

```
%esperscenario

ParkingEvent = {idMetre = 1, value = 1}
ParkingEvent = {idMetre = 2, value = 1}
ParkingEvent = {idMetre = 3, value = 0}
ParkingEvent = {idMetre = 4, value = 1}
ParkingEvent = {idMetre = 5, value = 0}

t=t.plus(1 second)

ParkingEvent = {idMetre = 1, value = 1}
ParkingEvent = {idMetre = 2, value = 0}
ParkingEvent = {idMetre = 3, value = 0}
ParkingEvent = {idMetre = 4, value = 1}
ParkingEvent = {idMetre = 5, value = 1}

t=t.plus(30 seconds)
```

Figura 56. Escenario creado en Esper Notebook

Creado el escenario, y por supuesto definidos antes los esquemas y patrones de eventos en Esper EPL, ejecutamos todo y obtenemos los eventos complejos que se generan. Podemos ver un ejemplo de los eventos generados en la Figura 57.

| Time | Statement | Stream | Output Event |
|-------------------------|---------------------------|--------|---------------------------------------------------------------|
| 1970-01-01 00:00:00.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18000000, idMetre=1} |
| 1970-01-01 00:00:00.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18000000, idMetre=2} |
| 1970-01-01 00:00:00.000 | FreeParkingSpot | Insert | FreeParkingSpot={timestamp=18000000, idMetre=3} |
| 1970-01-01 00:00:00.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18000000, idMetre=4} |
| 1970-01-01 00:00:00.000 | FreeParkingSpot | Insert | FreeParkingSpot={timestamp=18000000, idMetre=5} |
| 1970-01-01 00:00:01.000 | TotalFreeParkingSpots | Insert | TotalFreeParkingSpots={timestamp=18001000, totalPlazas=2} |
| | TotalOccupiedParkingSpots | Insert | TotalOccupiedParkingSpots={timestamp=18001000, totalPlazas=3} |
| 1970-01-01 00:00:01.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18001000, idMetre=1} |
| 1970-01-01 00:00:01.000 | FreeParkingSpot | Insert | FreeParkingSpot={timestamp=18001000, idMetre=2} |
| 1970-01-01 00:00:01.000 | FreeParkingSpot | Insert | FreeParkingSpot={timestamp=18001000, idMetre=3} |
| 1970-01-01 00:00:01.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18001000, idMetre=4} |
| 1970-01-01 00:00:01.000 | OccupiedParkingSpot | Insert | OccupiedParkingSpot={timestamp=18001000, idMetre=5} |

Figura 57. Eventos complejos generados en Esper Notebook

De esta manera y cambiando los escenarios, cambiando el tipo de evento simple generado en ellos así como el tiempo entre un suceso y otro, se comprobó durante la primera iteración el correcto funcionamiento de todos los patrones.

8.3.2. Mule ESB

Sobre la aplicación de Mule ESB también hemos ido realizado pruebas unitarias de cada componente que añadíamos a los distintos flujos, tanto en la segunda iteración descrita en la sección 3.2.2 en la que se implementaron los cuatro flujos que tiene la aplicación, como en la cuarta iteración descrita en sección 3.2.4 donde se realizaron ciertas mejoras en algunos flujos.

Las distintas pruebas que se han realizado son:

- **Recepción de eventos simples**

Mediante el componente Logger, podemos ver por pantalla que se reciben los eventos simples correctamente desde la conexión con RabbitMQ usando el protocolo AMQP. Se puede ver un ejemplo de esto en la Figura 58.

```
.LoggerMessageProcessor: Mensajes{"eventName": "ParkingEvent", "idMetre": 5, "value": 1}
```

Figura 58. Logger de recepción de eventos simples

- **Despliegue de esquemas y patrones en el motor Esper**

En el mismo componente de Esper desarrollado en Java está definido que si un patrón se despliega correctamente, se imprima por consola un mensaje descriptivo, que se puede observar en la Figura 59 y en la Figura 60.

```
*** Adding new EPL @public @buseventtype create json schema TrafficJamEvent (idMetre int, speed double, carPlate string);
```

Figura 59. Logger de despliegue de esquema sobre motor CEP

```
*** Adding new EPL @public INSERT INTO FreeParkingSpot SELECT current_timestamp() as timestamp, idMetre FROM ParkingEvent.win:time(1 seconds) WHERE value = 0;
```

Figura 60. Logger de despliegue de patrón sobre motor CEP

- **Generación de eventos complejos**

Al igual que antes, en el componente de Esper desarrollado en Java, está implementado que si se genera un evento complejo se imprima por consola un mensaje descriptivo, que se puede observar en la Figura 61.

```
** Complex event 'OccupiedParkingSpot' detected: {timestamp=1674636842549, idMetre=5}
```

Figura 61. Logger de generación de evento complejo en motor CEP

- **Consumo, envío y almacenamiento de eventos complejos**

Como hacíamos para probar la recepción de eventos simples, aquí mediante un componente Logger mostramos por consola un mensaje descriptivo si consumimos un evento complejo, como podemos ver en la Figura 62.

```
LoggerMessageProcessor: SALIDA CONSUMIDOR EVENTOS COMPLEJOS ***** {SpeedOver15={carPlate=NVu73S0, idMetre=15, speed=78.0, timestamp=1674636842658}}
```

Figura 62. Logger de consumo de evento complejo por flujo de Mule ESB

Para comprobar después que se han enviado correctamente a la cola de mensajería de RabbitMQ solo deberemos obtener los mensajes recibidos en la cola, como se ve en la Figura 63.

