Bi-Weekly Progress Report

Project Short Name: Genesis: Configuration Management

Your Name: Douglas Swanson

Number of hours worked week 1: 6 hours week 2: 18 hours

Report Due Date: 9/18/22

1. Tasks and Targets set for the report period
   1. Gather information about the data that the services will be required to handle
   2. Create a data model for this data, for both services
   3. Start writing the first service, Genesis Scenario Management
   4. Continue Researching Services
2. Adjustment and revisions made against the original tasks and targets. Explain the reasons if adjustments were made.
   1. Instead of starting to write the first service I spent time researching and learning the technology stack. This was done to better understand all of the components that will be required in order to successfully build this tool.
   2. Also tacked on learning how Kubernetes works and how to build out a cluster to see if it will actually be the tool that will be used as the service broker.
3. Summary of Accomplishments and the contribution to the project.
   1. Completed the Genesis Scenario data model which will be managed by the first service.
   2. Completed the Spacecraft data model which will be managed by the second service.
   3. Completed research of RESTful services.
   4. Completed a course on Full Stack React.
4. Elaboration of the work done.
   1. The first iteration of the Genesis Scenario model looks like this:
      1. Module Template

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| Key | int | Unique identifier for this module template |
| Name | varchar(25) | A human readable identifier for this template |
| Version | varchar(10) | The version of software this template belongs to |
| FilePath | Text | The directory file path to the folder containing the template’s files. |
| Description | Text | A more in depth description of the template and its purpose. |
| Deprecated | bool | A flag indicating that this template is old and has been replaced with a newer version. |

* + 1. Genesis Scenario

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| Key | int | Unique identifier for this scenario |
| Name | Varchar(25) | A human readable identifier for this scenario |
| Description | Text | An explanation of the purpose and or event that this scenario was used to support |
| Date Last Updated | Date Time | The last time that this scenario was updated by any user |
| Templates Used | Foreign Keys | The id’s of the templates used to create this scenario |

* 1. The first iteration of the spacecraft models looks like this:
     1. Spacecraft Template

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| Key | int | Unique identifier for this scenario |
| Name | Varchar(25) | A human readable identifier for this scenario |
| Constellation | Varchar(25) | An explanation of the purpose and or event that this scenario was used to support |
| Rx | double | The starting x position of the vehicle |
| Ry | double | The starting y position of the vehicle |
| Rz | double | The starting z position of the vehicle |
| Vx | double | The starting x velocity of the vehicle |
| Vy | double | The starting y velocity of the vehicle |
| Vz | double | The starting z velocity of the vehicle |

* 1. Turns out that RESTful services follow the following standards:
     1. Use HTTP Methods as they should be used
        1. I.E Get is used to get information where Post is used to add information etc.
     2. Services are stateless
        1. The service does the requested operations without needing to know what state the client is in.
        2. It the clients responsibility to track it’s own state.
     3. Use the URIs in a manner that self documents the data in which the service will use.
        1. I.E. to add a new spacecraft, the URI might look something like this
           1. .\spacecraft\add
     4. Will exclusively used the JSON format to accept request data and return response data
        1. To be used in the payload of the HTTP request/response
  2. In the full stack React course I took I learned how to:
     1. Use typescript
     2. Create a React application
     3. Connect the React application to a service using GraphQL
     4. Use hooks within the React application to only redraw the components on the screen that deal with the data that has just been updated
        1. In other words, setup the application to “React” to data changes
     5. Took a look at one method of deploying an application
     6. Wrote a Node js service to supply data

1. Conclusion Remarks
   1. This was a productive week overall despite not being able to start building out the first service. I learned a bunch of information about the architecture and design of React applications and what steps will be required to successfully build the two applications for the front end of this project. Also learned how to build a service in Node. Took sometime this week to study the data that will be managed by this Genesis project and than built out the data models that the services will use to perform their specified duties.
2. References used
   1. <https://developer.ibm.com/articles/ws-restful/> (IBM article on RESTful services)
   2. Service-Oriented Computing Textbook
   3. <https://learning.oreilly.com/library/view/soa-made-simple/9781849684163/ch02.html#ch02lvl4sec02> (SOA Made Simple Book)
   4. <https://learning.oreilly.com/library/view/full-stack-react-typescript/9781839219931/B15508_12_Final_JC_ePub.xhtml> (Full-Stack React, Typescript, and Node course book)