18-5-2022

Servicio de localización de autobuses

Tecnologías para la Integración de Soluciones.

Dr. José Rafael Rojano Cáceres

GUTIÉRREZ RODRIGUEZ JOSÉ ÁNGEL
DOMÍNGUEZ HERRERA JORGE
HERNÁNDEZ LÓPEZ JAVIER
UNIVERSIDAD VERACRUZANA

Contenido

Introducción

En la presente documentación se plasma el desarrollo de un servicio de localización de autobuses planteado por alumnos de la Universidad Veracruzana donde se pretende poner en práctica los aprendizajes adquiridos acerca de temas referentes a la integración de soluciones.

Este proyecto es un conjunto de microservicios que se pretenden implementar con el objetivo de formar una infraestructura capaz de permitir a la ciudadanía xalapeña tener localizado cada autobús en servicio con el fin de permitirles eficientizar y organizar mejor su proceso antes, durante y después de abordar de autobús a lo largo de las diversas rutas de la ciudad.

El objetivo del documento es transmitir al lector la problemática observada, justificar su relevancia y describir a detalle la estructura del servicio anteriormente mencionado, con el fin de transmitir al lector la pertinencia e importancia de implementar este proyecto, siendo siempre claros con los costos que conlleva realizarlo pero también resaltando los beneficios que representará tanto para los pasajeros (usuarios del transporte público en la ciudad) como para los empresarios y dueños de las diferentes unidades de transporte.

Motivación

Este proyecto nace con la motivación de mejorar la calidad de vida de la población xalapeña proveyendo a la ciudad de Xalapa con una infraestructura que permita a los ciudadanos tener un conocimiento constante acerca de cuáles autobuses de transporte público se pueden abordar en tiempo real en las diferentes paradas de sus diferentes rutas.

De forma más concreta, se busca facilitar y eficientizar el proceso que una persona sigue desde el momento de salir de su casa hasta realizar un abordo de autobuses en la ciudad anteriormente mencionada.

También se tiene el objetivo de ayudar a las empresas de transporte público a ofrecer un mejor servicio en la ciudad dándole un valor agregado el transporte colectivo por encima de los demás tipos de transporte. Por último, pero no menos importante, a nivel de equipo, tenemos una misión, la cual es hacer uso de los conocimientos

adquiridos en esta experiencia educativa para proponer mejoras entorno a situación anteriormente planteada.

Problemática

En la actualidad la infraestructura de la ciudad no es apta para proveer a su ciudadanía de servicios que mantengan un control de la ubicación de las unidades pertenecientes al servicio de transporte urbano público, mejor conocidos como autobuses, por ello se carece de eficiencia en el proceso mediante el cual la población aborda los autobuses necesarios para trasladarse a lo largo de las diferentes rutas establecidas por las empresas de transporte de la ciudad.

En forma específica, la población no tiene la capacidad de optimizar el uso de su tiempo mientras aborda su autobús, puesto que al desconocer a qué distancia se encuentra la unidad y en qué tiempo puede abordarla se ve limitada al tomar decisiones sobre el uso del tiempo disponible y su optimización para llevar a cabo, o no, otras actividades que considere pertinentes antes de abordar.

Una segunda problemática para abordar es que los dueños de las empresas de transporte público no logran entender completamente la demanda de autobuses a lo largo de las rutas de la ciudad, esto tiene como consecuencia una mala distribución de unidades (autobuses) a lo largo de la ciudad donde se destinan autobuses a rutas que no los necesitan y de la misma forma existen rutas que carecen de las unidades suficientes para satisfacer la demanda.

Además, se ha intentado proveer a la ciudad de aplicaciones que mediante GPS ayuden a localizar de manera exacta cada unidad de transporte (autobús) de la ciudad, sin embargo, al necesitarse conexión por internet o datos móviles para recibir y enviar las ubicaciones del GPS se requiere un fuerte gasto de recursos por parte de los usuarios de las unidades de transporte urbano lo cual no resulta óptimo económicamente hablando.

