Programación para las Comunicaciones

Práctica 3:
Pasarelas de Acceso a Servicio de Información Meteorológica

Pedro José Fernández Escámez

3 de junio de 2025

Índice

1. Introducción

El presente documento detalla el desarrollo de la Práctica 3 de la asignatura Programación para las Comunicaciones. El objetivo principal ha sido extender la funcionalidad del cliente desarrollado en la Práctica 2, transformándolo en un middleware o broker de información meteorológica. Este middleware no solo continúa recopilando y almacenando los datos de múltiples estaciones meteorológicas (servidores de la Práctica 2), sino que también expone esta información y permite la interacción con los servicios subyacentes a través de diversas pasarelas de comunicación. Se han implementado tres mecanismos de acceso principales: una pasarela web mediante HTTP/HTTPS, una API RESTful para interacciones programáticas, y un servicio basado en correo electrónico (SMTP/IMAP) para consultas remotas.

2. Funcionalidades

El middleware desarrollado implementa las siguientes funcionalidades principales, todas ellas partiendo del cliente de la Práctica 2 que actúa como punto central de recolección y gestión de datos:

2.1. Pasarela HTTP/HTTPS

Se ha implementado un servidor web embebido que opera sobre HTTPS (utilizando TLS con un keystore JKS proporcionado) en el puerto `8080` por defecto. La implementación actual se centra en HTTPS para mayor seguridad, sirviendo contenido a través de `https://localhost:8080`.

- **Portal de Acceso Principal (`/index.html`):** Al acceder a la raíz del servidor o a `/index.html`, se presenta una página HTML generada dinámicamente. Esta página actúa como un portal, ofreciendo enlaces y descripciones de las distintas formas de acceder a los servicios implementados, incluyendo la pasarela HTML de meteorología y la API REST.
- **Visualización de Datos Meteorológicos (`/meteorologia.html`):** Este endpoint devuelve una página HTML que muestra los últimos datos recibidos y almacenados por el broker de todas las estaciones meteorológicas activas. La página está diseñada para refrescarse automáticamente cada 10 segundos, permitiendo al usuario observar las actualizaciones de los datos en tiempo real.

2.2. API REST

Integrada en el mismo servidor HTTPS, se ha desarrollado una API REST que opera bajo la ruta base `/apirest/`. Todas las respuestas de esta API se devuelven en formato JSON.

- **Endpoint `/apirest/muestra_valores`:**
  - **Método:** GET.
  - **Descripción:** Devuelve un objeto JSON que contiene los últimos `DistributionMessage` almacenados para cada servidor meteorológico identificado por su ID.
  - **Respuesta Ejemplo:** `{"S1": {...}, "S2": {...}}`.
- **Endpoint `/apirest/cambia_parametro`:**
  - **Método:** GET.
  - **Descripción:** Permite enviar comandos de control a un servidor meteorológico específico de la Práctica 2. Los parámetros se especifican en la URL.
  - **Parámetros URL Ejemplo:** `serverId=S1&command=SET_FREQUENCY&frequency=2000` para cambiar la frecuencia del servidor S1. Otros comandos implementados incluyen `SET_ENCODING`, `TOGGLE_SENDING_DATA`, `SET_VARIABLE_UNIT`.

- **Respuesta:** Devuelve un objeto JSON `ResponseMessage` indicando el resultado de la operación.

2.3. Servicio de Correo Electrónico (SMTP/IMAP)

El cliente/broker implementa un servicio de correo electrónico que permite consultar los datos meteorológicos.

- **Recepción de Peticiones:** El servicio monitoriza una cuenta de Gmail configurada (ej. `pjfe.ppc@gmail.com`) utilizando el protocolo IMAP (o POP3, según configuración en `SmtpConfig`), revisando la bandeja de entrada en busca de nuevos correos no leídos a intervalos regulares (ej. cada 30-60 segundos).
- **Procesamiento de Comandos:** Se procesan los correos cuyo asunto sea exactamente `GET_DATA`.
- **Respuesta por Correo:** Al recibir una petición válida, el servicio responde al remitente con un correo electrónico en formato MIME multiparte.
- **Contenido de la Respuesta:**
  - Una parte de texto plano (`text/plain`) con un resumen legible de los últimos datos de todas las estaciones.
  - Adjuntos con los datos completos de cada estación, tanto en formato JSON como en formato XML.
- **Gestión de Duplicados:** Se implementa un mecanismo para evitar el procesamiento múltiple de correos, utilizando los UIDs de los mensajes en IMAP.

