Plan de pruebas

Introducción:

En el presente documento se hará una especificación de las pruebas necesarias para acreditar el correcto funcionamiento del código. Estas pruebas se dividirán según sean de aceptación, de sistema, de integración y unitarias.

Diagrama de componentes orientativo

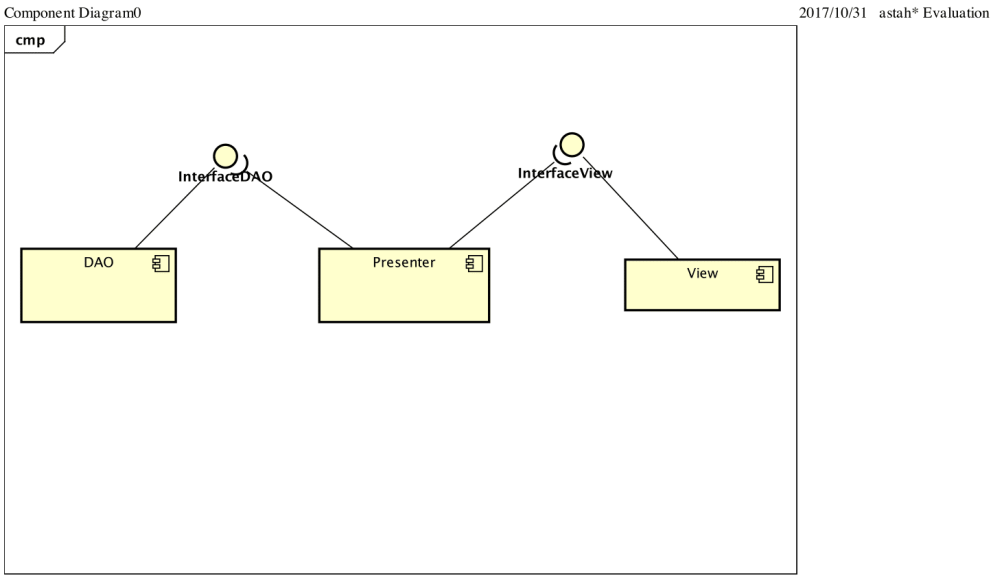
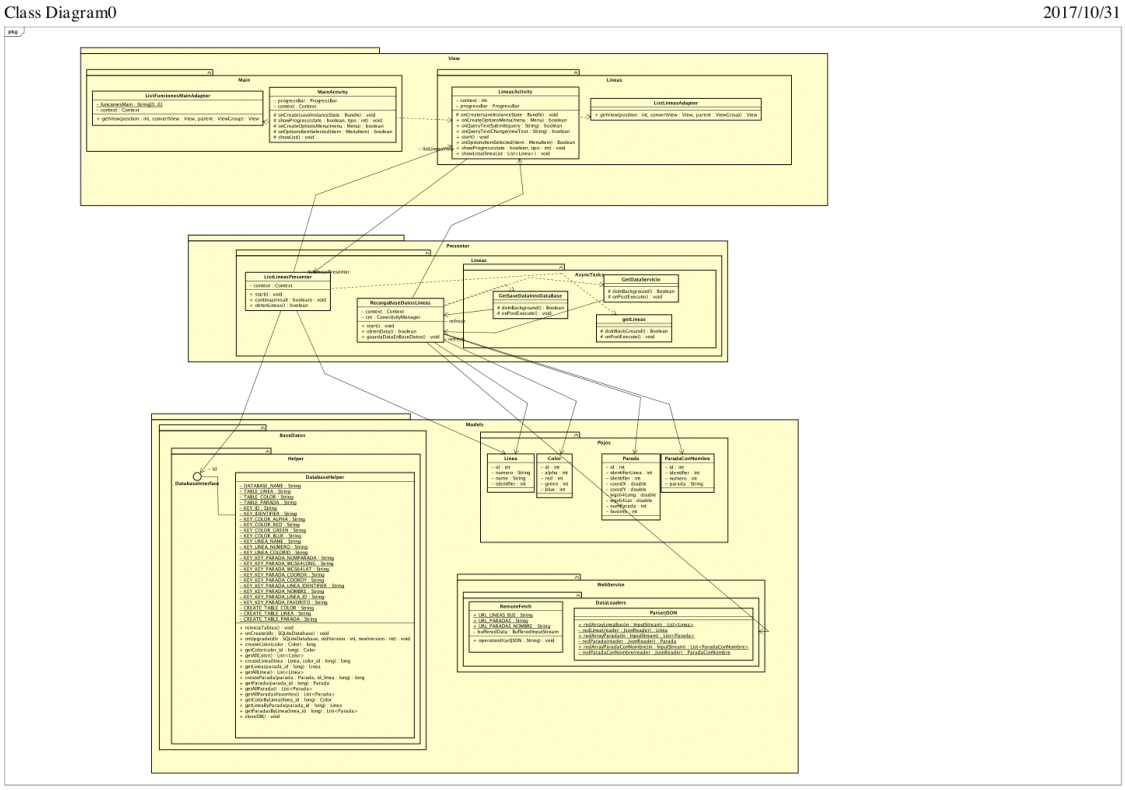


