



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	3
Resumen	4
1 Introducción	5
1.1 Motivación	5
1.1.1 Ineficiencias operativas detectadas	5
1.1.2 Oportunidad de un esquema Request-to-Pay	7
1.2 Objetivos	8
1.3 Fases y Métodos	8
1.4 Medios necesarios empleados para el desarrollo	9
2 Antecedentes y estado del arte	10
2.1 Evolución de los medios de pago hacia SEPA	10
2.2 El papel del EPC en la estandarización y el surgimiento de Request-to-Pay	10
2.3 Funcionamiento técnico de SEPA Request-to-Pay (SRTP)	11
2.4 RTP dentro del ecosistema de pagos: arquitectura en capas	20
3 Diseño e Implementación	25
3.1 Fundamentos Teóricos	25
3.2 Tecnologías utilizadas	26
3.2.1 Estructura y funcionamiento	27
3.2.2 Protocolos y estándares implementados	27
3.3 Emulación del prototipo <i>Request To Pay</i>	27
3.3.1 Diseño de las funcionalidades	27
3.3.2 Modelos de procesamiento de solicitudes	27
3.4 Herramientas de desarrollo	27
3.4.1 Lenguajes y frameworks	27
3.4.2 Pruebas y validación	27
4 Métodos	28
4.1 Preparación de los datos de entrada	28
4.1.1 Simulación de solicitudes HTTP	28
4.1.2 Configuración de escenarios de prueba	28
4.2 Evaluación del sistema	28
4.2.1 Métricas de rendimiento	28
4.2.2 Diseño de pruebas	28
4.2.3 Análisis de resultados	28
4.3 Optimización del servidor	28
4.4 Validación cruzada de la implementación	28

5	Resultados y discusión	29
5.1	Presentación de los resultados	29
5.1.1	Comparativa con los estándares del EPC	29
5.1.2	Análisis de casos de uso	29
5.2	Discusión de los resultados	29
6	Conclusiones y líneas futuras	30
6.1	Conclusiones	30
6.2	Limitaciones y Líneas Futuras	30
A	Códigos	31
	Índice de Figuras	33
	Índice de Tablas	34
	Índice de Códigos	35

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

Tras analizar la operativa SDD nacional se han detectado una serie de ineficiencias resumidas en los siguientes 5 puntos:

1. Modelo off-line y necesidad de mandato físico

- El pago se inicia sin interacción *online*; El deudor debe firmar y remitir **mandato SEPA**² y el acreedor debe custodiarlo durante el periodo de vida del contrato y hasta 14 meses tras la última transacción.
- No existe un estándar eMandate³ interoperable, cada banco utiliza su propio estándar.

Consecuencias: Fricción en la venta digital, costes de back-office (archivado, auditorías y gestión) y riesgo legal si el mandato no aparece ante una devolución.

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

²Autorización que un deudor otorga a un acreedor para permitir el cobro automático de pagos mediante SDD.

³Un eMandate es la versión digital de un mandato.

2. Derecho a devolución prolongado

- *Cobro autorizado*: el deudor dispone de **8 semanas** para devolver sin causa (“*no-questions-asked*”).
- *Cobro no autorizado*: hasta **13 meses** para reclamar si el banco emisor no puede demostrar mandato válido.

Consecuencias: gran incertidumbre sobre la firmeza del ingreso, reservas de liquidez, provisiones contables y alto nivel de *friendly fraud*⁴.

3. Ciclos de cobro lentos

- En esquema CORE⁵: envío al banco **D-5** para primera domiciliación y **D-2** para recurrencias; liquidación interbancaria **+2 días**.
- 6–8 días naturales entre petición y abono firme, incompatibles con venta inmediata o entrega digital.

Consecuencias: Tesorería imprevisible (cash-flow) y riesgo de prestar servicio sin cobro confirmado.

4. Costes y complejidad de las R-transactions⁶

- Existen distintos códigos de R-transactions y cada uno de ellos implica un flujo distinto, lo que dificulta la gestión.

Consecuencias: Necesidad de equipos específicos de conciliación y recobro, lo que conlleva un coste directo y pérdida de productividad.

5. Ausencia de autorización fuerte (SCA)⁷

- El SDD se basa en consentimiento previo, no aplica SCA al momento del cargo.

Consecuencias: Mayor riesgo de cargos disputados y se desaprovechan métodos de identidad digital ya existentes.

En conclusión. 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 online, incapaz de ofrecer experiencias de pago instantáneas comparables a tarjeta, monederos o Bizum.

⁴servicio consumido y posterior devolución.

⁵CORE es el esquema estándar de domiciliación bancaria SEPA para pagos entre empresas y consumidores.

⁶transacciones de rechazo, devolución o reembolso asociadas a pagos fallidos o no autorizados.

⁷La SCA (Strong Customer Authentication) es un mecanismo de seguridad que exige verificar al usuario con al menos dos factores.

1.1.2 Oportunidad de un esquema Request-to-Pay

El estándar *SEPA Request-to-Pay* (SRTP)⁸ se perfila como la evolución natural de la domiciliación SEPA. Sus ventajas técnicas frente al SDD son:

- a) **Autenticación reforzada y consentimiento digital inmediato**
El acreedor emite un request que el deudor aprueba en su banca o *wallet* mediante SCA. Este gesto sustituye al mandato físico y genera una prueba electrónica de consentimiento, firmada y almacenada dentro del PSP del pagador.
- b) **Irrevocabilidad y mitigación de fraude *post-servicio***
Tras la aceptación, el pago se realiza mediante SCT Inst⁹. Al no existir derecho de devolución automática, se elimina el *friendly fraud* asociado a la devolución de recibos y se reducen provisiones por impago.
- c) **Liquidez *real-time* y conciliación automática**
La disponibilidad de fondos en ≤ 10 s permite planificar tesorería al instante. Los identificadores de extremo a extremo y las referencias estructuradas ISO 20022 se transmiten sin perderse entre sistemas, de modo que la conciliación queda totalmente automatizada.
- d) **Simplificación operativa**
Desaparecen las R-TRANSACTIONS, la custodia de mandatos y las tareas de back-office. El flujo se limita a dos mensajes (request y aceptación) y, opcionalmente, una transferencia instantánea, con clara trazabilidad extremo a extremo.
- e) **Flexibilidad comercial y costes reducidos**
SRTP admite cobros únicos, recurrentes y fraccionados desde web o app vía enlaces profundos, QR o API, y al ser un pago mediante SCT Inst, las comisiones bancarias son muy inferiores a las de tarjeta o a las de gestión de devoluciones SDD.

En síntesis, SRTP traslada las ventajas históricas de la domiciliación—bajo coste y cobertura paneuropea—al entorno digital e inmediato, resolviendo los puntos técnicos que hoy limitan la competitividad del SDD en España y en la zona SEPA.

