

TECNOLÓGICO DE COSTA RICA



ESCUELA DE INGENIERÍA EN COMPUTACIÓN

IC8041 – Desarrollo de aplicaciones para dispositivos móviles

Proyecto 1 – Bus Urban System

ELABORADO POR:

JUAN JOSÉ ARAYA CASTRO	2015138766
MAURO MENDEZ MORA	2015146966
FAURICIO NAVARRO FUENTES	2015183026

PROFESOR:
Andrei Fuentes Leiva

I SEMESTRE
CARTAGO, 2018.

Propósito de la Aplicación	3
Funcionalidades	4
Wireframes	4
Descripción de diseño de alto nivel	9
Descripción de los web services	10
Interacción con sistemas externos	11

Este documento tiene como función el documentar externamente el proyecto Bus Urban System (BUS). Dicho trabajo se realizó como proyecto #1 del curso de Desarrollo de aplicaciones móviles en el Instituto Tecnológico de Costa Rica con el profesor Andrei Fuentes.

Propósito de la Aplicación

El proyecto BUS nació de una necesidad que tenemos en un país como Costa Rica, donde los buses no tienen un horario determinado, están atrasados por el tráfico o no se conoce la ruta del bus en particular. Nuestro grupo de trabajo quiso desarrollar lo que viene hacer una solución a estos problemas, implementando una aplicación móvil en el cual el usuario final pueda observar una ruta de bus específica, pueda observar cuáles son las paradas, pueda ubicarse a sí mismo en el mapa para poder llegar a una parada cercana y que además pueda ver por donde viene el bus más cercano.

Para describir la aplicación, el proyecto consiste en un sistema en el cual el usuario del bus o los choferes tendrán que identificarse en la app (loguearse). Para estos dos tipos de usuarios se mostrarán pantallas diferentes, el usuario del bus podrá navegar entre las rutas disponibles y al entrar en una se abrirá un mapa mostrando la ruta y todo lo antes mencionado. El chofer de bus entrará, encontrará las rutas desplegadas como el usuario de bus; sin embargo, al entrar en una, le dará la opción de iniciar una ruta y finalizar la misma. El iniciar una ruta le permitirá empezar a compartir su ubicación para el usuario del bus pueda verlo y además lo dirigirá hacia una aplicación externa de navegación para que pueda ayudarse; una vez terminada la ruta podrá hacer click a terminar ruta para dejar de compartir su ubicación.

Además de los usuarios anteriores, la aplicación tendrá un tercer usuario el cual será el administrador del proyecto, el cual tiene el rol de mantener los datos dentro de la aplicación. Podrá agregar rutas, choferes, paradas, entre otros; así mismo, podrá consultarlas, actualizarlas o eliminarlas. Para estas funcionalidades, la aplicación mostrará las pantallas correspondientes.

Los requerimientos a seguir en la elaboración del proyecto son los siguientes:

1. La aplicación deberá ser desarrollada en Android nativo, ya sea Java o Kotlin.
2. Se deberá implementar un backend propio, que no sea BaaS, y que sea accesible por dominio público.
3. El backend mencionado en el punto anterior, deberá tener un backoffice web de administración con su propia autenticación.
4. La aplicación debe implementar "user login".
5. Así mismo, la aplicación debe manejar autenticación y autorización a nivel de API.
6. Debe conectarse con al menos dos APIs externos, sin contar el backend propio.
7. Se deberá usar una de las siguientes características del teléfono: GPS, acelerómetro, cámara, compás.
8. El app debe tener al menos 15 activities.
9. Debe manejar métricas de uso con Mixpanel y Fabric.
10. Los apartados tratados en el propósito y funcionalidades de la aplicación se deben cumplir exitosamente.

