

4.1.	Banco de pruebas	29
4.1.1.	Diseño del entorno de pruebas	29
4.1.2.	Metodología experimental	30
4.1.3.	Resultados y observaciones	31
4.2.	Pruebas de la API REST	32
4.2.1.	Objetivos y alcance	32
4.2.2.	Metodología de prueba	32
4.2.3.	Resultados obtenidos	33
Medición	33	
Comando	34	
Respuesta	35	
4.3.	Pruebas de componentes	36
4.3.1.	Enfoque general	36
4.3.2.	Resultados por componente	37
4.4.	Pruebas del frontend	38
4.4.1.	Objetivos	38
4.4.2.	Metodología	39
4.4.3.	Resultados y observaciones	43
4.5.	Prueba final de integración	43
4.5.1.	Metodología de la prueba	44
4.5.2.	Resultados obtenidos	47
4.6.	Comparación con otras soluciones	48
5.	Conclusiones	51
5.1.	Resultados y metas	51
5.2.	Trabajo futuro	52
	Bibliografía	53

4.1.	Banco de pruebas	29
4.1.1.	Diseño del entorno de pruebas	29
4.1.2.	Metodología experimental	30
4.1.3.	Resultados y observaciones	31
4.2.	Pruebas de la API REST	32
4.2.1.	Objetivos y alcance	32
4.2.2.	Metodología de prueba	32
4.2.3.	Resultados obtenidos	33
Medición	33	
Comando	34	
Respuesta	35	
4.3.	Pruebas de componentes	37
4.3.1.	Enfoque general	37
4.3.2.	Resultados por componente	37
4.4.	Pruebas del frontend	38
4.4.1.	Objetivos	38
4.4.2.	Metodología	39
4.4.3.	Resultados y observaciones	43
4.5.	Prueba final de integración	43
4.5.1.	Metodología de la prueba	44
4.5.2.	Resultados obtenidos	47
4.6.	Comparación con otras soluciones	48
5.	Conclusiones	51
5.1.	Resultados y metas	51
5.2.	Trabajo futuro	52
	Bibliografía	53

Índice de figuras

1.1. Diagrama en bloques del sistema.	4
2.1. Contador de tránsito DTEC ¹ .	6
2.2. Módulo RS-232/TTL ² .	7
2.3. Microcontrolador ESP32-C3 utilizado en los nodos de campo ³ .	7
2.4. Módulo SIM800L ⁴ .	8
3.1. Diagrama de arquitectura del sistema y el flujo de datos.	12
3.2. Diagrama de secuencia del flujo de datos.	13
3.3. Diagrama de conexión entre los módulos del sistema.	15
3.4. Fotografía contador de tránsito DTEC instalado en campo ⁵ .	16
3.5. Diagrama de flujo de información del backend.	17
3.6. Diagrama con la disposición de los controladores y flujo de dependencias.	18
3.7. Diagrama de estructura de los componentes por pantalla.	20
3.8. Diagrama de flujo de comunicación con el backend.	21
3.9. Flujo de trabajo del firmware del ESP32-C3.	23
3.10. Arquitectura de despliegue del sistema en entorno cloud on-premise.	25
4.1. Fotografía del banco de pruebas.	30
4.2. Registros de detecciones vehiculares almacenados en la base de datos MySQL.	31
4.3. Intercambio de mensajes entre el backend y el nodo de campo mediante MQTT.	31
4.4. Solicitud POST /api/medicion con todos los campos completos.	33
4.5. Solicitud POST /api/medicion con campos faltantes.	34
4.6. Solicitud POST /api/comando exitosa.	34
4.7. Solicitud POST /api/comando con error por datos incompletos.	34
4.8. Solicitud POST /api/respuesta exitosa.	35
4.9. Solicitud POST /api/respuesta con error de validación.	35
4.10. Interacción entre el nodo de campo (ESP32-C3 + SIM800L) y el backend.	38
4.11. Pantalla de inicio de sesión del frontend.	40
4.12. Panel principal con visualización del listado de dispositivos.	40
4.13. Panel de visualización del dispositivo.	41
4.14. Panel de visualización del dispositivo: preparación para la ejecución de un comando y su ejecución.	41
4.15. Panel de visualización del dispositivo: ejecución de un comando.	42
4.16. Panel de visualización del dispositivo: comando ejecutado y la respuesta pendiente.	42
4.17. Panel de visualización del dispositivo: comando ejecutado y su respuesta.	42
4.18. Panel de visualización del historial de mediciones.	43