⁸Iniciativa del European Payments Council que define un flujo de solicitud (*request*) y aceptación de pagos en tiempo real, apoyado en mensajería **ISO 20022** (p. ej. `pain.013/pain.014`) y agnóstico respecto al instrumento de liquidación posterior.

⁹SCT INST: transferencia inmediata SEPA con liquidación 10 s.

Aspecto	SDD	SRTP (+ SCT Inst)
Autorización	Mandato off-line	Consentimiento digital (SCA) en tiempo real
Plazo de devolución	8 semanas / 13 meses	No aplica (operación irrevocable)
Disponibilidad de fondos	5–8 días	Segundos (<10 s)
Coste operativo	Alto (mandatos, R-CODES)	Bajo (mensajería ISO 20022, sin excepciones)
Cobertura <i>e-commerce</i>	Limitada	Óptima (API / móvil)
Riesgo de fraude	Medio-Alto (devolución)	Bajo (SCA + irreversibilidad)

1.2 Objetivos

El **objetivo general** del Trabajo Fin de Grado es entregar un *stack* de software que simule un proveedor *end-to-end* del esquema *SEPA Request-to-Pay* (SRTP), alineado con la versión 4.0 del *SRTP Scheme Rulebook* [?] y las guías técnicas del EPC (*EPC137* y *EPC164*) [?, ?]. El prototipo debe ser instalable con **Docker Compose**, exponer una API HTTP/JSON conforme a *OpenAPI 3.1* y cubrir las cuatro operaciones core (*create*, *reject*, *response*, *cancel*) del flujo SRTP.

Para acotar y medir el trabajo se definen las siguientes **metas específicas**:

1. Implementar en **Node 20 LTS/Express** los *endpoints* REST de alta disponibilidad y su contraparte *callback* para notificaciones asíncronas (**Socket.IO**).
2. Desarrollar un módulo de firma, sellado temporal y validación X.509 (QSeal/QWAC) para garantizar requisitos de *identificación*, *autenticación* y *autorización* definidos por el *API Security Framework* [?].
3. Persistir estado y auditoría en una base **SQLite** ligera (**SQLAlchemy**) con modelo de datos alineado a los *datasets* DS-02, DS-07 y DS-10 del Rulebook.
4. Entregar una colección **Postman** y un **runner** CI (GitHub Actions) que ejecute casos de prueba de integración, incluyendo validación de esquemas JSON contra *schemas* oficiales del EPC.
5. Documentar las divergencias norma → implementación y proponer una hoja de ruta para su homologación futura en el *EDS*.

1.3 Fases y Métodos

Se adopta un ciclo **ágil**, con *sprints* de dos semanas y reuniones *review/retro*. Cada iteración termina con un incremento funcional desplegado en **Docker Hub** y su etiqueta asociada en **Git**.

Fase 1 – Análisis • Lectura detallada de los documentos EPC 014, 137 y 164 y extracción de requisitos funcionales, de seguridad y de interoperabilidad.

- Modelado de actores en un diagrama de cuatro esquinas (Payee / Payer / PSP_{Payee} / PSP_{Payer}), identificando puntos de confianza y certificados requeridos.

- Priorización de *user-stories* y definición de *Definition of Done*.

Fase 2 – Diseño e implementación • Arquitectura **Clean Architecture** sobre **Express**: `routes.js`, `services.js`, `models.py`.

- Middleware de firma y verificación con `crypto.subtle` y librerías **OpenSSL**; generación de certificados de prueba `make cert`.
- Persistencia en **SQLite** mediante **Sequelize**; migraciones automáticas.
- Websocket **Socket.IO** encapsulado en `ext-socketio.py` para notificaciones *push* de estado.

Fase 3 – Pruebas y validación • Suite **Jest** + **supertest** para pruebas unitarias y de integración.

- Colección **Postman** con *scripts* pre/post-request que firman, estampan fecha y validan contra esquemas.
- Inyección de fallos con **toxiproxy**: retardos, caídas de red y respuestas 4xx/5xx para cubrir los flujos síncrono y asíncrono.
- Informe de cobertura y reporte **SonarQube** en el pipeline **CI**.

1.4 Medios necesarios empleados para el desarrollo

- **Software de desarrollo:** **Node.js** 20 LTS, **Express** 4, **Socket.IO** 4, **Sequelize** 6, **Jest**, **Postman** v10, **Docker** 24, **Docker Compose** v2, **Git** y **GitHub Actions**.
- **Herramientas de apoyo:** **OpenSSL** 3 para gestión de certificados, **toxiproxy** para pruebas de resiliencia, **Spectral** OCI para linting de especificaciones **OpenAPI**.
- **Documentación oficial:** **SRTP Scheme Rulebook** v4.0 [?], **SRTP related API Specifications** v3.1 [?], **API Security Framework** v2.0 [?], **ISO 20022** *pacs/pain/camt*, directivas **PSD2/eIDAS**.
- **Hardware y S.O.:** Portátil x86-64, 16 GB RAM, SSD 512 GB, **Ubuntu** 22.04 LTS, conexión simétrica de 300 Mbps; virtualización **Docker Desktop**.
- **Repositorios y control de versiones:** Organización privada en **GitHub**; **branch protection** y **semantic-versioning**.

2 Antecedentes y estado del arte

2.1 Evolución de los medios de pago hacia SEPA

El ecosistema europeo de pagos ha experimentado una profunda transformación en las últimas décadas, pasando de sistemas nacionales heterogéneos a un marco unificado bajo la iniciativa **SEPA**. Antes de SEPA, cada país operaba infraestructuras y normas propias para transferencias bancarias y adeudos, lo que complicaba los pagos transfronterizos dentro de Europa. Con la introducción del euro y el objetivo de un mercado único, surgió la necesidad de armonizar los instrumentos de pago. El Consejo Europeo impulsó la creación de SEPA a través del Reglamento (UE) 260/2012, que fijó la migración obligatoria a los nuevos esquemas paneuropeos de transferencia y adeudo en fechas límite (febrero de 2014 para la zona euro).

Así, en 2008 se lanzó el esquema **SEPA Credit Transfer (SCT)** para transferencias de crédito en euros, y en 2009 el **SEPA Direct Debit (SDD)** para adeudos domiciliados [?]. Estos esquemas sustituyeron progresivamente a los medios nacionales, unificando formatos (por ejemplo, el uso obligatorio de *IBAN*) y reglas de funcionamiento en todos los países SEPA. Posteriormente, para atender las demandas de inmediatez, la transferencia instantánea **SCT Inst** (*SEPA Instant Credit Transfer*) entró en funcionamiento en 2017, permitiendo abonar al beneficiario en menos de 10 s. La implantación de SCT Inst ha sido voluntaria hasta ahora, aunque recientemente la UE ha aprobado su obligatoriedad progresiva en 2025 para acelerar su adopción [?].

