Developers & Dragons
Software Architecture Specification
10/08/2021

1. Proposed software architecture

1.1. Overview

This application architecture uses a web-hosted database to access information used to fill in the web application. Alongside the web app is a supplemental discord bot that will perform game functions and access the database.

1.2. Subsystem decomposition – Identify the subsystems and the responsibilities of each. You should use component diagrams.

Discord: Used as an API interface for a bot

Web Application: Used as a central hub for information and the main interface with the database

1.3. Hardware/software mapping – How will subsystems be assigned to hardware? You should use deployment diagrams.

Nodejs: The main programming language for the Discord bot

Sequelize: The internal database for the bot

Python: The main language for the web application

Flask: The python based system used to create a virtual environment

Vultr: Web hosting service to act as the housing for all of these things

FileZilla: Used to drag and drop files between local machines and vultr

Git: Keeping things up to date with correct stuff.

1.4. Persistent data management – Identify the data which will be persistent. Describe the file system or database to be used, including a complete database design.

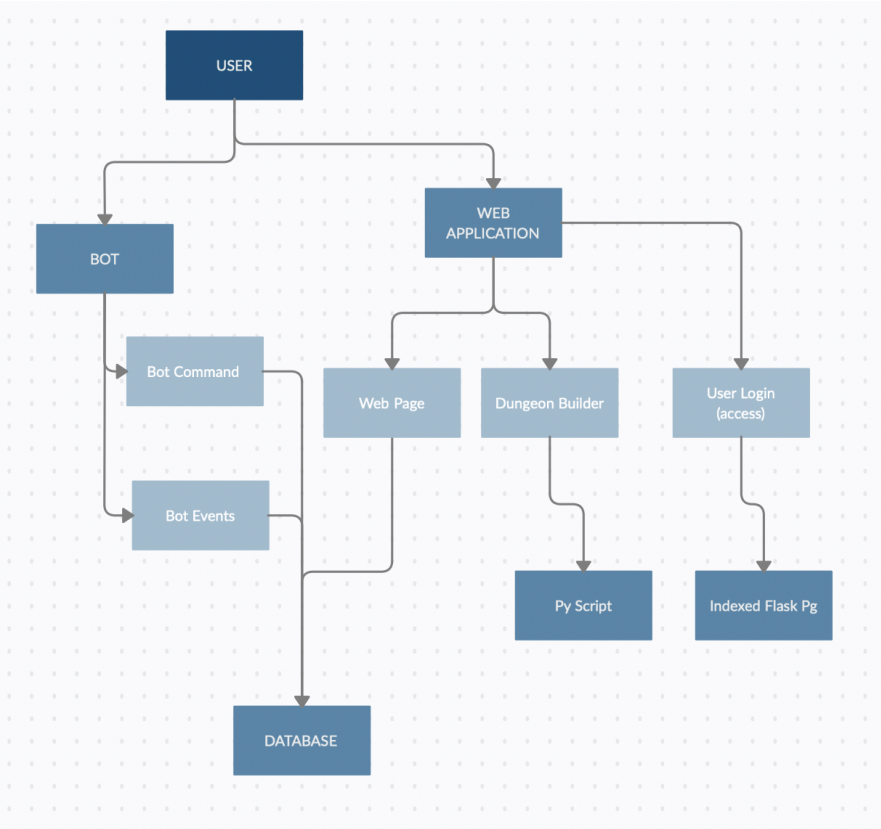
Book found information will be the only persistent database.

Other data will include character information and various user information

1.5. Access control and security – For each different actor (user, system administrator, etc.) describe the operations they will be enabled to use. Describe authentication and security provisions.

System Administrators will have read and write access to the database containing both bot and web app applications. Users on the other hand will have read access to data to the bot and web application and limited access to write to specific tables related to that user. Security will mostly involve the hashing of the end users sensitive information (username, email, phone number (optional).

1.6. Global software control – Describe the control flow (e.g. procedural, event-driven, threaded). Procedural control flows should be described using activity diagrams. Event-driven flows are best described using sequence and state diagrams (use UML diagram standards).



1.7. Boundary conditions – describe how the system will be started up, initialized and shut down. How will it respond to errors and exceptions? Any daily/weekly/monthly/yearly efforts necessary? All organizations buy new computers every 3-4 years, how to migrate to new server? Be able to bulk dump all data to file and bulk load all data from same file.

This system will be started up by running the required flask command within the Vultr service. This will generate the virtual environment. Then a process manager will be initialized to bring the bot to life, as well as the database. The shutdown process will be

the same, but in reverse. Errors will be managed with a series of try/catch systems to log errors, but not kill the process. Most of the efforts will be automated and handled ad hoc. Migrating to a new server will be as easy as dragging and dropping files with FileZilla, or pulling from git.