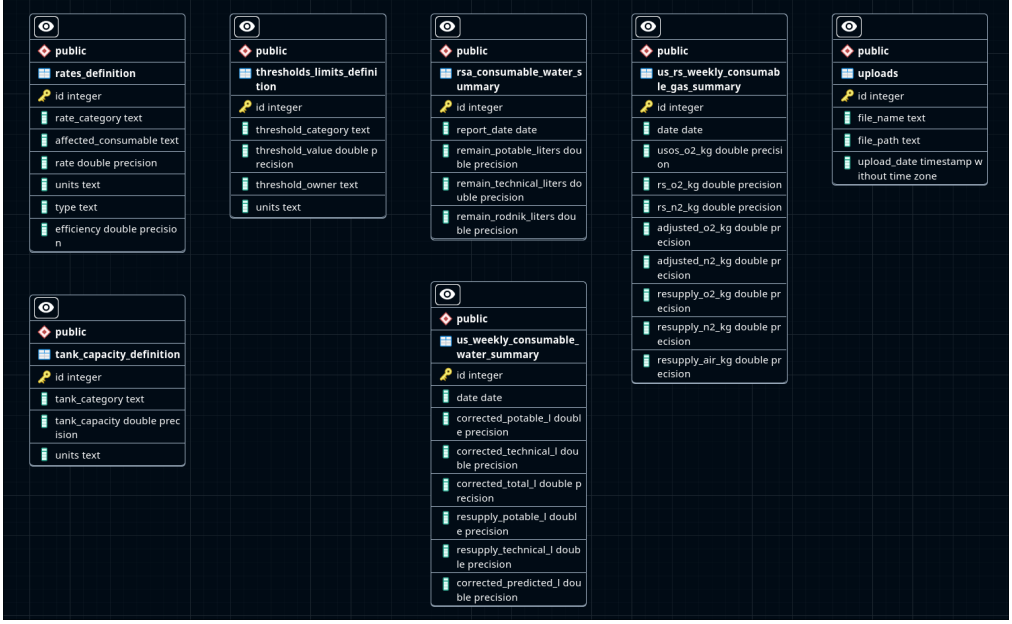
Inventory Management System Overview



Inventory Management Detail



Other Tables



v. Access control and security

Access matrix

Role	Analyses	View Files	Upload Data
Viewer	Read		
Analyst	Read/Write/Execute	Read	Read/Write/Execute
Admin	Read/Write/Execute	Read	Read/Write/Execute

vi. Global software control

Global software control for this system will be handled with a server-client architecture implemented using the Emmett web framework. This will allow the server to respond to HTTP requests from a browser with HTML templates. These templates will be augmented using the HTMX library.

Requests from the browser will be handled asynchronously by the server, allowing the system to respond to a larger number of requests than a synchronous server. This is made possible because the Emmett framework follows the `ASGI`, or Asynchronous Server Gateway Interface convention. This convention allows web server software to respond to numerous concurrent requests without the need to enlist additional processes, which will save expensive computing resources.

Synchronization between the browser and server will be handled like traditional HTTP servers. Rather than issuing responses using an intermediate language like `JSON`, server endpoints will respond with HTML. This will be possible by using HTMX, a small JavaScript library that makes asynchronous calls to a web server and replaces appropriate sections of the `DOM` with the server's response. This removes the need for a client-side JavaScript framework and reduces the overall complexity of the application by ensuring all application state remains on the server.