Message 1

The server reported 358 messages remaining.

| | |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Exchange | outputMule |
| Routing Key | |
| Redelivered | |
| Properties | <ul style="list-style-type: none">timestamp: 1674636842message_id: d0c003c0-9c8d-11ed-b05b-0a0027000003reply_to: amq.gen--YSJqFrPLUEOmvyAfm02-wpriority: 0delivery_mode: 2headers: MULE_CORRELATION_GROUP_SIZE: -1 MULE_CORRELATION_SEQUENCE: -1 MULE_ENDPOINT: amqp://outputMule/amqp-queue.OutputMessages MULE_ROOT_MESSAGE_ID: d0c003c0-9c8d-11ed-b05b-0a0027000003 MULE_SESSION: r00ABXNyACNvcmbXVsZ5S5ZkNzaW9uLkRlZmF1bHRNdWxluZVZvc2lubi7rdtEW7GGKAwAFWgAFdmFsaWRMAA1mbG93Q29uc3RydWNoAAmTG9yZy9tdWxluZVZwaS9jb25zdHJ1Y3QvRmxvd0NvbnN0cnVjdDdtMAAJpZHQAEkxqYX |

content_encoding: UTF-8

Payload
113 bytes
Encoding: string

{OzoneAlert={nitrogenDioxideValue=8, ozoneValue=196, idMetre=24, sulfurDioxideValue=91, timestamp=1674636842456}}

Figura 63. Mensaje recibido en cola de RabbitMQ

Por último, para probar que han sido almacenados en la base de datos, solo deberemos hacer la consulta *SQL SELECT * FROM <tabla de la base de datos>;*

En la Figura 64 podemos comprobar cómo han sido almacenados correctamente los eventos complejos:

```
MariaDB [cadizrespiraeventreport]> select * from polucionevents;
```

| id | polucionEvent |
|----|---------------------------------------------------------------------------------------------------------------------------|
| 1 | {SulfurDioxideAlert={nitrogenDioxideValue=7, ozoneValue=110, idMetre=23, sulfurDioxideValue=63, timestamp=1674636842751}} |
| 2 | {OzoneAlert={nitrogenDioxideValue=5, ozoneValue=192, idMetre=21, sulfurDioxideValue=62, timestamp=1674636842549}} |
| 3 | {OzoneAlert={nitrogenDioxideValue=0, ozoneValue=108, idMetre=24, sulfurDioxideValue=32, timestamp=1674636842456}} |
| 4 | {SulfurDioxideAlert={nitrogenDioxideValue=4, ozoneValue=178, idMetre=22, sulfurDioxideValue=79, timestamp=1674636842845}} |
| 5 | {OzoneAlert={nitrogenDioxideValue=6, ozoneValue=130, idMetre=21, sulfurDioxideValue=94, timestamp=1674636842658}} |
| 6 | {SulfurDioxideAlert={nitrogenDioxideValue=6, ozoneValue=130, idMetre=21, sulfurDioxideValue=94, timestamp=1674636842658}} |
| 7 | {SulfurDioxideAlert={nitrogenDioxideValue=5, ozoneValue=192, idMetre=21, sulfurDioxideValue=62, timestamp=1674636842549}} |
| 8 | {OzoneAlert={nitrogenDioxideValue=4, ozoneValue=178, idMetre=22, sulfurDioxideValue=79, timestamp=1674636842845}} |
| 9 | {OzoneAlert={nitrogenDioxideValue=7, ozoneValue=110, idMetre=23, sulfurDioxideValue=63, timestamp=1674636842751}} |
| 10 | {OzoneAlert={nitrogenDioxideValue=8, ozoneValue=196, idMetre=24, sulfurDioxideValue=91, timestamp=1674636842456}} |
| 11 | {SulfurDioxideAlert={nitrogenDioxideValue=8, ozoneValue=196, idMetre=24, sulfurDioxideValue=91, timestamp=1674636842456}} |
| 12 | {SulfurDioxideAlert={nitrogenDioxideValue=8, ozoneValue=182, idMetre=24, sulfurDioxideValue=66, timestamp=1674636843048}} |
| 13 | {OzoneAlert={nitrogenDioxideValue=8, ozoneValue=182, idMetre=24, sulfurDioxideValue=66, timestamp=1674636843048}} |
| 14 | {SulfurDioxideAlert={nitrogenDioxideValue=4, ozoneValue=172, idMetre=22, sulfurDioxideValue=53, timestamp=1674636843159}} |
| 15 | {OzoneAlert={nitrogenDioxideValue=4, ozoneValue=172, idMetre=22, sulfurDioxideValue=53, timestamp=1674636843159}} |
| 16 | {SulfurDioxideAlert={nitrogenDioxideValue=0, ozoneValue=72, idMetre=24, sulfurDioxideValue=45, timestamp=1674636843253}} |
| 17 | {SulfurDioxideAlert={nitrogenDioxideValue=7, ozoneValue=54, idMetre=21, sulfurDioxideValue=74, timestamp=1674636843347}} |
| 18 | {SulfurDioxideAlert={nitrogenDioxideValue=4, ozoneValue=56, idMetre=25, sulfurDioxideValue=69, timestamp=1674636843456}} |
| 19 | {OzoneAlert={nitrogenDioxideValue=8, ozoneValue=180, idMetre=21, sulfurDioxideValue=25, timestamp=1674636843550}} |
| 20 | {SulfurDioxideAlert={nitrogenDioxideValue=4, ozoneValue=146, idMetre=21, sulfurDioxideValue=55, timestamp=1674636843657}} |
| 21 | {OzoneAlert={nitrogenDioxideValue=4, ozoneValue=146, idMetre=21, sulfurDioxideValue=55, timestamp=1674636843657}} |
| 22 | {SulfurDioxideAlert={nitrogenDioxideValue=4, ozoneValue=84, idMetre=24, sulfurDioxideValue=71, timestamp=1674636843750}} |

Figura 64. Eventos complejos almacenados en Base de Datos

8.3.3. API REST

Los métodos de la API REST, tanto los implementados en la segunda iteración (descrita en el punto 3.2.2) como los implementados en la cuarta (apartado 3.2.4), han sido probados con Postman , que permite, entre otras cosas, implementar pruebas para APIs.