En la actualidad, los esquemas SEPA (transferencias estándar e inmediatas, adeudos básicos y B2B) concentran la mayoría de pagos bancarios en euros dentro de Europa [?]. Este salto hacia la unificación de pagos fue liderado por la propia industria bancaria europea. En 2002 los bancos constituyeron el **European Payments Council (EPC)**, órgano de autorregulación que diseña y gestiona los esquemas SEPA. El EPC publicó las primeras *rulebooks* de SCT y SDD en 2008–2009, estableciendo estándares comunes de mensaje (*ISO 20022*¹⁰) y calendarios de liquidación. Cabe destacar que el EPC no es un organismo legislativo de la UE ni un regulador, sino una asociación del sector bancario que actúa de facto como ente normalizador: especifica las reglas de los esquemas utilizados por los **PSP** (*Payment Service Providers* o proveedores de servicios de pago) y coopera con los bancos centrales para operar las infraestructuras de compensación [?]. Gracias a esta colaboración público-privada, a partir de 2014 se completó con éxito la migración de millones de pagos nacionales al formato SEPA, eliminando diferencias entre pagos domésticos y transfronterizos en euros.

2.2 El papel del EPC en la estandarización y el surgimiento de Request-to-Pay

El **European Payments Council (EPC)** ha desempeñado un rol central en la estandarización de los instrumentos de pago SEPA. Tras la implementación de SCT y SDD, el EPC continuó explorando mejoras para la era digital, en línea con las iniciativas del Eurosistema para fomentar pagos electrónicos paneuropeos más eficientes. En noviembre de 2018, el *Euro Retail Payments Board (ERPB)*¹¹ —órgano del BCE que orienta la estrategia de pagos minoristas— lanzó un

¹⁰ISO 20022: estándar internacional para el intercambio de mensajes financieros, basado en XML, empleado en los esquemas SEPA para definir las estructuras de datos de transferencias, adeudos, etc.

¹¹ERPB: foro de alto nivel presidido por el BCE que reúne a autoridades y sector financiero para impulsar la integración y modernización de los pagos minoristas en Europa.

llamado a la acción para desarrollar el concepto de *Request to Pay* (R2P) como nuevo servicio en la zona SEPA [?].

Atendiendo esta petición, el EPC creó un grupo de trabajo y comenzó a diseñar un esquema formal de Request to Pay durante 2019–2020. Tras una consulta pública, en noviembre de 2020 se publicó el primer *SRTP Scheme Rulebook* (versión 1.0) y se abrió el registro de participantes. El esquema **SEPA Request-to-Pay (SRTP)** entró en vigor el *15 de junio de 2021*, marcando un hito en la evolución de SEPA más allá de los instrumentos tradicionales [?]. Al igual que SCT Inst, la adhesión al esquema RTP es voluntaria; sin embargo, su desarrollo cuenta con fuerte apoyo institucional al considerarse un potenciador de los pagos instantáneos y digitales en Europa. El EPC continúa gestionando y actualizando el esquema (versión 4.0 en 2023), con la expectativa de que Request-to-Pay se integre gradualmente como componente clave del panorama de pagos europeos.

2.3 Funcionamiento técnico de SEPA Request-to-Pay (SRTP)

Request-to-Pay (RTP) es un servicio de mensajería financiera que actúa como capa de solicitud previa al pago. A diferencia de los instrumentos tradicionales (transferencias o adeudos) que mueven fondos, RTP no mueve dinero por sí mismo: permite a un beneficiario (*Payee*) enviar electrónicamente una solicitud de pago a un pagador (*Payer*), quien puede aceptarla o rechazarla antes de iniciarse la transacción monetaria [?]. El servicio funciona 24×7 y añade al flujo de pago un intercambio estructurado de datos (importe, concepto, vencimiento, identidad de las partes, etc.) previo al envío de fondos.

Los mensajes SRTP viajan en tiempo real formateados según ISO 20022¹², lo que facilita su integración con las plataformas SEPA existentes.

Modelo de cuatro esquinas. SRTP adopta la clásica arquitectura *4-corner model*:

¹²Las mensajes SRTP siguen las definiciones ISO 20022 específicas del esquema, permitiendo interoperabilidad con mensajes de pago **pacs/pain** de SCT/SCT Inst.

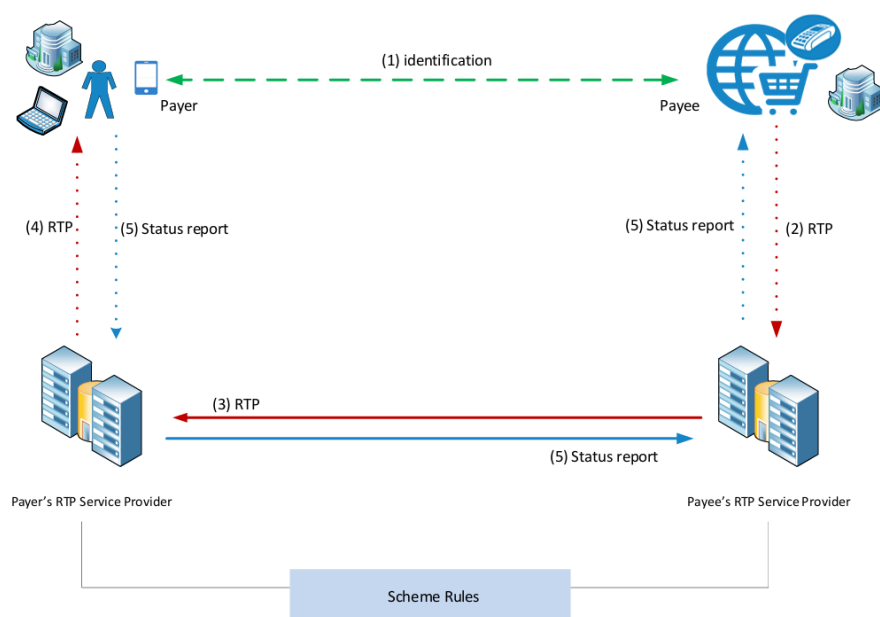


Figura 1: 4CornerModel

Cuadro 1: Pasos del flujo del esquema SEPA Request-to-Pay (SRTP) y su descripción

Paso	Descripción
1. Identificación	Una primera interacción que permite establecer la comunicación entre pagador y beneficiario a través de sus PSP.
2. Envío SRTP al PSP del beneficiario	El beneficiario envía la SRTP a su propio proveedor de servicios. Este mensaje contiene todos los datos esenciales del esquema RTP.
3. Envío SRTP al PSP del pagador	El PSP del beneficiario retransmite la SRTP al PSP del pagador.
4. Presentación SRTP al pagador	La SRTP se muestra al pagador en el canal o dispositivo acordado (aplicación móvil, navegador web, etc.).
5. Informe de estado	La aceptación o el rechazo de la SRTP por parte del pagador se envía de vuelta al beneficiario a través de sus respectivos PSP.