Funcionalidades

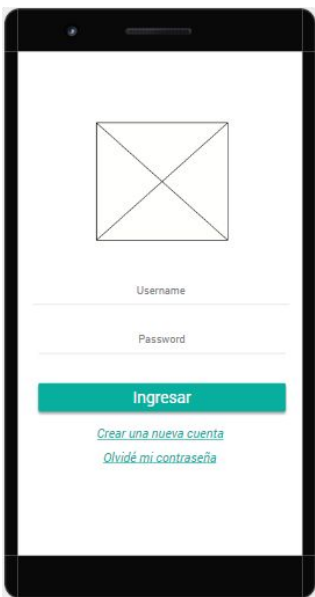

Como se mencionó en el apartado anterior, las funcionalidades de la aplicación dependerán del usuario que esté utilizando la aplicación. Sin embargo, al iniciar el proyecto todos los usuarios tendrán la función de **iniciar sesión**, ya sea con el sistema de logueo que implementamos o con una cuenta de Google.

Seguidamente, tanto el chofer de bus como el usuario del bus podrán **visualizar las empresas de bus, las rutas y las paradas** disponibles en la aplicación. Después de esto las funcionalidades de estos dos usuarios se diferencian: el chofer de bus tendrá la funcionalidad de **ingresar en una ruta**, para poder **iniciar la ruta y terminar la misma**. El iniciar la ruta lo que hará será empezar a compartir la ubicación del bus a lo largo del trayecto y también abrirá una aplicación de navegación externa. El terminar la ruta, parará el compartir de la ubicación del bus para permitir que el bus inicie otra ruta. Para el usuario de bus al **ingresar en una ruta**, la aplicación le mostrará un mapa con la ubicación del cliente, la ubicación del chofer, la ubicación de cada parada en la ruta y la demarcación de la ruta desde principio hasta su final.

Por último, se encuentran una serie de funcionalidades para dar mantenimiento a los datos de la aplicación, para esto el usuario administrador, podrá ingresar a la aplicación haciendo una autenticación y podrá **crear, consultar, modificar y eliminar** los respectivos datos de **empresas de bus, rutas, paradas y choferes**.

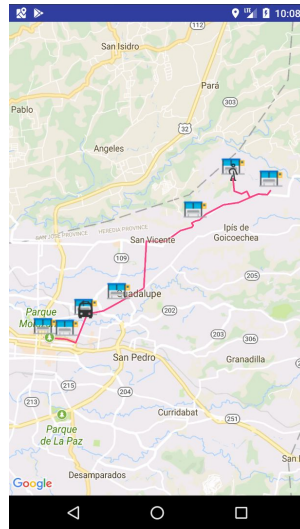
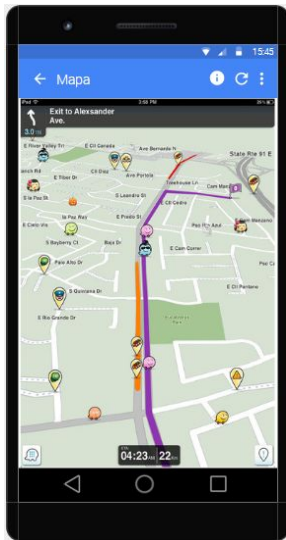
Wireframes

A continuación, se presentarán las interfaces que tendrá la aplicación. Se comparará los wireframes que se habían prediseñado con los implementados realmente.

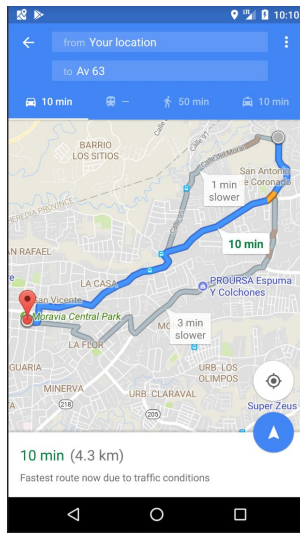
Prediseñado	Real	Descripción
		Pantalla de inicio de sesión: Se ingresa el nombre de usuario y contraseña para poder ingresar al app. La aplicación cuenta con 3 diferentes tipos de usuarios: Administrador , chofer y cliente. Además, la pantalla cuenta con la opción de crear una nueva cuenta y recuperar la contraseña. Se le agregó el logueo con Google.

