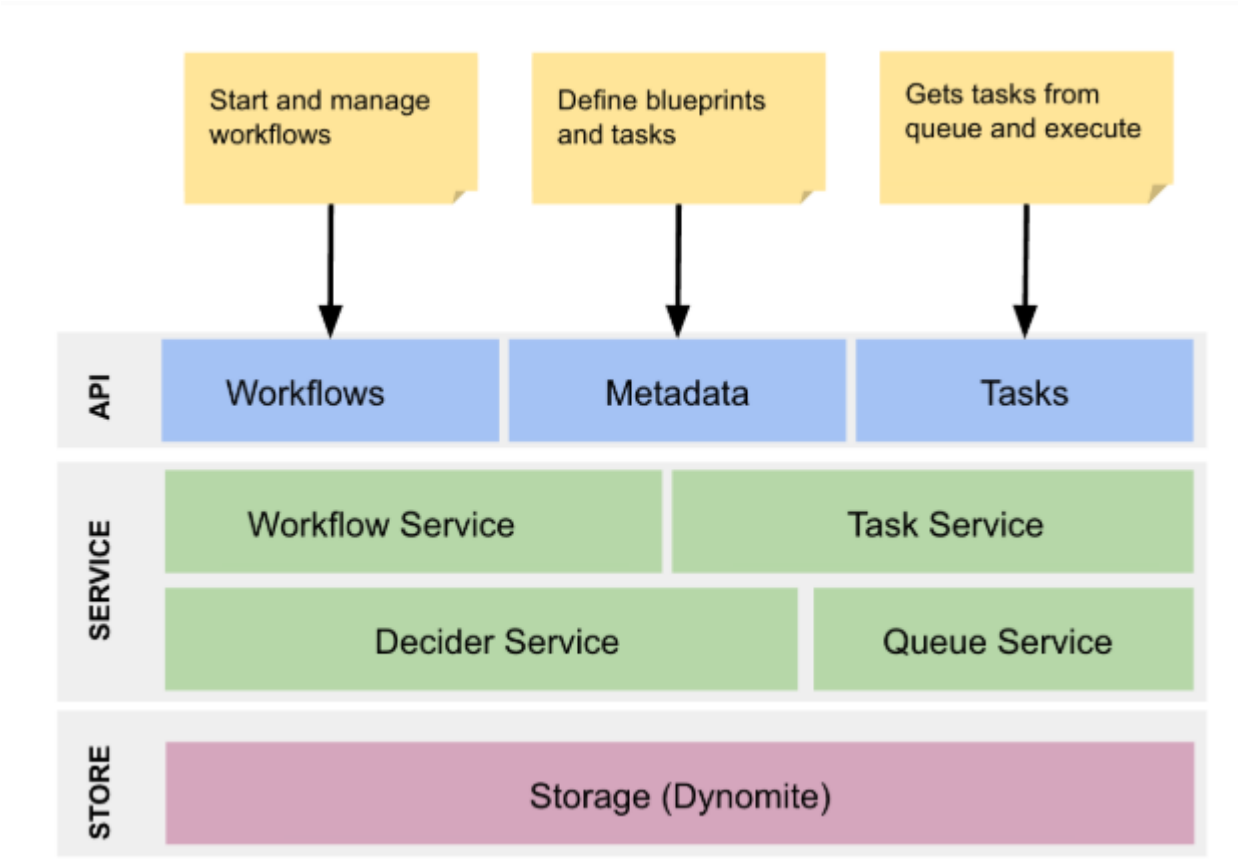
Conductor è un orchestratore di servizi, creato principalmente per orchestrare i servizi offerti da Netflix. Presenta le seguenti caratteristiche:

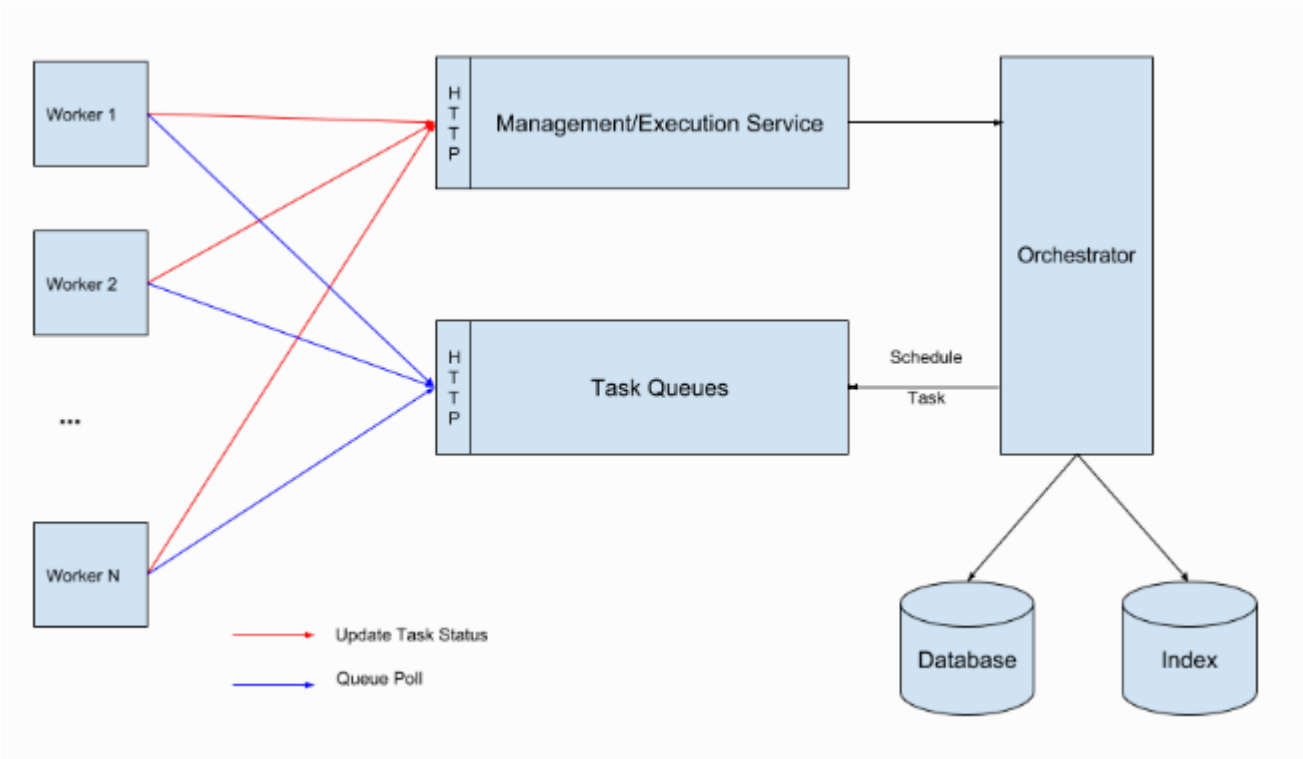
- permette la creazione di processi complessi in cui ogni task viene implementato come un microservizio
- il flusso di esecuzione è definito in JSON
- garantisce tracciabilità e visibilità all'interno dei flussi di esecuzione
- permette il riuso di microservizi esistenti
- mette a disposizione un'interfaccia grafica per visualizzare il flusso di esecuzione
- permette la sincronizzazione dei task
- è scalabile in quanto è capace di gestire molti flussi di esecuzione contemporaneamente
- la cominucazione è realizzata mediante richieste HTTP oppure RPC.

Nella realizzazione di Conductor è stata preferita l'orchestrazione alla coreografia in quanto la prima permetteva una scalabilità migliore.

Di seguito è mostrata l'architettura ad alto livello di Conductor.



Conductor segue il modello di comunicazione basato su RPC in cui i workers sono in esecuzione su una macchina separata dal server. I workers comunicano con il server mediante richieste HTTP.