El Operational Scheme Manager (OSM) es la pieza operativa que el EPC ha previsto para ofrecer un servicio centralizado de directorio -el EPC Directory Service (EDS) que facilite la interoperabilidad y el alcance mutuo entre los participantes del esquema SRTP. La OSM actúa como registro autorizado de todos los PSP y demás actores adheridos al esquema y su misión es garantizar que, en todo momento, cualquier PSP pueda localizar el endpoint correcto del PSP objetivo, verificar que está homologado y enviar de forma segura los mensajes SRTP. Por tanto, para que un PSP

pase a formar parte activa del esquema SEPA Request-to-Pay deberá en primer lugar superar con éxito las pruebas de conformidad ISO 20022 y obtener la homologación técnica, tras lo cual acceder al portal de la Operational Scheme Manager para registrar su ficha operativa —incluyendo URL de endpoint, roles SRTP, certificados TLS, claves de firma y datos de contacto 24×7— y aceptar los términos de uso del EPC Directory Service, de modo que la OSM valide la disponibilidad y correcta respuesta de sus endpoints y publique la información correspondiente en el EDS público, quedando accesible al resto de participantes; finalmente, el PSP actualiza sus canales de banca electrónica y/o sus APIs de iniciación de pagos para presentar las solicitudes RTP a sus usuarios, garantizar la autenticación fuerte conforme a PSD2 y gestionar de forma integrada los flujos de aceptación, rechazo y cancelación

Flujo básico. El flujo básico de un intercambio RPT es el siguiente:

Paso/Función	Descripción
1 Crear y enviar RTP	El beneficiario crea el "SEPA Request to Pay" SRTP en el formato normalizado (en un formato acordado bilateralmente con su proveedor). Contiene todos los elementos obligatorios y elementos opcionales que puedan ajustarse al flujo en función de las condiciones comerciales. El beneficiario lo envía al proveedor de servicios SRTP del beneficiario.
2 Validar RTP	El proveedor de servicios SRTP del beneficiario realiza una primera validación del SRTP. Esto incluye, por ejemplo, validación técnica, de seguridad y de formato (por ejemplo, comprobación del IBAN).
2B Rechazo	Si la validación en el paso 2 no tiene éxito, el proveedor de servicios SRTP del beneficiario notifica al beneficiario el rechazo del SRTP, crea un informe de estado negativo y lo envía al beneficiario en el formato acordado con este.
3 Completar y enviar RTP	En caso de validación correcta en el paso 2, el proveedor de servicios SRTP del beneficiario enriquece el SRTP con los elementos necesarios para el enrutamiento en el espacio entre proveedores de servicios SRTP y añade un sello de tiempo.
4 Enrutar RTP	El SRTP se envía al proveedor de servicios SRTP del pagador en función de los mecanismos de enrutamiento establecidos por el PSP.
5 Validar RTP	El proveedor de servicios SRTP del pagador valida el SRTP, incluye la comprobación del identificador del pagador. Esto puede incluir la validación específica del pagador (por ejemplo, si el pagador ha optado por no participar en el servicio, el SRTP es rechazado por defecto).
5B Rechazo	Si la validación en el paso 5 no tiene éxito, el proveedor de servicios SRTP del pagador rechaza el SRTP. El proveedor de servicios SRTP del beneficiario y el beneficiario son informados de este rechazo mediante un código de motivo de no aceptación del RTP.
5C Confirmación positiva funcional	Después de una validación externa en el paso 5, el proveedor de servicios SRTP confirma al pagador que el proveedor de servicios SRTP ha completado con éxito el procedimiento benefical. Esta confirmación es obligatoria solo en el caso de que el beneficiario o el proveedor de servicios SRTP no haya confirmado previamente la positividad funcional.
6 Enviar RTP	En el caso de validación correcta en el paso 5, el proveedor de servicios SRTP envía el documento SRTP al pagador en el formato acordado (el SRTP puede ser convertido en este paso).
7 Evaluar RTP	El pagador decide si aceptar o rechazar el SRTP, determinando el próximo curso de acción.
7B Positivo	Si el pagador decide aceptar el SRTP, se envía una respuesta positiva al proveedor de servicios SRTP por parte del pagador.
<i>(continúa en la siguiente página)</i>	

Paso/Función	Descripción
7C Negativo	Si el pagador rechaza el SRTP, se envía una respuesta negativa al proveedor de servicios SRTP por parte del pagador.
8 Crear/modificar y mandar informe RTP	El proveedor de servicios SRTP crea un informe informativo basado en la decisión del pagador. Si la decisión es negativa (7B/7C), el informe se envía de vuelta al pagador para su revisión. Si el SRTP ya ha sido aceptado o rechazado, surge un caso excepcional donde no se espera un acuse de recibo. El informe debe considerar la fecha de expiración del SRTP. En este caso, el proveedor de servicios SRTP es responsable de notificar al beneficiario la decisión del proveedor de servicios SRTP, incluyendo el código correspondiente. Como resultado, el beneficiario debe representar el SRTP o utilizar otro canal.
9 Enviar informe de estado	El proveedor de servicios SRTP envía un informe de estado actualizado (positivo o negativo) al pagador a través del mismo canal utilizado para el SRTP original, utilizando mecanismos establecidos para la actualización.
10 Proceder y remitir informe	El proveedor de servicios SRTP del beneficiario procesa el informe de estado recibido (positivo o negativo), informa al beneficiario y decide los pasos siguientes previo acuerdo con el beneficiario.
11 Procesar informe de estado	El beneficiario ejecuta las acciones finales tras la recepción del informe de estado: actualización del estado final del registro SRTP, preparación del pago SRTP conciliación, etc.
12 Generar solicitud de actualización de estado	El beneficiario y el proveedor de servicios SRTP del beneficiario pueden enviar una Solicitud de Actualización de Estado si no se ha recibido respuesta hasta la Fecha/Hora de Expiración.
13 Enrutar la solicitud de actualización de estado	La solicitud de actualización de estado al proveedor de servicios SRTP del pagador se enruta a través de la misma vía utilizada para el SRTP original basándose en los mecanismos de enrutamiento establecidos.
14 Validar solicitud de actualización de estado	Tras la recepción de la solicitud de actualización de estado, el Proveedor de Servicios SRTP del pagador comprueba la validez de la Solicitud.
14B Respuesta a la solicitud de actualización de estado	El Proveedor de Servicios SRTP del pagador responde al proveedor de servicios SRTP del beneficiario y, si procede (a través del proveedor de servicios SRTP del beneficiario), al beneficiario (por ejemplo, respuesta del SRTP original no recibida, el pagador aún no ha respondido, etc.).
15 Enrutar la solicitud de actualización de estado	En caso de que el pagador aún no haya respondido al SRTP inicial, el proveedor de servicios SRTP del pagador puede enviar la solicitud de actualización de estado al Pagador.
<i>Fin de la tabla</i>	

Hay algunos detalles acerca del esquema que conviene aclarar:

1. Rechazo de un SRTP o una Solicitud de Cancelación (Reject)

Un Reject”se produce cuando un SRTP o una Solicitud de Cancelación no es aceptada antes de ser enviada al siguiente participante en la cadena de pago. El mensaje de rechazo sigue la misma ruta que el SRTP original sin modificar ningún dato, y se incluye un registro con los detalles necesarios para asegurar un rastro de auditoría. Además, el mensaje de rechazo lleva un código de motivo que explica la razón del rechazo. La identificación del SRTP original se hace mediante la referencia única incluida por el proveedor de servicios del receptor. Los rechazos se envían de manera instantánea por el proveedor de servicios SRTP que no puede procesar la solicitud, y estos rechazos pueden ser generados automáticamente en función de comprobaciones técnicas o comerciales, sin intervención del pagador.