Por ejemplo, en los métodos implementados en la segunda iteración como el que devolvía el número de eventos producidos, se ha comprobado que la petición es válida (código de estado 200) y que la respuesta sigue el formato correcto, además de que el número de eventos es mayor a 0. Todo esto se puede observar en la Figura 65, donde se ve el código de la prueba y el resultado obtenido.



Figura 65. Prueba de método `getNumberOfEvents` en Postman

Para los métodos implementados en la cuarta iteración, que se encargan de devolver los eventos complejos también se implementaron pruebas en Postman.

Por ejemplo, para comprobar el método que devuelve todos los eventos de aparcamiento almacenados en la base de datos, se prueba que la respuesta a la petición a la API es válida (código 200) y que el JSON que se recibe como respuesta tiene el formato esperado.

En la Figura 66 se puede ver el código de implementación de la prueba, así como abajo los resultados de la misma tras enviar la petición a la API.



Figura 66. Prueba de método `getAparcamientos` en Postman

8.3.4. Aplicación móvil

Durante el desarrollo de la aplicación, en concreto en la tercera y en la quinta, descritas en el apartado 3.2.3 y 3.2.5, respectivamente, se realizaron pruebas unitarias por cada módulo añadido a la aplicación.

Primero, en la iteración descrita en el apartado 3.2.3, donde se implementaron las distintas pantallas de la aplicación, pero centrándonos sólo en el apartado visual y en la navegación entre ellas, se probó que todos los textos e imágenes se mostraban correctamente, que desde cualquier pantalla se podía navegar a otra a través del menú lateral. También se probó que se podía acceder desde la pantalla de eventos a la lista de eventos de cada tipo (sin mostrar todavía la lista eventos reales) pulsando en cualquiera de los tres botones, y de cada elemento de la lista a la vista de información detallada del evento.

Después, en la última iteración, descrita en el apartado 3.2.5, se probaron cada una de las funcionalidades añadidas:

- Se muestran correctamente las listas de eventos de cada tipo (aparcamiento, tráfico y polución). La lista no se corta sino que se puede “scrollear” hacia abajo para verla completa.
- Se muestra de manera correcta y amigable la información detallada de cada evento complejo generado.
- El mapa de aparcamientos libres se carga correctamente, centrándose en la ciudad de Cádiz y mostrando marcadores verdes únicamente en las plazas de aparcamiento que se encuentran libres. Al pulsar en un marcador, se abre una

ventana de información que indica el identificador del sensor y la hora a la que se generó el evento señalando que no estaba ocupado. También aparecen las opciones para ver la ubicación y generar una ruta hacia la misma, que nos lleva a la aplicación de Google Maps.

- El mapa de atascos se carga también correctamente, mostrando marcadores en las zonas donde se encuentran los atascos que existen actualmente en la ciudad, y con las mismas funcionalidades que la vista anterior.

8.3.5. Pruebas de integración

Una vez se ha probado cada componente por separado, se probó el comportamiento de los componentes al combinarse entre ellos. En nuestro sistema, hay pruebas de integración que carecen de mucho sentido, ya que a lo que precisamente se dedica un ESB como el de Mule, es a integrar datos de distintas fuentes y distintos componentes. Por tanto, los módulos probados en las pruebas unitarias tenían como función precisamente integrar distintos componentes, por lo que la integración de estos ya ha sido probada, como por ejemplo:

- La aplicación Mule se conecta correctamente con una cola de mensajería de RabbitMQ para recibir los eventos simples.
- La aplicación Mule se conecta correctamente con otra cola de mensajería de RabbitMQ para enviar los eventos complejos generados.
- La aplicación Mule se conecta correctamente con una base de datos desplegada en un servidor MySQL local para ejecutar consultas de inserción de los eventos complejos generados.

Sin embargo, sí que ha habido que realizar pruebas de integración de otros componentes que no aparecen en nuestros flujos de Mule ESB. Las pruebas realizadas han verificado lo siguiente:

- La aplicación instalada en un dispositivo Android se conecta sin problema al servicio REST desplegado en el servidor del ordenador donde se ejecuta la aplicación de Mule, permitiendo realizar peticiones a la API. Para que se dé la conexión, el dispositivo móvil y el ordenador deben encontrarse en la misma red.
- El simulador nITROGEN se conecta correctamente con la cola de mensajería de RabbitMQ para enviarle los eventos simples simulados.

8.4. Verificación de la calidad

En esta sección, se explicará cómo se han comprobado los requisitos no funcionales que se recogieron en el apartado 4.2, que hablaba de la garantía de calidad.

Las verificaciones llevadas a cabo son las siguientes:

- Se han simulado desde nITROGEN eventos que no cumplían con el formato esperado y al ejecutar nuestra aplicación Mule ESB, se comprueba que son directamente descartados al no cumplir con la expresión regular del componente *Regex*.

- Los requisitos de interoperabilidad han sido comprobados al realizar las pruebas de integración en las que vimos que los distintos componentes se combinan de forma exitosa.
- En cuanto a la operabilidad, se han realizado observaciones de campos a algunos usuarios de distinto perfil y todos han podido navegar por la aplicación sin problema, ya que la interfaz se les hacía familiar.
- En cuanto a la transferibilidad, mediante el *Device Emulator* de Android Studio, que permite ejecutar la aplicación en diferentes dispositivos Android emulados, se ha podido comprobar que la interfaz se adapta distintas resoluciones de pantalla.
- Se han probado los tiempos de carga mediante la herramienta Firebase Performance Monitoring [20], un servicio que ayuda a obtener información del rendimiento de aplicaciones en tiempo real.
- Se ha llevado a cabo un análisis estático del código mediante exploración, comprobando que efectivamente estaba debidamente documentado, y era fácilmente extensible.