Índice de figuras

1.1. Diagrama en bloques del sistema.	4
2.1. Contador de tránsito DTEC ¹ .	6
2.2. Módulo RS-232/TTL ² .	7
2.3. Microcontrolador ESP32-C3 utilizado en los nodos de campo ³ .	7
2.4. Módulo SIM800L ⁴ .	8
3.1. Diagrama de arquitectura del sistema y el flujo de datos.	12
3.2. Diagrama de secuencia del flujo de datos.	13
3.3. Diagrama de conexión entre los módulos del sistema.	15
3.4. Fotografía contador de tránsito DTEC instalado en campo ⁵ .	16
3.5. Diagrama de flujo de información del backend.	17
3.6. Diagrama con la disposición de los controladores y flujo de dependencias.	18
3.7. Diagrama de estructura de los componentes por pantalla.	20
3.8. Diagrama de flujo de comunicación con el backend.	21
3.9. Flujo de trabajo del firmware del ESP32-C3.	23
3.10. Arquitectura de despliegue del sistema en entorno cloud on-premise.	25
4.1. Fotografía del banco de pruebas.	30
4.2. Registros de detecciones vehiculares almacenados en la base de datos MySQL.	31
4.3. Intercambio de mensajes entre el backend y el nodo de campo mediante MQTT.	32
4.4. Solicitud POST /api/medicion con todos los campos completos.	34
4.5. Solicitud POST /api/medicion con campos faltantes.	34
4.6. Solicitud POST /api/comando exitosa.	35
4.7. Solicitud POST /api/comando con error por datos incompletos.	35
4.8. Solicitud POST /api/respuesta exitosa.	35
4.9. Solicitud POST /api/respuesta con error de validación.	36
4.10. Interacción entre el nodo de campo (ESP32-C3 + SIM800L) y el backend.	38
4.11. Pantalla de inicio de sesión del frontend.	40
4.12. Panel principal con visualización del listado de dispositivos.	40
4.13. Panel de visualización del dispositivo.	41
4.14. Panel de visualización del dispositivo: preparación para la ejecución de un comando y su ejecución.	41
4.15. Panel de visualización del dispositivo: ejecución de un comando.	42
4.16. Panel de visualización del dispositivo: comando ejecutado y la respuesta pendiente.	42
4.17. Panel de visualización del dispositivo: comando ejecutado y su respuesta.	42
4.18. Panel de visualización del historial de mediciones.	43

4.1. Banco de pruebas

31

4.1.3. Resultados y observaciones

Los resultados experimentales demostraron que el sistema fue capaz de:

- Mantener la integridad de los datos en escenarios de conectividad inestable.
- Asegurar la entrega de eventos por medio de la cola FIFO implementada en el firmware.
- Ejecutar comandos remotos y recibir respuestas de forma confiable.
- Reanudar la transmisión después de cortes de red sin pérdida de información.

Se presentan a continuación ejemplos representativos de los resultados obtenidos durante la prueba. En la figura 4.2 se muestra la tabla `Medicion` en phpMyAdmin [39], con los registros almacenados en la base de datos MySQL. Cada uno es una detección vehicular procesada por el sistema, con su fecha, valor, carril y clasificación. Esto evidencia que el backend recibió y registró correctamente las mediciones enviadas por el nodo, incluso tras interrupciones de red.

medicionId	fecha	valor	carril	clasificacionId	dispositivoId	createdAt
328896	2025-10-15 17:00:00	33	2	1	1	2025-11-06 12:01:54
328897	2025-10-15 17:05:00	14	1	2	1	2025-11-06 12:02:11
328898	2025-10-15 17:10:00	38	2	2	1	2025-11-06 12:03:00
328899	2025-10-15 17:15:00	31	1	1	1	2025-11-06 12:03:51
328900	2025-10-15 17:15:00	31	1	1	1	2025-11-06 12:03:57
328901	2025-10-15 17:20:00	24	1	2	1	2025-11-06 12:05:48
328902	2025-10-15 17:25:00	17	2	2	1	2025-11-06 12:05:58
328903	2025-10-15 17:30:00	39	2	1	1	2025-11-06 12:06:21
329008	2025-10-15 17:45:00	25	2	2	1	2025-11-07 15:06:11

FIGURA 4.2. Registros de detecciones vehiculares almacenados en la base de datos MySQL.

En la figura 4.3 se muestra un intercambio entre el servidor y el nodo. El backend envía un comando al dispositivo en `dispositivo/id/comando` y el nodo responde en `dispositivo/id/respuesta` con el cmdId y el valor asociado, validando la comunicación bidireccional.

SELECT * FROM 'Comando'
<input type="checkbox"/> Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
cmdId fecha tipoComando valor dispositivoId createdAt
16 2025-11-06 12:03:50 1 resetear 1 2025-11-06 12:03:50
SELECT * FROM 'Respuesta'
<input type="checkbox"/> Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
Respid fecha cmdId valor dispositivoId createdAt
7 2025-11-06 12:03:50 16 OK 1 2025-11-06 12:06:09

FIGURA 4.3. Intercambio de mensajes entre el backend y el nodo de campo mediante MQTT.

