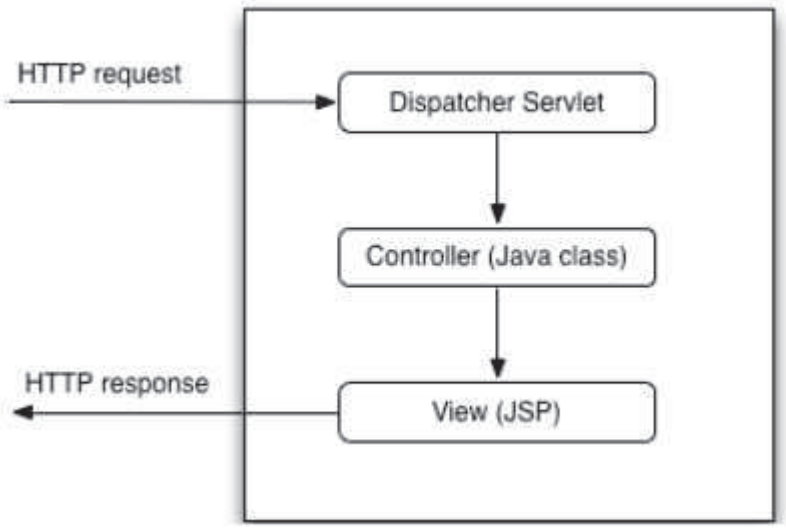
ciascun messaggio in maniera univoca in modo che il client dell'ESB possa individuare univocamente la risposta relativa alla richiesta inviata.
Per rendere ERMES-QIP capace di ricevere messaggi HTTP dall'esterno, esso è stato inserito in un server Apache Tomcat ed è stata configurata la seguente servlet:

```
<servlet>
    <servlet-name>webapp</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/servlet-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
```

Tale servet è della classe "org.springframework.web.servlet.DispatcherServlet": classe offerta dal framework Spring; trattasi di un *dispatcher* centrale per le richieste HTTP, smista agli *handler* registrati ("param-value") richieste web perché siano processate, fornendo la necessaria gestione delle eccezioni; si basa sul meccanismo dei JavaBean.



Figuta 5. ...

Il lavoro del DispatcherServlet consiste in pratica nel prendere una URI in entrata e di trovare la giusta combinazione di gestori (in genere i metodi di classe *Controller*) e

View (generalmente JSP) che si combinano per formare una pagina, una risorsa, un servizio che dovrebbe essere trovato in quella posizione.
In questo caso, il DispatcherServlet attiva l'HTTP inbound gateway/adapter; il messaggio inviato verso ERMES-QIP, infatti, arriva, passando per la servlet, nell'inbound HTTP adapter, la cui configurazione è la seguente:

```
<http:inbound-gateway id="httpInboundGateway"
    request-channel="receiveChannel"  reply-channel="outboundChannel"
    path="/inboundChannel.html"

    supported-methods="POST"  reply-timeout="50000"
    error-channel="errorChannel"/>
```

Si definiscono, oltre all'ID, il canale di riferimento e il *path*. Questo path si riferisce alla URI che deve scegliere il mittente perchè il messaggio venga accolto effettivamente da quell'inbound gateway. Vengono definiti i metodi supportati (POST) e si specifica che è richiesto il payload del tipo "java.lang.String", tipologia compatibile con il formato JSON dei messaggi in arrivo. Si definisce il tempo massimo di *timeout* e il canale di default per la gestione degli errori. Dall'inbound HTTP gateway il messaggio passa nel Json-to object adapter, in cui il payload viene trasformato da JSON a oggetto Java di tipo Request:

```
<int:json-to-object-transformer type="it.model.Request"
    input-channel="receiveChannel" output-channel="toRouterChannel">
  </int:json-to-object-transformer>
```

Successivamente, il messaggio entra in un router perché possa essere indirizzato verso il giusto livello:

```
<int:router input-channel="toRouterChannel" ref="ERMESRouter"
    method="select" default-output-channel="nonMatchesChannel" />

  <bean id="ERMESRouter" class="it.router.ERMESRouter" />
```