Quindi:

- I workers sono remote systems e comunicano con conductor server mediante richieste HTTP (oppure tramite qualsiasi meccanismo che supporti RPC)
- Le Task Queues sono usate per schedulare i tasks.
- Per la persistenza dei dati viene utilizzato DynomiteDB.

Per il nostro progetto Conductor è stato installato su una macchina Linux. In particolare è stato installato un "In-Memory Server" e quindi i dati non vengono veramente memorizzati sul disco, ma sono persi una volta spento il server.

In particolare i passi seguiti sono:

- **https://github.com/Netflix/conductor** per clonare la repo scaricando in formato zip
- **sudo unzip nomefile.zip**
- **sudo chmod 777 conductor-master -R** : in modo da avere tutti i permessi sulla cartella.
- Dal sito **https://search.maven.org/#search%7Cga%7C1%7Cnetflix%20conductor** scaricare il file con estensione all.jar relativo a:
  - GroupID: com.netflix.conductor
  - artifact: conductor-server-all
  - versione: 1.6+
- Creare nella directory conductor-master/server la cartella build
- Creare nella directory build la cartella libs
- copiare nella directory conductor-master/server/build/libs il file scaricato precedentemente
- installare docker-compose (sudo apt-get install docker-compose);
- a questo punto effettuare i comandi:
  - **cd docker**
  - **sudo docker-compose build (attendere la terminazione)**
  - **sudo docker-compose up**

Per interagire con l'interfaccia grafica è necessario aprire un browser e digitare localhost:5000, mentre per interagire con Swagger, è necessario digitare localhost:8080.

Tramite l'interfaccia grafica è possibile osservare i task e i workflow presenti in Conductor e il loro stato di esecuzione (Running, Complete, Failed, Terminate), mentre con Swagger è possibile gestire i task e i workflow per crearne di nuovi o modificarli.

0.1 Lavorare con Conductor

Conductor si basa principalmente su due concetti principali:

- Workflow
- Task.

0.1.1 Workflow

I workflows sono definiti utilizzando un linguaggio di dominio basato su JSON (JSON based DSL) e includono i task che sono eseguiti come parte del workflow.

0.1.2 Tasks

I task devono essere necessariamente registrati prima di essere eseguiti in un workflow. Si hanno due categorie di task:

- System Task
- Worker Task

0.1.2.1 System Tasks

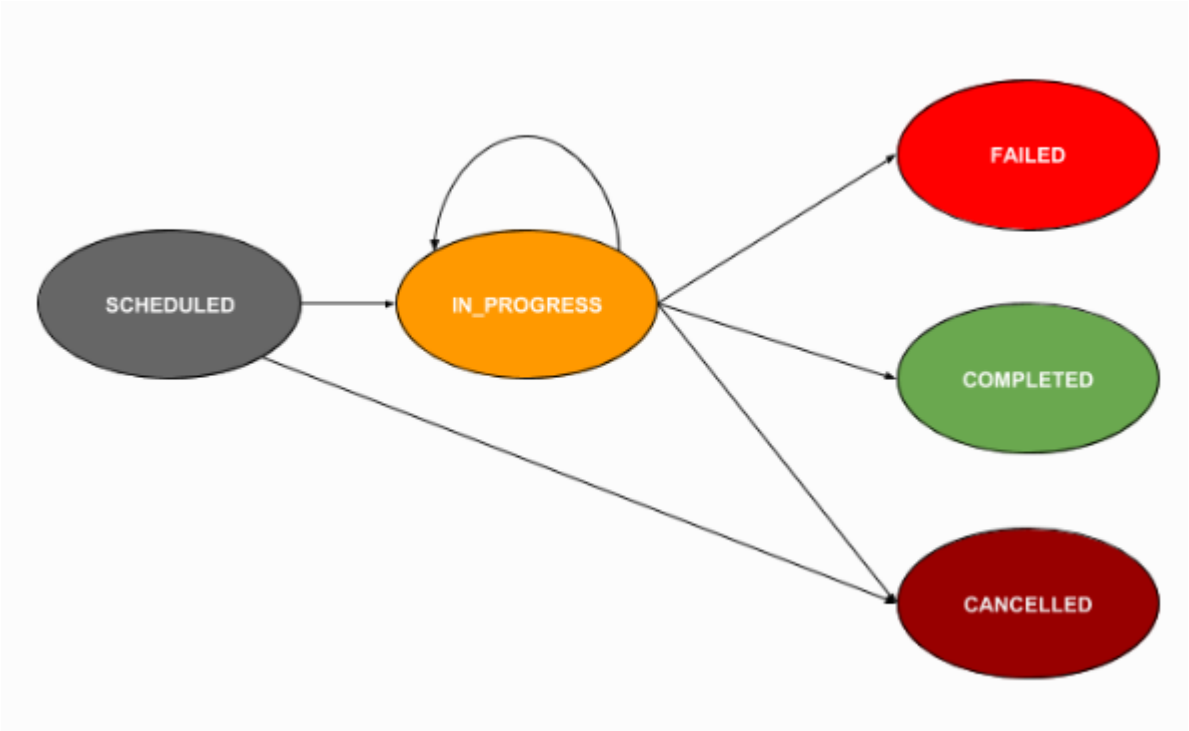
I System Tasks sono eseguiti all'interno della JVM di Conductor e sono gestiti da quest'ultimo. Abbiamo vari tipi di System Tasks:

Task	Scopo
Dynamic Task	definito in maniera dinamica in base all'input
Decision	Implementa uno switch case
Fork	Definisce l'esecuzione parallela di un insieme di task.
Dynamic Fork	I task da eseguire in parallelo sono decisi in base all'input.
Join	Unione di rami paralleli
Sub Workflow	Necessario per la realizzazione di workflow innestati
Wait	task che rimane nello stato di IN_PROGRESS fino ad un trigger esterno
HTTP	usato per chiamare microservizi tramite richieste HHTP
Event	Produce un evento

0.1.2.2 Worker tasks

I worker tasks sono implementati dalle applicazioni ed eseguono in un ambiente separato dal Conductor. Possono essere implementanti in qualsiasi linguaggio di programmazione, comunicanco con il Conductor server tramite delle REST ASPI. Nel modello sono definiti "SIMPLE".

Di seguito è riportato uno schema dei possibili stati di un task o di un workflow.



0.2 Conductor nel nostro progetto

0.2.1 Tasks

Come già detto in precedenza per poter realizzare un workflow è necessario definire a priori i task che fanno parte di quest'ultimo. In particolare nel nostro progetto sono stati definiti i seguenti task per la realizzazione dei workflow:

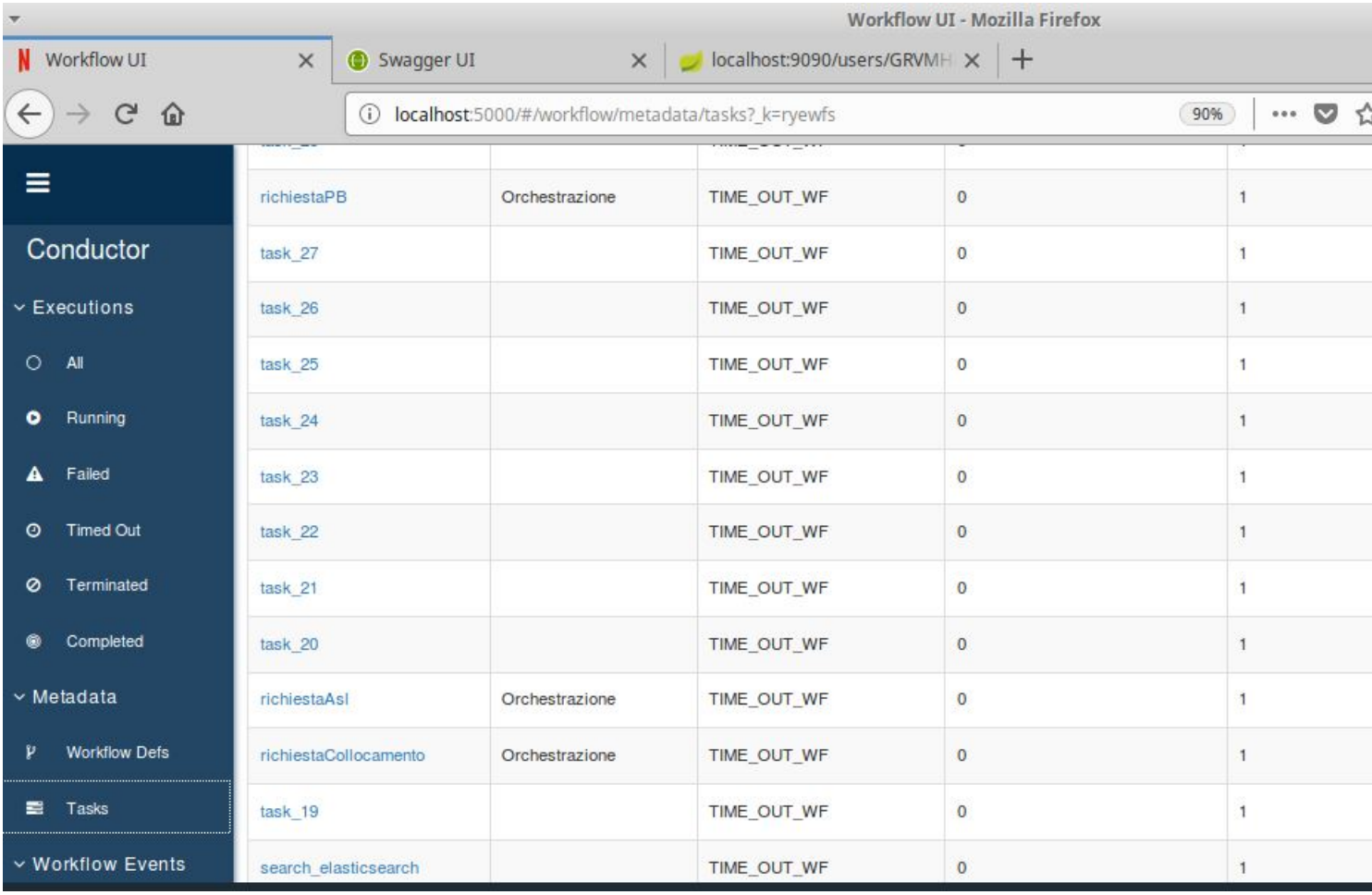


Figura 1: Tasks definiti

- richiestaPB : tale task è stato realizzato per poter effettuare una richiesta al servizio Pagine bianche.
- richiestaAsl : tale task è stato realizzato per poter effettuare una richiesta al servizio Asl.
- richiestaCollocamento : tale task è stato realizzato per poter effettuare una richiesta al servizio Collocamento.

Di seguito sono riportati i codici JSON relativi alla realizzazione di ogni task:

Per creare i task si è utilizzata l'interfaccia grafica Swagger, in paricolare i passi eseguiti sono i seguenti :

- Collegarsi all'interfaccia Swagger localhost:8080
- Tra le varie liste di operazioni disponibili selezionare il menù Metadata Management
- Selezionare "Create new task definition(s)"

```
[
  {
      "ownerApp": "Orchestrazione",
      "createTime": 0,
      "updateTime": 0,
      "createdBy": "MAP",
      "updatedBy": "MAP",
      "name": "richiestaPB",
      "description": "richiesta al servizio PB",
      "retryCount": 1,
      "timeoutSeconds": 0,
      "inputKeys": [
          "cf"
      ],
      "outputKeys": [
          "utente"
      ],
      "timeoutPolicy": "TIME_OUT_WF",
      "retryLogic": "FIXED",
      "retryDelaySeconds": 60,
      "responseTimeoutSeconds": 60,
      "concurrentExecLimit": 0,
      "inputTemplate": {}
  }
]
```

```
[
  {
      "ownerApp": "Orchestrazione",
      "createTime": 0,
      "updateTime": 0,
      "createdBy": "MAP",
      "updatedBy": "MAP",
      "name": "richiestaAsl",
      "description": "richiesta al servizio Asl",
      "retryCount": 1,
      "timeoutSeconds": 0,
      "inputKeys": [
          "cf"
      ],
      "outputKeys": [
          "utente"
      ],
```

```
      "timeoutPolicy": "TIME_OUT_WF",
      "retryLogic": "FIXED",
      "retryDelaySeconds": 60,
      "responseTimeoutSeconds": 60,
      "concurrentExecLimit": 0,
      "inputTemplate": {}
  }
]
```

```
[
  {
      "ownerApp": "Orchestrazione",
      "createTime": 0,
      "updateTime": 0,
      "createdBy": "MAP",
      "updatedBy": "MAP",
      "name": "richiestaCollocamento",
      "description": "richiesta al servizio Collocamento",
      "retryCount": 1,
      "timeoutSeconds": 0,
      "inputKeys": [
          "cf"
      ],
      "outputKeys": [
          "utente"
      ],
      "timeoutPolicy": "TIME_OUT_WF",
      "retryLogic": "FIXED",
      "retryDelaySeconds": 60,
      "responseTimeoutSeconds": 60,
      "concurrentExecLimit": 0,
      "inputTemplate": {}
  }
]
```

0.2.2 Workflows

Dopo aver definito i vari task abbiamo definito due diversi workflow :