4.1. Banco de pruebas

31

4.1.3. Resultados y observaciones

Los resultados experimentales demostraron que el sistema fue capaz de:

- Mantener la integridad de los datos en escenarios de conectividad inestable.
- Asegurar la entrega de eventos por medio de la cola FIFO implementada en el firmware.
- Ejecutar comandos remotos y recibir respuestas de forma confiable.
- Reanudar la transmisión después de cortes de red sin pérdida de información.

Se presentan a continuación ejemplos representativos de los resultados obtenidos durante la prueba. En la figura 4.2 se muestra la tabla `Medicion` en phpMyAdmin [39], con los registros almacenados en la base de datos MySQL. Cada uno es una detección vehicular procesada por el sistema, con su fecha, valor, carril y clasificación. Esto evidencia que el backend recibió y registró correctamente las mediciones enviadas por el nodo, incluso tras interrupciones de red.

medicionId	fecha	valor	carril	clasificacionId	dispositivoId	createdAt
328896	2025-10-15 17:00:00	33	2	1	1	2025-11-06 12:01:54
328897	2025-10-15 17:05:00	14	1	2	1	2025-11-06 12:02:11
328898	2025-10-15 17:10:00	38	2	2	1	2025-11-06 12:03:00
328899	2025-10-15 17:15:00	31	1	1	1	2025-11-06 12:03:51
328900	2025-10-15 17:15:00	31	1	1	1	2025-11-06 12:03:57
328901	2025-10-15 17:20:00	24	1	2	1	2025-11-06 12:05:48
328902	2025-10-15 17:25:00	17	2	2	1	2025-11-06 12:05:58
328903	2025-10-15 17:30:00	39	2	1	1	2025-11-06 12:06:21
329008	2025-10-15 17:45:00	25	2	2	1	2025-11-07 15:06:11

FIGURA 4.2. Registros de detecciones vehiculares almacenados en la base de datos MySQL.

En la figura 4.3 se presenta un ejemplo del intercambio de mensajes entre el servidor y el nodo de campo.

En este caso, el backend envía un comando remoto al dispositivo mediante una publicación en el tópico MQTT `dispositivo/id/comando`. El nodo recibe ese comando, lo procesa localmente y responde a través del dispositivo/`id/response`.

El mensaje de respuesta incluye el campo de cmdId y el valor asociado al comando ejecutado, lo que permite verificar el funcionamiento bidireccional del sistema.

4.2. Pruebas de la API REST

Esta sección presenta las pruebas realizadas sobre la API REST implementada en el backend del sistema. El propósito fue validar la interacción entre los componentes principales (backend, base de datos y broker MQTT) y comprobar la integridad, la seguridad y el rendimiento de las operaciones ofrecidas por los endpoints.

4.2.1. Objetivos y alcance

Los objetivos específicos que guiaron la planificación de las pruebas fueron los siguientes:

- Verificar la implementación correcta de los endpoints asociados a dispositivos, mediciones, comandos, respuestas y usuarios.
- Confirmar la persistencia y consistencia de los datos en la base de datos MySQL.
- Validar el esquema de autenticación y autorización mediante tokens JWT.
- Evaluar la integración con el broker MQTT para la publicación y recepción de mensajes.
- Medir el tiempo de respuesta y la estabilidad del servicio en diferentes condiciones de red y carga.
- Comprobar el manejo de errores y la coherencia de las respuestas ante solicitudes inválidas.

El alcance incluyó operaciones sincrónicas (consultas, altas, modificaciones y eliminaciones) y asíncronas (envío y recepción de comandos MQTT) con el fin de cubrir todos los flujos funcionales.

4.2.2. Metodología de prueba

El proceso de validación se realizó con la herramienta Postman [40]. Se elaboraron colecciones de solicitudes y scripts de prueba en la pestaña Tests, que verificaron los códigos de estado HTTP, la estructura de las respuestas y el contenido de los mensajes.

El Collection Runner [41] permitió ejecutar los casos de prueba en distintos entornos: desarrollo local, red simulada GPRS e integración con el broker MQTT. Los resultados se exportaron en formato JSON y se analizaron mediante la extensión Newman [42].

El middleware Morgan registró las solicitudes HTTP, mientras que el sistema de logging Winston almacenó eventos críticos del backend, como errores de conexión, tiempos de procesamiento y publicaciones MQTT. La trazabilidad obtenida permitió optimizar parámetros como la concurrencia de conexiones MySQL y la retención de mensajes MQTT.

Para evaluar la tolerancia a fallos, se interrumpieron deliberadamente las conexiones del broker MQTT y del enlace GPRS. Los mensajes en cola se reenviaron al restablecer la red sin generar duplicaciones ni pérdidas.

SELECT * FROM `Comando`					
Permitido [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]					
cmdid	fecha	tipoComando	valor	dispositivoId	createdAt
16	2025-11-06 12:03:50	resetear	1	1	2025-11-06 12:03:50

SELECT * FROM `Respuesta`					
Permitido [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]					
respid	fecha	cmdid	valor	dispositivoId	createdAt
7	2025-11-06 12:03:50	16	OK	1	2025-11-06 12:06:09

FIGURA 4.3. Intercambio de mensajes entre el backend y el nodo de campo mediante MQTT.