Come si nota nella configurazione XML precedente, il router, etichettato come "ErmesRouter" ha come canale di input il "toRouterChannel". La logica del router viene implementata nella classe "ERMESRouter" creata come un bean nella configurazione precedente:

```
package it.router;

import org.springframework.integration.Message;

import it.model.Request;

public class ERMESRouter {
  public String select(Message<Request> message) {
    Request t = message.getPayload();
    if (t.getOriginAdress().equals(""))
      return "updateChannel";
    else {


      if (t.getTag().equals("jDoraRequest"))
        return "jDoraChannel";
      if (t.getTag().equals("interServicesMessaging"))
        return "InterServicesMessagingChannel";
      if (t.getTag().equals("ClientRequest"))
        return "clientRequestsChannel";
      return "nonMatchesChannel";


    }

  }
}
```

Fondamentale perché ERMES-QIP sia sempre aggiornato è la funzionalità *Udpdate* che permette all'ESB di acquisire tutte le informazioni necessarie per svolgere al meglio il suo ruolo, in riferimento a tutti e tre i livelli.

Di seguito tre immagini che riassumono la struttura dei tre livelli di ERMES-QIP:

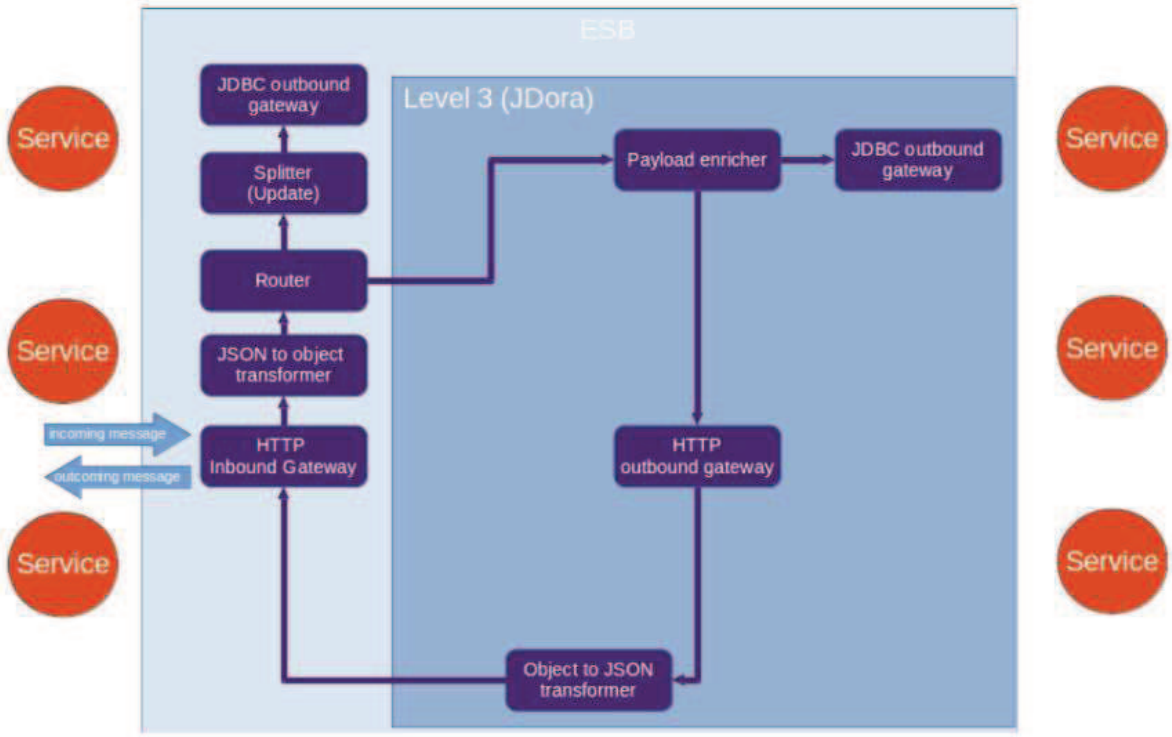


Figura 6. ...

