Universidad de Valladolid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Trabajo de Fin de Grado

Grado en Ingeniería de Tecnologías de Telecomunicación

Implementación de un Servidor HTTP para Emular un Prototipo del Sistema Request To Pay del EPC

Autor: Alonso
Sandoval Martínez
Tutores:
D. Tutor1
D. Tutor2

Valladolid, MES 202X

Índice

Agradecimientos

Agradezco a mis tutores, familia y amigos por su apoyo durante la realización de este trabajo. Su orientación y motivación han sido fundamentales para completar este proyecto.

Resumen

Este trabajo presenta la implementación de Como se muestra en [**?**], el protocolo SEPA permite... un servidor HTTP para emular un prototipo del sistema Request To Pay desarrollado por el European Payments Council (EPC), abordando...

Palabras clave— Servidor HTTP, Request To Pay, EPC, Telecomunicaciones, Pagos Electrónicos

1 Introducción

Para comprender el entorno actual de los pagos en Europa, conviene arrancar por la *Single Euro Payments Area* (SEPA): un espacio comunitario en el que todos los pagos en euros se rigen por los mismos estándares técnicos y normas operativas, de modo que enviar dinero de un país a otro resulta tan ágil y claro como una transferencia nacional. SEPA estableció protocolos de mensajería comunes, armonizó los plazos de liquidación y fijó reglas uniformes de protección al usuario, creando la base sobre la que se despliegan hoy los servicios de pago más innovadores.

En los últimos diez años, la digitalización de los servicios financieros ha cambiado por completo cómo particulares y empresas gestionan sus transacciones dentro de ese marco SEPA. Las transferencias instantáneas, las API abiertas de los bancos y el auge del comercio electrónico han disparado la demanda de procesos de cobro que sean sencillos, transparentes y en tiempo real. No obstante, los instrumentos de pago tradicionales —tarjetas, transferencias convencionales o domiciliaciones— nacieron en un contexto muy distinto y todavía arrastran limitaciones que penalizan tanto la experiencia de usuario como la eficiencia operativa.

Aquí es donde entra en juego *Request-to-Pay* (RTP). Este servicio de mensajería permite al beneficiario enviar al pagador una solicitud de pago digital estructurada, con todos los detalles (importe, concepto, vencimiento), y recibir en segundos una respuesta —aceptación, rechazo o aplazamiento— antes de iniciar el movimiento de fondos. RTP no sustituye los métodos de pago existentes, sino que actúa como una capa de orquestación sobre la infraestructura SEPA (y, en especial, los pagos inmediatos) y los canales de banca online, facilitando la conciliación, reduciendo la fricción en el cobro y modernizando la experiencia tanto para empresas como para consumidores.

1.1 Motivación

La domiciliación bancaria regulada por el esquema *SEPA Direct Debit* (**SDD**)[1] desde 2014, sigue siendo el método principal para cobros recurrentes en España. No obstante, su estructura, pensada para un entorno de procesos *offline*—genera hoy inconvenientes que chocan con las demandas de inmediatez, seguridad y experiencia de usuario fluida que caracterizan la economía digital actual.

1.1.1 Ineficiencias operativas detectadas

La Tabla 1 resume los siete ejes problemáticos identificados en la operativa SDD nacional.

[1] *SEPA Direct Debit* es el instrumento paneuropeo de cargo en cuenta regulado por el *European Payments Council*.

Consecuencias agregadas. Estas ineficiencias se traducen en (i) estructura de costes elevada por devoluciones y personal especializado; (ii) liquidez incierta—los ingresos se confirman con días de retraso y pueden desaparecer meses después—y (iii) freno a la economía digital, incapaz de ofrecer experiencias de pago instantáneas comparables a tarjeta, monederos o Bizum.

1.1.2 Oportunidad de un esquema Request-to-Pay

El estándar *SEPA Request-to-Pay* (**SRTP**)[7] emerge para modernizar los pagos de cuenta-a-cuenta. Sus aportaciones clave frente al SDD tradicional son:

a) **Autenticación reforzada en línea** (**SCA**), eliminando la firma previa de mandatos.

b) **Mandato digital implícito**: la aceptación del *request* actúa como consentimiento ejecutable.

c) **Carácter irrevocable** tras la aprobación, acotando las devoluciones *post-servicio*.

d) **Liquidación instantánea** mediante **SCT InstantSCT Instant**: transferencia inmediata SEPA con disponibilidad de fondos ($\leq 10\,s$)., que proporciona certidumbre financiera en segundos.

e) **Reducción significativa de las *R-transactions*** y simplificación del *back-office* operativo.

En síntesis, SRTP traslada las ventajas históricas de la domiciliación—bajo coste y alcance paneuropeo—al entorno digital *real-time*, resolviendo los siete puntos críticos que hoy lastran la competitividad del SDD en España y, por extensión, en la zona SEPA.