9. Conclusiones y trabajo futuro

En este capítulo haremos una valoración final de cómo ha ido el desarrollo del proyecto y lo que nos ha supuesto personalmente, después haremos un análisis del grado de cumplimiento de los objetivos marcados inicialmente, para acabar hablando de las posibles mejoras del mismo que se podrían realizar en un futuro.

9.1. Valoración del proyecto

La idea de este proyecto nació con la intención de crear un sistema que ayudara a monitorizar los niveles de contaminación de una ciudad, focalizado en Cádiz y abarcando sobre todo el principal causante de que se superen en la ciudad a menudo los niveles recomendados de contaminación, la movilidad.

La gran presencia de vehículos en una ciudad con una superficie pequeña hace que diariamente se produzcan atascos, que hacen que se emitan más emisiones que cuando hay una circulación normal. Además, todo esto es alimentado también por la falta de aparcamiento, que hace que muchos conductores se pasen un gran tiempo en circulación simplemente buscando aparcamientos, lo que provoca también que se contamine más de lo necesario.

Una herramienta que ayudara a monitorizar estas situaciones y así poder actuar rápidamente para reducir su impacto: disolviendo atascos, detectando niveles de polución de aire inusuales o ayudando a los conductores en la tarea de encontrar aparcamiento, mejoraría la calidad de vida de la gente a la vez que aportaría en la más que necesaria tarea de poner freno al cambio climático.

Además, el hecho de usar tecnologías emergentes y de enorme potencial como el procesamiento de eventos complejos, aporta un gran valor a esta herramienta y haría a la ciudad seguir la tendencia de las Smart Cities, implantando un modelo que podría servir después de referencia para otras ciudades.

Personalmente, este proyecto me ha aportado bastante a nivel académico, ya que, aunque se han usado herramientas y lenguajes ya conocidos como Java, Android Studio o SQL, también se ha profundizado en tecnologías como el Procesamiento de Eventos Complejos y los Buses de Servicios Empresariales que hasta hace apenas unos meses eran completamente desconocidos.

Como conclusión, este proyecto ha sido muy útil, no sólo por el aprendizaje de nuevas tecnologías y en tener una pequeña aproximación a lo que es el proceso íntegro de desarrollo de un proyecto de ingeniería, sino para ser conscientes de todo lo que pueden aportar las tecnologías y los sistemas de la información a la sociedad, y más en concreto al desarrollo de las Smart Cities y la lucha contra el cambio climático.

9.2. Cumplimiento de los objetivos propuestos

Al comienzo del proyecto, se establecieron unos objetivos concretos que debían cumplirse para dar el proyecto como acabado. Ahora, valoraremos el grado de cumplimiento de cada uno de esos objetivos, para comprobar que el proyecto ha sido ejecutado con éxito:

- **Implementar una arquitectura software que permita tratar una serie de datos relacionados con el tráfico, el aparcamiento y la contaminación atmosférica, leyendo los datos de una cola de mensajería y procesándolos en tiempo real con un motor CEP** → Este era el objetivo principal del sistema, que podemos decir que se ha cumplido satisfactoriamente. Haciendo uso de Mule ESB, recibimos una serie de eventos desde una cola de RabbitMQ, procesamos esos eventos en el motor CEP de Esper integrado en nuestro sistema, generándose eventos complejos que nos proporcionan información más valiosa.
- **Dotar al sistema de persistencia, almacenando los eventos complejos, para poder acceder en cualquier momento al registro de las situaciones detectadas** → Este objetivo también se ha logrado. Desde nuestro sistema desarrollado en Mule ESB, consumimos los eventos complejos generados en el motor CEP y por un lado almacenamos en una base datos, y por otro lado los enviamos a una cola de mensajes para que puedan ser consumidos por un sistema externo.
- **Desarrollo de una aplicación Android que permita visualizar situaciones de interés detectadas de una manera amigable para el usuario** → También ha sido logrado exitosamente este objetivo. Nuestra aplicación para dispositivos Android muestra la información extraída de los eventos complejos de manera amigable y comprensible.
- **Localizar geográficamente en un mapa de la ciudad las plazas de aparcamiento libres y los atascos existentes a través de la aplicación Android** → Este objetivo también ha sido cumplido. Haciendo uso de la API de Google Maps, mostramos los aparcamientos libres y los atascos detectados en nuestro sistema sobre el mapa de la ciudad.

9.3. Trabajo futuro

Desde un principio, se podía intuir claramente que las posibilidades que podía llegar a ofrecer un sistema como el desarrollado eran infinitas, también fue por esta precisa razón por la que se acotaron claramente los objetivos, para no intentar abarcar demasiado dada la limitación de tiempo y recursos disponibles.

- Instalación de sensores en la ciudad de manera que se sustituyan los datos simulados que actualmente recibe nuestro sistema por datos reales. Habría que instalar sensores de los tres tipos:
 - Sensores en todas las plazas de la bolsa de aparcamiento de la ciudad que detecten si hay un vehículo estacionado en ella o no.
 - Sensores similares a los radares de la DGT instalados sobre los semáforos de la ciudad, sobre todo en los puntos donde sea más habitual que se produzca un atasco.
 - Sensores repartidos por distintos puntos de la ciudad que midan los niveles de polución del aire.
- Desarrollo de aplicación para dispositivos con iOS como sistema operativo, ya que hay una gran cantidad de usuarios para los que nuestro sistema no será accesible al estar solo disponible para Android.