Come si evince dalla precedente figura, i mittenti a questo livello sono i servizi mentre il ruolo del ricevente è ovviamente assunto dallo stesso ESB. Tale livello supporta la componente distribuita per la condivisione delle classi (e delle sue istanze) JDora. JDora permette di scambiare una classe Java da un entità server ad un'entità client:
il client conosce solo la denizione di classe attraverso l'interfaccia pubblica inclusa nel suo pacchetto. Il client chiama il metodo di loadClass del NetworkClassLoader che è in grado di contattare il server remoto e scaricare la classe richiesta. Il NetworkClassLoader è ora in grado di leggere la definizione delle classi ottenute e risolvere la classe che sarà disponibile localmente. Il ruolo dell'ESB in questo caso
è semplicemente quello di risolvere delle richieste in cui, dato un elenco di classi, si richiedono gli URL dei relativi class owner. Attraverso la funzionalità Update esiste la possibilità, da parte dei servizi, di inviare un messaggio all'ESB contenente tutti i nomi delle classi di cui essi stessi sono owner, in modo tale che ERMES-QIP sia sempre aggiornato riguardo tutte le classi utilizzate dai servizi. Il router invia in questo livello messaggi con tag "JDora Request", i quali andranno verso un message-enricher collegato ad un JDBC-outbound-gateway: il message enricher inserisce nel campo resolved adress del messaggio l'indirizzo del class owner della classe riportata nel content; ciò avviene per mezzo di una query fatta al service registry per mezzo del JDBC outbound gateway. A questo punto il messaggio ritorna ad un http-inbound-gateway così che possa tornare al servizio richiedente. Ovviamente tale scelta architetturale risulta di fatto come una sorta di ibrido tra un

sistema SOA e un sistema con le system condiviso. J-DORA può essere utilizzato per dare una risposta immediata al problema di condivisione delle classi nei servizi; è necessario però prevedere durante gli sviluppi futuri che tale funzionalità venga inglobata in ERMES-QIP utilizzando il livello 3 Direct. In tal modo si arriva ad avere anche in questo livello, accoppiamento lasco tra i servizi. Di seguito un esempio della richiesta che un servizio esegue nei confronti di ERMES-QIP:

```
{
    "tag": "jDoraRequest",
    "content": [
        ""
    ],
    "originAdress":
    "http://192.168.56.102:8080/resolvedAdressPOST",
    "resolvedAdress": "class.goal",
    "id": ""
}
```

Quindi, la risposta che ERMES-QIP restituisce:

```
{
  "tag": "jDoraRequest",
  "content": [
    ""
  ],
  "resolvedAdress": "http://192.168.56.101:8080/serviceOwner",
  "originAdress": "http://localhost:8080/service",
  "id": null
}
```