1.2 Objetivos

El objetivo general del trabajo es desarrollar un servidor HTTP que emule un proveedor SRTP de extremo a extremo, cumpliendo con:

1. el *SRTP Scheme Rulebook* v4.0,
2. las especificaciones de mensajes en formato JSON descritas en `EPC137-22`, y

[6]El mandato es la orden firmada por el pagador que autoriza futuros cargos.
[6]Mandato electrónico interoperable.
[6]**CORE**: esquema SDD aplicable a consumidores.
[6]Operaciones rechazadas, devueltas o revocadas
[6]**SCA**: *Strong Customer Authentication* exigida por PSD2.
[7]Iniciativa del European Payments Council que define un flujo de solicitud y aceptación de pagos en tiempo real, independiente del medio de liquidación subyacente.

3. los requisitos de identificación, autenticación y autorización del *API Security Framework*.

Como metas específicas se plantean: (i) exponer endpoints REST para los flujos genéricos (creación, rechazo, respuesta y cancelación de un SRTP); (ii) implementar validación de mensaje y sellado temporal; (iii) generar casos de prueba automáticos con Postman; y (iv) documentar las divergencias entre la norma y el prototipo.

1.3 Fases y Métodos

El proyecto se aborda en tres iteraciones:

Fase 1 – Análisis Revisión de los documentos EPC, extracción de requisitos funcionales y de seguridad, y modelado de los actores (Payee, Payer, PSPs) en un diagrama de cuatro esquinas.

Fase 2 – Diseño e implementación Arquitectura Node.js con Express y Socket.IO para notificaciones push; módulo de firma/validación X.509; base de datos ligera (SQLite) para persistir solicitudes.

Fase 3 – Pruebas y validación Colección Postman con pruebas unitarias y de integración, validación de esquemas JSON contra *OpenAPI* y simulación de retrasos/redirecciones para cubrir los flujos síncrono y asíncrono.

Cada iteración finaliza con una reunión de revisión y un backlog de mejoras, siguiendo principios ágiles (*time-boxed sprints* de dos semanas).

1.4 Medios necesarios empleados para el desarrollo

- **Software**: Node.js 20 LTS, Express, Jest, Postman, Docker Desktop y Git para control de versiones.
- **Documentación**: EPC014-20 v4.0 (Rulebook), EPC137-22 (API specs), EPC164-22 (API Security) y guías ISO 20022.
- **Hardware / SO**: Portátil con procesador x86-64, 16 GB RAM, sistema operativo Linux (Ubuntu 22.04) y conexión a Internet.
- **Otras herramientas**: directorio de certificados de prueba (OpenSSL), contenedor nginx como *reverse proxy* TLS.

1.5 Estructura del documento

El resto de la memoria se organiza así:

1. **Antecedentes**: estado del arte en pagos SEPA, directiva PSD2 e ISO 20022.

2. **Diseño e implementación**: descripción detallada de la arquitectura, modelos de datos y endpoints.

3. **Métodos y experimentación**: plan de pruebas, métricas de rendimiento y escenarios de interoperabilidad.

4. **Resultados y discusión**: análisis de cumplimiento de requisitos y comparación con soluciones comerciales.

5. **Conclusiones y trabajos futuros**: aportaciones, posibles extensiones (FIDO2, pagos fraccionados) y líneas de investigación.

6. **Bibliografía y anexos**: normativa EPC, scripts de prueba y guía de instalación del prototipo.

2 Antecedentes y estado del arte

Esta sección proporciona una visión general del contexto técnico y normativo del proyecto...

2.1 Visión general del sistema *Request To Pay*

El sistema *Request To Pay* es una iniciativa del EPC para facilitar los pagos electrónicos...

2.2 Tecnologías de servidores HTTP

Se revisan las principales tecnologías y frameworks utilizados para desarrollar servidores HTTP...

2.3 Prototipos y estándares del EPC

El EPC ha establecido una serie de estándares que se tomaron como referencia para este proyecto...

3 Diseño e Implementación

Aquí se detalla el proceso de diseño e implementación del servidor HTTP...

3.1 Arquitectura del servidor HTTP

3.1.1 Estructura y funcionamiento

El servidor se diseñó para manejar solicitudes y respuestas HTTP de manera eficiente...

3.1.2 Protocolos y estándares implementados

Se implementaron protocolos como HTTP/1.1 y se consideraron estándares de seguridad...

3.2 Emulación del prototipo *Request To Pay*

3.2.1 Diseño de las funcionalidades

Se replicaron las funcionalidades clave del sistema *Request To Pay*...

3.2.2 Modelos de procesamiento de solicitudes

La lógica de negocio se diseñó para procesar solicitudes de pago...

