COSC 350

**WEB APPLICATION INFRASTRUCTURE**

A breakdown of network and infrastructure settings of the Lockbox Application.

horizontal line

# Placeholder image

# Group members:

# David Nsoesie, Albert V. Casas, Anthony Gillis Jr., Ehimare Uiyoshioria

# 

# 

# Abstract

As the amount of internet users grows, new technologies are emerging. IAAS (Infrastructure as a service) also grows as industry powerhouses like Amazon, Microsoft and Google continue to be the easy solution for up and coming businesses to host, deploy and manage their apps, servers and sites across the globe. These companies ensure consumers and competition are protected in the emerging cloud-based economy. The cloud providers guarantee that the resources will be available around the clock. The providers assure that the period of unavailability of resources is very low. Recently the cloud users have increased rapidly; therefore the providers also have increased, basically increasing the complexity of the infrastructure. In this short report we will discuss how IAAS assisted in the deployment of the lockbox application and certain practices that were followed as the client API was built to ensure scalability and readability.

## What’s a Rest Api?

An API is an application programming interface - in short, it’s a set of rules that lets programs talk to each other, exposing data and functionality across the internet in a consistent format (Mahesh). REST stands for Representational State Transfer. This is an architectural pattern that describes how distributed systems can expose a consistent interface. When people use the term ‘REST API,’ they are generally referring to an API accessed via HTTP protocol at a predefined set of URLs. These URLs represent various resources - any information or content accessed at that location, which can be returned as JSON, HTML, audio files, or images. Often, resources have one or more methods that can be performed on them over HTTP, like GET, POST, PUT and DELETE. (Mahesh)

There are a few very useful advantages to using and building Rest Api’s, some more useful than others. Rest Api’s feature

* Stateless client/server protocols
* Object manipulation from the URI/URL
* A Uniform interface
* A Layer system
* Use of hypermedia

For developers it provides a few key advantages:

* Separation between the client and the server.
* Visibility, readability and scalability
* The Api is independent of the type of platform or languages

Separating the client and server during development has one very glaring advantage. It allows each development team to scale the product without too much of a problem (BBVAOPEN4U). They are able to migrate to other servers and make database changes without worrying about the contract the Api is using to fulfill requests; depending on the overall extent of these changes. The separation makes it easier to have the front and back-end on different servers making frontend and backend development teams more flexible and easy to manage with proper communication.

### HTTP

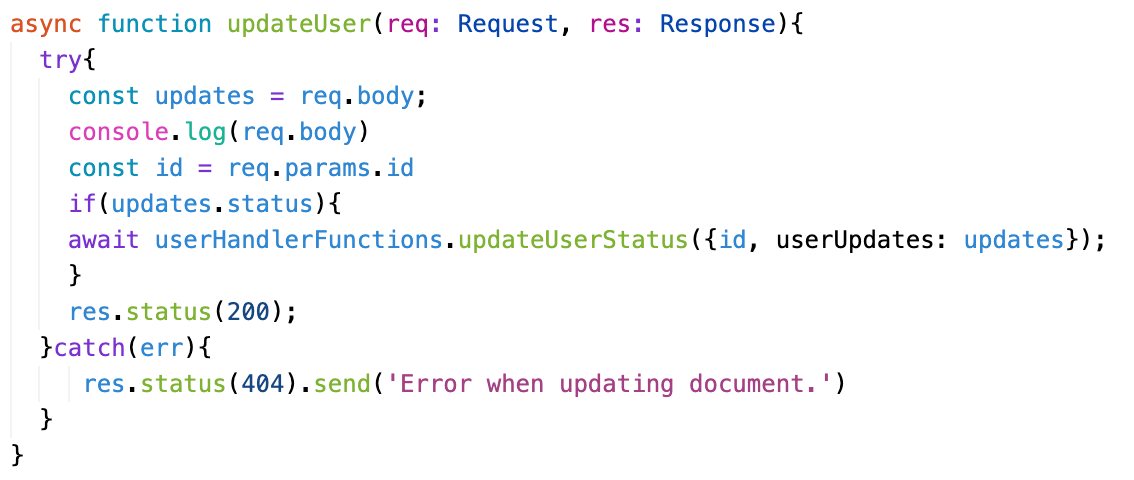
HTTP is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser (MDN docs). A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts, and more. Refer to figure (1.1)



1.1

Clients and servers communicate by exchanging individual messages instead of utilizing streams of data. Requests are made by the web browser, also known as the client and the messages sent by the server as a response are therefore called responses.