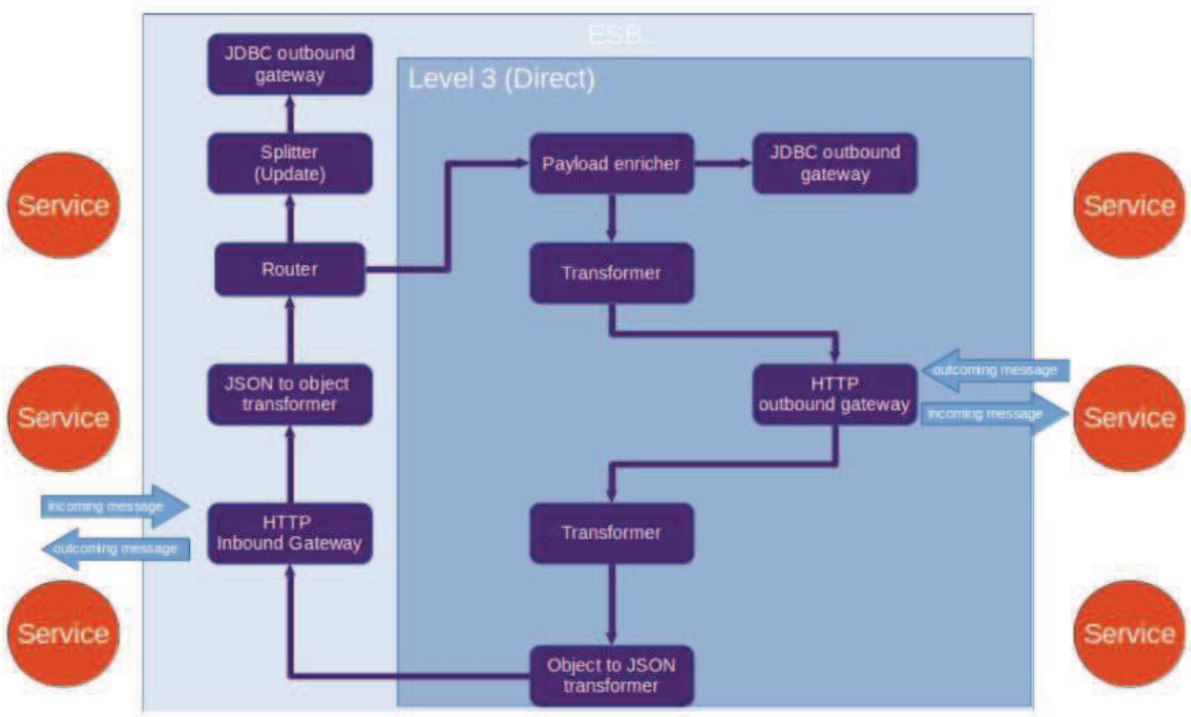


Figura 7. …

Come precedentemente detto, questa soluzione per il livello 3 è quella più ortodossa per ciò che riguarda l'utilizzo di un ESB. Una volta arrivato il messaggio con tag "level3Direct" dal router, si trasforma il payload da JSON ad un oggetto Request. Per questo livello l'oggetto Request, nel content conterrà in ordine:

- il progetto di riferimento;
- l'oggetto che si vuole ottenere.

Grazie al pattern già visto, si inserisce nel resolved adress l'URL del servizio che possiede l'oggetto in questione. Nel service registry, per risolvere questa richiesta, saranno presenti associazioni tra i vari tipi di oggetti e le URL dei vari servizi di supporto. A questo punto il messaggio entra nel trasformer per diventare un "Level3Request".

Tale oggetto contiene il progetto di riferimento, l'oggetto da richiedere, ovviamente l'indirizzo di destinazione e quello di origine, un id ed infine una stringa denominata "*version*". Tale stringa è fondamentale se si prevede un lavoro parallelo di più utenti sullo stesso oggetto. La logica relativa al sistema di *versioning* può facilmente risiedere nell'ESB e permetterebbe a ciascun utente di gestire le modiche parallele. Si potrebbe pensare, per esempio, alla possibilità di eseguire dei veri e propri *branch* paralleli che poi si andrebbero ad unificare tramite fasi di *merge*.

```
1  public class Level3Request {
     private String tag;
3    private String project;
     private String object;
5    private String data;
     private UUID id;
```

```
7    private String version;
     private String destinationAdress;
9  public Level3Request(String tag, String project, String object, String
   destinationAdress, String data, UUID id,
         String version) {
11     super();
       this.tag = tag;
13     this.data = data;
       this.id = id;
15     this.version = version;
       this.object=object;
17     this.project=project;
       this.destinationAdress=destinationAdress;
19   }
...
```

Continuando ad analizzare il cammino dell'oggetto "Level3Request" appena creato, questo viene inviato, tramite http-outbound-gateway, al servizio che contiene l'oggetto richiesto. Tale servizio risponde inviando all'ESB, in formato JSON, l'oggetto che viene inserito nel content di un oggetto Request; infine, ERMES-QIP risponde al servizio richiedente inviando l'oggetto Request all'http-inbound-gateway iniziale che consegna la risposta. Di seguito è riportato un esempio di HTTP POST che il client invia ad ERMES:

```
{
    "tag": "level3Direct",
    "content": [
        "ProjectMorpheus",
        "Project-info"
    ],
    "originAdress": "abc",
    "resolvedAdress": "",
    "id": ""
}
```