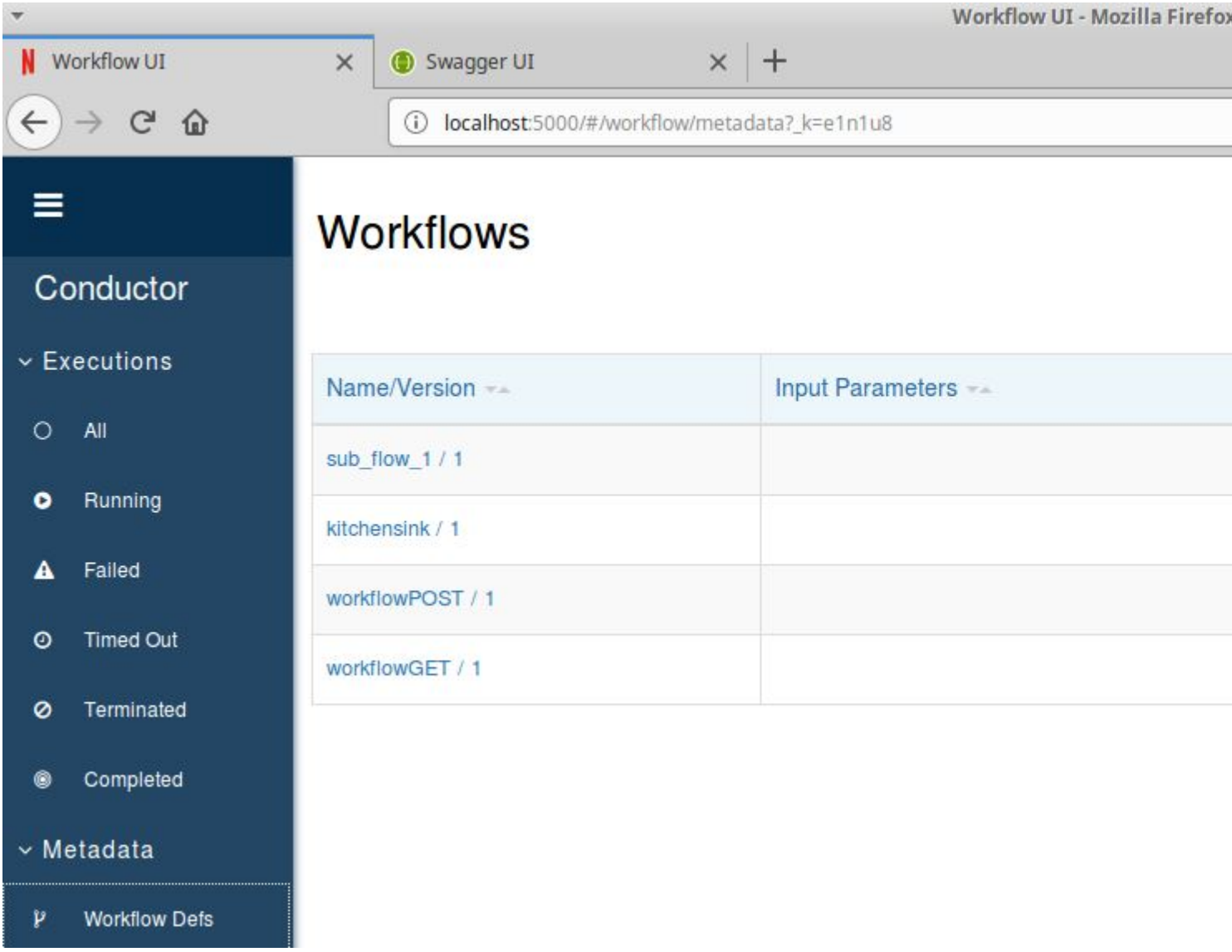


Figura 2: Workflow definiti

- workflowGet : per la realizzazione di tale workflow si è fatto uso non solo di task da noi definiti, ma anche di alcuni system task ovvero “Decision” , “Fork” e “Join”. In particolare questo workflow definisce il flusso per poter realizzare le seguenti funzionalità :
  - Effettuare una chiamata ad un singolo servizio tra Salute-Info-Lavoro per ricevere informazioni relative ad un utente.
  - Effettuare delle chiamate in maniera parallela ai tre servizi disponibili per ricevere tutte le informazioni disponibili relative ad un utente.
- workflowPost : tale workflow definisce il flusso per poter realizzare le seguenti funzionalità :

- Effettuare una chiamata ad un singolo servizio tra Salute-Info-Lavoro per poter registrare le informazioni relative ad un utente.
- Effettuare delle chiamate in maniera parallela ai tre servizi disponibili, distribuendo le varie informazioni relative ad un utente al fine registrarle.

Di seguito è riportato il codice e lo schema relativo ad ogni workflow:

Per creare i workflows si è utilizzata l'interfaccia grafica Swagger, in paricolare i passi eseguiti sono i seguenti :

- Collegarsi all'interfaccia Swagger localhost:8080
- Tra le varie liste di operazioni disponibili selezionare il menù Metadata Management
- Selezionare "Create new workflow definition"

0.2.2.1 Workflow Get

```
{
  "ownerApp": "Orchestrazione",
  "createTime": 0,
  "updateTime": 0,
  "createdBy": "MAP",
  "name": "workflowGET",
  "description": "orchestrazione dei servizi",
  "version": 1,
  "tasks": [
  {
  "name": "decision",
  "taskReferenceName": "decision",
  "inputParameters": {
    "scelta": "${workflow.input.scelta}"
  },
  "type": "DECISION",
  "caseValueParam": "scelta",
  "decisionCases": {
    "info": [{
          "name": "richiestaPB",
              "taskReferenceName": "richiestaPB",
            "inputParameters": {
            "http_request":{
                "uri":  "http://172.17.0.1:8081/utente/${workflow.input.cf}",
                  "method": "GET"
                }
                  },
            "type": "HTTP"
```

```
    }],
    "salute": [{
          "name": "richiestaAsl",
            "taskReferenceName": "richiestaAsl",
            "inputParameters": {
            "http_request":{
                "uri":  "http://172.17.0.1:8082/user/${workflow.input.cf}",
                  "method": "GET"
                }
                  },
            "type": "HTTP"
    }],
    "lavoro": [{
          "name": "richiestaCollocamento",
              "taskReferenceName": "richiestaCollocamento",
            "inputParameters": {
            "http_request":{
                "uri":  "http://172.17.0.1:9090/users/${workflow.input.cf}",
                  "method": "GET"
                }
                  },
            "type": "HTTP"
    }],
    "all": [{
      "name": "fork_join",
         "taskReferenceName": "forktask",
        "type": "FORK_JOIN",
        "forkTasks": [
        [{
          "name": "richiestaPB",
          "taskReferenceName": "richiestaPBRef",
          "inputParameters": {
            "http_request":{
            "uri":  "http://172.17.0.1:8081/utente/${workflow.input.cf}",
             "method": "GET"
              }
          },
          "type": "HTTP"
        }],
      [{
        "name": "richiestaAsl",
            "taskReferenceName": "richiestaAslRef",
            "inputParameters": {
            "http_request":{
```

```
            "uri":  "http://172.17.0.1:8082/user/${workflow.input.cf}",
              "method": "GET"
            }
                 },
             "type": "HTTP"
            }],
      [{
        "name": "richiestaCollocamento",
              "taskReferenceName": "richiestaCollocamentoRef",
              "inputParameters": {
              "http_request":{
            "uri":  "http://172.17.0.1:9090/users/${workflow.input.cf}",
              "method": "GET"
                }
                           },
              "type": "HTTP"
                }]
        ]},
          {
               "name": "joint",
              "taskReferenceName": "joint",
              "type": "JOIN",
              "joinOn" :[
                   "richiestaPBRef",
                   "richiestaAslRef",
              "richiestaCollocamentoRef"
                ]
            }]
      }
    }],
    "outputParameters": {
      "infoU": "${richiestaPB.output.response.body}",
      "saluteU": "${richiestaAsl.output.response.body}",
    "lavoroU": "${richiestaCollocamento.output.response.body}",
      "infoAll": "${richiestaPBRef.output.response.body}",
      "saluteAll": "${richiestaAslRef.output.response.body}",
    "lavoroAll": "${richiestaCollocamentoRef.output.response.body}",

  },
    "schemaVersion": 2
}]
```