Diagrama de clases orientativo



Pruebas unitarias.

Las pruebas que realizaremos sobre la base de datos implementada será sobre la clase DatabaseHelper en la cual se realizaran las operaciones de guardado y obtención relacionadas con las líneas de buses, las paradas de buses y sobre el color relacionado a cada línea.

Se probará el método creaColor(Color) con los siguientes datos:

	Alpha	Red	Green	Blue	Resultado
C1	23	245	12	24	Su Id
C2	249	23	123	98	Su Id
C3	-1	-1	256	-1	-1
C4	2	123	123	123	Su Id

Luego se probara a obtener los colores, para el método getColor(long color_id) se probara con los resultados de la parte anterior como parámetro de este método. Para el método gelAllColor() se comprobara que la lista que retorna contiene los colores del apartado anterior.

A continuación, probaremos las líneas, se ingresarán los siguientes datos:

	Numero	Nombre	Identificador	Color	Resultado
L1	1	uno	1	C1	Su Id
L2	2	dos	2	C2	Su Id
L3	3	tres	234	C1	Su Id
L4	4	cuatro	999	C4	Su Id

Luego se probara a obtener las líneas, para el método getLinea(long linea_id) se probara con los resultados de la parte anterior como parámetro de este método. Para el método gelAllLinea() se comprobara que la lista que retorna contiene las líneas del apartado anterior.

A continuación probaremos el método getColorbyLinea(long linea_id) usando como parámetros las líneas P1, P2 y P3 y asegurándonos que retorna los colores indicados en la tabla.

Realizaremos pruebas unitarias para comprobar el correcto funcionamiento de la función de ParseJSON en el caso de obtener todas las líneas de un json y convertirlo a una lista de líneas. Para su prueba se usará un fichero que contiene lo siguiente:


```
{"summary":{"items":33,"items_per_page":1000,"pages":1,"current_page":1},"resources":[
    {
      "ayto:numero":"20",
      "dc:name":"ESTACIONES-BARRIO LA TORRE",
      "dc:modified":"2017-09-11T23:00:00.976Z",
      "dc:identifier":"20",
      "uri":"http://datos.santander.es/api/datos/lineas_bus/20.json"},
    {
      "ayto:numero":"19",
      "dc:name":"ESTACIONES-RICARDO L. ARANDA",
      "dc:modified":"2017-09-11T23:00:00.976Z",
      "dc:identifier":"19",
      "uri":"http://datos.santander.es/api/datos/lineas_bus/19.json"}]
}
```


También probaremos del ParseJSON el resto de métodos para parsear paradas de bus, para ello se contará con pequeños json para comprobar su funcionamiento, esos ficheros contienen para la prueba de las paradas:

```
},
"resources": [
  {
    "wgs84_pos:long": "-3.8700401309569807",
    "gn:coordY": "4811697.5",
    "gn:coordX": "429713.84",
    "ayto:linea": "19",
    "dc:modified": "2017-10-30T02:40:01.616Z",
    "wgs84_pos:lat": "43.45300506221107",
    "ayto:parada": "489",
    "dc:identifier": "75",
    "uri": "http://datos.santander.es/api/datos/lineas_bus_paradas/75.json"
  },
  {
    "wgs84_pos:long": "-3.788291427047982",
    "gn:coordY": "4812815.5",
    "gn:coordX": "436339.12",
    "ayto:linea": "19",
    "dc:modified": "2017-10-30T02:40:01.615Z",
    "wgs84_pos:lat": "43.46366420538056",
    "ayto:parada": "36",
    "dc:identifier": "34",
    "uri": "http://datos.santander.es/api/datos/lineas_bus_paradas/34.json"
  },
  {
    "wgs84_pos:long": "-3.7930615771148126",
    "gn:coordY": "4812737.5",
    "gn:coordX": "435952.47",
    "ayto:linea": "20",
    "dc:modified": "2017-10-30T02:40:01.615Z",
    "wgs84_pos:lat": "43.46292891213499",
    "ayto:parada": "37",
    "dc:identifier": "35",
    "uri": "http://datos.santander.es/api/datos/lineas_bus_paradas/35.json"
  }
]
```