3.3 Herramientas de desarrollo

3.3.1 Lenguajes y frameworks

Se utilizó [especificar lenguaje/framework, e.g., Node.js] para el desarrollo...

3.3.2 Pruebas y validación

Se emplearon herramientas como Postman para validar la funcionalidad del servidor...

4 Métodos

Esta sección describe los métodos empleados para desarrollar y evaluar el sistema...

4.1 Preparación de los datos de entrada

4.1.1 Simulación de solicitudes HTTP

Se simularon solicitudes HTTP utilizando herramientas como curl...

4.1.2 Configuración de escenarios de prueba

Se definieron escenarios de prueba que cubren casos de uso típicos...

4.2 Evaluación del sistema

4.2.1 Métricas de rendimiento

Se evaluaron métricas como latencia y *throughput*...

4.2.2 Diseño de pruebas

Se diseñaron pruebas unitarias, de integración y de carga...

4.2.3 Análisis de resultados

Los resultados obtenidos se analizaron para identificar cuellos de botella...

4.3 Optimización del servidor

Se implementaron mejoras para optimizar el rendimiento del servidor...

4.4 Validación cruzada de la implementación

La implementación se comparó con los estándares del EPC...

5 Resultados y discusión

Esta sección presenta los resultados obtenidos y su análisis...

5.1 Presentación de los resultados

5.1.1 Comparativa con los estándares del EPC

La implementación cumple con los requisitos establecidos por el EPC...

5.1.2 Análisis de casos de uso

Se analizaron casos de uso reales para evaluar el comportamiento del sistema...

5.2 Discusión de los resultados

Se discuten las implicaciones de los resultados y las limitaciones encontradas...

6 Conclusiones y líneas futuras

En esta sección se resumen los logros y se proponen mejoras futuras...

6.1 Conclusiones

El proyecto logró implementar con éxito un servidor HTTP para *Request To Pay*...

6.2 Limitaciones y Líneas Futuras

Entre las limitaciones se encuentra la escalabilidad del sistema, que podría mejorarse...

Tabla 1: Principales ineficiencias del SDD en España

Eje del problema	**Qué ocurre**	**Impacto operativo / de negocio**
Mandato *offline*[2]y su custodia	El pagador debe firmar un mandato. El acreedor deberá archivarlo durante toda la vida del contrato + 13 meses tras el último cargo. No existe un estándar *eMandate*[3]; cada banco aceptaa formatos distintos.	Fricción en la venta digital (el usuario abandona o rechaza la firma). Costes de archivado y riesgo legal si el mandato no se localiza en caso de devolución.
Cobro lento y sin confirmación en tiempo real	Plazos CORE[4]D-5 primera/única y D-2 recurrente; liquidación D+1–D+2. Sin certeza de fondos hasta la compensación.	Se presta el servicio antes de saber si hay fondos, por tanto hay necesidad de provisiones por fallos de cobro y por tanto imposibilidad de vender bienes digitales o físicos con entrega inmediata y riesgo cero
Derecho de devolución amplio	El deudor puede devolver: 8 semanas si había mandato; 13 meses si no lo había.	Ingresos "no definitivos"durante 13 meses, costes indirectos de recobro y reputación. El cliente puede beneficiarse del servicio y recuperar.[el] dinero.
Complejidad de las *R-transactions*[5]	Hay distintos códigos de R-transactions y cada una de ellas implica un flujo distinto lo que dificulta mucho el proceso; Los bancos repercuten comisión por cada devolución.	Se necesitan equipos específicos de conciliación y recobro lo que genera un coste directo y una gran pérdida de productividad.
Fraude e impago *post-servicio*	El cliente puede consumir el servicio y ordenar la devolución dentro de plazo (*friendly fraud*).	Pérdida del servicio/producto y del importe; disputas difíciles: el banco prioriza al cliente salvo mandato perfecto.
Ausencia de Autenticación Reforzada (**SCA**)[6]	El cargo se basa en consentimiento previo; no hay SCA en el momento de pago.	Mayor riesgo de disputas y desaprovechamiento de identidad digital (OTP, biometría).
Limitaciones B2B y *cross-border*	El esquema B2B elimina la devolución "sin preguntas", pero exige validación previa y apenas lo usan particulares.	Complejo de habilitar y poco escalable fuera de España por la heterogeneidad bancaria.

A Códigos

```
fID1 = fopen('myoutput1.txt','r');
for n = 1:4
    b = fscanf(fID1,'%7u %7u %7u \r',3);
    btotal = b(1)+b(2)+b(3);
    fprintf('%7u + %7u + %7u = %7u \r', b(1), b(2), b(3), btotal)
end
```

Listing 1: Código de ejemplo.

Índice de Figuras

Índice de Tablas

Índice de Códigos