2.4. Arquitectura del Middleware y Justificación

El cliente de la Práctica 2 ha evolucionado para convertirse en un componente central (broker) que no solo recibe y almacena los últimos datos de las estaciones meteorológicas mediante UDP broadcast, sino que también gestiona las peticiones de acceso remoto. La arquitectura se basa en los siguientes componentes principales:

- **Cliente/Broker Principal:** Mantiene el estado actual de los datos de los servidores, gestiona la comunicación con los servidores de la P2 (envío de comandos de control) y orquesta las pasarelas.
- **Servidor HTTPS Embebido (`HttpGatewayServer`):** Maneja todas las peticiones HTTP/S. Se crea un pool de hilos para atender múltiples peticiones concurrentemente. Este servidor delega las peticiones a manejadores específicos según la ruta.
- **Manejadores HTTP (`*Handler`):** Clases como `IndexHtmlHandler`, `MeteorologyHtmlHandler`, y `RestApiHandler` procesan las peticiones para las rutas correspondientes.
- **Servicio de Correo Electrónico (`EmailService`):** Se ejecuta en un hilo separado (`EmailPollingThread`) que sondea periódicamente la bandeja de entrada del correo configurado usando JavaMail (API `javax.mail`).

Esta arquitectura permite desacoplar las diferentes formas de acceso y centralizar la lógica de negocio y el almacenamiento de datos en el broker. El uso de hilos dedicados para el sondeo de correo y el manejo de peticiones HTTP/S asegura que la aplicación sea responsiva.

2.5. Configuración de Direcciones IP y Puertos

- **Pasarela HTTPS (Cliente/Broker):** Escucha en `https://localhost:8080`. El puerto 8080 es fijo para esta práctica.

- **Servidores Meteorológicos (Práctica 2):**
  - Puerto de difusión (Broadcast): `5000` (fijo, UDP).
  - Puertos de control (Unicast): Variables, configurados al lanzar cada servidor (ej. `5001`, `5002`, `5003`, UDP).
- **Servicio de Correo Electrónico:**
  - Servidor SMTP: Configurable (ej. `smtp.gmail.com`, puerto `587` para TLS/STARTTLS).
  - Servidor IMAP/POP3: Configurable (ej. `imap.gmail.com`, puerto `993` para IMAPS).

3. Implementación

3.1. Protocolos y Formato de Tramas

3.1.1. HTTP/HTTPS y API REST

Se utiliza el protocolo HTTPS (HTTP sobre TLS) para la comunicación con el gateway web. Las peticiones a la API REST utilizan principalmente el método GET. Los datos intercambiados con la API REST utilizan el formato JSON. Las páginas web se sirven como HTML.