Segue poi un esempio per ciò che riguarda la risposta che ERMES-QIP restituisce al client:

```
{
  "tag": "level3Direct",
  "content": [
    "{\n    \"assignDate\": null,\n    \"attachments\": [\n    ],
    \n    \"creationDate\": \"18/02/2015\",\n    \"creator\": {
    \n        \"firstName\": \"Riccardo\"
...
],
  "resolvedAdress": "",
  "originAdress": "",
  "id": null
}
```

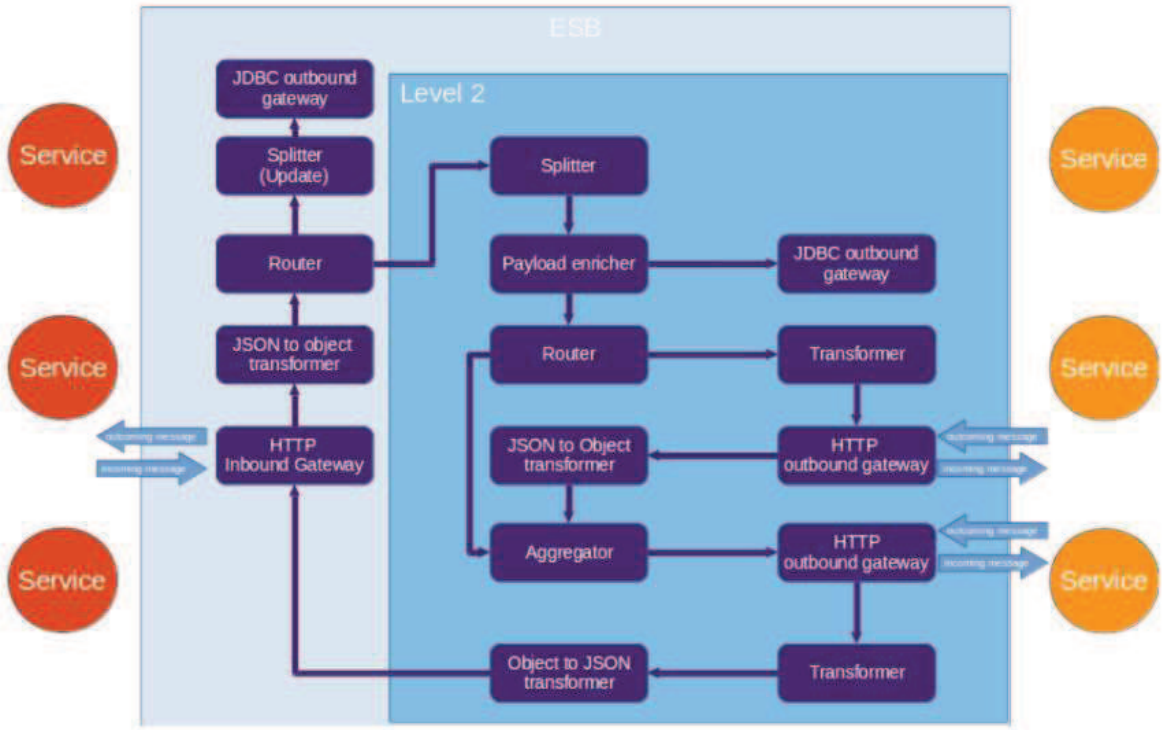


Figura 8. ...

In questo livello la comunicazione avviene tra differenti servizi web. La situazione più frequente è quella per la quale un servizio A richiede ad un servizio B l'elaborazione di dati presenti in un terzo servizio C; l'ESB assume il ruolo di un vero e proprio middleware, evitando la comunicazione point-to-point tra le varie entità in gioco. Le richieste presenti in questo livello sono di due differenti tipologie:

- **Request**, classe già vista, è utilizzata dal servizio richiedente per comunicare con l'ESB e dall'ESB per rispondere al servizio medesimo. In questo livello l'oggetto Request contiene due identificatori all'interno dell'attributo content, in un primo campo che identifica l'azione che si richiede su quei dati, il secondo invece identica proprio l'insieme di dati necessari per la computazione. Attraverso modiche minime è ovviamente possibile richiedere servizi multipli e in cascata.
- **Level2Service**, viene utilizzata dall'ESB per tre scopi: richiedere un insieme di dati, richiedere una determinata computazione ed infine per inviare la risposta del servizio invocato all'ESB.

```
public class Level2Request {
  private String originAdress;
  private String destinationAdress;
  private ArrayList<Double> data;
  private UUID id;
  public RequestForService() {
    super();
  }
  public Level2Request(String originAdress, String destinationAdress,
      ArrayList<Double> data, UUID id) {
    super();
    this.originAdress = originAdress;
    this.destinationAdress = destinationAdress;
    this.data = data;
    this.id=id;
  }
...
```

Si ipotizzi che un messaggio di richiesta dal servizio ad ERMES-QIP contenga nell'attributo content due identificatori: uno indicante quale operazione il richiedente vuole sia eseguita. L'altro indicante dove sono i dati necessari per l'operazione di cui prima.
Quando il router instrada il messaggio verso il livello 2, tale messaggio viene inizialmente inviato verso uno splitter. Lo splitter si occupa di creare due diversi messaggi: un primo che andrà a recuperare i dati da processare (data); un secondo che porterà questi dati al servizio per processarli (operation).
Dopo lo splitting, attraverso il classico sistema enricher JDBC-oubound-Gateway, ERMES-QIP assegna il corretto tag a ciascun messaggio e risolve l'identificatore con il servizio destinatario.
Il pacchetto di tipo *data* viene inviato attraverso un http-outbound-gateway ad un determinato servizio, il quale risponderà con un oggetto requestForServices comprendente i dati richiesti. Questo messaggio entra in un aggregator insieme al messaggio operation. Il messaggio che ne risulta avrà tutti gli attributi del messaggio operation e l'attributo content dal messaggio data. A questo punto l'ESB è pronto per inviare il messaggio così costruito al servizio che eseguirà il calcolo prestabilito. Il servizio ha due diverse possibilità: può rispondere inserendo il risultato di tipo stringa nell'attributo *data* dell'oggetto RequestForServices; può informare l'ESB sulla locazione del risultato della computazione, lasciando nullo l'attributo *data* e ponendo nell'attributo destination adress un URL.

Di seguito un esempio del body di un messaggio HTTP POST che richiede all'ESB di calcolare la somma di alcuni dati associati all'identificatore "data.progetto2":

```
{
    "tag": "interServicesMessaging",
    "content": [
        "operation.sum",
        "data.progetto2"
    ],
    "originAdress":
    "http://192.168.56.101:8080/level2ResponseToService",
    "resolvedAdress": "",
    "id": ""
}
```