Solución

Este proyecto plantea la solución de dotar a Xalapa con una infraestructura que permita implementar un conjunto de microservicios capaces de mantener ubicada a cada unidad que ofrece servicio de transporte urbano independientemente de la empresa y ruta a la que pertenezcan; ofreciendo a los usuarios un listado en tiempo

real de los autobuses que se encuentran más cercanos al pasajero; de igual manera será capaz de ofrecer a los dueños de las diferentes empresas un concentrado que muestre la demanda de autobuses a lo largo de las rutas manejadas.

El primer microservicio se llama “Ubícame” y estará instalado en cada una de las diversas antenas situadas en diferentes puntos de la ciudad; cada autobús se conectará a su antena más cercana, donde el microservicio creará un registro del autobús y se guardará dentro de la memoria de la antena, este mismo microservicio recibirá la petición de los usuarios del transporte urbano para conocer los autobuses que se encuentran en el área de la antena y finalmente otras antenas que también tengan instalado el servicio búscame podrán pedir que se elimine un autobús de la memoria de la antena para darlo de alta en otra.

Un segundo microservicio se llama “Abordos” y se encarga de registrar los abordos que se realizan, guardando el identificador del autobús y la antena a la cual estaba conectado cuando el usuario lo abordó; este microservicio concentra el historial de todos los abordos efectuados en la ciudad y permite a los dueños de las empresas de autobuses conocer la demanda del servicio en las diferentes zonas y rutas que se cubren a lo largo de la ciudad.

Finalmente existe un tercer microservicio encargado de redirigir al autobús a su antena más cercana para ejecutar la petición al microservicio búscame creando un alta en la antena más cercana y una eliminación de carro en la antena donde se encuentra actualmente el autobús.

Estimación de costos del proyecto.

Salarios

Puesto	Cantidad	Pago mensual	Meses por pagar	Total pagado
Programador	3	10000	6	180,000
Encargado de instalación de infraestructura	12	8000	6	576,000
Encargado de documentación	3	6000	6	108,000

Subtotal	864,000
35% de prestaciones	302,400
Total	1,166,400

Contrataciones por honorarios

Concepto	Cantidad	Pago mensual	Meses por pagar	Total pagado
Contador	1	1200	12	14,400
Jurídico	1	2000	6	12,000
Total				26,400