4.2. Pruebas de la API REST

Esta sección presenta las pruebas realizadas sobre la API REST implementada en el backend del sistema. El propósito fue validar la interacción entre los componentes principales (backend, base de datos y broker MQTT) y comprobar la integridad, la seguridad y el rendimiento de las operaciones ofrecidas por los endpoints.

4.2.1. Objetivos y alcance

Los objetivos específicos que guiaron la planificación de las pruebas fueron los siguientes:

- Verificar la implementación correcta de los endpoints asociados a dispositivos, mediciones, comandos, respuestas y usuarios.
- Confirmar la persistencia y consistencia de los datos en la base de datos MySQL.
- Validar el esquema de autenticación y autorización mediante tokens JWT.
- Evaluar la integración con el broker MQTT para la publicación y recepción de mensajes.
- Medir el tiempo de respuesta y la estabilidad del servicio en diferentes condiciones de red y carga.
- Comprobar el manejo de errores y la coherencia de las respuestas ante solicitudes inválidas.

El alcance incluyó operaciones sincrónicas (consultas, altas, modificaciones y eliminaciones) y asíncronas (envío y recepción de comandos MQTT) con el fin de cubrir todos los flujos funcionales.

4.2.2. Metodología de prueba

El proceso de validación se realizó con la herramienta Postman [40]. Se elaboraron colecciones de solicitudes y scripts de prueba en la pestaña Tests, que verificaron

4.2. Pruebas de la API REST

33

4.2.3. Resultados obtenidos

Las pruebas confirmaron la estabilidad y solidez de la API REST. Todas las operaciones CRUD se ejecutaron correctamente y devolvieron respuestas en formato JSON con los códigos HTTP correspondientes, lo que permite destacar lo siguiente:

- Autenticación: las solicitudes sin token o con credenciales inválidas fueron rechazadas con los códigos 401 y 403.
- Integración MQTT: los comandos se publicaron en los tópicos dispositivo/{id}/comando, y las respuestas se recibieron en dispositivo/{id}/respuesta, que actualizó los estados en la base de datos.
- Persistencia: no se registraron pérdidas ni duplicaciones de datos en la base de datos MySQL.
- Rendimiento: el tiempo de respuesta promedio fue de 210 ms en entorno local y de 550 ms bajo simulación GPRS, con un máximo de 1,2 s en carga alta.
- Manejo de errores: los mensajes de error fueron claros y usaron códigos estandarizados (400, 404, 423, 500).

A continuación, se presentan las pruebas realizadas en el backend, destinadas a verificar el correcto funcionamiento de los servicios implementados.

Medición

La entidad Medicion constituye el núcleo del sistema, ya que representa los datos enviados por el firmware al backend. A continuación, se muestran las pruebas realizadas sobre el endpoint POST /api/medicion, donde se validó la correcta recepción y verificación de los campos obligatorios.

En la figura 4.4 se observa el envío correcto de los datos requeridos: fecha, valor, carril, clasificacionId y dispositivold. El backend almacena la medición y retorna una confirmación en formato JSON.

```

POST http://127.0.0.1:8000/api/medicion
{
  "fecha": "2025-11-13T21:15:00-03:00",
  "valor": 125,
  "carril": 2,
  "clasificacionId": 1,
  "dispositivoId": 1
}
201 Created
878 ms 512 B
{
  "message": "Medición creada con éxito",
  "status": 1,
  "data": [
    {
      "medicionId": 329461,
      "fecha": "2025-11-13T00:15:00.000Z",
      "valor": 125,
      "carril": 2,
      "clasificacionId": 1,
      "dispositivoId": 1,
      "createdAt": "2025-11-13T01:30:28.463Z"
    }
  ]
}
  
```

FIGURA 4.4. Solicitud POST /api/medicion con todos los campos completos.

4.2. Pruebas de la API REST

33

los códigos de estado HTTP, la estructura de las respuestas y el contenido de los mensajes.

El Collection Runner [41] permitió ejecutar los casos de prueba en distintos entornos: desarrollo local, red simulada GPRS e integración con el broker MQTT. Los resultados se exportaron en formato JSON y se analizaron mediante la extensión Newman [42].

El middleware Morgan registró las solicitudes HTTP, mientras que el sistema de logging Winston almacenó eventos críticos del backend, como errores de conexión, tiempos de procesamiento y publicaciones MQTT. La trazabilidad obtenida permitió optimizar parámetros como la concurrencia de conexiones MySQL y la retención de mensajes MQTT.

Para evaluar la tolerancia a fallos, se interrumpieron deliberadamente las conexiones del broker MQTT y del enlace GPRS. Los mensajes en cola se reenviaron al restablecer la red sin generar duplicaciones ni pérdidas.