		<p>Pantalla que se muestra al dar click en la opción de crear una nueva cuenta. Para poder registrarse el usuario debe ingresar el nombre, ubicación , nombre de usuario y contraseña.</p>
		<p>Esta pantalla se utiliza cuando el usuario selecciona la opción de “Olvidé mi contraseña”. Los requisitos para recuperar la contraseña son el correo electrónico y el nombre de usuario</p>
		<p>Esta pantalla sería la utilizada por el administrador la cual puede observar tanto choferes, empresas, rutas y paradas que se despliega en la parte inferior de la pantalla con la posibilidad de agregar, editar o eliminar alguno de todas las posibles opciones mencionadas anteriormente.</p>

		<p>Pantallas para agregar: Esta pantalla permite ingresar la información de un nuevo dato a la base de datos, existirá una pantalla para agregar usuario, empresa y ruta. Ya que son pantallas con estructura similar simplemente con un cambio en los campos para ingresar información se omitieron.</p>
		<p>Pantalla par modificar: Esta pantalla permite modificar un dato ya existente en la base de datos, existirá una pantalla para modificar chofer, usuario, empresa y ruta. Ya que son pantallas con estructura similar simplemente con un cambio en los campos para la información existente se omitieron.</p>
		<p>Como usuario cliente y chofer se muestra la lista de las empresas registradas y al seleccionarlasy se desplegará las rutas que están registradas en esa empresa. Así mismo, se hará una lista de las rutas y de las paradas.</p>



Esta pantalla corresponde a la pantalla cliente en la cual el cliente puede ver la ubicación actual de los buses y tener así una idea de cuánto falta para que llegue a la parada más cercana.



Esta pantalla corresponde a la pantalla de chofer que se le mostrará la ruta que debe seguir o al menos la que ya selecciono con sus respectivas paradas.



Esta es una pantalla que tuvimos que agregar para que el chofer pueda indicar cuando empieza o termina una ruta.

Backoffice

username

password

Login

Olvidaste tu contraseña

Soporte

Django administration

Email:

Password:

Log in

HomeChoferesUsuariosEmpresasRutasParadas

Home

Choferes

Usuarios

Empresas

Rutas

Paradas

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

BUSDRIVER

Bus drivers

+ Add

Change

BUSEMPRESA

Bus empresas

+ Add

Change

BUSPARADA

Bus paradas

+ Add

Change

BUSRUTA

Bus rutas

+ Add

Change

BUSUSER

Bus users

+ Add

Change

HomeChoferesUsuariosEmpresasRutasParadas

Choferes

Nombre	Empresa	Placa	Editar	Eliminar
Mauro	Buses Coronado	11111111		
Juan Jose	Buses Alajuela	22222222		
Faustino	Buses Cartago	33333333		

Select bus empresa to change

Action:

Go

0 of 2 selected

☐ BUS EMPRESA

☐ Lumaca sa

☐ TUASA

2 bus empresas

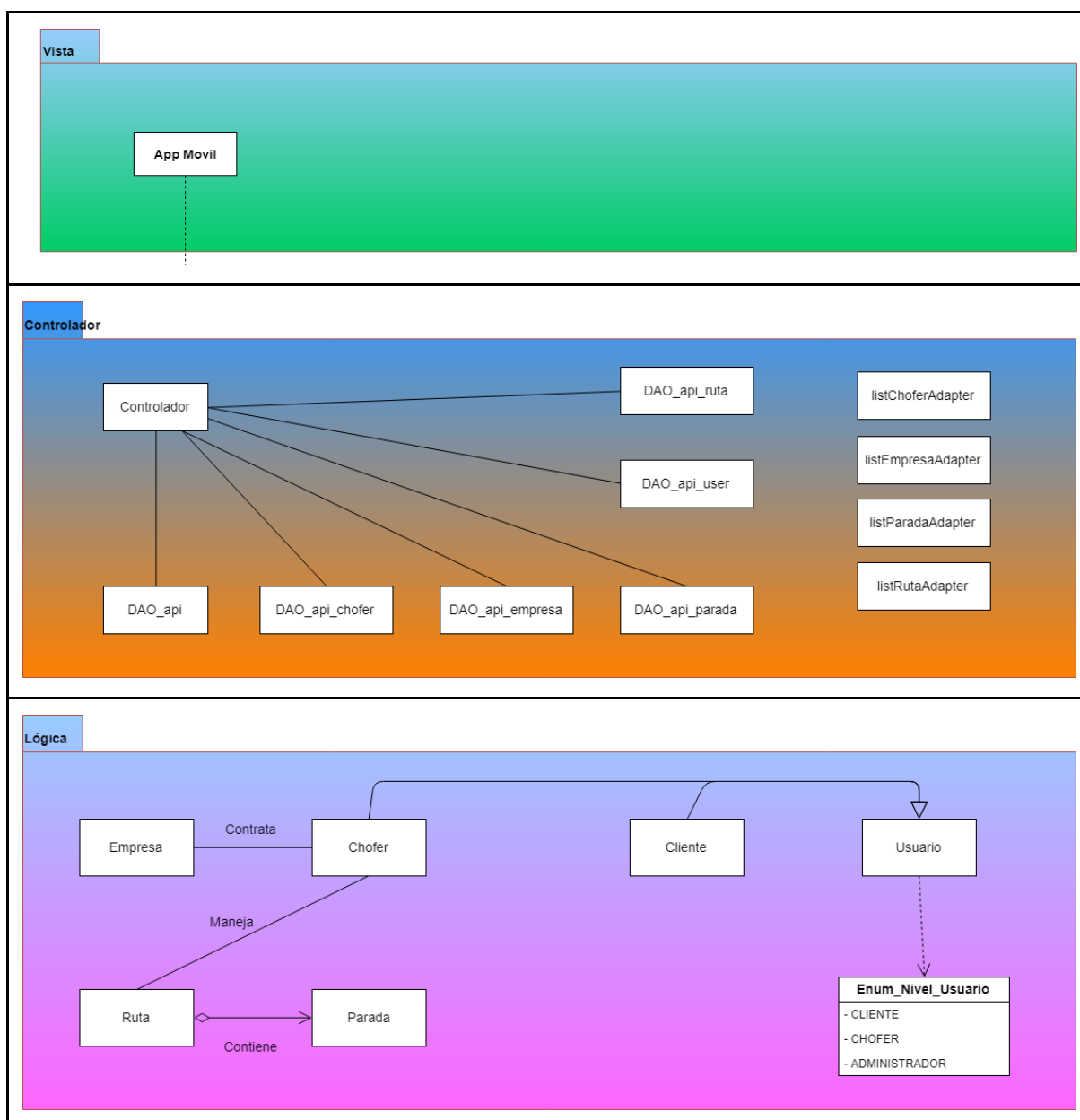
Nota Aclaratoria: Se omitieron pantallas (activities) que son esencialmente las mismas pero con diferentes datos. Por ejemplo: Visualizar empresas y visualizar choferes. En total se realizaron 22 pantallas.

Descripción de diseño de alto nivel

En la siguiente sección se presentará un diagrama a alto nivel que muestra cómo está organizada la aplicación internamente. El componente de Vista contiene todos los archivos correspondientes a Android, al no tener relación entre ellos, los representamos como una entidad que utiliza el backend.