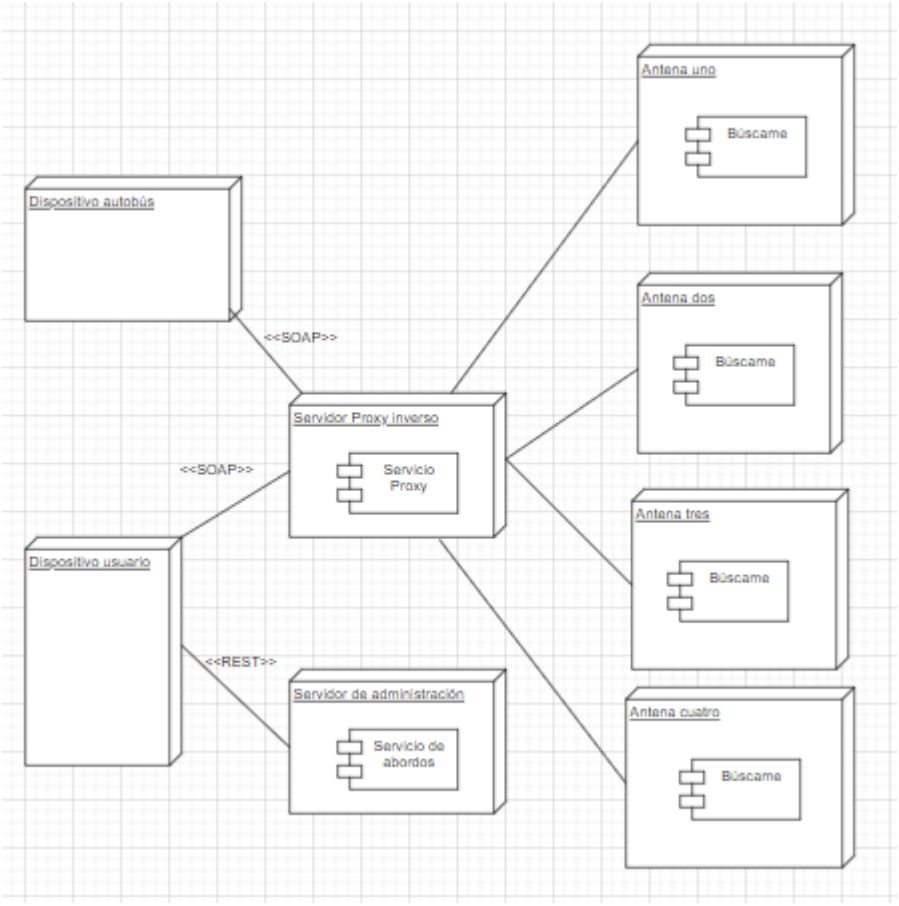
Gastos por servicios informáticos

Concepto	Cantidad	Pago mensual	Meses por pagar	Total pagado
Servicio de hosting (incluye dominio)	6	79.99	12	5,759.28
Total				5,759.28

Costos totales

Concepto	Costo
Salarios	1,166,400
Contrataciones por honorarios	26,400
Gastos por servicios informáticos	5,759.28
Total	1,198,559.28

Diagrama de despliegue



Documentación del microservicio “Búscame”

El microservicio *búscame* debe ser accedido mediante el EndPoint de alguna de sus cuatro antenas (dependiendo de a cuál se quiera conectar), dicho puntos de conexión se presentan a continuación:

Antena uno: https://buscame-autobuses.herokuapp.com/autobuses/

Antena dos: https://buscame2.herokuapp.com/autobuses/

Antena tres: https://buscame3.herokuapp.com/autobuses/

Antena cuatro: https://buscame4.herokuapp.com/autobuses/

Así mismo, se consume y responde mediante la tecnología SOAP, codificada en XML; en él se contienen cuatro funciones a las cuales se les puede realizar una petición y se presentan a continuación:

Crear autobús

Esta petición se debe realizar cuando el autobús se conecte a su antena más cercana con el fin de crear un registro sobre su estancia en la zona aproximada, la función a

llamar se llama *CrearRequest* y se deben enviar cinco parámetros que contengan la información del autobús que se creará en la antena, estos datos son id, ruta, empresa, conductor y placa; el primer parámetro debe ser un entero, los restantes son cadenas de texto. A continuación, un ejemplo de cómo realizar una petición a la función:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <CrearRequest xmlns="https://buscame.autobuses.com/buscame">
         <id>1</id>
         <ruta>Xalapa-Banderilla</ruta>
         <empresa>ATB</empresa>
         <conductor>Jorge Domínguez</conductor>
         <placa>XL-4852</placa>
      </CrearRequest>
   </Body>
</Envelope>
```

En cuanto a la respuesta que se recibirá: esta respuesta será dada en un tipo de dato booleano, si regresa *true* significará que el autobús se creó correctamente, en caso contrario regresará *false*. A continuación, el ejemplo de una respuesta de la función:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns2:CrearResponse xmlns:ns2="https://buscame.autobuses.com/buscame">
         <ns2:response>true</ns2:response>
      </ns2:CrearResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listar autobuses

Esta petición debe realizarse cuando un usuario del transporte urbano se conecta a su antena más cercana con el fin de conocer qué carros se encuentran en la misma zona, se debe llamar a la función *ListarCarrosRequest* y no necesita que se envíen parámetros. A continuación, se muestra un ejemplo de cómo realizar la petición:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
```

```
    <ListarCarrosRequest
xmlns="https://buscame.autobuses.com/buscame">[any]</ListarCarrosRequest>
  </Body>
</Envelope>
```