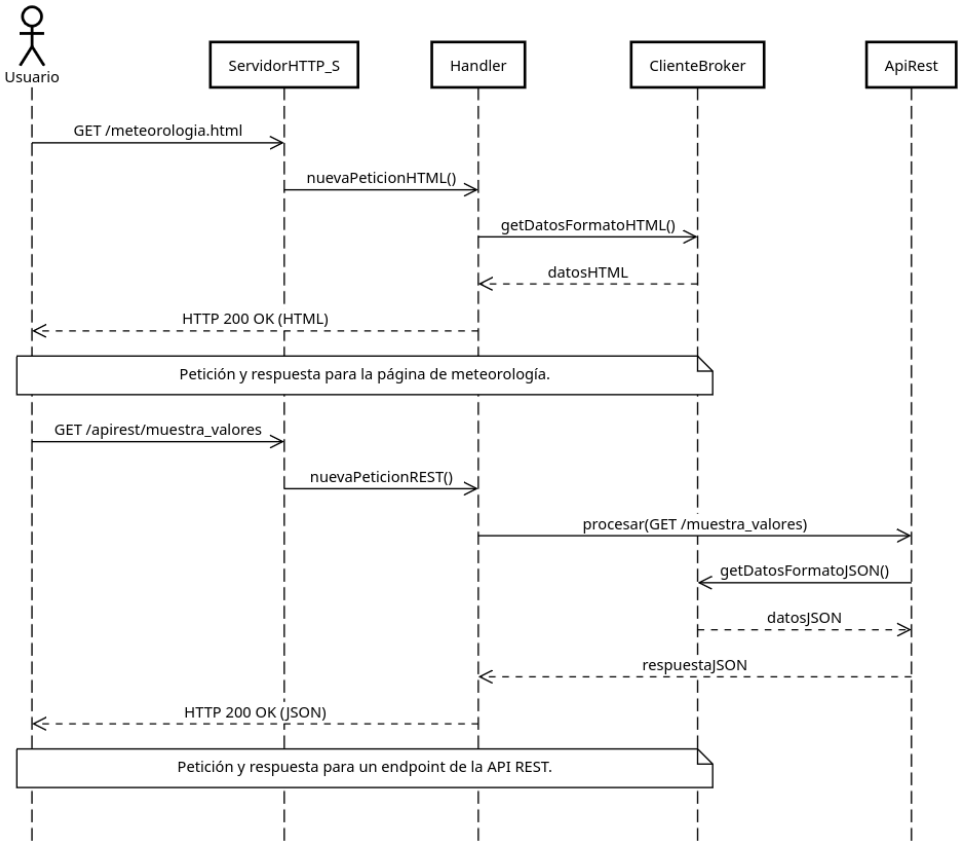

Figura 1: Diagrama de Secuencia: Acceso a `/meteorologia.html` y API REST.

3.1.2. Servicio de Correo Electrónico

El protocolo SMTP se utiliza para el envío de correos de respuesta por parte del servicio. Para la recepción de correos (sondeo de la bandeja de entrada), se utiliza IMAP (o POP3, según configuración).

Las peticiones se identifican por el asunto del correo (ej. ”GET_DATA”). Las respuestas son correos en formato MIME multiparte (`multipart/mixed`):

- Una parte `text/plain` con el resumen de los datos.
- Una o más partes `application/json` como adjuntos, una por cada servidor, con los datos serializados.
- Una o más partes `application/xml` como adjuntos, una por cada servidor, con los datos serializados.

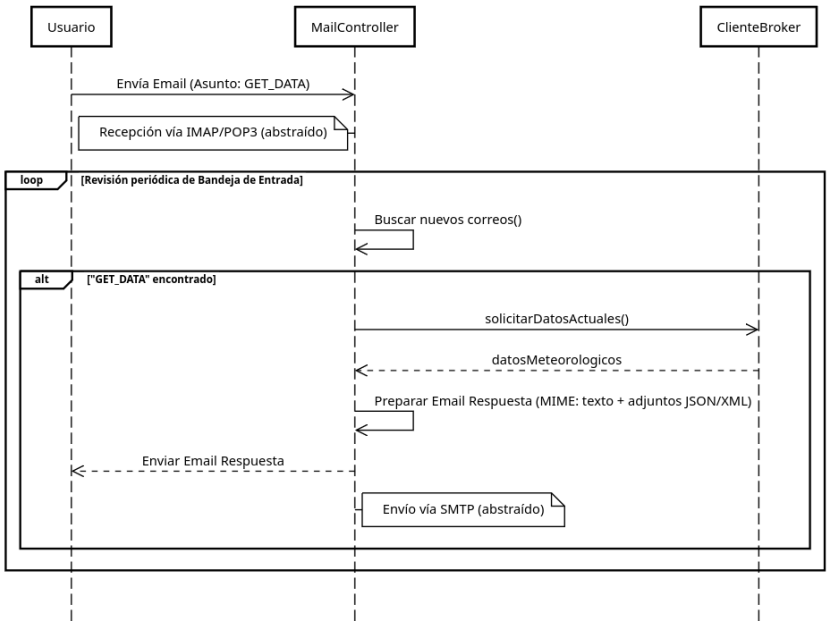

Figura 2: Diagrama de Secuencia: Interacción por Correo Electrónico.

3.2. Operaciones de E/S y Modelos Utilizados

- **Pasarela HTTPS:** Se utiliza la clase `com.sun.net.httpserver.HttpsServer` del JDK para crear un servidor HTTPS embebido. Este maneja las conexiones TCP y la capa TLS/SSL. Utiliza un modelo basado en hilos (pool) para atender peticiones concurrentes de forma eficiente.
- **Servicio de Correo:** Se emplea la API JavaMail (`javax.mail`) para la interacción con los servidores SMTP e IMAP/POP3. Las operaciones de red (envío y recepción de correos) son bloqueantes pero se gestionan dentro del hilo de sondeo del `EmailService` para no afectar al resto de la aplicación.
- **Comunicación con Servidores P2:** El cliente/broker mantiene la comunicación UDP (DatagramSockets) establecida en la Práctica 2 para recibir broadcasts y enviar mensajes de control unicast.