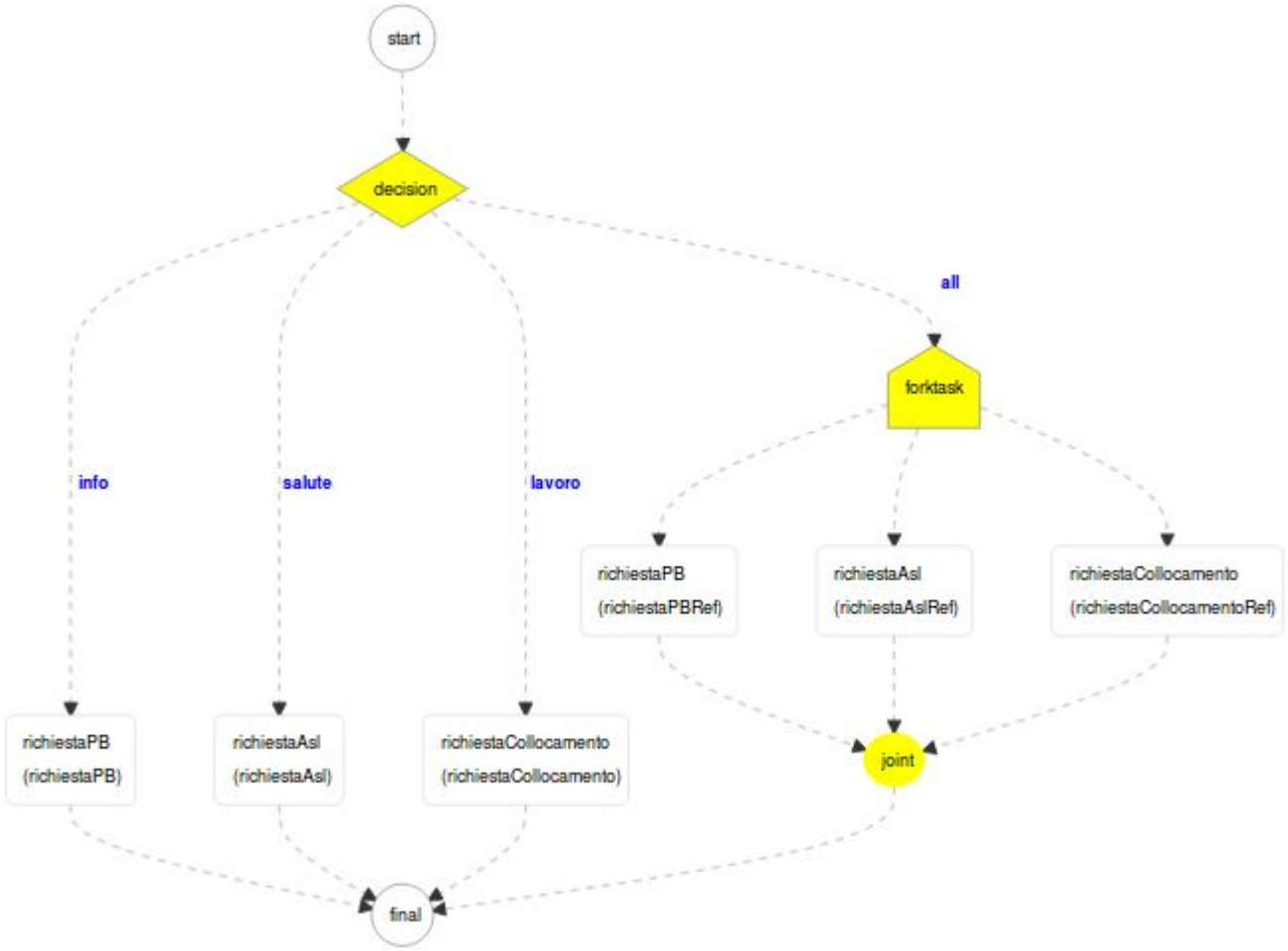


Figura 3: WorkflowGet

0.2.2.2 Workflow Post

```
{
  "ownerApp": "Orchestrazione",
  "createTime": 0,
  "updateTime": 0,
  "createdBy": "MAP",
  "name": "WorkFPostAPB",
  "description": "orchestrazione dei servizi",
  "version": 1,
  "tasks": [
    {
  "name": "decision",
  "taskReferenceName": "decision",
  "inputParameters": {
    "scelta": "${workflow.input.scelta}"
    },
```

```
  "type": "DECISION",
  "caseValueParam": "scelta",
  "decisionCases": {
    "info": [{
        "name": "richiestaPB",
        "taskReferenceName": "richiestaPB",
        "inputParameters": {
          "http_request":{
            "uri":  "http://172.17.0.1:8081/aggiungi",
              "method": "POST",
            "contentType": "application/json",
            "body":{
              "cf": "${workflow.input.codiceFiscale}",
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "indirizzo": "${workflow.input.indirizzo}",
              "numerofisso": "${workflow.input.numFisso}",
              "cellulare": "${workflow.input.cellulare}",
              "email": "${workflow.input.email}"
            }
          }
         },
         "type": "HTTP"
    }],
    "salute": [{
        "name": "richiestaAsl",
          "taskReferenceName": "richiestaAsl",
          "inputParameters": {
              "http_request":{
            "uri":  "http://172.17.0.1:8082/user/add",
            "method": "POST",
            "contentType": "application/json",
            "body":{
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "dataNascita": "${workflow.input.dataNascita}",
              "sesso": "${workflow.input.sesso}",
              "codiceFiscale": "${workflow.input.codiceFiscale}",
              "medicoBase": "${workflow.input.medicoBase}"
            }
          }
              },
              "type": "HTTP"
      }],
```

```
    "lavoro": [{
        "name": "richiestaCollocamento",
        "taskReferenceName": "richiestaCollocamento",
        "inputParameters": {
              "http_request":{
            "uri":  "http://172.17.0.1:9090/users/add",
              "method": "POST",
            "contentType": "application/json",
            "body":{
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "cfis": "${workflow.input.codiceFiscale}",
              "data_nascita": "${workflow.input.dataNascita}",
              "lavoro_attuale": "${workflow.input.lavoroAtt}",
              "lavoro_precedente": "${workflow.input.lavoroPrec}"
            }
          }
              },
          "type": "HTTP"
    }],
    "all": [{
       "name": "fork_join",
      "taskReferenceName": "forktask",
       "type": "FORK_JOIN",
         "forkTasks": [
        [{
          "name": "richiestaPB",
        "taskReferenceName": "richiestaPBPar",
        "inputParameters": {
          "http_request":{
            "uri":  "http://172.17.0.1:8081/aggiungi",
              "method": "POST",
            "contentType": "application/json",
            "body":{
              "cf": "${workflow.input.codiceFiscale}",
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "indirizzo": "${workflow.input.indirizzo}",
              "numerofisso": "${workflow.input.numFisso}",
              "cellulare": "${workflow.input.cellulare}",
              "email": "${workflow.input.email}"
            }
          }
         },
```

```
                "type": "HTTP"
      }],
      [{
        "name": "richiestaAsl",
          "taskReferenceName": "richiestaAslPar",
          "inputParameters": {
              "http_request":{
            "uri":  "http://172.17.0.1:8082/user/add",
            "method": "POST",
            "contentType": "application/json",
            "body":{
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "dataNascita": "${workflow.input.dataNascita}",
              "sesso": "${workflow.input.sesso}",
              "codiceFiscale": "${workflow.input.codiceFiscale}",
              "medicoBase": "${workflow.input.medicoBase}"
            }
          }
              },
                "type": "HTTP"
                  }],
      [{
        "name": "richiestaCollocamento",
        "taskReferenceName": "richiestaCollocamentoPar",
        "inputParameters": {
              "http_request":{
            "uri":  "http://172.17.0.1:9090/users/add",
              "method": "POST",
            "contentType": "application/json",
            "body":{
              "nome": "${workflow.input.nome}",
              "cognome": "${workflow.input.cognome}",
              "cfis": "${workflow.input.codiceFiscale}",
              "data_nascita": "${workflow.input.dataNascita}",
              "lavoro_attuale": "${workflow.input.lavoroAtt}",
              "lavoro_precedente": "${workflow.input.lavoroPrec}"
            }
          }
              },
          "type": "HTTP"
        }]
          ]},
          {
```

```
            "name": "joint",
            "taskReferenceName": "joint",
            "type": "JOIN",
            "joinOn" :[
              "richiestaPBPar",
              "richiestaAslPar",
        "richiestaCollocamentoPar"
      ]
      }]
    }
  }],
    "outputParameters": {
      "infoU": "${richiestaPB.output.response.body}",
      "saluteU": "${richiestaAsl.output.response.body}",
    "lavoroU": "${richiestaCollocamento.output.response.body}",
      "infoAll": "${richiestaPBPar.output.response.body}",
      "saluteAll": "${richiestaAslPar.output.response.body}",
    "lavoroAll": "${richiestaCollocamentoPar.output.response.body}",

  },
  "schemaVersion": 2 }
```