La respuesta dada por el servidor será mediante un conjunto de objetos, pudiendo entenderse como un arreglo de autobuses, donde cada autobús tiene los datos id, ruta, empresa, conductor y placa. A continuación, se muestra un ejemplo de respuesta recibida:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:ListarCarrosResponse xmlns:ns2="https://buscame.autobuses.com/buscame">
      <ns2:carro>
        <ns2:id>1</ns2:id>
        <ns2:ruta>Ruta 1</ns2:ruta>
        <ns2:empresa>Empresa I</ns2:empresa>
        <ns2:conductor>Jorge Domínguez</ns2:conductor>
        <ns2:placa>SGR-56</ns2:placa>
      </ns2:carro>
      <ns2:carro>
        <ns2:id>2</ns2:id>
        <ns2:ruta>RUTA NUEVA</ns2:ruta>
        <ns2:empresa>ATB</ns2:empresa>
        <ns2:conductor>Ángel Gutiérrez</ns2:conductor>
        <ns2:placa>OPL-78</ns2:placa>
      </ns2:carro>
    </ns2:ListarCarrosResponse>
  </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Quitar carro

Esta petición se realiza cuando se pretende conectar al autobús a otra antena y sacarlo de la antena actual, se debe llamar a la función *QuitarCarroRequest* mandando solamente el parámetro llamado “id” que debe ser de tipo entero. A continuación, se presenta un ejemplo de petición:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <QuitarCarroRequest xmlns="https://buscame.autobuses.com/buscame">
         <id>2</id>
      </QuitarCarroRequest>
   </Body>
</Envelope>
```

La respuesta se da en términos de un dato booleano el cual tendrá el valor *true* en caso de haber ejecutado la función correctamente y *false* en el caso contrario. A continuación, un ejemplo de una respuesta recibida

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns2:QuitarCarroResponse xmlns:ns2="https://buscame.autobuses.com/buscame">
         <ns2:response>true</ns2:response>
      </ns2:QuitarCarroResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Documentación del microservicio “Historial de abordos” – REST

Este servicio consiste en obtener el historial de abordos, de todos los pasajeros que hayan abordado las unidades, correspondientes a las 4 antenas, que se mencionan en el proyecto. Ambas funciones que se mencionan a continuación, están conectadas

a una base de datos en la nube que lleva por nombre Postgres SQL, las cuales permitirán disponer de la información en todo momento. También, se encuentran ya alojadas en heroku.

Por otra parte este microservicio deberá ser accedido mediante la URL https://historialabordos.herokuapp.com

El servicio considera dos funciones:

1.- **Guardar abordó**, el cual es con tecnología API REST y funciona mediante una petición POST, en esta función, se pedirán 2 parámetros, el cual es el nombre del autobús abordado y el segundo el nombre del pasajero que lo abordó. Sin embargo habrá dos atributos más, fecha y hora, pero no serán requisitados ya que se llenarán al momento de realizar la petición automáticamente.

Este microservicio se podrá acceder mediante la URL https://historialabordos.herokuapp.com/GuardarAbordo/NombrePasaje/NombrePasajero

```
@RestController
public class AbordosController {
    @Autowired
    private AbordosRepository iabordos;

    //CREAR ABORDOS
    @RequestMapping(value = "/GuardarAbordo/{nombreAutobus}/{nombrePasajero}", method = RequestMethod.POST)
    public String GuardarAbordo(@PathVariable("nombreAutobus") String nombreA, @PathVariable("nombrePasajero") String nombreP)
    {
            if(nombreA != null && nombreP != null){
                LocalDate todaysDate = LocalDate.now();
                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
                LocalTime time = LocalTime.now();
                String hora = time.format(formatter);