2. Respuestas a un SRTP (positiva o negativa)

Cuando un pagador responde a un SRTP, puede aceptar (respuesta positiva) o rechazarlo (respuesta negativa). En ambos casos, el mensaje sigue la misma ruta que el SRTP original, sin alterar los datos, y se envía instantáneamente entre los proveedores de servicios SRTP. Las respuestas negativas incluyen un código de motivo que especifica la razón del rechazo, mientras que las respuestas positivas simplemente confirman la aceptación de la solicitud. Como en el caso de los rechazos, se mantiene un registro detallado de los datos relevantes para asegurar la trazabilidad del proceso y la transparencia en la comunicación entre los proveedores de servicios.

3. Solicitud de Cancelación del SRTP

Una “Request for cancelation” RfC puede ser iniciada por el y se transmite al pagador a través de los proveedores de servicios SRTP. La solicitud sigue la misma ruta que el SRTP original, sin modificar los datos, y debe incluir un código de motivo (atributo AT-R106) que justifique la cancelación. Esta solicitud puede realizarse hasta la fecha de expiración del SRTP, a menos que ya haya sido rechazado, cancelado o expirado. El proveedor de servicios SRTP del pagador verifica la validez de la solicitud antes de reenviarla, y si no se puede procesar, envía una respuesta negativa. Si la cancelación se ejecuta correctamente, el proveedor del pagador envía una respuesta positiva de manera instantánea, manteniendo siempre un registro para garantizar la trazabilidad del proceso.

Casos de uso representativos. El **SEPA Request-to-Pay** se concibió como un servicio versátil, capaz de cubrir desde pagos cotidianos de bajo valor hasta cobros empresariales complejos. No obstante, el caso de uso considerado más transformador—y sobre el que se centra este TFG—es la *sustitución del adeudo directo SEPA (SDD) en pagos recurrentes*. Aun así, existen otros escenarios relevantes que merece la pena describir para contextualizar el alcance potencial de SRTP.

1. Punto de Venta Físico (POS)

- **Descripción:** Este caso de uso de Request to Pay se emplea en tiendas físicas, donde el payee (el comercio) envía una solicitud de pago al payor (el cliente) utilizando un código QR o una tecnología NFC (Near Field Communication). Al escanear el código QR con su móvil o usar NFC en la terminal de pago, el cliente es redirigido a su aplicación bancaria para autorizar el pago de manera inmediata.
- **Proceso de Identificación:**

- *Identificación del Payor:* El payor se autentica directamente en su aplicación bancaria, generalmente mediante su número de cuenta bancaria, número de tarjeta, o métodos de autenticación biométrica o PIN, según lo permita la aplicación del banco.
- *Identificación del Payee:* El payee está identificado en el sistema mediante un ID de comercio asociado a la terminal de pago o al código QR/NFC escaneado por el cliente. Estos elementos proporcionan los datos necesarios para vincular la solicitud de pago al comercio correspondiente.

- **Proceso de Pago:**

- El cliente escanea el código QR o se conecta mediante NFC a la terminal de pago.
- La aplicación bancaria del cliente recibe la solicitud de pago con el monto y la referencia.
- El cliente revisa la información y autoriza el pago, un proceso rápido y conveniente, crucial en entornos físicos donde la velocidad es esencial.

- **Diferencias Clave:**

- La rapidez y conveniencia son fundamentales en este caso de uso. El proceso de identificación y autorización es sencillo y rápido, requiriendo solo la confirmación del cliente a través de su banco, típicamente con autenticación biométrica o PIN.
- Es ideal para transacciones de bajo valor donde la experiencia del cliente es un factor determinante.

2. Comercio Electrónico (E-commerce)

- **Descripción:** En este caso, el payee (el comercio electrónico) envía una solicitud de pago al payor (el cliente) durante el proceso de checkout o mediante un enlace de pago enviado a través de una aplicación bancaria o correo electrónico. El cliente es redirigido a su aplicación bancaria, donde debe autenticarse y revisar los detalles antes de aprobar el pago.

- **Proceso de Identificación:**

- *Identificación del Payor:* La identificación implica un proceso de autenticación multifactor (por ejemplo, contraseña, token de seguridad o autenticación biométrica) para garantizar la seguridad de la transacción, realizado en la aplicación bancaria o la página de pago del comercio.
- *Identificación del Payee:* El payee se identifica mediante un ID de comercio electrónico que incluye su nombre comercial, identificador fiscal, número de cuenta o un identificador único en la plataforma de pagos.

- **Proceso de Pago:**

- El cliente recibe la solicitud de pago a través de un enlace en el checkout o en su aplicación bancaria.
- El cliente se autentica en su aplicación bancaria, revisa el monto, la referencia y los detalles del comerciante, y aprueba el pago.
- Una vez validada, el dinero se transfiere de manera segura.

- **Diferencias Clave:**

- A diferencia del punto de venta físico, el proceso en comercio electrónico es más lento debido a la autenticación adicional y la revisión en línea.
- La seguridad es prioritaria, dado el mayor riesgo de fraude en transacciones digitales.

3. Facturación Electrónica (E-invoicing)

- **Descripción:** Común en empresas que envían facturas electrónicas, el payee (la empresa emisora) envía una solicitud de pago con los detalles de la factura al payor (el cliente). El cliente recibe la solicitud en su correo electrónico o aplicación bancaria, pudiendo revisar los detalles antes de decidir cuándo pagar.
- **Proceso de Identificación:**
 - *Identificación del Payor:* El payor se identifica mediante su número de cliente, correo electrónico o número de cuenta bancaria vinculado a la empresa emisora.
 - *Identificación del Payee:* El payee se identifica por su NIF (Número de Identificación Fiscal) o un ID de facturación electrónica único, asegurando la verificación de la entidad receptora.
- **Proceso de Pago:**
 - El cliente recibe la factura y la solicitud de pago por correo o notificación bancaria.
 - Tras validar la información, autoriza el pago a través de su aplicación bancaria, completando la transacción.
- **Diferencias Clave:**
 - Ofrece flexibilidad, permitiendo al cliente revisar la factura antes de pagar.
 - La identificación del payee incluye detalles fiscales, incrementando la seguridad en la validación.