2.1

As you can see in figure 2.1, API, written in typescript has this function “updateUser” which takes in two arguments, a request and response. The request argument is used to pull any incoming data that the client may be making. From there, once the request has been fulfilled the API will then respond to the client with the Response argument it was provided with and either send a response with a 200 status or a 404 if there was an error while said request was being made. 

Designed in the early 1990s, HTTP is an extensible protocol which has evolved over time. It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection, though any reliable transport protocol could theoretically be used. Due to its extensibility, it is used to not only fetch hypertext documents, but also images and videos or post content to servers

### 

### Backend Infrastructure( Microservices )

Typical monolithic applications are built using different layers—a user interface (UI) layer, a business layer, and a persistence layer. A central idea of a microservices architecture is to split functionalities into cohesive “verticals”—not by technological layers, but by implementing a specific domain. Figure 1.2 depicts a reference architecture for a typical microservices application on AWS.



Typical microservices application on AWS



1.2

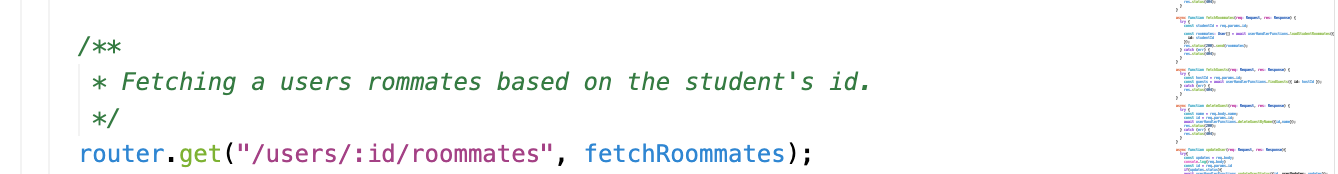
### LockBox Application Infrastructure

Lockbox is an application that serves as a management tool for two types of people at universities; students and Building administrators. The main motivation for building the application was to create an interface that students on a campus community can use to interact and communicate with their roommates without actually having to reach out to anyone.

On the administrative side we wanted to create an interface that would enable campus employees to monitor traffic throughout the university and have a safe area to store that information. It also allows these users to manage access to buildings across the campus community.

The Application is hosted on heroku and is running in a private vpc under the address 10.0.0.0/24. Within this VPC there’s a load balancer that allows our Lockbox API monolith to be accessed by end-users. Our front end application accesses this API over http to make requests and perform crud operations on data in mongoDB. Unlike the infrastructure on display in figure 1.2. Our structure does not place a mongodb server within the vpc. We instead provide the configured vpc with the ip of our mongodb server after we have whitelisted the external IP of our gateways in Mongo. With our mongodb server running externally, and all the api logic housed on AWS. We decided to host the client on a static server, keeping the client and server completely separate.

The route structure of our api makes it very easy to make requests and serve JSON to our front end application. The lockbox api is written in javascript/Typescript, utilizing node.js runtime.



1.3

Once an API request is made from the client to the route in 1.3 “{{API-URL/users/:id/roommates}}” the application will call the function in the callback which will return whatever is found in the database as JSON. As shown in figure 1.4



1.4

The Lockbox api can be accessed through the following URL: <https://sheltered-refuge-78289.herokuapp.com/>

The Lockbox client can be accessed through the following URL: <https://limitless-eyrie-01672.herokuapp.com/>

References

“An Overview of HTTP.” *MDN Web Docs*, https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

BBVAOpen4U. “REST API: What Is It, and What Are Its Advantages in Project Development?” *BBVAOpen4U*, 23 Mar. 2016, https://bbvaopen4u.com/en/actualidad/rest-api-what-it-and-what-are-its-advantages-project-development.

Haldar, Mahesh. “RESTful API Designing Guidelines - The Best Practices.” *By*, https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9.

*HTTP/1.1: Request*, https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html.

Richardson, Chris. “Building Microservices Using an API Gateway.” *NGINX*, 8 Feb. 2019, https://www.nginx.com/blog/building-microservices-using-an-api-gateway/.

“Whitepapers.” *Amazon*, Earthpledge Foundation, https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/simple-microservices-architecture-on-aws.html.