El componente de Controlador contiene las entidades para representar la Lógica con la Vista, por lo tanto, utilizando los datos guardados, rellena la aplicación móvil visualmente. También está encargado de recibir las peticiones del usuario desde la Vista para mantener la base de datos actualizada.

El componente de Lógica contiene las identidades que representan los datos de manera persistente, este componente contiene las relaciones y atributos que se almacenan en la base de datos.



Descripción de los web services

Para el API se utilizó Django como la plataforma de la aplicación debido a la facilidad que ofrece a la hora de realizar el Backoffice, nuestra aplicación utiliza distintas aplicaciones que ofrece Django como lo fue Rest Framework para implementar la el token a los usuarios ya que este Rest Framework permite el realizar el bloqueo de la información por token, pero esta información se logró Tokenizar para los usuarios como ejemplo para crear un usuario se puede hacer desde:

<https://bus-api-moviles.herokuapp.com/api/user/create/>

Se requiere primer nombre, primer apellido, email y una contraseña para registrarlo en nuestra base de datos como se puede observar en las siguientes imágenes:

User Create OPTIONS

Use this endpoint to register new user.

POST /api/user/create/

HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "first_name": "Andrei",
  "last_name": "Fuentes",
  "email": "andreifu@example.com",
  "id": 17
}
```

Raw data HTML form

First name

Last name

Email

Password

POST

Contraseña: Moviles123

Luego de loguearse para obtener el token se requiere ingresar las credenciales de acceso en el siguiente link:

<https://bus-api-moviles.herokuapp.com/api/user/login/>

si se ingresan credenciales correctos se obtendrá una vista como:

The screenshot shows a web interface for an API endpoint titled 'Obtain Json Web Token'. At the top, there is a breadcrumb 'Api Root / Obtain Json Web Token'. The main heading is 'Obtain Json Web Token' with an 'OPTIONS' button to its right. Below the heading, a description states: 'API View that receives a POST with a user's username and password.' and 'Returns a JSON Web Token that can be used for authenticated requests.' A light blue box indicates the method and path: 'POST /api/user/login/'. Below this, the response details are shown: 'HTTP 200 OK', 'Allow: POST, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON object:

```
{  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxNywidXN1cm5hbWUiOiJhbmRyZWlmdUBleGFtcGxlIiwiaWF0Ij0iMTUzIiwiaXNjaWkiOiJ1b3RlciJ9"}
```

. At the bottom, there are two tabs: 'Raw data' (selected) and 'HTML form'. The 'HTML form' tab contains a login form with 'Email' and 'Password' labels, input fields, and a 'POST' button.

En caso de no ser correctos o estar vacíos devolvieron un json con el error y como se puede observar en la imagen de arriba tiene HTTP 200 OK de correcto en caso de error devolverá HTTP 400, una vez que tengamos el token existe un link para observar la información del usuario pero siempre desde web se verá de esta manera:

<https://bus-api-moviles.herokuapp.com/api/user/view/>

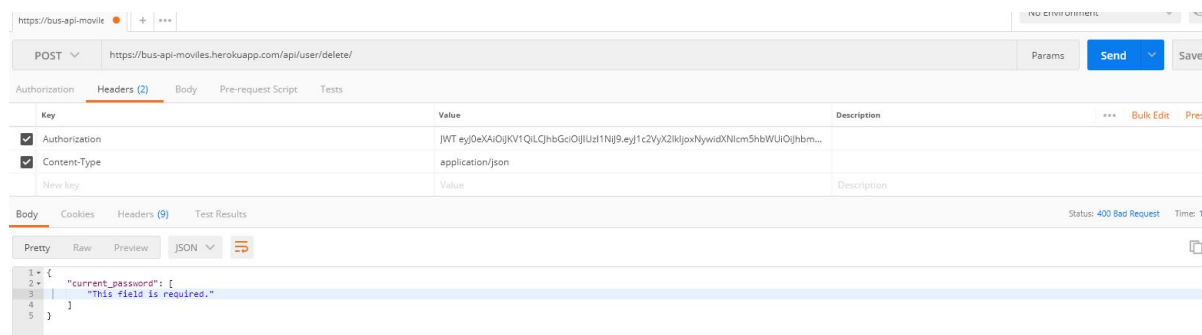


Como se puede observar aquí también permite el update pero se debe enviar el formato del json completo ahora se verá un ejemplo desde Postman con el token anterior:



Como se puede observar en la imagen esta vez nos devolvió un HTTP 200 OK en vez del error desde web que es HTTP 401 Unauthorized tal y como se solicitó, para el caso de delete existe el link:

<https://bus-api-moviles.herokuapp.com/api/user/delete/>



Se puede observar que debe ser POST para eliminar con el current_password además del Token válido actualmente en caso de que el Token sea inválido y la contraseña correcta no realizará el delete

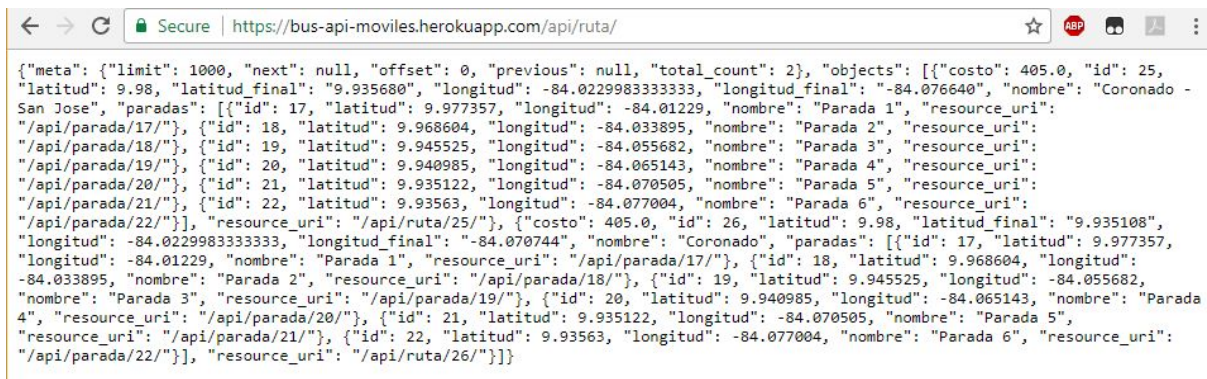
Por el motivo de que se encuentra Tokenizada esa información no era posible realizar listas de estas pero sin Tokenizar es accesible por:

<https://bus-api-moviles.herokuapp.com/api/users/>

Cabe destacar que en el link hay un formulario autogenerado por Rest Framework para “registrar usuario” con todos los atributos que este posee pero son más necesarios para el Backoffice que importantes para el usuario, el formulario registrará el usuario tal y como lo hace el link primeramente visto.

Luego de esto la implementación del Chofer, Empresa, Parada y Ruta se utilizó una aplicación llamada Tastypie que simplifica el update, delete, create y el generador de listas a un simple link tal y como por ejemplo para Rutas:

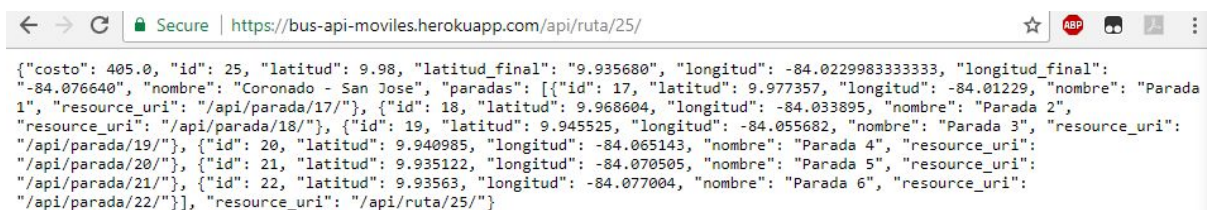
<https://bus-api-moviles.herokuapp.com/api/ruta/>