vii. Boundary conditions

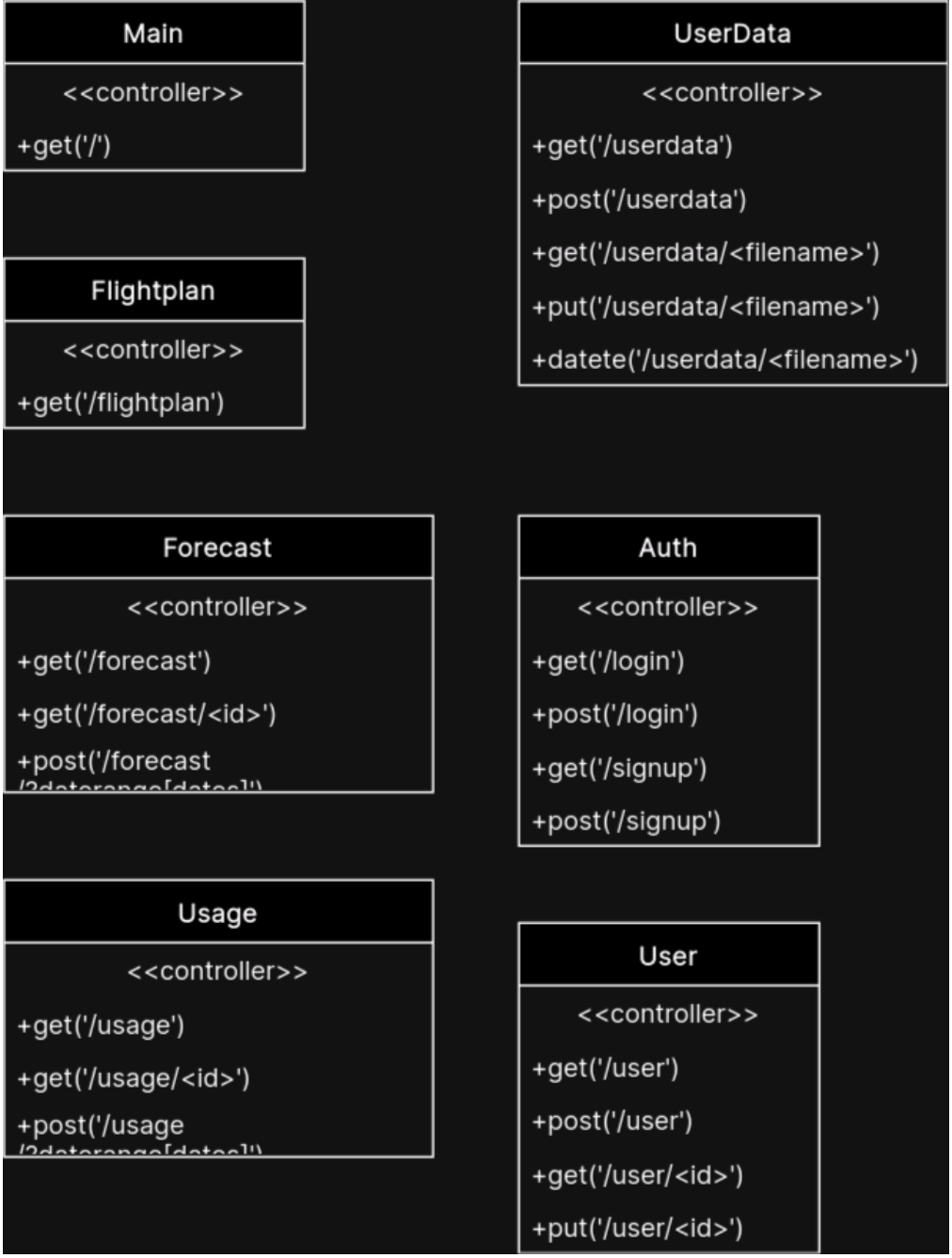
Startup The server software will run on what is essentially an infinite loop, waiting for HTTP requests. The true system, however, will start up when a client makes a request. If that request is targeted at a valid route, the request will be processed by a controller. If not, the system will return a 404 error.

Errors All exceptions will be collected in an error log for additional analysis by developers. If the exception is severe enough that it fails to process a client request properly, it will be returned to the client as an error message with an HTTP status code.