                Abordos abordo = new Abordos();
                abordo.setNombreAutobus(nombreA);
                abordo.setFecha(todaysDate);
                abordo.setHora(hora);
                abordo.setNombrePasajero(nombreP);
                iabordos.save(abordo);
                return "Usario agregado";
            }else{
                return "Ha ocurrido un problema, no se pudo agregar el usuario ";
            }
    }
```

Lo que se obtiene como respuesta sería un String, el cual puede tener dos posibles valores dependiendo si la función, se ejecutó con éxito, si fue así, mostrará **“Usuario agregado”**. Si no, te arrojará el mensaje de, **“Ha ocurrido un problema, no se ha podido crear el usuario”**.

2.- **Obtener historial**, al igual que el anterior será con tecnología API REST y funcionará mediante una petición GET, éste no necesita parámetros, ya que solo recupera todos los datos que se encuentren dentro de la Base de Datos. Este microservicio se podra acceder mediante la URL : https://historialabordos.herokuapp.com/ObtenerHistorial

```
    //ACCEDER AL HISTORIAL DE ABORDOS
    @RequestMapping(value = "/ObtenerHistorial", method = RequestMethod.GET)
    public Iterable<Abordos> obtenerHistorial() {
        return iabordos.findAll();
    }

}
```

En este caso, como respuesta se obtendrá la lista del historial de abordos. Ya sea que este vacía o muestre los datos que estén en la base de datos.

Para realizar las pruebas se utilizó una plataforma de API llamada **postman** en donde se asigna la URL y se especifica la petición con la que fue diseñada.

Plan de pruebas

Ficha técnica

Nombre del proyecto	Servicio de localización de autobuses
Organización o área encargada	Equipo 3 del grupo Tecnologías para la integración de soluciones 78935.
Líder o gerente del proyecto	Jorge Domínguez Herrera
Líder o gerente de pruebas	Dr. José Rafel Rojano Cáceres

Aprobaciones

Nombre	Cargo	Departamento
Jorge Domínguez Herrera	Líder o gerente del proyecto	Equipo 3 del grupo Tecnologías para la integración de soluciones 78935.
Dr. José Rafael Rojano Cáceres	Maestro de la experiencia educativa	Área docente

José Ángel Gutiérrez Rodríguez	Encargado de servicio “Historial de abordos”	Equipo 3 del grupo Tecnologías para la integración de soluciones 78935.
Javier Hernández López	Encargado de servicio “Historial de abordos”	Equipo 3 del grupo Tecnologías para la integración de soluciones 78935.

Resumen ejecutivo

El presente plan de pruebas tiene como objetivo describir la metodología y recursos utilizados para asegurar la eficacia de servicio de localización de autobuses y de su infraestructura, para ello se ejecutará un plan de prueba sobre los de microservicios que lo componen, llamados “Búscame” y “Historial de abordos”.

Si bien este plan no se puede categorizar completamente como un plan detallado sí especificará el alcance de las pruebas, las funcionalidades a probar, los tipos de pruebas a realizar, criterios de aceptación, entregables, y los recursos (humanos y no humanos) a utilizar.

En todo momento se usarán recursos gratuitos o con bajo costo de operación y no se requerirá más personal humano del que forma parte del equipo y el encargado de la evaluación.

Alcance de las pruebas

Elementos de prueba

Este plan de pruebas tiene como alcance testear los siguientes elementos de prueba:

- Microservicio “Búscame” (desplegado en cuatro antenas)
- Microservicio de “Historial de abordos” (desplegado en un solo servidor)
- Funcionamiento del servidor Proxy inverso (encargado de redireccionar entre antenas).

Funcionalidades para probar

Las funcionalidades por probar son las siguientes:

- Crear autobús del microservicio “Búscame” (en las cuatro antenas)
- Listar autobuses del microservicio “Búscame” (en las cuatro antenas)