```
{
  "meta": {
    "limit": 1000,
    "next": null,
    "offset": 0,
    "previous": null,
    "total_count": 2,
    "objects": [
      {
        "costo": 405.0,
        "id": 25,
        "latitud": 9.98,
        "latitud_final": "9.935680",
        "longitud": -84.02299833333333,
        "longitud_final": "-84.076640",
        "nombre": "Coronado - San José",
        "paradas": [
          {
            "id": 17,
            "latitud": 9.977357,
            "longitud": -84.01229,
            "nombre": "Parada 1",
            "resource_uri": "/api/parada/17/"
          },
          {
            "id": 18,
            "latitud": 9.968604,
            "longitud": -84.033895,
            "nombre": "Parada 2",
            "resource_uri": "/api/parada/18/"
          },
          {
            "id": 19,
            "latitud": 9.945525,
            "longitud": -84.055682,
            "nombre": "Parada 3",
            "resource_uri": "/api/parada/19/"
          },
          {
            "id": 20,
            "latitud": 9.940985,
            "longitud": -84.065143,
            "nombre": "Parada 4",
            "resource_uri": "/api/parada/20/"
          },
          {
            "id": 21,
            "latitud": 9.935122,
            "longitud": -84.070505,
            "nombre": "Parada 5",
            "resource_uri": "/api/parada/21/"
          },
          {
            "id": 22,
            "latitud": 9.93563,
            "longitud": -84.077004,
            "nombre": "Parada 6",
            "resource_uri": "/api/parada/22/"
          }
        ]
      },
      {
        "costo": 405.0,
        "id": 26,
        "latitud": 9.98,
        "latitud_final": "9.935108",
        "longitud": -84.02299833333333,
        "longitud_final": "-84.070744",
        "nombre": "Coronado",
        "paradas": [
          {
            "id": 17,
            "latitud": 9.977357,
            "longitud": -84.01229,
            "nombre": "Parada 1",
            "resource_uri": "/api/parada/17/"
          },
          {
            "id": 18,
            "latitud": 9.968604,
            "longitud": -84.033895,
            "nombre": "Parada 2",
            "resource_uri": "/api/parada/18/"
          },
          {
            "id": 19,
            "latitud": 9.945525,
            "longitud": -84.055682,
            "nombre": "Parada 3",
            "resource_uri": "/api/parada/19/"
          },
          {
            "id": 20,
            "latitud": 9.940985,
            "longitud": -84.065143,
            "nombre": "Parada 4",
            "resource_uri": "/api/parada/20/"
          },
          {
            "id": 21,
            "latitud": 9.935122,
            "longitud": -84.070505,
            "nombre": "Parada 5",
            "resource_uri": "/api/parada/21/"
          },
          {
            "id": 22,
            "latitud": 9.93563,
            "longitud": -84.077004,
            "nombre": "Parada 6",
            "resource_uri": "/api/parada/22/"
          }
        ]
      }
    ]
  }
}
```

La herramienta permite obtener individualmente cada dato simplemente con agregar el ID luego del último slash como por ejemplo:

<https://bus-api-moviles.herokuapp.com/api/ruta/25/>