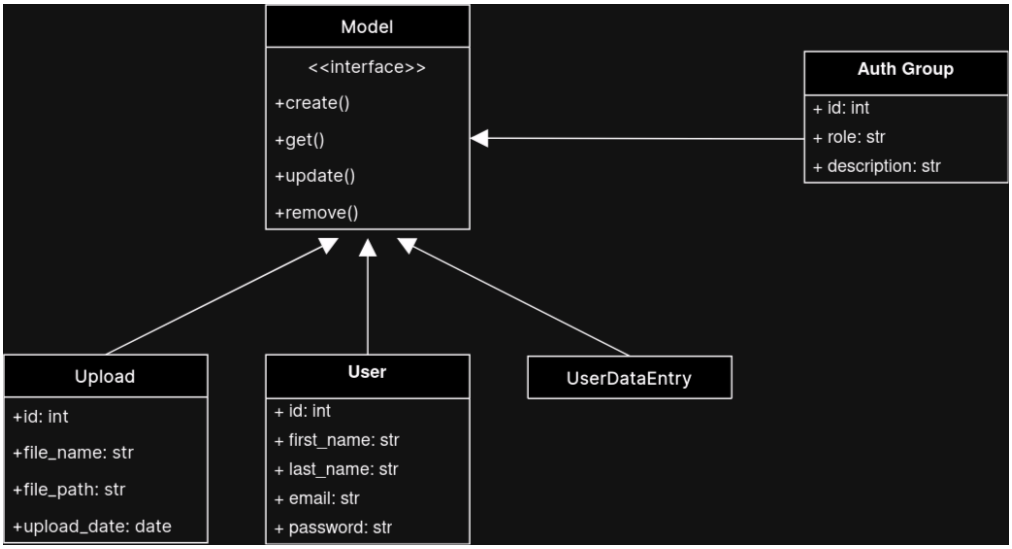
Shut down When a client ends a session, the system will ensure all active database connections used in client requests are closed and that any ongoing transactions are committed to the database.

4. Subsystem services

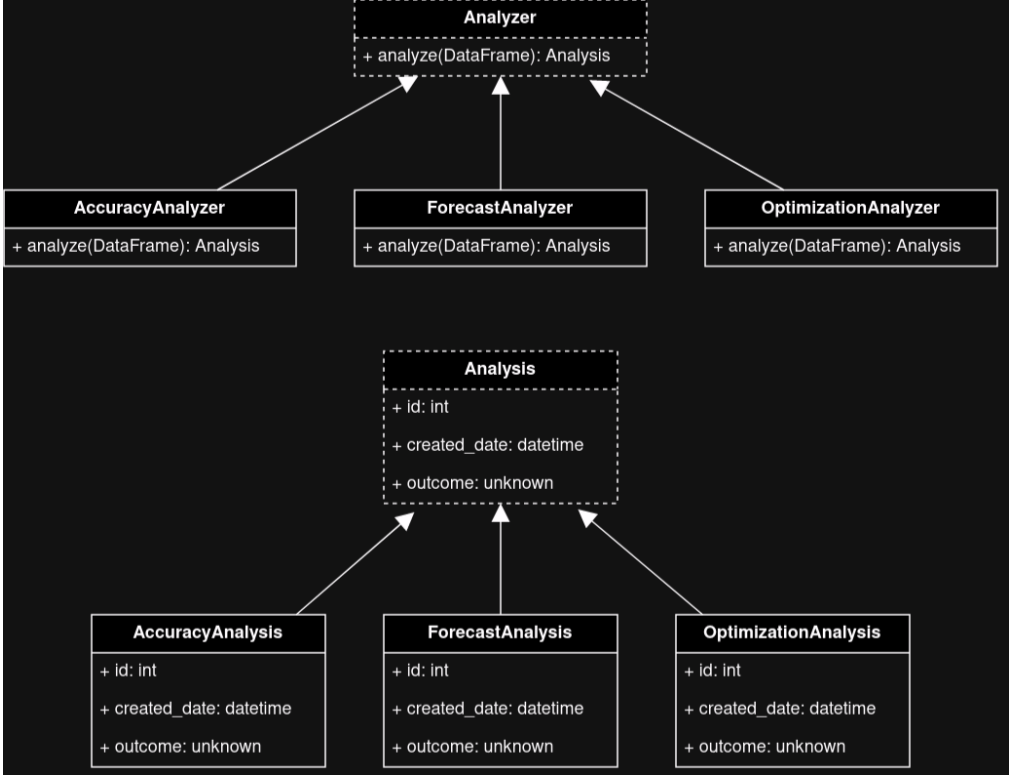
Controllers



Models



Analyzers and Analysis



Glossary

ISS: International Space Station

IMS: Inventory Management System - Barrios' ISS inventory management datastore

ORM: Object Relational Mapper - A framework-specific set of classes and methods for interacting with a database