- Desarrollo de una aplicación web que presente también la información de los eventos a usuarios que quieran acceder desde un PC o desde el navegador web de sus dispositivos móviles sin tener que descargarse la aplicación.
- Implementación de un registro de usuarios en la aplicación, que permita que diferenciar el uso de la aplicación por parte de autoridades que por el resto de aplicación. Las primeras podrían acceder así a información de carácter confidencial que estaría oculta para el resto de usuarios.

Bibliografía

- [1] P. S. M., «Cádiz, irrespirable: mantiene dos de los diez focos de más contaminación de Andalucía», *lavozdelsur.es*, 28 de junio de 2022. Accedido: 18 de enero de 2023. [En línea]. Disponible en:
https://www.lavozdelsur.es/actualidad/ecologia/cadiz-irrespirable-mantiene-dos-diez-focos-mas-contaminacion-andalucia_279018_102.html
- [2] Ecologistas en Acción, «La calidad del aire en el Estado español durante 2021», jun. 2022. Accedido: 18 de enero de 2023. [En línea]. Disponible en:
<https://www.ecologistasenaccion.org/wp-content/uploads/2022/06/informe-calidad-aire-2021.pdf>
- [3] J. Boubeta-Puig, J. Rosa-Bilbao, y J. Mendling, «CEPchain: A graphical model-driven solution for integrating complex event processing and blockchain», *Expert Systems with Applications*, vol. 184, p. 115578, dic. 2021, doi: 10.1016/j.eswa.2021.115578.
- [4] «Esper - EsperTech». <https://www.espertech.com/esper/> (accedido 25 de enero de 2023).
- [5] MuleSoft, ¿Qué es Mule ESB? <https://www.mulesoft.com/es/resources/esb/what-mule-esb> (accedido 18 de enero de 2023).
- [6] RabbitMQ, *Messaging that just works — RabbitMQ*. <https://www.rabbitmq.com/> (accedido 18 de enero de 2023).
- [7] Amazon Web Services, ¿Qué es API RESTful? | Guía sobre API RESTful para principiantes / AWS. <https://aws.amazon.com/es/what-is/restful-api/> (accedido 18 de enero de 2023).
- [8] MySQL, *MySQL*. <https://www.mysql.com/> (accedido 18 de enero de 2023).
- [9] Android Developers, *Download Android Studio & App Tools*. <https://developer.android.com/studio> (accedido 18 de enero de 2023).
- [10] «Download the Latest Java LTS Free». <https://www.oracle.com/es/java/technologies/downloads/> (accedido 25 de enero de 2023).
- [11] MuleSoft, *Anypoint Studio | Entorno de desarrollo integrado (IDE)*. <https://www.mulesoft.com/es/platform/studio> (accedido 18 de enero de 2023).
- [12] «Eclipse Downloads | The Eclipse Foundation». <https://www.eclipse.org/downloads/> (accedido 25 de enero de 2023).
- [13] <https://ucase.uca.es/nITROGEN/> (Accedido 25 de enero de 2023).
- [14] A. Garcia-de-Prado, G. Ortiz, J. Hernández, y E. Moguel, «Generación de Datos Sintéticos para Arquitecturas de Procesamiento de Datos del Internet de las Cosas».
- [15] «Balsamiq. Rapid, Effective and Fun Wireframing Software | Balsamiq». <https://balsamiq.com/> (accedido 25 de enero de 2023).
- [16] «Postman API Platform | Sign Up for Free», *Postman*. <https://www.postman.com> (accedido 24 de enero de 2023).
- [17] «Desarrolladores de Android», *Android Developers*. <https://developer.android.com/design?hl=es-419> (accedido 21 de enero de 2023).
- [18] «Color - Material Design», *Desarrollador Android*, 6 de marzo de 2015. <https://desarrollador-android.com/material-design/disenio-material-design/estilo/color/> (accedido 25 de enero de 2023).
- [19] «EsperTech - Leader in Complex Event Processing». <https://esperonline.net/> (accedido 25 de enero de 2023).

- [20] «Firebase Performance Monitoring», *Firebase*.
<https://firebase.google.com/docs/perf-mon?hl=es-419> (accedido 27 de enero de 2023).

A. Manual de instalación

i. Aplicación móvil

El único requisito para instalar la aplicación móvil es tener un dispositivo con una versión de Android igual o superior a Android 8.0 (nivel de API ≥ 26).

Podremos transferir nuestra aplicación al dispositivo móvil directamente desde el proyecto de Android Studio, sin tener que subir la aplicación a ningún repositorio, de dos formas:

- Mediante conexión USB, para lo que deberemos tener activada la depuración USB.
- Mediante conexión inalámbrica, deberemos tener activada la depuración inalámbrica.

Para ambos métodos deberemos tener activadas las opciones de desarrollador y tener activados los permisos para instalar aplicaciones con origen desconocido.

Según el modelo del dispositivo y la versión de Android que use, estas dos opciones se activan de una forma u otra, ya que cambian las opciones de los ajustes.

Se muestra a continuación un ejemplo de cómo activar la depuración USB y la depuración inalámbrica, pero puede variar como se ha comentado, se aconseja buscar antes cómo se activan estas opciones,

1. Entra en “Ajustes” > “Acerca del dispositivo” o “Acerca del teléfono” > “Info software
2. Presiona 7 veces de forma consecutiva la opción “Número de compilación”
3. Regresa a “Ajustes”
4. Presiona “Opciones de desarrollador”
 - a. Presiona “Depuración de USB”
 - b. Presiona “Depuración inalámbrica”

Si la opción elegida es la depuración inalámbrica, tienes dos opciones, usar un código QR o usar un código de vinculación.