4.2.3. Resultados obtenidos

Las pruebas confirmaron la estabilidad y solidez de la API REST. Todas las operaciones CRUD se ejecutaron correctamente y devolvieron respuestas en formato JSON con los códigos HTTP correspondientes, lo que permite destacar lo siguiente:

- Autenticación: las solicitudes sin token o con credenciales inválidas fueron rechazadas con los códigos 401 y 403.
- Integración MQTT: los comandos se publicaron en los tópicos dispositivo/{id}/comando, y las respuestas se recibieron en dispositivo/{id}/respuesta, que actualizó los estados en la base de datos.
- Persistencia: no se registraron pérdidas ni duplicaciones de datos en la base de datos MySQL.
- Rendimiento: el tiempo de respuesta promedio fue de 210 ms en entorno local y de 550 ms bajo simulación GPRS, con un máximo de 1,2 s en carga alta.
- Manejo de errores: los mensajes de error fueron claros y usaron códigos estandarizados (400, 404, 423, 500).

A continuación, se presentan las pruebas realizadas en el backend, destinadas a verificar el correcto funcionamiento de los servicios implementados.

Medición

La entidad Medicion constituye el núcleo del sistema, ya que representa los datos enviados por el firmware al backend. A continuación, se muestran las pruebas realizadas sobre el endpoint POST /api/medicion, donde se validó la correcta recepción y verificación de los campos obligatorios.

En la figura 4.4 se observa el envío correcto de los datos requeridos: fecha, valor, carril, clasificacionId y dispositivold. El backend almacena la medición y retorna una confirmación en formato JSON.

En la figura 4.5 se muestra la respuesta ante una solicitud incompleta, donde falta al menos uno de los campos obligatorios. El sistema devuelve un mensaje en formato JSON indicando la causa del error.

```

POST 127.0.0.1:8000/medicion
Body
{
    "values": 129,
    "carril": 2,
    "clasificacionId": 1,
    "dispositivoId": 1
}
{
    "message": "falta(s), valor, carril, clasificacionId dispositivoId son obligatorios",
    "status": 0
}
  
```

FIGURA 4.5. Solicitud POST /api/medicion con campos faltantes.

Comando

La entidad Comando permite enviar instrucciones remotas a los nodos de campo a través del broker MQTT. Las pruebas se realizaron sobre el endpoint POST /api/comando, que verifica tanto la creación exitosa como la validación de datos.

La figura 4.6 muestra una solicitud válida con los campos fecha, valor y dispositivoid, que genera un nuevo comando y lo envía al dispositivo indicado.

```

POST 127.0.0.1:8000/comando
Body
{
    "fecha": "2025-11-13T21:15:00-03:00",
    "valor": "reset",
    "dispositivoId": 1
}
{
    "message": "Comando creado con éxito y publicado.",
    "status": 1,
    "data": [
        {
            "carril": 29,
            "fecha": "2025-11-13T00:15:00-0002",
            "tipoComando": 1,
            "valor": "reset",
            "dispositivoId": 1,
            "createdAt": "2025-11-13T02:29:50-0002"
        }
    ]
}
  
```

FIGURA 4.6. Solicitud POST /api/comando exitosa.

En la figura 4.7 se observa una solicitud con parámetros faltantes.

```

POST 127.0.0.1:8000/comando
Body
{
    "fecha": "2025-11-13T21:15:00-03:00",
    "carril": 29
}
{
    "message": "Faltan datos obligatorios",
    "status": 0
}
  
```

FIGURA 4.7. Solicitud POST /api/comando con error por datos incompletos.

```

POST 127.0.0.1:8000/medicion
Body
{
    "fecha": "2025-11-13T21:15:00-03:00",
    "valor": 129,
    "carril": 2,
    "clasificacionId": 3,
    "dispositivoId": 1
}
{
    "message": "Medición creada con éxito",
    "status": 1,
    "data": [
        {
            "medicionId": 329801,
            "fecha": "2025-11-13T00:15:00-0002",
            "valor": 129,
            "carril": 2,
            "clasificacionId": 1,
            "dispositivoId": 1,
            "createdAt": "2025-11-13T01:29:00-0002"
        }
    ]
}
  
```

FIGURA 4.4. Solicitud POST /api/medicion con todos los campos completos.

En la figura 4.5 se muestra la respuesta ante una solicitud incompleta, donde falta al menos uno de los campos obligatorios. El sistema devuelve un mensaje en formato JSON indicando la causa del error.

```

POST 127.0.0.1:8000/medicion
Body
{
    "values": 129,
    "carril": 2,
    "clasificacionId": 1,
    "dispositivoId": 1
}
{
    "message": "falta(s), valor, carril, clasificacionId dispositivoId son obligatorios",
    "status": 0
}
  
```

FIGURA 4.5. Solicitud POST /api/medicion con campos faltantes.

