PyKlatchat Whitepaper

Contents

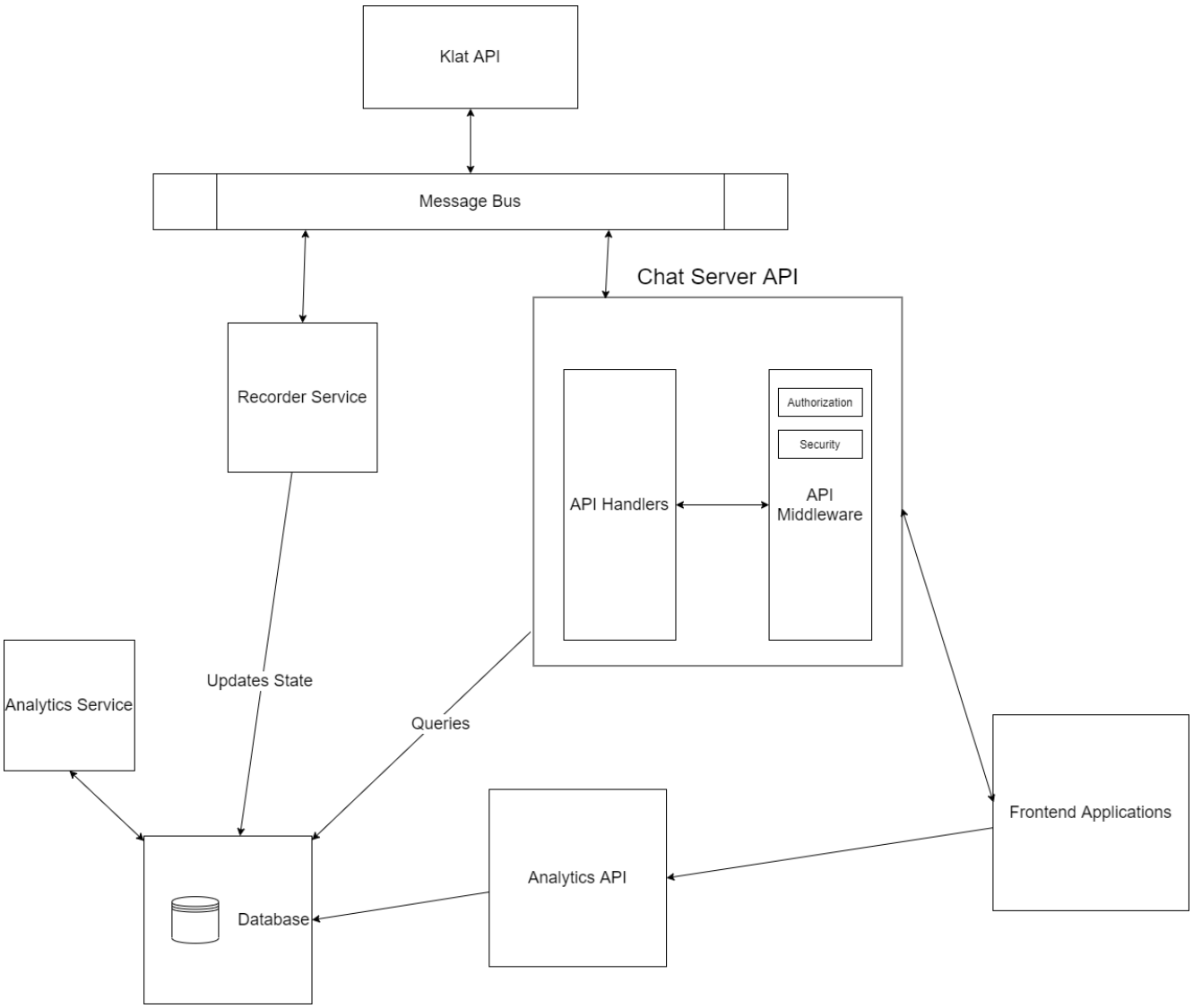
Abstract

PyKlatchat is an ideological continuation of the existing klatchat application which aims at ground-up redesigning of the existing implementation to increase modularity, extensibility, and speed capabilities. It was inspired by facing the fact that the system has grown over the ages and there is a need to have a fresh look at its design.

Suggested Architecture

Suggested Architecture (see picture 1) was designed considering following key principles:

- Single Responsibility Principle
- Scalability Potential
- Clarification of the module's interdependencies
- Flexibility Principle (ability of the system to be easily refactored in case of changing requirements, or customization requests from customers)



Picture 1: Graphical representation of suggested architecture

Suggested Modules Description

Message Bus

Module implementing Message Bus architectural pattern, it is used as a unified runtime data holder used for Input/Output communication with backends.

Klat API

Klat API is an interface for message exchange with Neon AI backend, system implies bidirectional interaction between those and suggests mechanisms for multiple possible attachment for those. Klat API is attached to the system by aforementioned **Message Bus**.

Recorder Service

Recorder Service is a standalone unit which is responsible of holding persistency of the system by constant modifying PyKlatchat database based on receiving incoming requests. For scalability purposes needs to be replicated and/or managed by message broker.

Chat Server API

Web service responsible is listening on Frontend Services requests, process them through the middleware logic (e.g. ensures authorization correctness and provides security validation) and fulfil them based on database and message bus state. In turn, once Frontend Services receive stateful requests (e.g. user wrote a message to the chat) Chat Server pushes them to **Message Bus**. To provide the most efficient implementation it is considered stateless while all the stateful actions are delegated to the **Recorder** via **Message Bus**.

Analytics Services

Analytics Service is a conceptual suggestion for the potential requirements e.g.

“Order chat participant by the efficiency formula” or “Aggregate active conversation in descending order” etc.

In order not to overload the system with additional responsibilities such tasks were delegated to the Analytics Services which, depending on the use-case, are running periodically or constantly executing and updating dedicated database tables with calculation results.

Analytics API

Web Service responsible for serving information about current analytics state.

Frontend Applications

Common name for any representation-oriented web services, which are to be used by the users directly. E.g. current implementations include Klatchat and Klatchat Nano.

These are bounded solely to the client-side logic and are intended to hold as little server-side logic as possible.

System Development Tools

Technologies for development were selected to conform the stack which is already being used in Neon AI applications.

Desired stack is considered as following:

- Python 3
- Mycroft Message Bus
- Socket IO for Klat API communication
- Flask (WSGI interface) or Fast API (ASGI interface) as web application framework.
- ORM for Database Communication, it is suggested to implement a single ORM that will be used by all the services to hold data stability and simplifying testing modules.

Development Phases

Phase Name	Estimated Time Period (days)	Description
Message Bus Implementation	7 - 10 days	Developing of suggested Message Bus structure
Some database redesign	To be discussed	Clean up database to hold only relevant data.
ORM library creation	2-4 days	Develop universal ORM library (optionally to be SQL, no-SQL agnostic)
Migration to the new Chat Server API	14-20 days	Logic migration from existing Chat Server to the new Chat Server API module
Middleware Configuration	2-4 days	Attach authorization and other related middleware to the Chat Server
Recorder Implementation	3-5 days	Implement Recorder Service and ensures in its stability on increasing load.
Frontend Application Redesign	To be discussed	Codebase optimization for the frontend side
Analytics Service and API Implementation	To be discussed	Integrate requirements for analytics and implement them