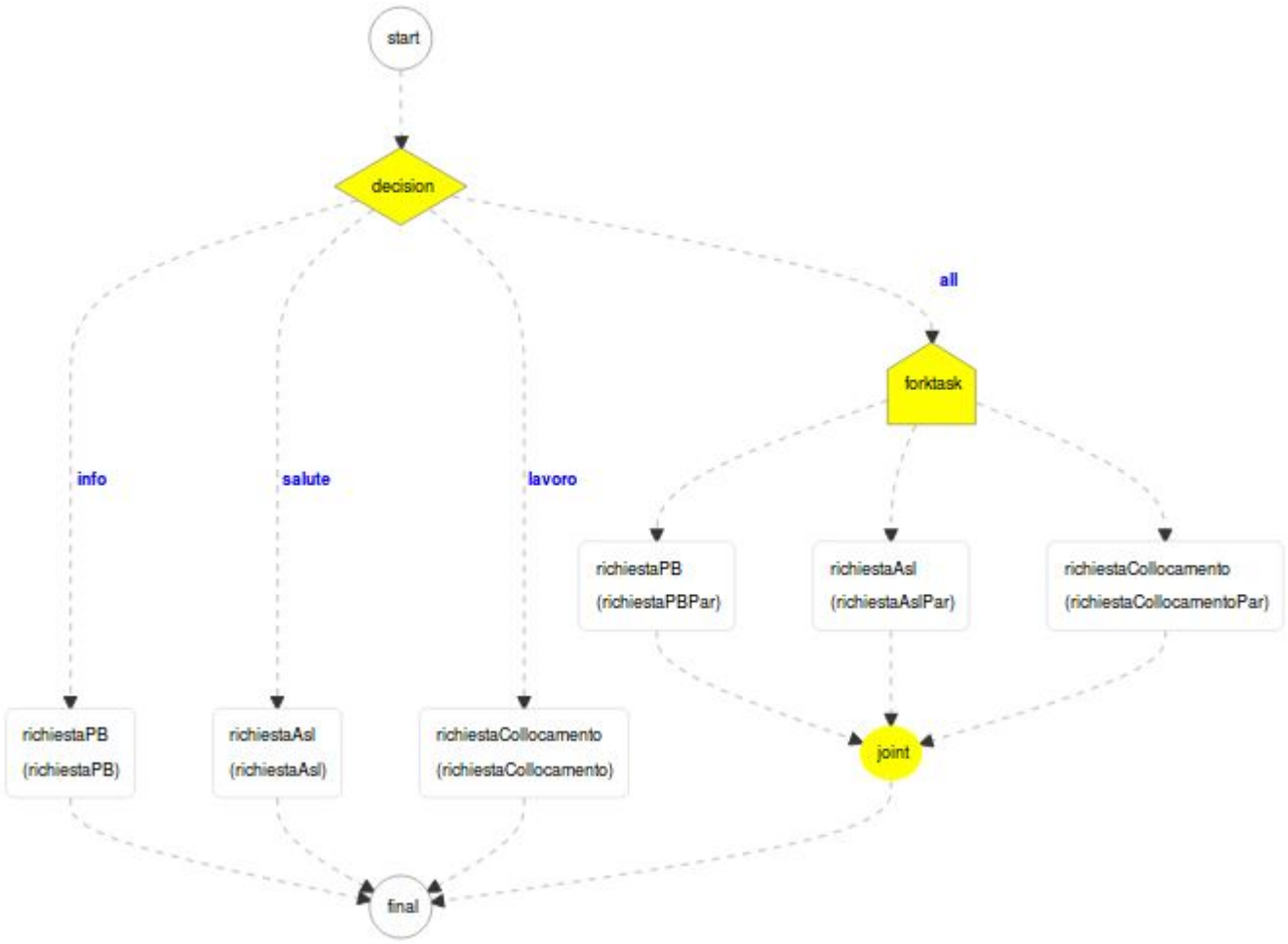


Figura 4: WorkflowPost

0.2.3 Esempi di esecuzione

In tale sezione sono riportati alcuni esempi di utilizzo dei servizi , task e workflow creati. In particolare di seguito sono riportati i passi generici per poter effettuare una richiesta :

- Collegarsi all'interfaccia Swagger localhost:8080
- Tra le varie liste di operazioni disponibili selezionare il menù Workflow Management
- Selezionare "Start a new workflow with StartWorkflowRequest, which allows task to be executed in a domain"

0.2.3.1 Richiesta informazioni Salute

Esempio di richiesta al servizio Asl:

```
{
    "name": "workflowGET",
    "version": 1,
```

```
    "input": {
    "scelta": "salute",
        "cf": "GRVMHL94"
                },
    "taskToDomain": {}
}
```

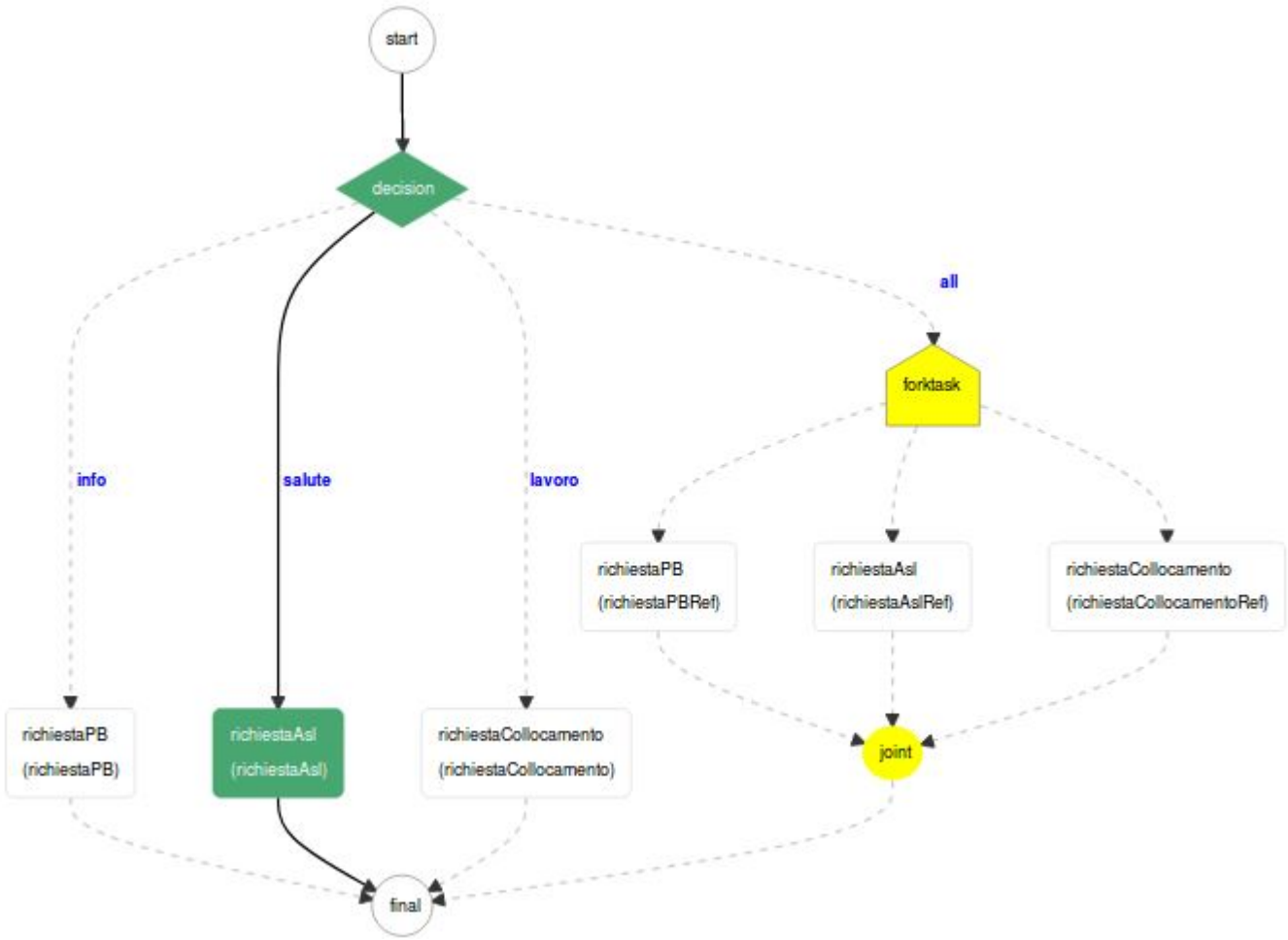


Figura 5: Workflow richiesta Asl

Workflow Input


```
{
   "scelta": "salute",
   "cf": "GRVMHL94"
}
```


Workflow Output


```
{
   "infoU": null,
   "saluteU": {
      "nome": "Michela",
      "cognome": "Gravina",
      "dataNascita": "03/11/1994",
      "sesso": "femmina",
      "codiceFiscale": "GRVMHL94",
      "medicoBase": "Giovanni Marzullo"
   },
   "lavoroU": null
```