Comando

La entidad Comando permite enviar instrucciones remotas a los nodos de campo a través del broker MQTT. Las pruebas se realizaron sobre el endpoint POST /api/comando, que verifica tanto la creación exitosa como la validación de datos.

La figura 4.6 muestra una solicitud válida con los campos fecha, valor y dispositivoid, que genera un nuevo comando y lo envía al dispositivo indicado.

4.2. Pruebas de la API REST

35

Respuesta

La entidad **Respuesta** almacena los mensajes enviados por los nodos de campo en respuesta a los comandos recibidos. Se evaluó el endpoint `POST /api/respuesta`, y se comprobó la asociación correcta con el comando original y el dispositivo.

La figura 4.8 muestra un ejemplo de respuesta registrada exitosamente, con los campos fecha, cmdId, valor y dispositivoId.

```
POST 127.0.0.1:8000/respuesta
{
  "fecha": "2025-11-12T21:15:00-03:00",
  "cmdId": 29,
  "valor": "reset",
  "dispositivoId": 1
}
```

FIGURA 4.8. Solicitud POST /api/respuesta exitosa.

En la figura 4.9 se observa la respuesta generada cuando alguno de los campos requeridos no fue proporcionado.

```
POST 127.0.0.1:8000/respuesta
{
  "cmdId": 29
}
```

FIGURA 4.9. Solicitud POST /api/respuesta con error de validación.

A continuación, se presenta la tabla 4.1 con los resultados de las pruebas de los endpoints REST, donde se registraron las respuestas del sistema, los códigos HTTP obtenidos y el tiempo medio de procesamiento para cada operación:

4.2. Pruebas de la API REST

35

Respuesta

```
POST 127.0.0.1:8000/comando
{
  "fecha": "2025-11-12T21:15:00-03:00",
  "cmdId": 1,
  "valor": "reset",
  "dispositivoId": 1
}
```

FIGURA 4.6. Solicitud POST /api/comando exitosa.

En la figura 4.7 se observa una solicitud con parámetros faltantes.

```
POST 127.0.0.1:8000/comando
{
  "cmdId": 29
}
```

FIGURA 4.7. Solicitud POST /api/comando con error por datos incompletos.

Respuesta

La entidad **Respuesta** almacena los mensajes enviados por los nodos de campo en respuesta a los comandos recibidos. Se evaluó el endpoint `POST /api/respuesta`, y se comprobó la asociación correcta con el comando original y el dispositivo.

La figura 4.8 muestra un ejemplo de respuesta registrada exitosamente, con los campos fecha, cmdId, valor y dispositivoId.

```
POST 127.0.0.1:8000/respuesta
{
  "fecha": "2025-11-12T21:15:00-03:00",
  "cmdId": 29,
  "valor": "reset",
  "dispositivoId": 1
}
```

FIGURA 4.8. Solicitud POST /api/respuesta exitosa.

En la figura 4.9 se observa la respuesta generada cuando alguno de los campos requeridos no fue proporcionado.

TABLA 4.1. Resultados de las pruebas realizadas sobre los principales endpoints de la API REST mediante Postman.

Endpoint	Tipo	Resultado	Código HTTP	Tiempo medio (ms)
GET /dispositivo	GET	Consulta correcta de todos los dispositivos	200	215
GET /dispositivo/{id}	GET	Recuperación exitosa de un dispositivo específico	200	225
POST /dispositivo	POST	Alta de nuevo dispositivo	201	245
GET /medicion/dispositivo/{id}	GET	Consulta de mediciones por dispositivo	200	230
POST /comando	POST	Publicación de comando en MQTT	201	310
GET /comando/{id}	GET	Consulta de estado de comando	200	520
POST /respuesta	POST	Registro de respuesta	201	245
GET /respuesta/{id_com}	GET	Recuperación de respuesta asociada	200	225
POST /usuario/login	POST	Autenticación válida (JWT)	200	180
GET /usuario	GET	Acceso restringido (JWT)	403	190

Los ensayos confirmaron que la API REST cumple los criterios de fiabilidad, seguridad y desempeño definidos en el diseño. El uso de colecciones automatizadas permitió repetir las pruebas en distintos entornos y documentar los resultados con precisión.

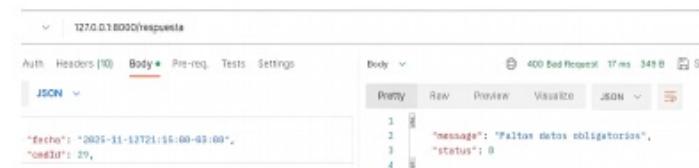
4.3. Pruebas de componentes

Las pruebas de componentes tuvieron como propósito verificar la integración entre los módulos del sistema (firmware, backend, broker MQTT, base de datos y frontend) y asegurar el correcto comportamiento de manera individual y conjunta.