4. Pagos Recurrentes

- **Descripción:** Ideal para suscripciones o pagos periódicos (streaming, software en la nube, gimnasios), el payor autoriza la primera transacción para suscribirse. Los pagos posteriores se gestionan automáticamente según la periodicidad acordada, sin intervención adicional.
- **Proceso de Identificación:**
 - *Identificación del Payor:* El payor autoriza la primera transacción con autenticación en su aplicación bancaria (cuenta, contraseña o biométrica).
 - *Identificación del Payee:* El payee se identifica por su ID de suscripción y número de cuenta para configurar los pagos recurrentes.
- **Proceso de Pago:**
 - El cliente autoriza el pago inicial.
 - Los cobros posteriores se realizan automáticamente según la periodicidad (mensual, anual, etc.).
- **Diferencias Clave:**
 - Se distingue por la automatización tras la autorización inicial, reduciendo la intervención del cliente.

- La comodidad y la automatización son sus principales ventajas.

5. Pagos de Grandes Montos

- **Descripción:** Utilizado en transacciones de alto valor (compras importantes, bienes de gran valor), este caso requiere métodos adicionales de autenticación para garantizar la seguridad.
- **Proceso de Identificación:**
 - *Identificación del Payor:* El payor usa autenticación robusta (OTP, tokens de seguridad o multifactor como PIN y biométrica) para validar la transacción.
 - *Identificación del Payee:* El payee es una entidad registrada con un identificador único (ID de comerciante), asegurando el destinatario correcto.
- **Proceso de Pago:**
 - El payor revisa el monto y autoriza el pago con autenticación adicional.
 - Tras verificar la identidad, el pago se completa.
- **Diferencias Clave:**
 - Requiere autenticación más fuerte debido al alto riesgo de fraude en grandes sumas.
 - La seguridad es el factor más crítico, priorizando la protección contra fraudes y errores.

2.4 RTP dentro del ecosistema de pagos: arquitectura en capas

El ecosistema de pagos SEPA (Zona Única de Pagos en Euros) se puede describir mediante capas jerárquicas, desde los usuarios finales hasta las infraestructuras financieras que ejecutan las transacciones. A continuación se detallan las principales capas del modelo actual de pagos SEPA:

Capa de usuarios finales En la capa más alta se encuentran el *pagador* (ordenante del pago) y el *beneficiario* (cobrador). Son quienes inician y reciben los pagos, respectivamente, ya sea personas o empresas involucradas en una transacción.

Capa de iniciación o interacción Es donde se produce la solicitud o autorización del pago a través de algún canal o interfaz. Por ejemplo, en pagos SEPA tradicionales el pagador suele autorizar una transferencia mediante la banca online o una aplicación móvil. En entornos de comercio electrónico o físico, esta capa correspondería al proceso de *checkout* (por ejemplo, introduciendo datos en una pasarela de pago online o mediante un TPV/POS en tienda).

Capa de proveedores de servicios de pago (PSP) Aquí actúan las entidades financieras o PSP de cada parte. Típicamente, el banco del pagador y el banco del beneficiario son quienes ofrecen las cuentas bancarias y servicios de pago a sus clientes. Estas entidades facilitan la emisión y recepción de órdenes de pago en nombre de los usuarios finales.

Capa de esquemas de pago Es el conjunto de reglas y estándares comunes que permiten la interoperabilidad entre todos los PSP. En SEPA, por ejemplo, existen los esquemas de transferencia crediticia (SCT, SCT Inst) y adeudo directo (SDD), definidos por el European Payments Council (EPC). Cada esquema asegura que, independientemente del banco involucrado, un pago en euros siga las mismas normas y formatos. Un esquema de pago no es el movimiento del dinero en sí, sino el conjunto de reglas, mensajes e infraestructura técnica que orquesta cómo se inician y gestionan esos pagos.

Capa de infraestructuras de compensación Son las plataformas interbancarias que procesan las transacciones según las reglas del esquema. En SEPA, existen cámaras de compensación automatizadas (CSM) y redes de pago instantáneo. Por ejemplo, el sistema STEP2 de EBA Clearing procesa transferencias SEPA diarias, mientras que para pagos instantáneos existen redes 24/7 como TIPS (del BCE) o RT1 (de EBA Clearing). Estas infraestructuras se encargan de intercambiar las órdenes de pago entre el banco emisor y el banco receptor, aplicando compensación multilateral si procede.

Capa de liquidación Es la capa más baja, donde se realiza la liquidación final de fondos entre bancos. Normalmente ocurre en los bancos centrales u organismos de compensación: por ejemplo, las obligaciones netas resultantes de muchas transacciones SEPA pueden liquidarse en TARGET2 (sistema del Banco Central Europeo). En pagos inmediatos, la liquidación suele ser casi en tiempo real (por ejemplo, mediante TIPS que liquida cada operación al momento en cuentas del banco central).



Figura 2: Layer

Esta arquitectura por capas garantiza que un pago iniciado por un usuario en un banco pueda

llegar a otro usuario en distinto banco de forma segura, eficiente e interoperable en toda la zona euro. Cada capa agrega funciones específicas: los usuarios generan órdenes, los PSP las gestionan, los esquemas proporcionan las reglas comunes, y las infraestructuras las ejecutan y asientan los fondos en última instancia.

RTP es un nuevo servicio/esquema incorporado en la arquitectura SEPA que se sitúa principalmente en la capa de esquema de pago, actuando como una capa adicional de mensajería sobre los instrumentos de pago existentes.

Ejemplo comparado: pago con tarjeta vs. RTP

Como muestra el esquema comparativo, los pagos con tarjeta (ej. Visa/Mastercard) históricamente han operado con una arquitectura de funciones similar a la de SEPA, pero con diferencias en los actores y procesos de cada capa:

Usuarios (pagador/beneficiario) En tarjetas, el pagador es el *titular de la tarjeta* y el beneficiario es el *comercio* que recibe el pago. En esencia es equivalente al ordenante y beneficiario de una transferencia, con la diferencia de que el pagador utiliza un instrumento distinto (su tarjeta en lugar de una cuenta bancaria directa).

PSP / entidades En el modelo de cuatro partes de las tarjetas interviene el *banco emisor* (emite la tarjeta al pagador) y el *banco adquirente* (procesa pagos para el comerciante). Estos roles son análogos a la entidad del pagador y del beneficiario en SEPA, pero en el mundo tarjeta suelen implicar acuerdos específicos (p. ej. el comerciante contrata un adquirente para aceptar Visa/Mastercard). En cambio, en SEPA cualquier banco puede enviar o recibir transferencias para un cliente sin acuerdos individuales con cada comercio, ya que todos siguen el esquema común.