Figura 6: Input/Output richiesta asl

Esempio di richiesta ai servizi Asl, Pagine Bianche e Collocamento:

```
{
    "name": "workflowGET",
    "version": 1,
    "input": {
    "scelta": "all",
        "cf": "GRVMHL94"
                },
    "taskToDomain": {}
}
```

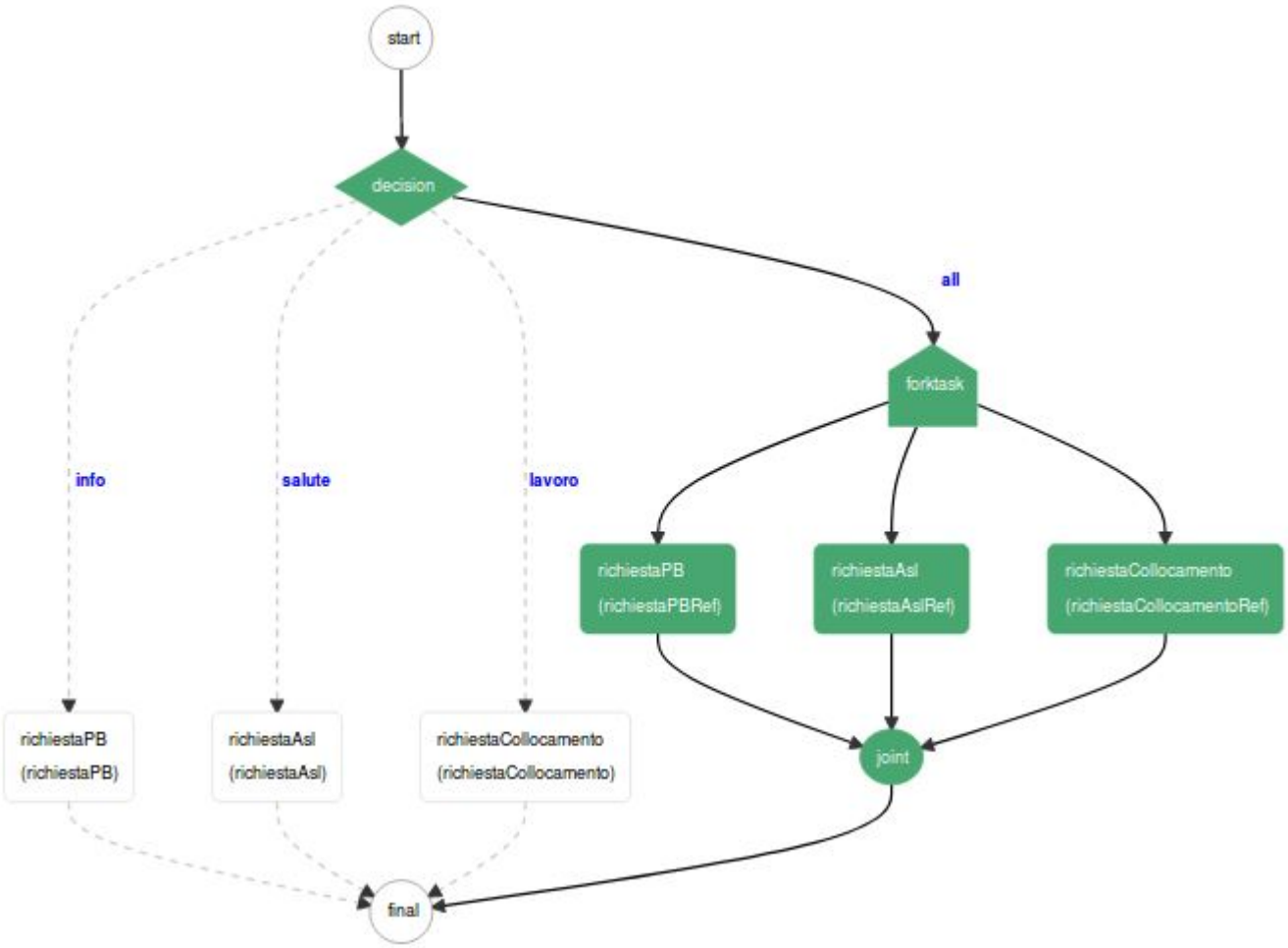


Figura 7: Workflow richiesta Asl

Figura 8: Input/Output richiesta asl, pagine bianche e collocamento

Esempio di richiesta di memorizzazione nuovo utente al servizio Asl:

```
{
    "name": "workflowPOST",
   "version": 1,
    "input": {
    "scelta": "salute",
```

```
    "codiceFiscale": "GLLANN91",
    "nome": "Anna",
    "cognome": "Galli",
    "dataNascita": "23/09/1991",
    "indirizzo": "via Napoli",
     "email": "anna@liber.it",
     "lavoroPrec": "disoccupata",
     "lavoroAtt":  "impiegata",
     "medicoBase": "Giovanni Marsico",
      "sesso": "femmina",
      "numFisso": "0825884384",
      "cellulare": "3425678456"
  },
   "taskToDomain": {}
  }
```