A diferencia del banco de pruebas y de la validación de la API REST, esta etapa se centró en la integridad del flujo de datos completo, el manejo de errores y la coherencia operativa ante fallas o sobrecarga.

4.3.1. Enfoque general

El sistema se evaluó bajo un esquema progresivo:



```

1
2
3
4
"fecha": "2025-11-12T21:15:00-03:00",
"message": "Faltan datos obligatorios",
"codeId": 29,
"status": 0

```

FIGURA 4.9. Solicitud POST /api/respuesta con error de validación.

A continuación, se presenta la tabla 4.1 con los resultados de las pruebas de los endpoints REST:

TABLA 4.1. Resultados de las pruebas realizadas sobre los principales endpoints de la API REST mediante Postman.

Endpoint	Tipo	Resultado	Código HTTP	Tiempo medio (ms)
GET /dispositivo	GET	Consulta correcta de todos los dispositivos	200	215
GET /dispositivo/{id}	GET	Recuperación exitosa de un dispositivo específico	200	225
POST /dispositivo	POST	Alta de nuevo dispositivo	201	245
GET /medicion/dispositivo/{id}	GET	Consulta de mediciones por dispositivo	200	230
POST /comando	POST	Publicación de comando en MQTT	201	310
GET /comando/{id}	GET	Consulta de estado de comando	200	520
POST /respuesta	POST	Registro de respuesta	201	245
GET /respuesta/{id_com}	GET	Recuperación de respuesta asociada	200	225
POST /usuario/login	POST	Autenticación válida (JWT)	200	180
GET /usuario	GET	Acceso restringido (JWT)	403	190

Los ensayos confirmaron que la API REST cumple los criterios de fiabilidad, seguridad y desempeño definidos en el diseño. El uso de colecciones automatizadas permitió repetir las pruebas en distintos entornos y documentar los resultados con precisión.

4.3. Pruebas de componentes

37

1. Pruebas unitarias: destinadas a validar la funcionalidad de cada componente de software.
2. Pruebas de integración: diseñadas para verificar la comunicación entre módulos y la consistencia de los datos.
3. Pruebas de tolerancia a fallos: enfocadas en la recuperación automática ante desconexiones, errores de red o reinicios.

El entorno completo se desplegó en contenedores Docker independientes, lo que permitió reproducir escenarios de prueba con precisión y medir el impacto de fallas.

4.3.2. Resultados por componente

Cada módulo del sistema fue evaluado de forma independiente para verificar su funcionamiento, la integridad de los datos y la robustez ante fallos de conexión. Este proceso permitió analizar el comportamiento de cada componente en situaciones reales de operación, asegurando una arquitectura estable y confiable en todas las etapas del flujo de información. Se resumen los principales resultados obtenidos en las pruebas de cada componente:

- Firmware (ESP32-C3): se validó el análisis correcto de tramas RS-232, el almacenamiento temporal en colas FIFO y la publicación confiable de mensajes MQTT hacia el broker. Estas pruebas confirmaron que el dispositivo interpreta adecuadamente los datos recibidos y mantiene la consistencia de los paquetes aun en escenarios de mayor carga de transmisión.
- Broker MQTT: se ejecutaron desconexiones simuladas para medir la tolerancia a fallos. El sistema mantuvo la sesión activa y transmitió los mensajes pendientes una vez restablecida la conexión, lo que aseguró continuidad del servicio sin pérdida de información.
- Backend: se comprobó la correcta gestión de solicitudes REST, el procesamiento adecuado de cada operación y la sincronización efectiva con el broker MQTT para el envío y recepción de comandos y respuestas.
- Frontend: se verificó la comunicación bidireccional con la API REST, la presentación inmediata de la información obtenida y la visualización en el panel del estado actualizado de dispositivos y mediciones.
- Manejo de errores: los registros obtenidos mediante Winston y Morgan reflejaron reconexiones exitosas ante cortes de red y operaciones sin pérdida de datos, lo que confirmó un sistema con mecanismos sólidos de recuperación frente a situaciones adversas.

En la figura 4.10 se puede observar el intercambio de mensajes entre el nodo de campo y el servidor. En la parte superior de la imagen se muestra la consola del ESP32-C3, donde el firmware registra el envío de una medición a través del módulo SIM800L utilizando el protocolo MQTT. En la parte inferior, se visualiza la consola del backend desarrollada en Node.js, que recibe y procesa el mensaje, verificando los datos de la medición antes de almacenarlos en la base de datos.

4.3. Pruebas de componentes

37

4.3. Pruebas de componentes

Las pruebas de componentes tuvieron como propósito verificar la integración entre los módulos del sistema (firmware, backend, broker MQTT, base de datos y frontend) y asegurar el correcto comportamiento de manera individual y conjunta.

A diferencia del banco de pruebas y de la validación de la API REST, esta etapa se centró en la integridad del flujo de datos completo, el manejo de errores y la coherencia operativa ante fallas o sobrecarga.

