**CST-235 Java Application Programming Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | **REST-Based Services, and Message-Based Services Guide***.* | |
| **Date:** | *04/19/2020* | |
| **Revision:** | *This should be the revision, starting at 1.0, for your Report.* | |
| **Team:** | 1. *Donnell Sample* | |
|  | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | *https://github.com/Davon55/RestAPI.git* | |
| **Peer Review:** | *Y/N* | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Planning Documentation**

**Agile Scrum Product Backlog:**

*This needs to contain a URL to the Bitbucket Scrum Product Backlog Artifact.*

**Agile Scrum Sprint Backlog:**

*This needs to contain a URL to the Bitbucket Scrum Sprint Backlog Artifact. This current week's progress should be reflected in the above section of this Design Report.*

**Agile Scrum Burn Down Chart:**

*This needs to contain a URL to the Bitbucket Scrum Burn Down Chart Artifact.*

**Agile Retrospective Results:**

*The following table should be completed after each Retrospective on Things That Went Well (Keep Doing). An alternative to the following table is to use a mind mapping tool such as Coggle; if you use a mind mapping tool, you must include the URL or image file.*

|  |
| --- |
| **What Went Well** |
|  |
|  |
|  |

*The following table should be completed after each Retrospective on Things That Didn’t Go Well (Stop Doing) and What Would Be Done Differently Next Time With an Action Plan to Improve (Try Doing and Continuous Improvement). An alternative to the following table is to use a mind mapping tool such as Coggle; if you use a mind mapping tool, you must include the URL or image file.*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
|  |  |  |
|  |  |  |
|  |  |  |

**Design Documentation**

**Install Instructions:**

*Include step-by-step instructions for setting up your database, configuring, and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from Bitbucket and deploy the application to an externally hosted site.*

**General Technical Approach:**

*Describe your approach and design here in your own words. You should also summarize any meeting notes, brainstorming sessions, and so forth that you want to retain throughout the design of your project.*

**Key Technical Design Decisions:**

*Any final technical design decisions, such as framework decisions and so forth, should be documented here. List the technology/framework, its purpose in the design, and why it was chosen.*

**Known Issues:**

*Any anomalies or known issues in the code or functionality should be documented here.*

**Risks:**

*Any risks, unknowns, or general project elements that should be tracked for risk management should be documented here.*

**ER Diagram:**

*Include the image file of your ER database diagram.*

**DDL Scripts:**

*This should contain a link to Bitbucket where the DDL script can be downloaded from.*

**Sitemap Diagram:**

*Include the image file of your Sitemap diagram.*

**User Interface Diagrams:**

*Inlcude any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for why you are leaving this section as N/A.*

**Class Diagrams:**

*Insert any class diagrams here. Your class diagrams should be drawn correctly with the three appropriate class compartments, + plus and – minus to indicate accessibility, and also the data types for the state/properties as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for why you are leaving this section as N/A.*