```
{
  "costo": 405.0,
  "id": 25,
  "latitud": 9.98,
  "latitud_final": "9.935680",
  "longitud": -84.02299833333333,
  "longitud_final": "-84.076640",
  "nombre": "Coronado - San José",
  "paradas": [
    {
      "id": 17,
      "latitud": 9.977357,
      "longitud": -84.01229,
      "nombre": "Parada 1",
      "resource_uri": "/api/parada/17/"
    },
    {
      "id": 18,
      "latitud": 9.968604,
      "longitud": -84.033895,
      "nombre": "Parada 2",
      "resource_uri": "/api/parada/18/"
    },
    {
      "id": 19,
      "latitud": 9.945525,
      "longitud": -84.055682,
      "nombre": "Parada 3",
      "resource_uri": "/api/parada/19/"
    },
    {
      "id": 20,
      "latitud": 9.940985,
      "longitud": -84.065143,
      "nombre": "Parada 4",
      "resource_uri": "/api/parada/20/"
    },
    {
      "id": 21,
      "latitud": 9.935122,
      "longitud": -84.070505,
      "nombre": "Parada 5",
      "resource_uri": "/api/parada/21/"
    },
    {
      "id": 22,
      "latitud": 9.93563,
      "longitud": -84.077004,
      "nombre": "Parada 6",
      "resource_uri": "/api/parada/22/"
    }
  ]
}
```

Generar un JSON con todas las rutas que existen en la base con un límite de 1000 ya que es el máximo que soporta este método de API, en este link es solo el necesario realizar POST, PUT, DELETE y GET con el JSON completo para realizar tales operaciones, la parte negativa de esto es que no se logró tokenizar esta parte del API, ya que utiliza otro método que no es el JWT TOKEN utilizado para los usuarios además de que se intentó replicar el método de token en los distintos modelos pero las plantillas que el framework traían no se lograron editar por lo que se optó por un API funcional sobre ninguno.

Para el caso del Backoffice se debe acceder a cualquiera de las siguientes rutas:

<https://bus-api-moviles.herokuapp.com>

<https://bus-api-moviles.herokuapp.com/admin>

Con las credenciales:

Correo: admin@admin.com

Contraseña: Admin123

Django administration

WELCOME, ADMIN@ADMIN.COM. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#) [Change](#)

BUSDRIVER

Bus drivers [+ Add](#) [Change](#)

BUSEMPRESA

Bus empresas [+ Add](#) [Change](#)

BUSPARADA

Bus paradas [+ Add](#) [Change](#)

BUSRUTA

Bus rutas [+ Add](#) [Change](#)

BUSUSER

Bus users [+ Add](#) [Change](#)

Recent actions

My actions

- asdas@ada.com
Bus user
- Coronado
Bus ruta
- Coronado - San Jose
Bus ruta
- Coronado
Bus ruta
- Coronado
Bus ruta
- Coronado - San Jose
Bus ruta
- Coronado
Bus ruta
- Coronado
Bus ruta
- Coronado
Bus ruta
- Coronado
Bus ruta

El cual permite la edición, eliminación y adición de información desde esta plataforma, excepto el añadir usuario ya que la contraseña no pasa por su respectiva encriptación desde esta plataforma.

Interacción con sistemas externos

En esta sección se detallaron los sistemas en el entorno del proyecto. Dichos elementos son los siguiente:

- Google:
 - Google Maps: Se utiliza el API de Google para poder realizar el detalle de la ruta, las paradas y del bus en el usuario cliente. Para el usuario chofer, se utiliza este API para ir a la pantalla de navegación y que se le indique direcciones.
 - Google Login: Se utiliza el API de Google para loguearse en la aplicación sin la necesidad de crearse una cuenta utilizando la aplicación. Sin embargo, esta funcionalidad se permitirá solo para clientes, los choferes y los administradores tendrán que usar el logeo propiciado por la aplicación.
- Heroku: Se utilizó Heroku para el alojamiento de la aplicación, de esta forma está en la nube y puede ser accesada desde cualquier lugar remotamente, utilizando la aplicación.
- Mixpanel: Se utilizó el sistema Mixpanel para tener métricas de uso en la aplicación, con este sistema podemos hacer rastreos de eventos en la aplicación y así saber estadísticas de la aplicación; por ejemplo: qué elementos visuales se usan más que otros.
- Fabric: Al igual que Mixpanel, Fabric nos permite tener métricas de uso de la aplicación y también un rastreo de los errores que se le presenten al usuario.