ASGI: Asynchronous Server Gateway Interface - An asynchronous calling convention for Python web servers

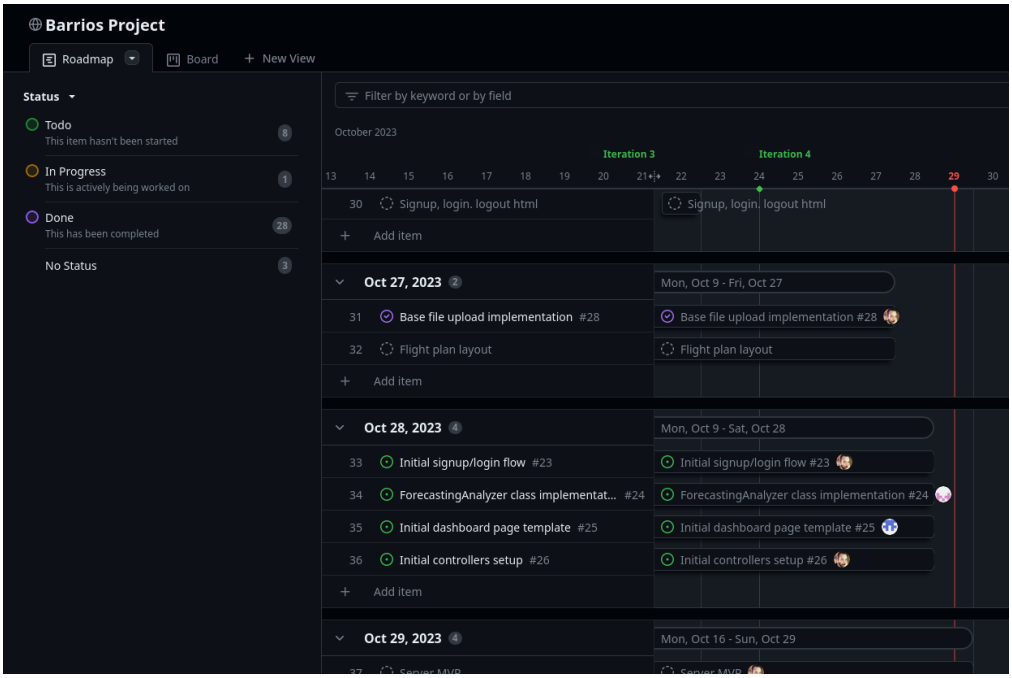
HTTP: Hypertext Transfer Protocol - The messaging and networking protocol used to request and deliver hypermedia on the web

HTML: Hypertext Markup Language - A language used to define hypertext documents for use by a web browser

DOM: The Document Object Model - A data structure representing an HTML document and all of the elements contained therein.

PostgreSQL: A popular relational database

Appendix


Barrios Project
Roadmap
Board
New View
Status
Todo
This item hasn't been started
8
In Progress
This is actively being worked on
1
Done
This has been completed
28
No Status
3
Filter by keyword or by field
October 2023
Iteration 3
Iteration 4
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
30 Signup, login. logout html
Signup, login. logout html
Add item
Oct 27, 2023 2
Mon, Oct 9 - Fri, Oct 27
31 Base file upload implementation #28
Base file upload implementation #28
32 Flight plan layout
Flight plan layout
Add item
Oct 28, 2023 4
Mon, Oct 9 - Sat, Oct 28
33 Initial signup/login flow #23
Initial signup/login flow #23
34 ForecastingAnalyzer class implementat... #24
ForecastingAnalyzer class implementation #24
35 Initial dashboard page template #25
Initial dashboard page template #25
36 Initial controllers setup #26
Initial controllers setup #26
Add item
Oct 29, 2023 4
Mon, Oct 16 - Sun, Oct 29