Entonces, en Android Studio pulsamos en Run > Select Device... > Pair Devices Using Wi-Fi y elegiremos también la opción. En Android Studio nos aparecerá un código QR, que escanearemos desde el dispositivo, o un código de vinculación, que introduciremos en nuestro dispositivo.

Una vez hecho esto, se construirá la aplicación y se instalará nuestro dispositivo. Si no tenemos activada la opción de instalar aplicaciones con orígenes desconocido, nos solicitará los permisos, aunque normalmente podremos activarlos en el momento.

Si no, podremos activar la opción antes de la instalación de esta forma (también puede variar según el modelo y la versión del SO):

1. Entra en “Ajustes” > “Protección de la privacidad”
2. Selecciona “Permisos especiales”
3. Elige la opción “Instalar aplicaciones desconocidas”

4. Elige la aplicación desde la que se instala la aplicación (explorador de archivos en nuestro caso) y concédele permiso.

ii. RabbitMQ

Para el uso de RabbitMQ, es necesario tener instalado Erlang, para ello, descargaremos la última versión de 64 bits para Windows disponible en el árbol de versiones de la página oficial: https://erlang.org/download/otp_versions_tree.html

Para el uso de RabbitMQ, es necesario tener instalado Erlang, para ello, descargaremos la última versión de 64 bits para Windows disponible en el árbol de versiones de la página oficial: https://erlang.org/download/otp_versions_tree.html

Hecho esto, actualizamos las variables de entorno del sistema, creando una nueva variable llamada ERLANG_HOME y situada en C:\Program Files\Erlang OTP.

Después, añadimos al PATH %ERLANG_HOME%\bin.

Instalada la versión de Erlang, y habiendo permitido las comunicaciones de Erlang en el cortafuegos cuando se solicite, ya podemos pasar a descargar la última versión del instalador de RabbitMQ, rabbitmq-server-{version}.exe desde <https://www.rabbitmq.com/install-windows.html>

Una vez instalado, también tendremos que editar las variables de entorno del sistema, creando la variable RABBITMQ_SERVER en C:\Program Files\RabbitMQServer\rabbitmq_server-{versión instalada}

Hecho esto, añadimos al PATH %RABBITMQ_SERVER%\sbin.

Con esto ya estará instalado el servidor de RabbitMQ y podemos comprobar el estado del servicio con rabbitmqctl status. Para parar el servicio podremos usar rabbitmqctl stop y para iniciarlo, rabbitmq-server start.

Para acceder a la consola gráfica, accedemos a <http://localhost:15672/> y usamos el usuario y contraseña por defecto “guest”.

iii. nITROGEN

Para la simulación de los datos deberemos descargar el archivo comprimido de nITROGEN entrando en <https://ucase.uca.es/nITROGEN/> y pulsando en “Download now”.

Una vez descargado, descomprimos el archivo descargado y ejecutamos el archivo “nITROGEN_GUI.exe”.

iv. Base de datos

Para almacenar los eventos complejos en la base de datos, deberemos descargar e instalar una base de datos SQL como MariaDB, descargable desde <https://mariadb.org/>

Una vez descargado en instalador, lo abrimos, hacemos click en “Siguiente”, aceptamos los términos de licencia, volvemos a hacer click dos veces en “Siguiente”, introducimos una contraseña para el root (la que se usa en la base de datos del proyecto es isi), marcamos utilizar UTF y procedemos a instalarla.

Una vez hecho esto, aceptamos las peticiones del cortafuegos, y pasamos a la creación de la base de datos y las tablas.

Para ello, abrimos la cmd, vamos a la carpeta donde instalamos la base de datos, escribimos `mysql -u root -p`, introducimos la contraseña establecida anteriormente y ejecutamos las siguientes sentencias:

```
CREATE DATABASE IF NOT EXISTS CadizRespiraEventReport;
USE CadizRespiraEventReport;
CREATE TABLE IF NOT EXISTS AparcamientoEvents (id INT NOT NULL
PRIMARY KEY AUTO_INCREMENT, aparcamientoEvent VARCHAR (4096));
CREATE TABLE IF NOT EXISTS TraficoEvents (id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT, traficoEvent VARCHAR (4096));
CREATE TABLE IF NOT EXISTS PolucionEvents (id INT NOT NULL PRIMARY
KEY AUTO_INCREMENT, polucionEvent VARCHAR (4096));
```

Hecho esto tendremos creadas las tres tablas que usaremos en nuestro sistema, podemos comprobarlo mediante la sentencia:

```
show tables;
```

v. Anypoint Studio y proyecto Mule

Para usar Anypoint Studio es necesario instalar JDK8, por ejemplo desde Adoptium <https://adoptium.net/es/temurin/releases/?version=8>

Una vez instalado el Java Development Kit 8, descargaremos la versión 6.6 de Anypoint Studio (para la versión 7 no hay servidor gratuito) entrando en <https://www.mulesoft.com/lp/dl/anypoint-mule-studio>, seleccionando “Previous versions”, y registrándonos.

Después de la descarga, lo descomprimos, abrimos el archivo de configuración AnypointStudio.ini y añadimos, después de la sentencia del launcher y antes de los vmargs, la siguiente sentencia, con la ruta donde hemos instalado JDK8:

```
-vm
C:/[Ruta de instalación de Java]/bin/javaw.exe
```

Hecho esto, abrimos Anypoint Studio y elegimos una ruta para el workspace. Cuando el cortafuegos solicite permitir el acceso a Anypoint Studio se lo concederemos.

Después, haremos clic en Help → Install new Software. En la ventana que se abrirá abrimos el desplegable “Work with” y seleccionamos Mule Runtimes for Anypoint Studio. Desplegamos entonces las opciones de Anypoint Studio Community Runtime (servidor gratuito) y seleccionamos Mule ESB Server Runtime 3.9.0 CE.