- Eliminar autobús del microservicio “Búscame” (en las cuatro antenas)
- Guardar abordo del microservicio “Historial de abordos”

- Consultar historial de abordos del microservicio “Historial de abordos”

Pruebas de regresión

No se aplicarán pruebas de regresión porque se aplicará el plan de pruebas a absolutamente todas las funcionalidades.

Funcionalidades que no se prueben

De la misma forma, no habrá funcionalidades que no se prueben.

Enfoque de pruebas (estrategia)

Pruebas funcionales

En el microservicio “Búscame” se probarán los siguientes criterios:

Funcionalidad	Criterio de aprobación	Dónde se va a aprobar	Con qué datos se va a probar
Crear autobús	Añade autobús a la llista y lo elimina de la antena anterior en caso de existir. Sin embargo, en caso de tener ID repetido no añade el autobús a la lista.	En las 4 antenas donde corre el servicio búscame.	Se probará con dos autobuses que tengan el mismo ID, así como con un autobús que no esté en la antena anterior y con un autobús que sí esté en una antena anterior.
Listar autobuses	Regresa la lista completa de los autobuses que están actualmente conectados a la antena donde se hizo la petición.	En las 4 antenas donde corre el servicio búscame.	Se probará con una antena que no tenga autobuses conectados y con otra antena que sí tenga autobuses conectados.
Eliminar autobús	Elimina el autobús con el ID indicado en la petición; en caso de no existir el ID, la antena	En las 4 antenas donde corre el servicio búscame.	Se va a probar con un ID que este activo en la lista de autobús, con uno que no lo esté y

	debe regresar true y no deben surgir errores.		con un ID que se encuentre repetido.
Guardar abordo	Registra cuando se aborda un autobús y en qué antena se realizó el abordo siempre que los datos requeridos estén completos.	Servidor donde se encuentre desplegado el microservicio “Historial de abordos”.	Se probará ingresando un abordo con todos los datos requeridos y otro con datos no requeridos.
Consultar historial de abordos	Regresa una lista de abordos realizados a lo largo del uso del servicio; en el día cero deberá regresar una lista vacía.	Servidor donde se encuentre desplegado el microservicio “Historial de abordos”.	Se realizará una prueba en el día cero (sin abordos registrados) y otra con abordos ya registrados

Pruebas no funcionales

Funcionalidad	Tipo de prueba	Descripción	Criterio de aprobación
Crear autobús	Prueba de estrés	Se crearán 50 peticiones por segundo (10 veces a las que una antena recibiría)	El servidor no se deberá caer (dejar de funcionar) o tardar más de 7 segundos en dar una respuesta.
Listar autobuses	Prueba de volumen	Se buscará obtener una llista de 100 autobuses	La petición deberá ser respondida en máximo 10 segundos.

Listar autobuses	Prueba de estrés	Se crearán 50 peticiones por segundo (2 veces a las que una antena recibiría)	El sistema no deberá dejar de funcionar.
Eliminar autobús	Prueba de estrés	Se crearán 50 peticiones por segundo (10 veces a las que una antena recibiría)	El servidor no se deberá caer (dejar de funcionar) o tardar más de 5 segundos en dar una respuesta.
Guardar abordo	Prueba de carga	Se realizarán 20 peticiones por segundo	La acción se deberá ejecutar en no más de 5 segundos.
Consultar historial de abordos	Prueba de carga	Se realizarán 20 peticiones por segundo	Se deberán obtener los resultados en no más de 10 segundos.
Consultar historial de abordos	Prueba de volumen	Se realizará la petición con al menos 100 datos registrados.	Se deberán presentar todos los datos requeridos en no más de 10 segundos.

Criterios de inicio y suspensión de pruebas

Criterios de inicio

- Los microservicios “Búscame” y “Historial de abordos” deberán estar completamente desarrollados.