3.3. Identificación de Hilos Implementados

La aplicación utiliza varios hilos para gestionar sus tareas concurrentes:

- **Hilo Principal del Cliente/Broker:** Ejecuta el método `main` y el bucle de `handleUserInput` para la consola.
- **Hilo `BroadcastListenerThread`:** Dedicado a escuchar continuamente los mensajes de difusión UDP de los servidores meteorológicos de la Práctica 2.

- **Pool de Hilos del `HttpsServer`:** El servidor HTTPS gestiona internamente un pool de hilos para atender las peticiones HTTP/S entrantes. Cada petición (para `/index.html`, `/meteorologia.html`, o la API REST) es manejada por uno de estos hilos.
- **Hilo `EmailPollingThread`:** Hilo dedicado del `EmailService` que sondea periódicamente la bandeja de entrada del correo electrónico en busca de nuevas peticiones.

3.4. Modelos de Datos (Formatos)

- **HTML (`/meteorologia.html`):** Página HTML generada dinámicamente que presenta los datos meteorológicos en formato tabular, con información de cada servidor y sus variables.
- **JSON (API REST):**
  - `/apirest/muestra_valores`: Devuelve un mapa donde las claves son los IDs de los servidores y los valores son los objetos `DistributionMessage` serializados a JSON.
  - `/apirest/cambia_parametro`: Devuelve un objeto `ResponseMessage` serializado a JSON indicando el resultado del comando.
- **Correo Electrónico (MIME Multipart):**
  - **Petición:** Asunto de texto plano (ej. "GET_DATA").
  - **Respuesta:**
    - Parte principal: `text/plain` con datos legibles.
    - Adjuntos: Archivos `.json` y `.xml` conteniendo el objeto `DistributionMessage` serializado para cada servidor.

3.5. Organización del Código

El código fuente se organiza en los siguientes paquetes principales:

- `client`: Contiene la clase `Client`, que es el punto de entrada principal y el broker.
- `common`: Clases compartidas para los mensajes (`AbstractMessage`, `DistributionMessage`, `ControlMessage`, `ResponseMessage`, `WeatherVariable`) y utilidades (`MessageUtils`). Estas son mayormente reutilizadas de la Práctica 2.
- `gateway`: Clases relacionadas con las pasarelas de acceso: `HttpGatewayServer`, los manejadores HTTP (`IndexHtmlHandler`, `MeteorologyHtmlHandler`, `RestApiHandler`), `EmailService` y `SmtpConfig`.