Se procederá a la instalación del servidor de Mule ESB y una vez finalizada, Anypoint Studio nos pedirá que reiniciemos la aplicación.

Una vez reiniciada, podemos proceder a importar el proyecto pulsando en File → Import → Anypoint Studio → Anypoint Studio generated Deployable Archive (.zip) y seleccionando nuestro proyecto.

Hecho esto e importado el proyecto, haciendo click sobre el proyecto y pulsando en Run As → Mule Application, podeos ejecutarlo sin problemas.

B. Manual de usuario

Una vez tengamos la aplicación instalada y todos los componentes necesarios ejecutándose, entraremos en la aplicación haciendo click en su icono.

Al acceder a la aplicación, nos encontraremos con la pantalla de bienvenida, que nos muestra información de la app: logo, nombre, objetivos, autor...



Al pulsar sobre el icono de las tres barras (metáfora visual que significa menú), se abrirá el menú principal de la aplicación y podremos navegar a la pantalla que queramos pulsando en su nombre o su icono.



Al acceder a la sección de Eventos, se mostrarán los tipos de eventos, y al pulsar en uno de los botones, se mostrará la lista de eventos del tipo elegido.



| ID | Evento | Fecha | Hora | Acción |
|-----|---------------------------|------------|----------|--------------|
| 107 | OccupiedParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 108 | FreeParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 109 | TotalFreeParkingSpots | 20-01-2023 | 00:16:03 | VER DETALLES |
| 110 | TotalOccupiedParkingSpots | 20-01-2023 | 00:16:03 | VER DETALLES |
| 111 | OccupiedParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 112 | OccupiedParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 113 | OccupiedParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 114 | FreeParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 115 | FreeParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |
| 116 | OccupiedParkingSpot | 20-01-2023 | 00:16:03 | VER DETALLES |

Al pulsar en el botón que indica “Ver detalles”, se mostrará información detallada del evento de la siguiente manera:

| | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 2:25    | |
|  Cádiz Respira | |
| ID Evento | 106 |
| Tipo Evento | FreeParkingSpot |
| Fecha y hora | 20-01-2023 00:16:03 |
| ID Sensor | 4 |

| | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 2:39    | |
|  Cádiz Respira | |
| ID Evento | 81 |
| Tipo Evento | SulfurDioxideAlert |
| Fecha y hora | 01-01-1970 01:00:00 |
| ID Sensor | 25 |
| Nivel de NO2 | 108 |
| Nivel de SO2 | 51 |
| Nivel de O3 | 108 |

Si accedemos a la sección “Aparcamientos”, se mostrará un mapa de Google con los aparcamientos que hay libres en la ciudad, de la siguiente manera:



Al pulsar en el marcador de una de las plazas de aparcamiento, se mostrará información del evento, y se mostrarán dos opciones en la esquina inferior derecha, para iniciar una ruta hacia la localización de la plaza de aparcamiento y para abrir la localización en Google Maps, como se ve en la siguiente imagen:



C. Boceto de la aplicación

Como ya explicamos en el apartado 5.4, el boceto realizado con Balsamiq que se podrá observar a partir de la siguiente página es navegable. Esto permite que aparte de ver la interfaz de usuario de la aplicación, podamos comprobar la navegabilidad de la misma.

Sin embargo, como esto nos sirve como prototipo de la aplicación, para hacernos una idea de lo que después queremos implementar, no han sido prototipadas todas las pantallas, ya que muchas son prácticamente iguales (cambiando solo el tipo de evento).

Además, no todos los botones son funcionales. Por tanto, para navegar por el boceto debemos tener en cuenta las siguientes consideraciones:

- Desde cualquier pantalla se puede abrir el menú lateral pulsando en el símbolo.
- Todos los enlaces del menú lateral son navegables.
- En la pantalla de eventos, solo es navegable el botón de aparcamientos.
- En la lista de eventos de aparcamientos, a la que se llega tras pulsar el botón que acabamos de mencionar, solo podremos acceder a los detalles del primero.
- En el mapa de aparcamientos, si pulsamos en el marcador de más arriba, se abrirá una ventana de información. Esto no está implementado en el mapa de atascos ya que es igual.
- El menú lateral no se puede cerrar como tal, se debe navegar a alguna vista a través de los enlaces.



12:25



Cádiz Respira



Cádiz Respira

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis lobortis dui vel quam pharetra, non rutrum sapien consequat. Maecenas sed lacinia leo. Integer turpis metus, rhoncus et odio at, aliquam posuere sem.

Nam sed **convallis lectus**, vitae sodales massa. Donec varius erat leo, a scelerisque sapien posuere blandit. Curabitur eget mollis velit. Sed eget erat vel ligula consequat ornare lobortis nec risus. Pellentesque habitant morbi tristique senectus et netus et malesuada.

ESTRATEGIA
2030
DESARROLLO SOSTENIBLE



Manuel Cano Crespo





12:25



ira

Cádiz Respira



Inicio



Eventos



Aparcamientos



Atascos

pharetra,
s sed
et odio

massa.
ien
lit. Sed
ortis nec
ue





Cádiz Respira

Aparcamiento

Tráfico

Polución





Cádiz Respira



Inicio



Eventos



Aparcamientos



Atascos







Cádiz Respira

| Id | Evento | Fecha y hora | Ver detalles |
|-------|---------------------|------------------------|------------------------------|
| 1 | FreeParkingSpot | 19/01/2023 19:12:53 | Ver detalles |
| 2 | FreeParkingSpot | 19/01/2023 19:12:53 | Ver detalles |
| 3 | OccupiedParkingSpot | 19/01/2023 19:12:53 | Ver detalles |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |





Cádiz Respira

-  Inicio
-  Eventos
-  Aparcamientos
-  Atascos

detalles

[detalles](#)

detalles

detalles

...