Y para la prueba de las paradas con nombre:


```
{
  "wgs84_pos:long": "-3.866588480444409",
  "ayto:numero": "499",
  "gn:coordY": "4811045.72",
  "gn:coordX": "429986.36",
  "ayto:sentido": "Centro",
  "vivo:address1": "Avda severo Ochoa",
  "dc:modified": "2017-10-29T23:20:03.158Z",
  "wgs84_pos:lat": "43.44716242117678",
  "ayto:parada": "Camarreal Peñacastillo",
  "dc:identifier": "42063",
  "uri": "http://datos.santander.es/api/datos/paradas_bus/42063.json"
},
{
  "wgs84_pos:long": "-3.8571753633683796",
  "ayto:numero": "500",
  "gn:coordY": "4810658.9",
  "gn:coordX": "430744.13",
  "ayto:sentido": "Centro",
  "vivo:address1": "Ortega y Gasset.28",
  "dc:modified": "2017-10-29T23:20:03.158Z",
  "wgs84_pos:lat": "43.44375026592359",
  "ayto:parada": "Ortega y Gasset.28",
  "dc:identifier": "42064",
  "uri": "http://datos.santander.es/api/datos/paradas_bus/42064.json"
},
{
  "wgs84_pos:long": "-3.8062240806383896",
  "ayto:numero": "505",
  "gn:coordY": "4814192.02",
  "gn:coordX": "434901.67",
  "ayto:sentido": "Puertochico",
  "vivo:address1": "Avenida de Cantabria nº 35",
  "dc:modified": "2017-10-29T23:20:03.158Z",
  "wgs84_pos:lat": "43.47593386282703",
  "ayto:parada": "Avenida de Cantabria nº 35",
  "dc:identifier": "50693",
  "uri": "http://datos.santander.es/api/datos/paradas_bus/50693.json"
}
```

Las pruebas unitarias que realizaremos sobre ListLineasPresenter son para probar los métodos obtenLineas ya que es el que llevan toda la lógica de la clase, ya que se encarga de obtener de la base de datos local la lista de las líneas de bus este método se ejecuta de forma asíncrona no en el hilo principal por lo que probaremos también el correcto funcionamiento de ese AsyncTask.
Se comprobará su correcto funcionamiento ejecutando el método y comprobando que la lista de líneas de bus en la clase se rellena con los datos correctos.
En el caso de que todo sea correcto se llenara la lista con líneas. En caso de que se produzca algún error retornara algo indicándolo.
Para comprobar que esta funciona correctamente se supone que esta precargado con los datos de las tablas anteriores así que la lista debería contener L1, L2, L3 y L4.

Pruebas de integración.

Primero se realizarán las pruebas de integración de la capa de presentación (ListLineasPresenter) con la base de datos (DatabaseHelper) haciendo las pruebas unitarias de la capa de presentación, pero con la base de datos local, cuando hay datos y cuando está vacía que debería indicarlo.

A continuación, se probará el componente RecargaBaseDatosLineas, para ello se probara la recarga que realiza sobre la base de datos, primero se probará en la situación en la que debería funcionar sin ningún error, y luego se le quitará la conexión a internet para comprobar que se produce un error.

Pruebas de aceptación.

Prueba 1: Comprobar líneas

1. El usuario selecciona la función Mostrar líneas en el menú principal.
2. El sistema obtendrá las líneas de la base de datos local.
3. Se mostrará un listado de todas las líneas, con el formato correcto.
4. Se hará click en una línea y se comprobará que lleva a la actividad de sus paradas.

Prueba 2: Recargar base de datos.

1. El usuario selecciona la función Mostrar línea en el menú principal.
2. El usuario selecciona la función de recargar la base de datos.
3. Se mostrará indicadores para mostrar que se está ejecutando, se notificara cuando termine y se mostraran los nuevos datos obtenido.

Prueba 3: Base de datos vacía.
1. El usuario selecciona la función Mostrar líneas en el menú principal.
2. Se intentan cargar los datos, pero al estar vacía, se notifica y se recomienda actualizarla.