**Service API Design:**

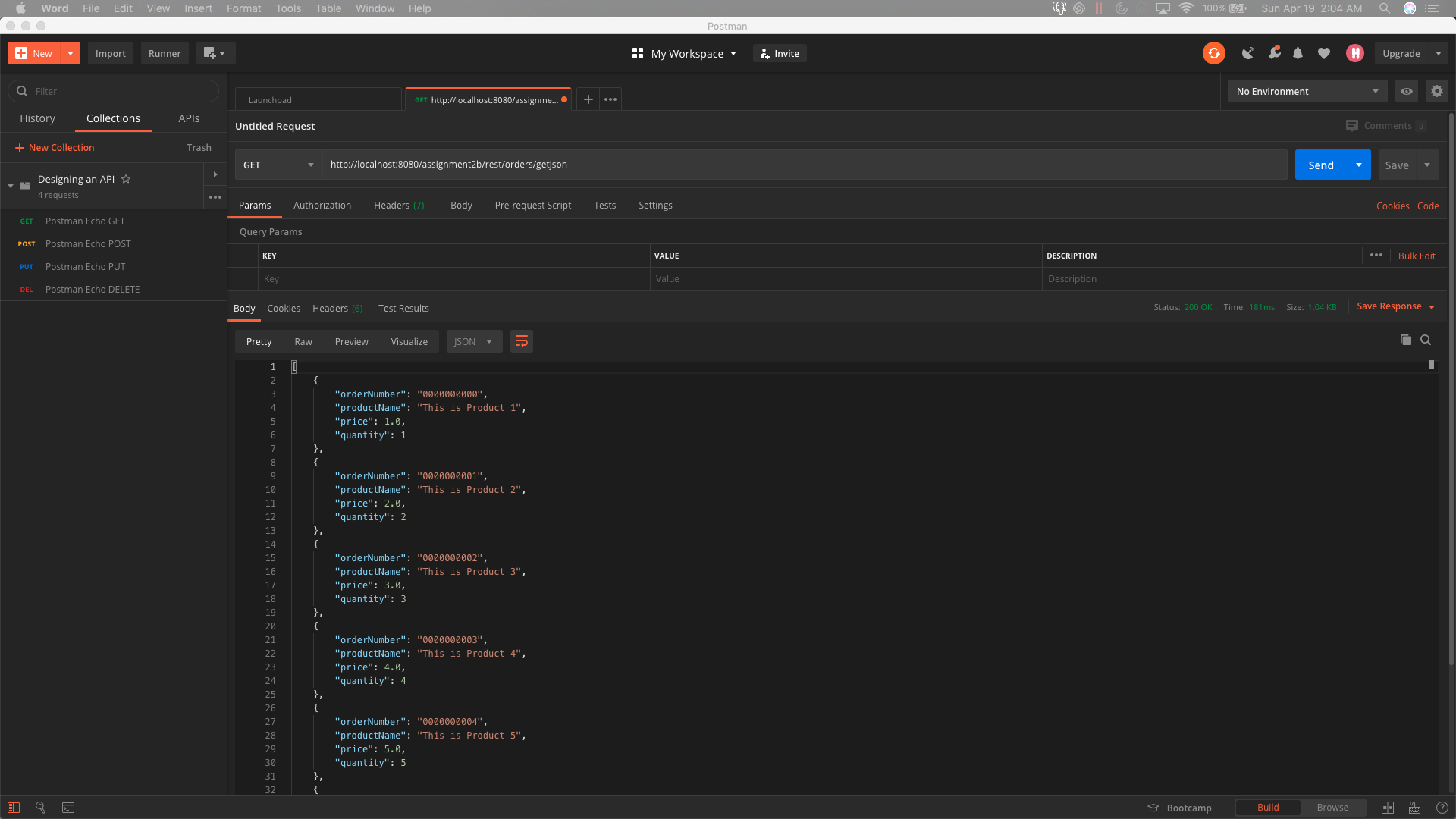
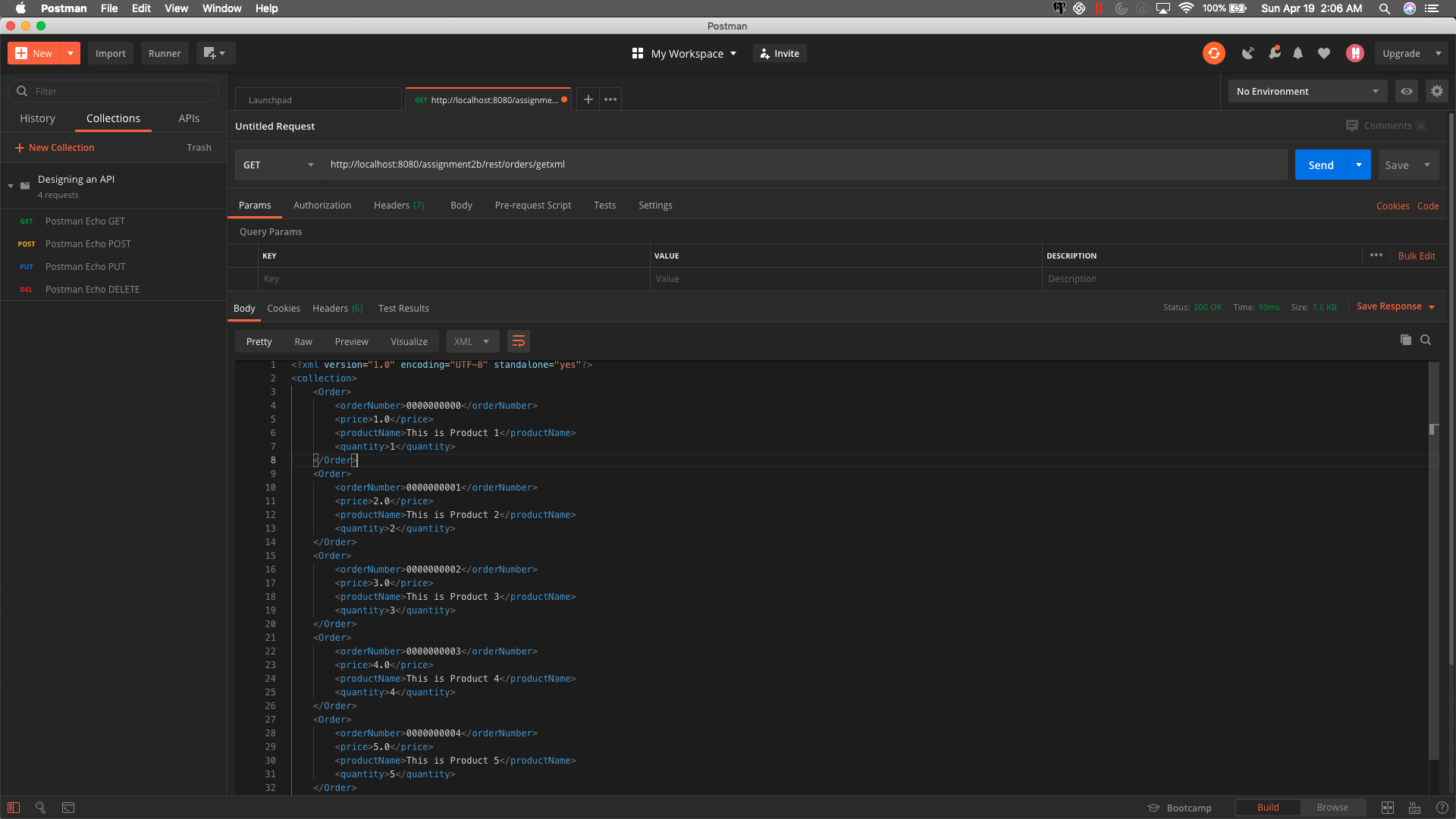
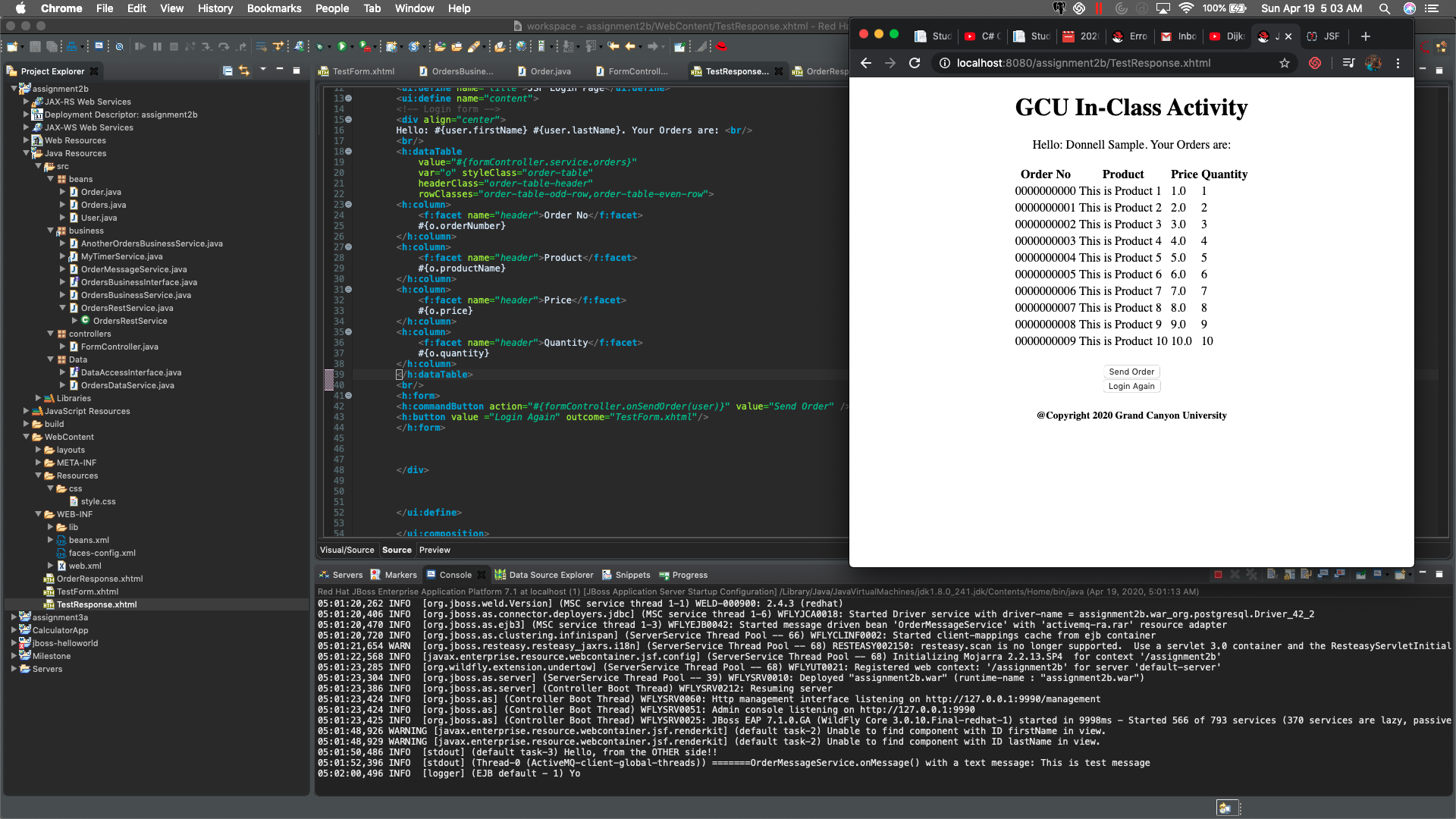
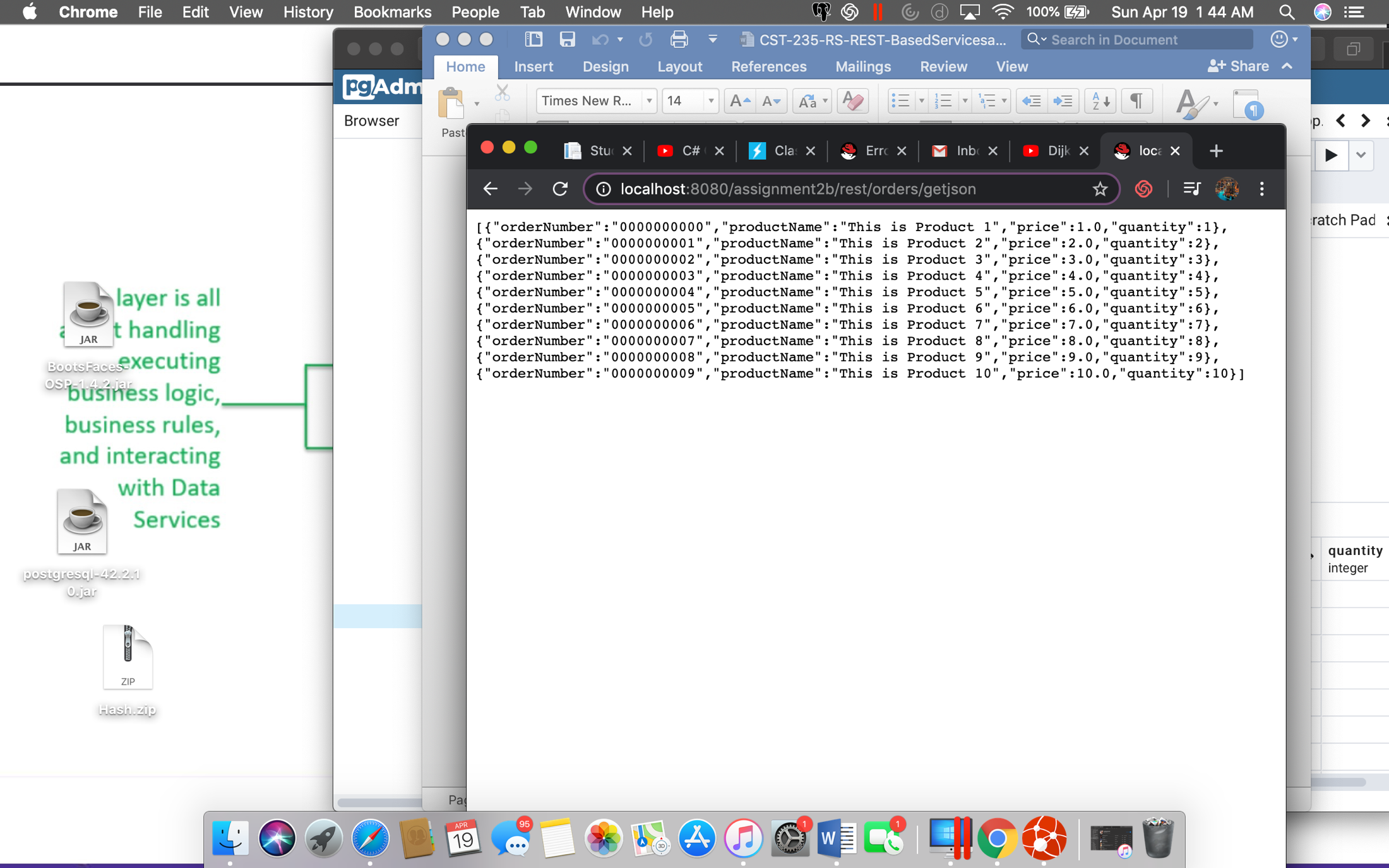
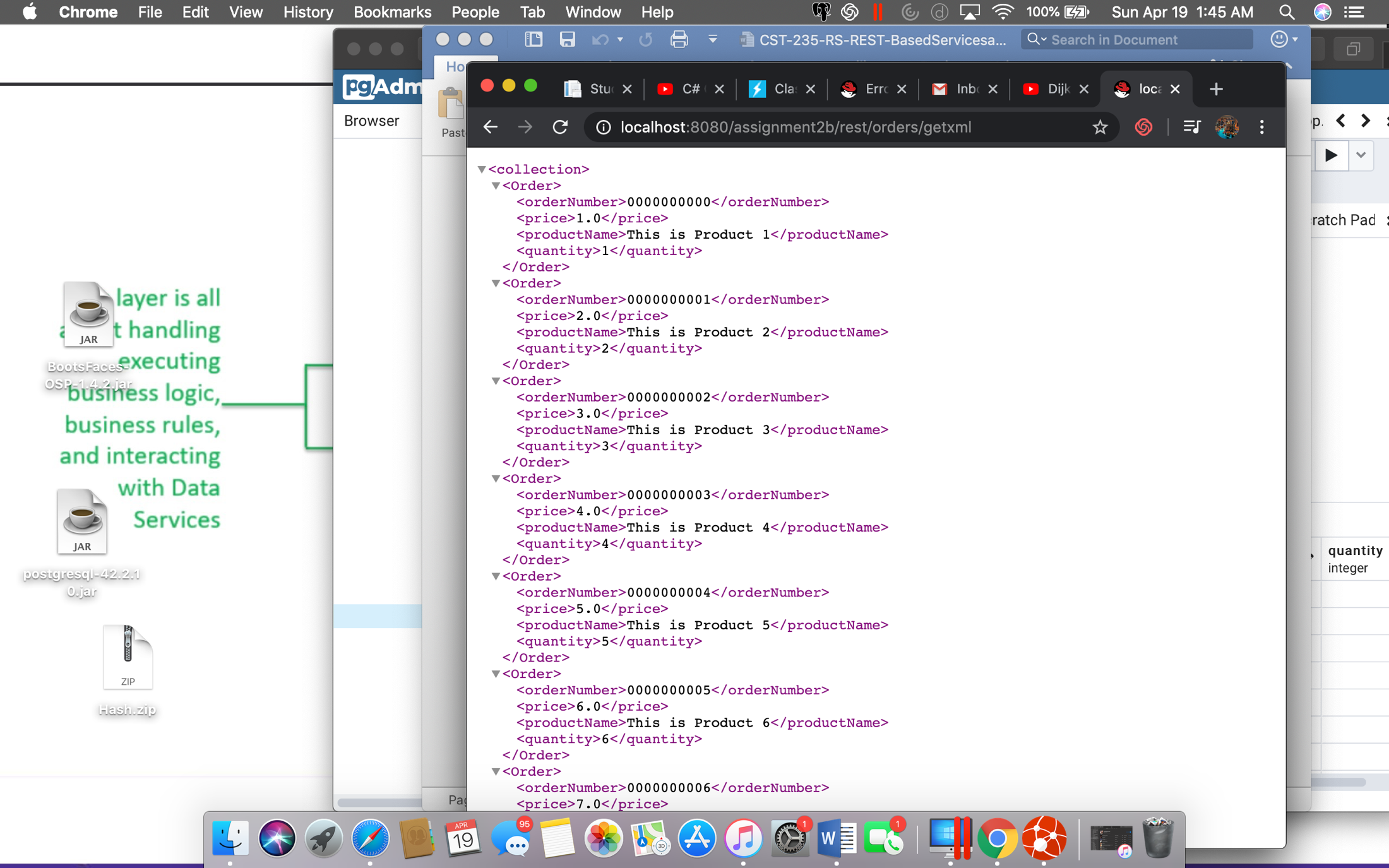
*This section should fully document any Third Party Service Interface APIs being consumed or application specific Service APIs being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third party developer to integrate with the service and API.*

**Security Design:**

*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

**Other Documentation:**

*For this project we took the approach of creating a REST-based service with a message based service. We first configured our ide to be able to create a REST and MESSAGING service. We then created a JAX-RS facet which utilized XML and JSON using GET. Tested both in postman for cleaner code. Multiple annotation were added to the class to complete this request. Next was the messaging service where another class was created in the business service, this class was ran by a message drive bean. A method was then created in the interface then implements in the business service, and then added code to test whether a text message or other type of message was being sent. A JSF page and Form controller was also create to send responses to other pages and display information. Multiple new annotations was put into place also the data access interface was injected.*

**