12:25







Cádiz Respira

| | |
|--------------|---------------------|
| ID Evento | 1 |
| Tipo Evento | FreeParkingSpot |
| Fecha y hora | 19/01/2023 19:12:53 |
| Id Sensor | 17 |





Cádiz Respira

-  Inicio
-  Eventos
-  Aparcamientos
-  Atascos





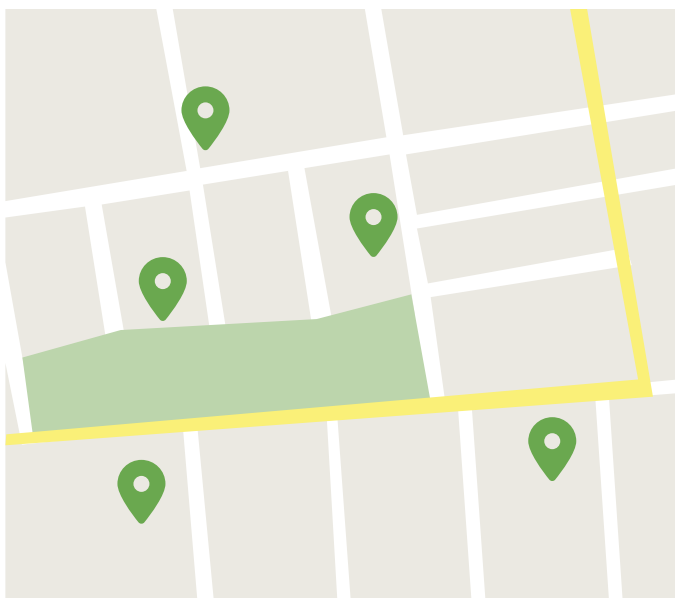
12:25



Cádiz Respira

Aparcamientos libres

Este mapa muestra los aparcamientos libres en Cádiz. Los puntos verdes indican la ubicación de los aparcamientos. El área sombreada en verde representa el centro histórico de la ciudad. La línea amarilla indica la zona de peaje.





12:25



Cádiz Respira

res

res

res

res

res

res



Inicio



Eventos



Aparcamientos



Atascos



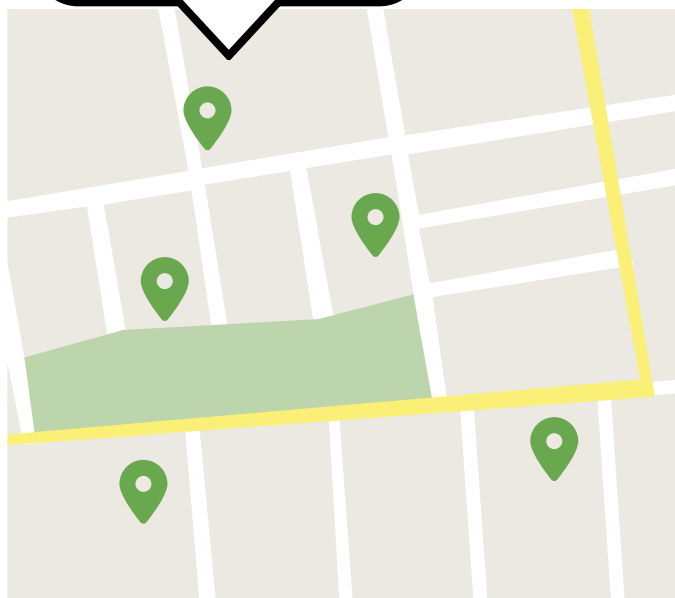
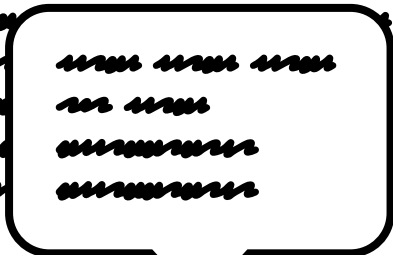


12:26



Cádiz Respira

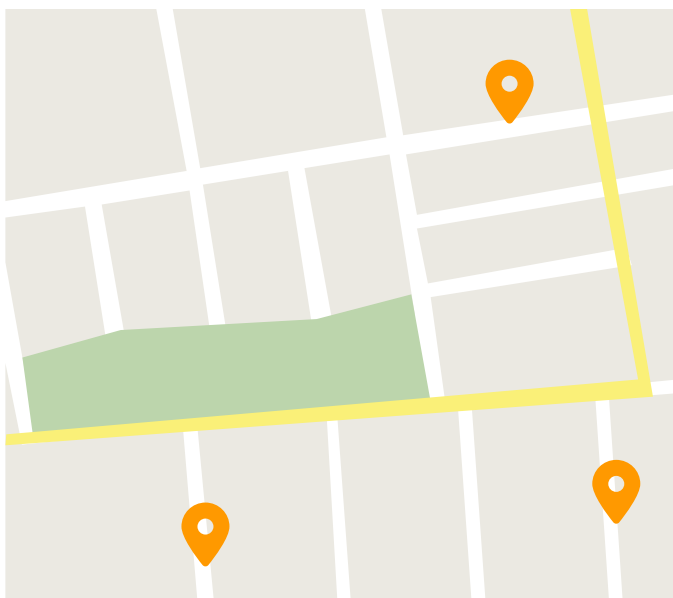
Aparcamientos libres





Atascos

Atascos en la zona de Cádiz Respira. Se muestran los puntos de congestión en la zona de estudio. Los puntos de congestión se indican con una X roja. Los puntos de congestión se indican con una X roja.









12:25



Cádiz Respira

-  Inicio
-  Eventos
-  Aparcamientos
-  Atascos