Esquema de pago Las tarjetas operan bajo esquemas propietarios como Visa, Mastercard, etc., que definen reglas, formatos de mensajes (p. ej. mensajes de autorización y liquidación) y que actúan también como redes de procesamiento. Son equivalentes a los esquemas SEPA en cuanto a que proveen interoperabilidad, pero controlados por empresas particulares. El esquema Visa/Mastercard indica cómo se autoriza una compra, cómo se liquida posteriormente y fija también aspectos comerciales como las tasas de intercambio entre emisor y adquirente. En SEPA, el esquema SRTP + SCT Inst provee una funcionalidad comparable de solicitud y pago, pero dentro de un marco colaborativo paneuropeo. De hecho, SRTP adopta un modelo muy similar al de tarjetas de cuatro partes (pagador, beneficiario y sus respectivos proveedores), tanto que el EPC prevé que incluso podrían llegar a existir comisiones de intercambio análogas en este ecosistema¹³ (aunque inicialmente SRTP nace sin tarifas de intercambio explícitas).

Infraestructura de compensación En los pagos con tarjeta, la red del esquema (p. ej. VisaNet) se encarga de la autorización instantánea de la transacción y de la compensación/clearing de las transacciones entre emisores y adquirentes. Visa o Mastercard centralizan el intercambio de mensajes financieros. En SEPA, por el contrario, las compensaciones suelen realizarse a través de múltiples infraestructuras (por ejemplo, cámaras como STEP2 o servicios inmediatos como TIPS) que no pertenecen a una sola empresa sino que son parte del ecosistema colaborativo europeo. Con Request to Pay, la mensajería de solicitud viaja por la red designada (p. ej.

¹³<https://redbridgedta.com>

la plataforma R2P de EBA Clearing) y el pago resultante se compensa a través de los *rails* SEPA existentes (p.ej. RT1/TIPS para instantáneas).

Liquidación final En ambos casos, finalmente hay un traspaso de fondos entre bancos. En las tarjetas, las marcas de tarjeta calculan las obligaciones netas entre cada banco emisor y adquirente y típicamente las liquidan al final del día a través de cuentas en un banco central u otros mecanismos interbancarios. En SEPA, cada transferencia individual (especialmente si es instantánea) puede liquidarse inmediatamente en el banco central. Desde el punto de vista de capas, ambos mundos terminan convergiendo en la necesidad de que el dinero se ajuste entre las cuentas de los bancos participantes.

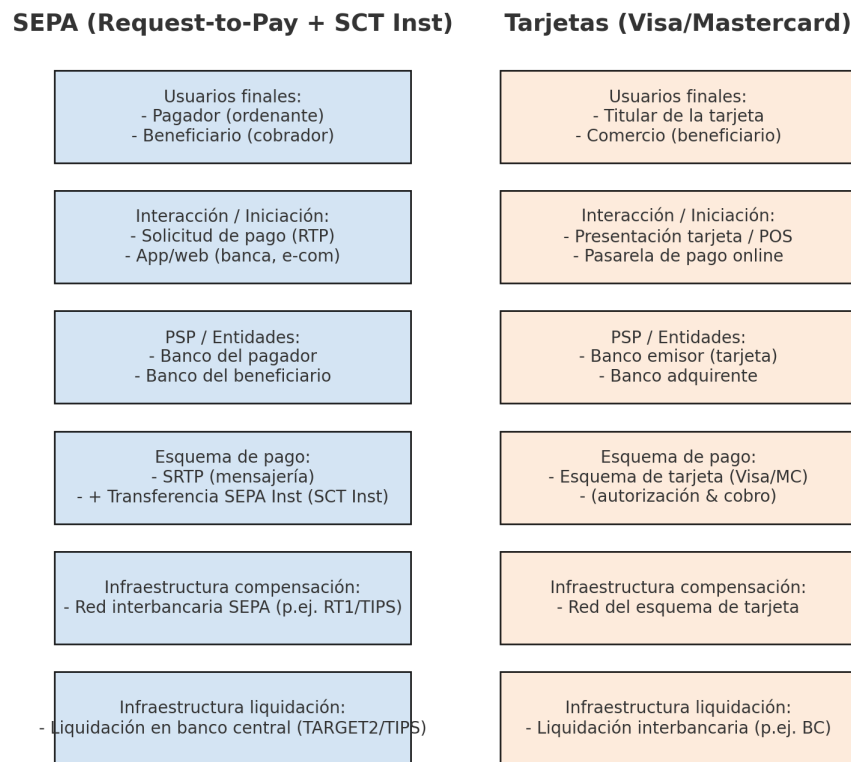


Figura 3: LayerComp

¿Qué simplifica o elimina Request to Pay respecto al modelo de tarjetas?

Principalmente, elimina intermediarios dedicados y procesos redundantes. Por ejemplo, en un pago SRTP + SCT Inst no es necesario un procesador/acquirente específico ni una red de tarjetas propietaria, ya que los propios bancos de pagador y beneficiario se comunican directamente mediante el esquema común¹⁴. Esto puede reducir costes de aceptación para el comercio (evitando comisiones elevadas de tarjetas) y simplifica la integración: el comercio recibe el dinero directamente en su cuenta bancaria vía SEPA, sin pasos intermedios de recibir fondos a través de entidades de tarjeta y luego liquidarlos. Además, no se requiere que el pagador proporcione datos sensibles como el PAN de tarjeta o incluso su IBAN al comercio; la solicitud llega por canales bancarios seguros y el cliente simplemente autoriza en su entorno bancario¹⁵. En resumen, Request to Pay se apoya en la infraestructura bancaria existente (cuentas y pagos inmediatos) para ofrecer una experiencia similar a la de tarjeta, pero con menos capas propietarias, aprovechando la red SEPA ya desplegada en toda Europa.

¹⁴<https://cpg.de>, <https://redbridgedta.com>

¹⁵<https://docs.monei.com>

3 Diseño e Implementación

Para implementar un sistema completo de *Request To Pay*, se ha desarrollado un servidor HTTP que expone una API RESTful y un cliente web en tiempo real. El objetivo de este apartado es detallar cómo se ha llevado a cabo esta implementación, así como las fases de desarrollo involucradas.

3.1 Fundamentos Teóricos

Antes de profundizar en la implementación, es fundamental aclarar los dos pilares teóricos en los que se basa el sistema:

- **Servidor HTTP:** Una aplicación que permanece a la escucha en un puerto de red, esperando conexiones TCP de clientes capaces de comunicarse mediante el protocolo HTTP. Una vez establecida la conexión, el servidor recibe un mensaje estructurado con los siguientes componentes:
 - *Línea de petición:* Indica el método (GET, POST, etc.), la ruta del recurso solicitado y la versión del protocolo.
 - *Cabeceras:* Aportan metadatos, como el tipo de contenido, credenciales o longitud del mensaje.
 - *Cuerpo* (opcional): Contiene datos adicionales, si el método lo requiere.