- El microservicio “Búscame” deberá estar desplegado en cuatro antenas.
- El microservicio “Historial de abordos” deberá estar conectado a una base de datos completamente funcional.
- Debe existir un microservicio de proxy inverso para redirigir entre las antenas que tienen implementado el microservicio “Búscame”.

- Deberá existir un script de automatización de pruebas.
- Los microservicios deberán contar con documentación que permita tener ejemplos de consumo.

Criterios de suspensión

- Se suspenderá la ejecución de pruebas si se suscitara la caída de cualquier servidor o exceso de peticiones permitidas.
- Se suspenderá la ejecución de pruebas si no se recibe respuesta después de 13 segundos de haber lanzado una petición (independientemente servicio y/o servidor).
- Se suspenderá la ejecución de pruebas si el servicio de proxy inverso se encuentra fuera de funciones.
- Se suspenderá la ejecución de pruebas si existe mal funcionamiento del script de automatización.
- Se suspenderá la ejecución de pruebas si se cambia la forma de consumo de algún servicio.

Entorno de ejecución de pruebas

Las pruebas se ejecutarán en un entorno cerrado y con acceso limitado solo a los interesados en ejecutar las pruebas, se realizará mediante scripts que permitan la automatización de las peticiones o uso de Chroniums u otra tecnología de automatización de pruebas y el equipo de cómputo donde se ejecutarán dichos scripts tendrán una conexión estable a internet.

Por el momento, no se contempla la ejecución de pruebas en un ambiente móvil o con conexión inestable a internet. También se controlará el ingreso de comida o bebida que pueda dañar el equipo durante la ejecución de las pruebas.

Matriz RACI

Matriz RACI para la ejecución de pruebas del "Servicio de localización de autobuses"

C	Consultado
R	Responsable
I	Informado

Funcionalidad para probar	Jorge Dguez.	Dr. Rojano	Javier Hdz.	Ángel Gtz.
Crear autobús	R	I	C	C
Listar autobuses	R	I	C	C
Eliminar autobús	R	I	C	C
Guardar abordo	C	I	R	R
Consultar historial de abordos	C	I	R	R

Cronograma

Cronograma de ejecución de pruebas

Funcionalidad	5/6/2022	6/6/2022	7/6/2022
Crear autobús			
Listar autobuses			
Eliminar autobús			
Guardar abordo			
Consultar historial de abordos			

Días de ejecución

Ejecución:

Para ejecutar el proyecto se ha puesto a disposición de los desarrolladores los archivos Dokcerfile utilizados para contener las aplicaciones, estos se encuentran en el repositorio oficial del proyecto, en GitHub y cada microservicio cuenta con su archivo es cual debe ser ejecutado en una máquina que tenga el sistema de contenedores “Docker” instalado. En ese mismo sentido, los archivos de Docker del microservicio “Búscame” y el microservicio “Historial de abordos” necesitarán un archivo JAR para generar el contenedor, el cual se encuentra dentro del mismo repositorio en la ruta relativa “/target” partiendo desde las carpetas donde se ubican los archivos de Docker. Por otro lado, el microservicio de proxy inverso que redirige entre las antenas requiere un archivo llamado “proxy.conf” el cual se encuentra en la misma carpeta donde está el Dockerfile.

De manera simple, usted puede usar la línea de comandos de Heroku para desplegar las aplicaciones en el sitio de hosting anteriormente mencionado. Usted solo deberá posicionarse en la carpeta donde se ubica el Dockerfile a ejecutar, abrir la línea de comandos y escribir las siguientes instrucciones:

- heroku container:login
- heroku container:push
- heroku container:release

De la forma anteriormente mencionada usted podrá ejecutar la aplicación de una manera sencilla y simple, también está la opción de crear la imagen correspondiente a partir del Dockerfile y ejecutarla en el servidor de su preferencia.