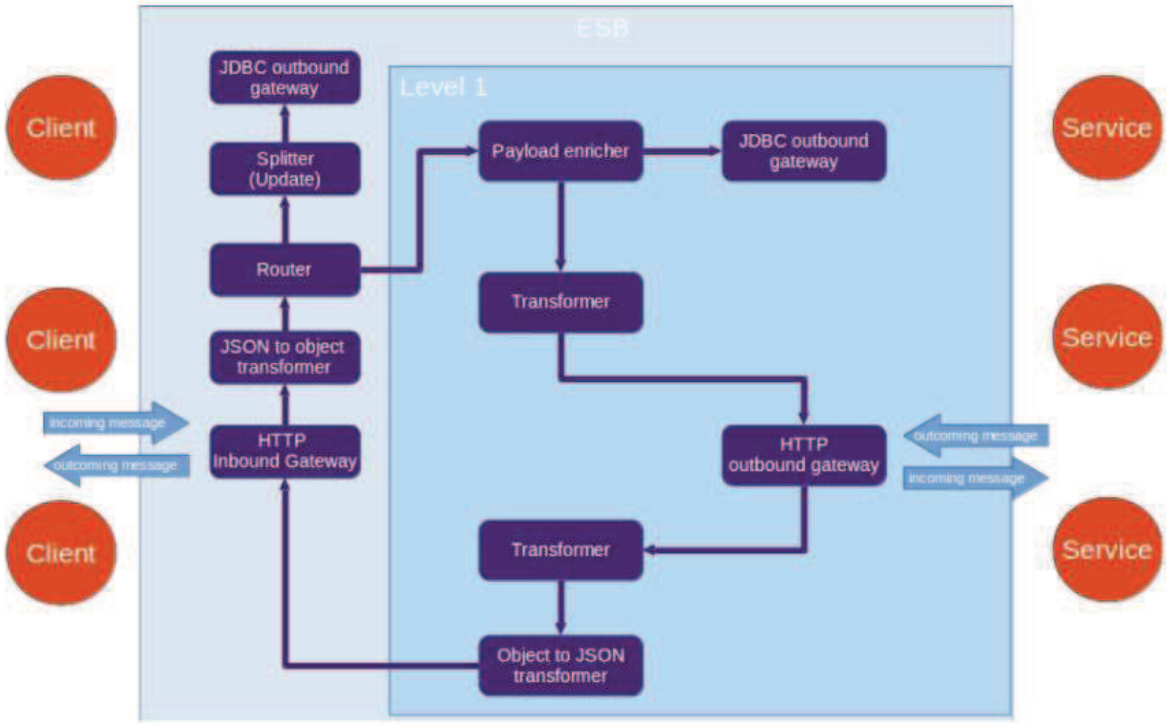


Figura 9. …

In questo livello la comunicazione avviene tra client e i vari servizi. L'ESB, come nel livello precedente, evita il collegamento point-to-point tra client e servizi operando da middleware e ottenendo in tal modo la trasparenza alla locazione. Si ha inoltre la

possibilità di modificare ed eliminare qualsiasi servizio senza che tale operazione pesi sul client o sugli altri servizi.
Anche in questo livello ci sono due differenti tipologie di messaggio:

- **Request**, classe già vista, è utilizzata qui dal client per eseguire verso ERMES-QIP una richiesta riguardante il dominio applicativo.
- **Level1Request**, usata dall'ESB per comunicare con un determinato servizio e ottenere ciò che è stato richiesto dal client.

```
public class Level1Request {

  private String originAdress;
  private String destinationAdress;
  private ArrayList<String> data;
  private UUID id;

  public Level1Request(String originAdress, String destinationAdress,
      ArrayList<String> data, UUID id) {
    super();
    this.destinationAdress = destinationAdress;
    this.data = data;
    this.originAdress = originAdress;
    this.id = id;

  }
...
```

Il cammino del messaggio etichettato come "Level1Request" è il seguente: per prima cosa va attraverso il pattern enricher, jdbc-inbound-gateway in cui viene risolto l'identificatore della funzionalità richiesta (presente nel campo ResolvedAdress) con l'URL del servizio capace di esaudirla. Successivamente, il messaggio entra in un Transformer e diventa un "Level1Request"; l'URL presente nell'attributo ResolvedAdress dell'oggetto Request viene inserito nell'attributo DestinationAdress dell'oggetto Level1Request. Il messaggio può essere dunque inviato al servizio destinatario. Questo risponderà con un altro messaggio, sempre del tipo "Level1Request", che conterrà nel campo *data*, in formato JSON, l'oggetto richiesto. A questo punto ad ERMES-QIP non rimane che trasformare il payload del messaggio in arrivo in una Request; il messaggio viene quindi inviato al canale di risposta dell'http-inbound-gateway generale in modo che questo possa rispondere al servizio mittente.

Di seguito il body di una HTTP POST inviata lato client ad ERMES-QIP.

```
{
    {
     "tag": "level1",
     "content": [
        "projectMorpheus",
        "goalObtainMoney"
     ],
     "originAdress": "client-URL",
     "resolvedAdress": "getGoalState",
     "id": ""
}
}
```

Il client prevede nel campo Content, in ordine:

- il progetto di riferimento;
- l'oggetto su cui si vuole sia compiuta l'azione.

In generale, si inseriscono in questo campo tutti i parametri richiesti dal servizio REST a cui si fa la richiesta. L'azione che si desidera venga compiuta, invece, viene inserita nel campo "resolvedAdress".
Nel service registry saranno presenti associazioni azione-URL del servizio capace di offrirla.