El servidor interpreta la ruta, determina qué componente interno debe procesarla, ejecuta la lógica correspondiente y genera una respuesta formada por:

- *Línea de estado:* Incluye un código de resultado (por ejemplo, 200 OK, 404 Not Found, 500 Internal Server Error).
- *Cabeceras:* Describen la respuesta.
- *Cuerpo* (opcional): Contiene datos, como HTML o JSON.

Tras enviar la respuesta, la conexión puede cerrarse o mantenerse activa para futuras peticiones, dependiendo de la versión del protocolo y las cabeceras de control. En esencia, el servidor HTTP funciona como el centro de operaciones que recibe todas las solicitudes y coordina el acceso a la lógica y los datos de la aplicación.

- **API RESTful:** Se basa en los principios de la arquitectura REST (*Representational State Transfer*¹⁶) aplicados al protocolo HTTP para exponer recursos de forma uniforme y predecible. Sus características principales son:
 - Cada entidad del dominio (por ejemplo, un usuario o una petición RTP) se representa mediante una URL estable.
 - Los verbos HTTP (POST, GET, PUT, DELETE) describen operaciones como creación, consulta, modificación o eliminación de recursos.
 - El servidor es *sin estado*, por lo que cada solicitud contiene toda la información necesaria, facilitando la escalabilidad horizontal.¹⁷

¹⁶Transferencia de Estado Representacional

¹⁷Es decir, permite añadir o retirar servidores sin necesidad de compartir sesiones en memoria.

- La uniformidad de los códigos de estado y los formatos de representación asegura que clientes heterogéneos consuman la API de manera predecible.
- Herramientas como cachés, control de versiones en URLs o cabeceras, y negociación de contenido permiten evolucionar la interfaz sin afectar a los consumidores existentes.

En conjunto, el servidor HTTP actúa como el camino por donde viajan las solicitudes, mientras que la API RESTful establece las reglas claras y fáciles de mantener para que ese camino conecte eficientemente la lógica del servidor con los diversos clientes que dependen de ella.

3.2 Tecnologías utilizadas

El desarrollo del proyecto se ha llevado a cabo en un entorno virtual Python que aísla las dependencias y permite reproducir la instalación mediante el comando `pip install -r requirements.txt`. Este enfoque asegura consistencia y portabilidad, facilitando tanto la colaboración entre desarrolladores como el despliegue en entornos diversos.

Backend. El núcleo del backend se fundamenta en Python 3 y Flask, un micro-framework que proporciona un servidor WSGI, un sistema de enrutado eficiente y una integración fluida con el estándar HTTP. Sobre esta base, se incorpora Flask-socketIO, un middleware que habilita la negociación de WebSockets, permitiendo servir tráfico HTTP y comunicación bidireccional en el mismo puerto de manera transparente. Para la gestión de datos, se emplea SQLAlchemy a través de Flask-SQLAlchemy, lo que facilita la representación de modelos de dominio como objetos Python, mientras que SQLite actúa como una base de datos ligera y autónoma durante el desarrollo, eliminando la necesidad de un servidor externo para transacciones básicas. Este núcleo se complementa con utilidades de la librería estándar de Python, incluyendo:

- `hashlib`, para firmar transacciones de estado mediante el algoritmo SHA-256;
- `datetime`, para registrar marcas temporales en los logs.

Frontend. En el lado del cliente, se ha implementado un frontend estático basado en HTML5, CSS3 y JavaScript ES6, priorizando la ligereza al evitar frameworks complejos. La maquetación adaptativa se logra mediante Bootstrap 5, cargado vía CDN, y la iconografía se enriquece con Font Awesome, también distribuido por CDN. La comunicación en tiempo real se establece con el cliente Socket.IO 4.x, que conecta vía WebSocket al mismo host y puerto que Flask, mientras que las peticiones REST se realizan de forma nativa con la Fetch API, sin depender de librerías adicionales.

Herramientas de desarrollo. El entorno de trabajo se ha centrado en Visual Studio Code como editor principal, aprovechando sus extensiones para optimizar la gestión del proyecto y el desarrollo del código. El control de versiones se ha gestionado con Git, utilizando ramas específicas para cada funcionalidad nueva, lo que asegura un desarrollo ordenado y trazable. Para las pruebas manuales de la API, se ha empleado Postman, donde se diseñó una colección de peticiones parametrizadas que serán detalladas en secciones posteriores del documento.

El uso de un entorno virtual junto con dependencias consolidadas establece una base robusta para el despliegue de la aplicación en entornos más exigentes, como contenedores, nubes públicas o servidores locales, preservando la integridad de su arquitectura fundamental.

3.2.1 Estructura y funcionamiento

El código del proyecto se estructura en dos componentes principales, **backend** y **frontend**, interconectados mediante los protocolos HTTP y WebSockets. Esta división, diseñada de manera intencionada, responde a la necesidad de lograr un desarrollo ordenado, eficiente y preparado para futuros crecimientos, considerando que trabajé en el proyecto de forma individual. Al separar el *backend*, encargado de la lógica de negocio, el manejo de datos y la comunicación con el cliente, del *frontend*, centrado en la interfaz de usuario y la experiencia interactiva, se obtiene una arquitectura clara y modular. Esta organización aporta múltiples ventajas: mejora la mantenibilidad al permitir identificar y corregir errores de manera localizada, simplifica la incorporación de nuevas funcionalidades sin alterar otras partes del sistema y refleja fielmente la arquitectura empleada en entornos de producción reales, lo que facilita una transición fluida hacia despliegues en contenedores, nubes públicas o servidores locales. Además, esta separación promueve la reutilización de código, ya que el *backend* puede servir a múltiples clientes (como aplicaciones móviles o de escritorio) y el *frontend* puede adaptarse a diferentes dispositivos sin modificar la lógica subyacente. A continuación, se describen los archivos que componen cada una de estas partes y su rol específico en el proyecto.

3.2.2 Protocolos y estándares implementados

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

3.3 Emulación del prototipo *Request To Pay*

3.3.1 Diseño de las funcionalidades

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

3.3.2 Modelos de procesamiento de solicitudes

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

3.4 Herramientas de desarrollo

3.4.1 Lenguajes y frameworks

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

3.4.2 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...

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.

Referencias

Índice de Figuras

1	4CornerModel	12
2	Layer	21
3	LayerComp	23

Índice de Tablas

1	Pasos del flujo del esquema SEPA Request-to-Pay (SRTP) y su descripción	12
---	---	----

Índice de Códigos

1	Código de ejemplo.	31
---	----------------------------	----