CS Capstone Technology Review

November 9, 2018

HyperRail App

Prepared for

OPENS Lab

Chet Udell

Prepared by

Group 25

Adam Ruark

Abstract

The central component of this project is the server that hosts this content. Of this, there are three pieces that need to be considered when deciding how to create this: how to handle user requests, how to store user data, and how to host the server. This paper discusses a few options for each of these problems and lists their advantages and disadvantages.

CONTENTS

1 Introduction

The goal of our team is to develop an application for users to configure the HyperRail system from anywhere. The application should also be able to save and load configurations and delegate many systems at once. One of the key components of this application is the web server. This server will act as the central hub for the connecting the user to the HyperRail system. This system will be in control of the user's data and verify that the configuration they create gets saved and sent to the robot succesfully. It will also monitor the status of each robot to ensure there is minimal downtime with the hardware. The realm of web hosting is a diverse field and there are many technologies out there for us to choose from. In this paper I will present three key questions regarding hosting a website and potential technologies to answer these problems.

2 How to Handle User Requests

The first task is determining how we need to route user requests and manage what the user can interact with. Below are some technologies that allow us to do this.

2.1 Django - Python Framework

Django is a widely used python library used to delegate a web server. This framework handles user requests, integration with databases, user authentication, and dynamic modeling of web pages before serving them [3]. Django itself comes with a simple server, however this is not recommended and should be paired with a similar service such as Spawning. Python is a fantastic language for getting a lot done in a little amount of code and is human readable too. This advantage means that the people that carry on this project after us will have very little difficulty molding our code into whatever plans they have for the future and it should be picked up easily. The downside of this technology is that we still rely on another technology to create and run the server.

2.2 NodeJS - JavaScript Framework

NodeJS is a very popular server-side JavaScript framework used mainly to delegate requests sent to them. This JavaScript native platform allows us to utilize the countless JavaScript modules available on NPM and also lets us use the same language for both server-side and client-side code [1]. It's fairly quick to get an application up and running and also quite versatile in what it can do. The advantage of this tool is that anyone taking on our work after us only has to be familiar with JavaScript and nothing else since both server-side and client-side code can be written in JavaScript. The downside of this is that the overall language has many caveats and the multitude of modules we'd use have their own patterns that would make understanding our code difficult at first glance. Overall this choice has a learning curve, but is very powerful once the user learns how to work with it.

2.3 PHP

PHP is super simple to get up and running and is great at serving out mostly static web pages very quickly. Creating a server in just as easy as in Javascript, but it also holds the advantage that any web pages served out can have inline PHP or JavaScript embedded in them [6]. This presents more opportunities in which technology we can use once development begins.

3 How to Store User Data

The next issue we run into is how to store the user data. There are many solutions to this problem out there each with their own benefits and downsides. Below is a discussion of a few technologies we could use for this.

3.1 MongoDB

MongoDB is a management tool that is used to host and maintain a database [5]. For a fee, MongoDB can host the databases for us and we just need to set up the architecture for it. Alternatively, we can download their engine and host a MongoDB instance on a local server instead. This has its own list of pros and cons but this means we don't have to pay a fee. MongoDB is also capable in integrating with all three frameworks discussed above making this a malleable solution.

3.2 Amazon DynamoDB

Amazon DynamoDB is Amazon's version of database hosting and is very similar to MongoDB. Amazon's databases must exist on their servers so there will be a third-party dependency [4]. However, unlike MongoDB, they do offer a free tier to their service as long as the database stays under 25GB in size. The advantage of this is that this also allows seamless integration with any other AWS service we want to take advantage of and is scalable and stable. The disadvantage of this method is that we are reliant on a third-party dependency and introduces complexity to our architecture.

3.3 JSON or XML files

Lastly, due to the nature of this project, we might not even need a true database. The overall data we will be storing is minimal in size and can be stored in simple object files such as JSON, XML, or YAML. This means that we can maintain a simple architecture and have full control of our data. This also requires that we build that logic to maintain this and it may be better to use another solution.

4 Where to host the Server

Lastly, after creating a server and/or database, this needs to be hosted somewhere that can be accessed from the internet. We have a few different options of doing this

4.1 OSU/Local Servers

The first option is to host it locally. This means that we run the server off of an on-site central system that we maintain. The advantage of this is that we have full control of our dependencies and hardware and are not limited by third-party restrictions. However, it is another dependency we will have to monitor and if the service goes down, it's on us to figure out why and fix it. Overall this is a simple and cheap solution and may be a good stepping point before worrying about using a bigger technology.

4.2 Amazon EC2

Amazon EC2 offers server instances where we can host whatever we want. These also provide fantastic integration with their DynamoDB servers if we choose to go that route. Additionally, they offer monitoring tools and scalable systems if this project grows larger than anticipated [2]. The downside of this tool is that Amazon does not offer this service for free (beyond a 12 month free period) and must be paid for. Additionally, this may be overkill for what we are attempting to accomplish.

4.3 HyperRail System

Lastly, we can ”host” these directly on the Arduinos controlling the HyperRail systems. While this isn't a true web technology, it keeps each system independent and non-reliant to internet connectivity. The downside of this is that maintaining multiple systems at once will be difficult or near impossible and keeping all HyperRail systems in sync is impossible. This will require an entirely different architecture than the one proposed by us, but it does carry a huge advantage in that internet connectivity is no longer a hard requirement. Due to the nature of where the HyperRail system will be use, this may be an idea we want to look into further.

5 Conclusion

Overall there are many different routes that we can take to handling the data that comes in and is ultimately used to run the HyperRail system. My suggestion for this application is a combination of NodeJS to create the server, JSON files to store user data, and our local servers to host the whole system. JSON files are native to JavaScript and should provide a homeogenous environment for the developer and our local servers are easier to maintain than relying on a third-party with data. There are other technologies out there that perform similar tasks and we may look into them down the road, but these offer much of the functionality we need and will most likely be used for this project.

References

[1] Amit Ashwini. *What is node.js use for in simple terms?* URL: https://www.quora.com/What-is-node-js-used-for-in-simple-terms.

[2] David Clinton. *What is EC2 and What Does It Do?* URL: https://medium.com/@dbclin/what-is-ec2-and-what-does-it-do-765db27bbca2.

[3] *Django Introduction*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

[4] Priyanka Varma. *What is Amazon DynamoDB and what is it used for?* URL: https://www.quora.com/What-is-Amazon-DynamoDB-and-what-is-it-used-for.

[5] *What is MongoDB?* URL: https://intellipaat.com/blog/what-is-mongodb/.

[6] *What is PHP?* URL: http://php.net/manual/en/intro-whatis.php.