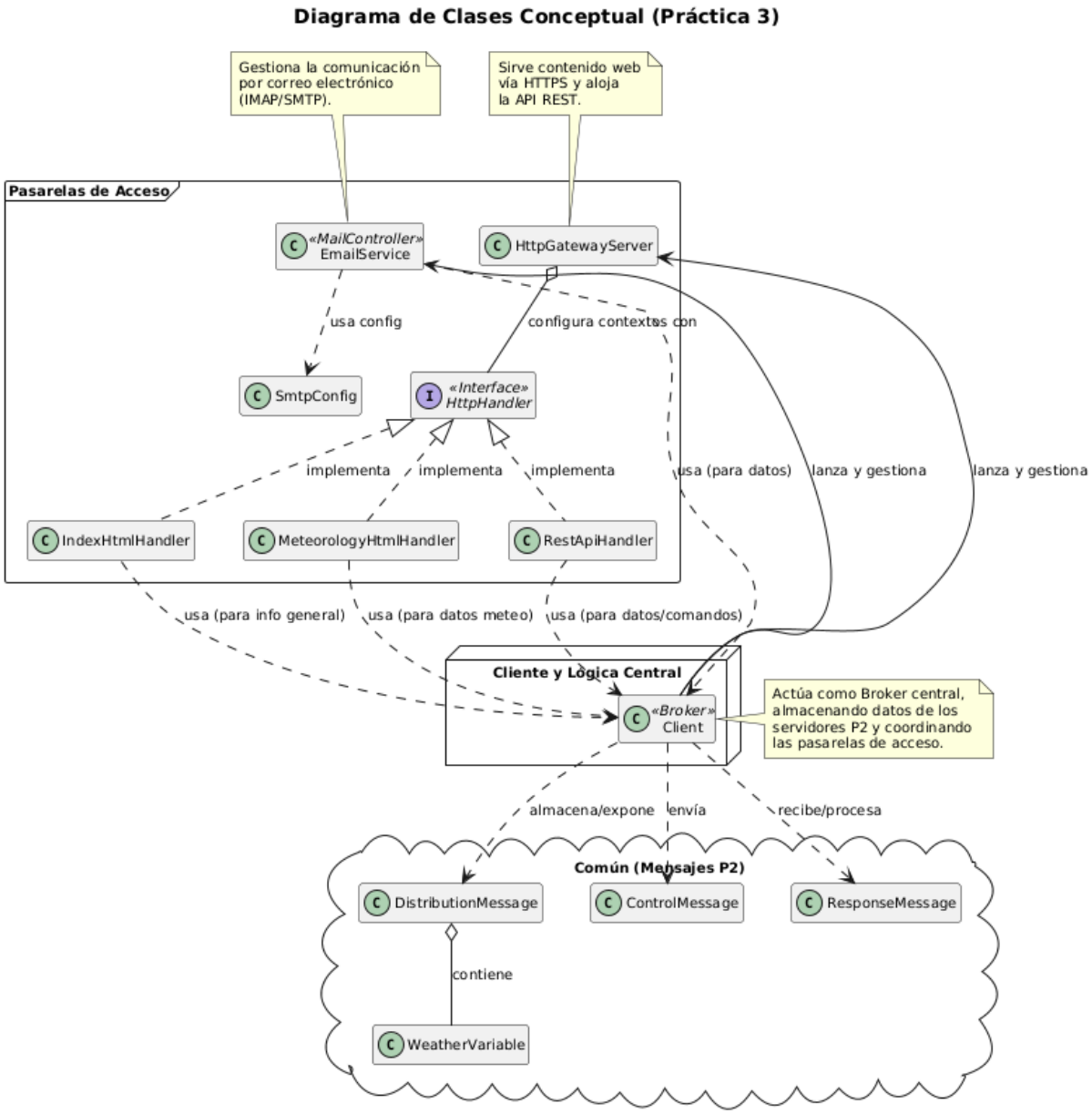

Figura 3: Diagrama de Clases Principales de la Práctica 3.

4. Ejecución

Para ejecutar la práctica, se deben seguir los siguientes pasos:

1. **Lanzar los Servidores Meteorológicos (Práctica 2):** Es necesario tener al menos tres instancias del servidor de la Práctica 2 ejecutándose. Cada servidor se lanza desde la línea de comandos, especificando su ID único y un puerto de control único. Por ejemplo:

```
# Compilar el proyecto si no se ha hecho: mvn clean install
java -cp server.Server S1 5001 temp C
java -cp server.Server S2 5002 hum %
java -cp server.Server S3 5003 viento m/s
```

2. **Lanzar el Cliente/Broker (Práctica 3):** Una vez que los servidores de la P2 están activos, se lanza

la clase principal del cliente/broker. Es importante haber configurado previamente las credenciales de correo y las contraseñas del keystore directamente en el código fuente (`Client.java` para `SmtpConfig`, y `HttpGatewayServer.java` para el keystore).

```
java -cp client.Client
```

3. **Acceso a las Pasarelas:**
   - **HTTP/HTTPS:** Abrir un navegador web y acceder a la URL `https://localhost:8080/`. Se presentará la página de inicio. Acceder a `https://localhost:8080/meteorologia.html` para ver los datos meteorológicos y verificar su actualización al recargar.
   - **API REST:** Utilizar un navegador o una herramienta cliente REST (como Postman o curl) para acceder a los endpoints. Por ejemplo:
     - `https://localhost:8080/apirest/muestra_valores` (GET)
     - `https://localhost:8080/apirest/cambia_parametro?serverId=S1&command=SET_FREQUENCY&frequency=3000` (GET)
   - **Servicio SMTP/IMAP:** Enviar un correo electrónico a la dirección configurada en `SmtpConfig` (ej. `pjfe.ppc@gmail.com`) con el asunto exacto `GET_DATA`. Verificar el correo de respuesta recibido, incluyendo el remitente, cuerpo y los adjuntos JSON/XML.