📋 Tabla de Endpoints - API Mock de Servicios REST

🔧 Endpoints de Configuración

Endpoint	Método	Descripción	Ejemplo de Prueba
`/configure-mock`	`POST`	Crear nueva configuración de mock	Ver ejemplos detallados abajo
`/configure-mock`	`GET`	Obtener todas las configuraciones	`curl http://localhost:3000/configure-mock`
`/configure-mock/stats`	`GET`	Obtener estadísticas de configuraciones	`curl http://localhost:3000/configure-mock/stats`
`/configure-mock/:id`	`DELETE`	Eliminar configuración específica	`curl -X DELETE http://localhost:3000/configure-mock/{ID}`
`/configure-mock`	`DELETE`	Limpiar todas las configuraciones	`curl -X DELETE http://localhost:3000/configure-mock`
`/health`	`GET`	Health check del servidor	`curl http://localhost:3000/health`

🎯 Endpoints de Mocks (Dinámicos)

Endpoint	Método	Descripción	Configuración Necesaria
`/*`	`ANY`	Cualquier ruta configurada como mock	Debe crearse primero con `/configure-mock`

🧪 Ejemplos Detallados de Pruebas

1. Health Check

bash

```
curl http://localhost:3000/health
```

Respuesta esperada:

json

```
{
  "status": "OK",
  "timestamp": "2025-07-12T02:38:24.715Z"
}
```

2. Crear Mock Básico

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/productos",
    "method": "GET",
    "statusCode": 200,
    "responseBody": {"productos": ["producto1", "producto2"]},
    "contentType": "application/json"
  }'
```

Respuesta esperada:

json


```
{
  "success": true,
  "data": {
    "id": "uuid-generado",
    "route": "/api/v1/productos",
    "method": "GET",
    "statusCode": 200,
    "responseBody": {"productos": ["producto1", "producto2"]},
    "contentType": "application/json",
    "createdAt": "2025-07-12T02:38:24.715Z"
  },
  "timestamp": "2025-07-12T02:38:24.715Z",
  "message": "Mock configuration created successfully"
}
```


Probar el mock creado:

bash

```
curl http://localhost:3000/api/v1/productos
```

---

3. Mock con Query Parameters

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/usuarios",
    "method": "GET",
    "queryParams": {"tipo": "admin"},
    "statusCode": 200,
    "responseBody": {"usuario": "admin", "permisos": ["leer", "escribir", "eliminar"]},
    "contentType": "application/json"
  }'
```

Probar:

```
bash

curl "http://localhost:3000/api/v1/usuarios?tipo=admin"
```

---

4. Mock con Body Parameters (POST)

```
bash

curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/login",
    "method": "POST",
    "bodyParams": {"email": "admin@test.com", "password": "123456"},
    "statusCode": 200,
    "responseBody": {"token": "abc123", "user": "admin"},
    "contentType": "application/json"
  }'
```

Probar:

```
bash

curl -X POST http://localhost:3000/api/v1/login \
  -H "Content-Type: application/json" \
  -d '{"email": "admin@test.com", "password": "123456"}'
```

---

5. Mock con Headers

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/profile",
    "method": "GET",
    "headers": {"authorization": "Bearer token123"},
    "statusCode": 200,
    "responseBody": {"name": "Usuario Admin", "role": "admin"},
    "contentType": "application/json"
  }'
```

Probar:

bash

```
curl http://localhost:3000/api/v1/profile \
  -H "Authorization: Bearer token123"
```

---

6. Mock con Lógica Condicional

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/data",
    "method": "GET",
    "statusCode": 200,
    "responseBody": "{{if query.user === \"admin\"}}{\\"data\\": \\"admin_data\\", \\"access\\": \\"full\\"}{{else}}{\\"data\\":
    "contentType": "application/json"
  }'
```

Probar como admin:

bash

```
curl "http://localhost:3000/api/v1/data?user=admin"
```

Probar como usuario normal:

bash

```
curl "http://localhost:3000/api/v1/data?user=normal"
```

---

7. Mock con Variables Dinámicas

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/timestamp",
    "method": "GET",
    "statusCode": 200,
    "responseBody": "{\\"timestamp\\": \\"{{timestamp}}\\", \\"uuid\\": \\"{{uuid}}\\", \\"random\\": {{random.number}}}",
    "contentType": "application/json"
  }'
```

Probar:

bash

```
curl http://localhost:3000/api/v1/timestamp
```

---

8. Mock con Diferentes Content-Types

XML Response:

bash

```
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/xml-data",
    "method": "GET",
    "statusCode": 200,
    "responseBody": "<?xml version=\"1.0\"?><data><message>Hello XML</message></data>",
    "contentType": "text/xml"
  }'
```

Plain Text Response:

bash

```bash
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/text-data",
    "method": "GET",
    "statusCode": 200,
    "responseBody": "Simple text response",
    "contentType": "text/plain"
  }'
```

---

9. Mock con Códigos de Error

bash

```bash
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/error",
    "method": "GET",
    "statusCode": 500,
    "responseBody": {"error": "Internal server error", "code": 500},
    "contentType": "application/json"
  }'
```

Probar:

bash

```bash
curl http://localhost:3000/api/v1/error
```

🔍 Comandos de Gestión

Ver todas las configuraciones:

bash

```bash
curl http://localhost:3000/configure-mock
```

Ver estadísticas:

bash

```
curl http://localhost:3000/configure-mock/stats
```

Eliminar configuración específica:

```
bash

# Primero obtén el ID con GET /configure-mock
curl -X DELETE http://localhost:3000/configure-mock/{ID_DE_LA_CONFIGURACION}
```

Limpiar todas las configuraciones:

```
bash

curl -X DELETE http://localhost:3000/configure-mock
```

🎯 Casos de Uso Avanzados

10. Mock para API REST Completa

Crear Usuario (POST):

```
bash

curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/users",
    "method": "POST",
    "statusCode": 201,
    "responseBody": {"id": "{{uuid}}", "name": "{{body.name}}", "email": "{{body.email}}", "created": "{{timestamp}}"},
    "contentType": "application/json"
  }'
```

Probar:

```
bash

curl -X POST http://localhost:3000/api/v1/users \
  -H "Content-Type: application/json" \
  -d '{"name": "Juan Pérez", "email": "juan@test.com"}'
```

Obtener Usuario (GET):

bash

```bash
curl -X POST http://localhost:3000/configure-mock \
  -H "Content-Type: application/json" \
  -d '{
    "route": "/api/v1/users/123",
    "method": "GET",
    "statusCode": 200,
    "responseBody": {"id": "123", "name": "Juan Pérez", "email": "juan@test.com"},
    "contentType": "application/json"
  }'
```

📝 Notas Importantes

1. **Orden de Prioridad**: Los mocks se evalúan en el orden que fueron creados
2. **Matching Exacto**: Todos los parámetros configurados deben coincidir exactamente
3. **Variables Disponibles**: `{{timestamp}}`, `{{uuid}}`, `{{random.number}}`, `{{query.param}}`, `{{body.param}}`, `{{header.param}}`
4. **Condicionales**: Formato `{{if condition}}true_content{{else}}false_content{{/if}}`
5. **Content-Type**: El header se establece automáticamente según la configuración

¡Ahora tienes una guía completa para probar todas las funcionalidades del API Mock! 🚀