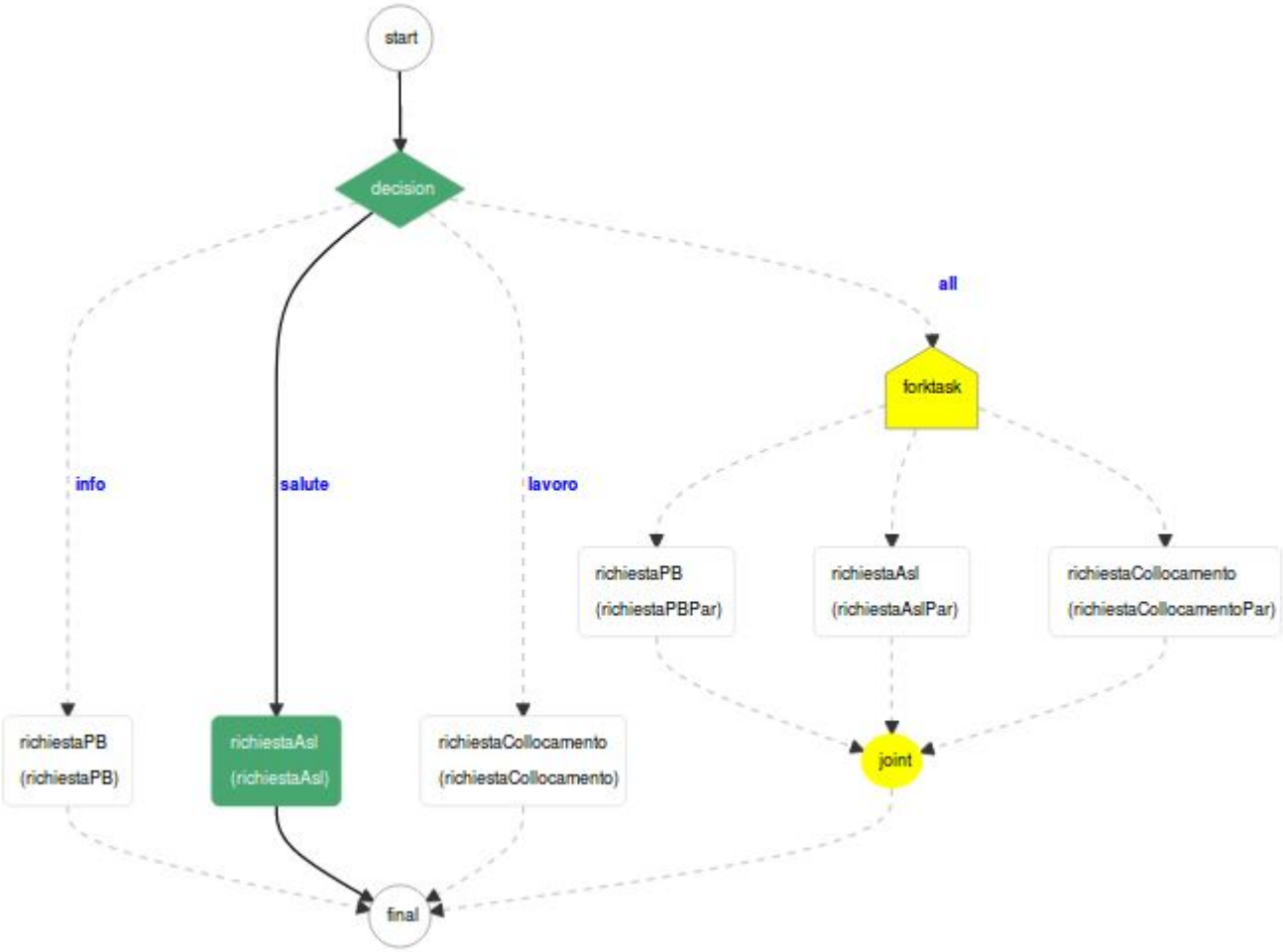


Figura 9: Workflow richiesta salvataggio utente Asl

Workflow Input


```
{
   "scelta": "salute",
   "codiceFiscale": "GLLANN91",
   "nome": "Anna",
   "cognome": "Galli",
   "dataNascita": "23/09/1991",
   "indirizzo": "via Napoli",
   "email": "anna@liber.it",
   "lavoroPrec": "disoccupata",
   "lavoroAtt": "impiegata",
   "medicoBase": "Giovanni Marsico",
```


Workflow Output


```
{
   "infoU": null,
   "saluteU": "true",
   "lavoroU": null,
   "infoAll": null,
   "saluteAll": null,
   "lavoroAll": null
}
```


Figura 10: Input/Output richiesta asl

Esempio di richiesta di memorizzazione nuovo utente ai servizi Asl, Pagine Bianche e Collocamento :


```
{
    "name": "workflowPOST",
   "version": 1,
    "input": {
    "scelta": "all",
      "codiceFiscale": "GLLANN91",
      "nome": "Anna",
      "cognome": "Galli",
      "dataNascita": "23/09/1991",
      "indirizzo": "via Napoli",
       "email": "anna@liber.it",
       "lavoroPrec": "disoccupata",
       "lavoroAtt":  "impiegata",
       "medicoBase": "Giovanni Marsico",
        "sesso": "femmina",
        "numFisso": "0825884384",
        "cellulare": "3425678456"
   },
    "taskToDomain": {}
   }
```

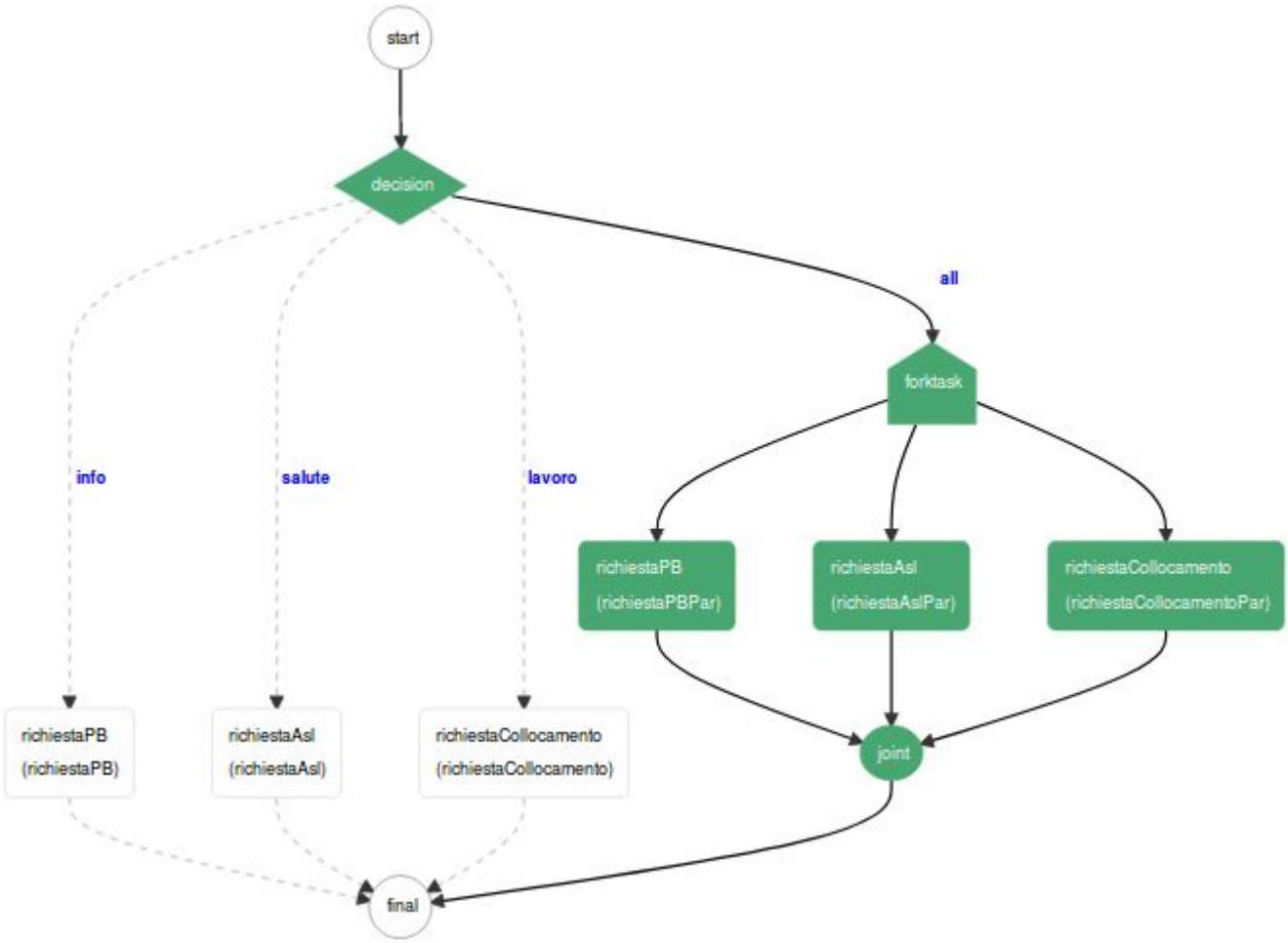


Figura 11: Workflow richiesta salvataggio utente Asl, Pagine Bianche e Collocamento

Workflow Input


```
{
   "scelta": "all",
   "codiceFiscale": "GLLANN94",
   "nome": "Anna",
   "cognome": "Galli",
   "dataNascita": "23/09/1991",
   "indirizzo": "via Napoli",
   "email": "anna@liber.it",
   "lavoroPrec": "disoccupata",
   "lavoroAtt": "impiegata",
   "medicoBase": "Giovanni Marsico",
```


Workflow Output


```
{
   "infoU": null,
   "saluteU": null,
   "lavoroU": null,
   "infoAll": "true",
   "saluteAll": "true",
   "lavoroAll": "true"
}
```


Figura 12: Input/Output richiesta asl, pagine bianche e collocamento