4.3.1. Enfoque general

El sistema se evaluó bajo un esquema progresivo:

1. Pruebas unitarias: destinadas a validar la funcionalidad de cada componente de software.
2. Pruebas de integración: diseñadas para verificar la comunicación entre módulos y la consistencia de los datos.
3. Pruebas de tolerancia a fallos: enfocadas en la recuperación automática ante desconexiones, errores de red o reinicios.

El entorno completo se desplegó en contenedores Docker independientes, lo que permitió reproducir escenarios de prueba con precisión y medir el impacto de fallas.

4.3.2. Resultados por componente

Cada módulo del sistema fue evaluado de forma independiente para verificar su funcionamiento, la integridad de los datos y la robustez ante fallos de conexión. Se resumen los principales resultados obtenidos en las pruebas de cada componente:

- Firmware (ESP32-C3): se validó el análisis de tramas RS-232, el almacenamiento temporal en colas FIFO y la publicación confiable de mensajes MQTT.
- Broker MQTT: se realizaron desconexiones simuladas. El sistema mantuvo la sesión y retransmitió los mensajes pendientes.
- Backend: se comprobó la correcta gestión de solicitudes REST y la sincronización con el broker MQTT.
- Frontend: se verificó la comunicación bidireccional con la API REST y la actualización en tiempo real de las mediciones.
- Manejo de errores: los registros de Winston y Morgan mostraron reconexiones exitosas sin pérdida de información.

En la figura 4.10 se puede observar el intercambio de mensajes entre el nodo de campo y el servidor. En la parte superior de la imagen se muestra la consola del ESP32-C3, donde el firmware registra el envío de una medición a través del módulo SIM800L utilizando el protocolo MQTT. En la parte inferior, se visualiza la consola del backend desarrollada en Node.js, que recibe y procesa el mensaje, verificando los datos de la medición antes de almacenarlos en la base de datos.

La figura 4.11 muestra la pantalla de inicio de sesión, donde el usuario debe ingresar sus credenciales para acceder al sistema.

The image shows a login interface with a header 'Login'. Below it is a form containing two input fields: one for the username with the value 'admin' and another for the password with the value '.....'. At the bottom is a large green button labeled 'INGRESAR'.

FIGURA 4.11. Pantalla de inicio de sesión del frontend.

Una vez autenticado, el usuario accede al panel principal mostrado en la figura 4.12, donde se presenta el listado de dispositivos registrados junto con la ubicación y el tipo de contador.

Listado de Dispositivos	
contador 1	<button>VER ></button>
Ubicación: Padre Banda	
Tipo: DTEG	
contador 2	<button>VER ></button>
Ubicación: Tres Arroyos	
Tipo: PADR	
contador 3	<button>VER ></button>
Ubicación: Gorozaga	
Tipo: DTEG	
contador 4	<button>VER ></button>
Ubicación: Pison	
Tipo: DTEG,ORD	

FIGURA 4.12. Panel principal con visualización del listado de dispositivos.

Al seleccionar un dispositivo del listado, el sistema despliega un panel detallado con la información específica de dicho equipo. La figura 4.13 ilustra la primera vista del panel de dispositivo, donde se presenta la información general de identificación del equipo (nombre, ubicación y tipo), el último comando enviado y su correspondiente respuesta, la última medición registrada y un botón que permite acceder a la visualización de las mediciones históricas. Esta interfaz facilita el análisis del estado del dispositivo y el acceso rápido a los datos relevantes sin necesidad de recorrer menús adicionales.

La figura 4.11 muestra la pantalla de inicio de sesión, donde el usuario debe ingresar sus credenciales para acceder al sistema.

The image shows a login interface with a header 'Login'. Below it is a form containing two input fields: one for the username with the value 'admin' and another for the password with the value '.....'. At the bottom is a large green button labeled 'INGRESAR'.

FIGURA 4.11. Pantalla de inicio de sesión del frontend.

Una vez autenticado, el usuario accede al panel principal mostrado en la figura 4.12, donde se presenta el listado de dispositivos registrados junto con la ubicación y el tipo de contador.

Listado de Dispositivos	
contador 1	<button>VER ></button>
Ubicación: Padre Banda	
Tipo: DTEG	
contador 2	<button>VER ></button>
Ubicación: Tres Arroyos	
Tipo: PADR	
contador 3	<button>VER ></button>
Ubicación: Gorozaga	
Tipo: DTEG	
contador 4	<button>VER ></button>
Ubicación: Pison	
Tipo: DTEG,ORD	

FIGURA 4.12. Panel principal con visualización del listado de dispositivos.

Al seleccionar un dispositivo del listado, el sistema despliega un panel detallado con la información específica de dicho equipo. La figura 4.13 ilustra la primera vista del panel de dispositivo, donde se muestran los datos generales de identificación (nombre, ubicación y tipo), el último comando enviado, su respuesta, la última medición registrada y un botón para visualizar las mediciones históricas.