
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laurea Magistrale in Informatica

# *ACMEat*

***Relazione del progetto del corso di***
***Ingegneria del Software Orientata ai Servizi***

*Studenti:*
**Lorenzo BALUGANI**
**Alberto PAPARELLA**
**Mae SOSTO**

*Docenti:*
**Prof. Ivan LANESE**
**Prof. Davide ROSSI**

Anno Accademico 2021 – 2022

# Indice

# 1 Introduzione

## 2 Descrizione del dominio e del problema

La società ACMEat propone ai propri clienti un servizio che permette di selezionare un menu da uno fra un insieme di locali convenzionati e farselo recapitare a domicilio.
Per poter usufruire del servizio il cliente deve inizialmente selezionare un comune fra quelli nei quali il servizio è attivo. A fronte di questa selezione ACMEat presenta la lista dei locali convenzionati che operano in quel comune e dei menù che offrono. Il cliente può quindi specificare locale e menù di suo interesse e una fascia oraria per la consegna (si tratta di fasce di 15 minuti tra le 12 e le 14 e tra le 19 e le 21).
Segue quindi una fase di pagamento che viene gestita attraverso un istituto bancario terzo al quale il cliente viene indirizzato. A fronte del pagamento l'istituto rilascia un token al cliente il quale lo comunica ad ACMEat, che a sua volta lo usa per verificare con la banca che il pagamento sia stato effettivamente completato. A questo punto l'ordine diventa operativo. I clienti possono comunque ancora annullare l'ordine ma non più tardi di un'ora prima rispetto all'orario di consegna. In tal caso ACMEat chiede alla banca l'annullamento del pagamento.
ACMEat conosce tutti i locali convenzionati nei vari comuni nei quali opera e i loro giorni e orari di operatività. Nel caso in cui un locale non sia disponibile in un giorno in cui dovrebbe normalmente essere aperto è responsabilità del locale stesso contattare ACMEat entro le 10 del mattino comunicando tale indisponibilità. Entro tale orario vanno anche comunicati cambiamenti dei menu proposti (in mancanza di tale comunicazione si assume che siano disponibili gli stessi del giorno precedente). I locali vengono anche contattati ad ogni ordine per verificare che siano effettivamente in grado di far fronte alla richiesta del cliente. In caso negativo l'accettazione dell'ordine si interrompe prima che si passi alla fase di pagamento.
Per la consegna ACMEat si appoggia a più società esterne: per ogni consegna vengono contattate tutte le società che abbiano sede entro 10 chilometri dal comune interessato specificando: indirizzo del locale dove ritirare il pasto, indirizzo del cliente cui recapitarlo e orario previsto di consegna. A fronte di questa richiesta le società devono rispondere entro 15 secondi specificando la loro disponibilità e il prezzo richiesto; ACMEat sceglierà fra le disponibili che avranno risposto nel tempo richiesto quella che propone il prezzo più basso. Nel caso in cui nessuna società di consegna sia disponibile l'ordine viene annullato prima che si passi alla fase di pagamento.

# 3 Modellazione delle comunicazioni

La prima fase di lavoro ha visto la realizzazione di una *coreografia* (riportata in Lst. 1) con l'obiettivo di modellare le comunicazioni dello scenario descritto nella Sez. 2. Tale coreografia è stata iterativamente raffinata in modo da migliorare il più possibile le sue proprietà di *correctedness*; per motivi di spazio, viene riportata solo l'ultima coreografia frutto di questo lavoro di raffinamento. La coreografia è stata poi proiettata in un *sistema di ruoli*, riportato nella Sez. 3.3, in cui vengono distinti i seguenti ruoli: *ACMEat*, la *banca*, il *cliente*, il *fattorino*, il *ristorante* e il *servizio di spedizione*. Infine, viene riportata una modellazione della coreografia anche attraverso un diagramma di coreografia BPMN.

## 3.1 Coreografia

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  SelezioneComune : Cliente -> ACMEat ;
  InvioListaLocali : ACMEat -> Cliente
)

CoreografiaOrdine :==   (
  Ordine : Cliente -> ACMEat ;
  Richiesta disponibilitá ristorante : ACMEat -> Ristorante;
  DisponibilitáR: Ristorante -> ACMEat;
  // Disponibile ?
  (
    // No
    Annullamento procedura : ACMEat -> Cliente
  ) + (
    // Sí
    (
      Richiesta disponibilitá SDS : ACMEat -> SDS;
      ( //In 15 secondi
        PreventivoDisponibilitá: SDS -> ACMEat +
        1
      )
    )* ;
  // Lista vuota ?
  (
    // Sí
    AnnullamentoC : ACMEat -> Cliente |
    AnnullamentoR : ACMEat -> Ristorante ;
  ) + (
    // No
    Contatta SDS costo minore: ACMEat -> SDS ;
    ACK: SDS -> ACMEat;
    CreazioneRichiestaPagamento: ACMEat -> Banca ;
```

```
39      ConfermaCreazioneTransazione: Banca -> ACMEat ;
40      RedirezionePagamento: ACMEat -> Cliente ;
41      // Modellazione timeout
42      (
43        Pagamento : Cliente -> Banca;
44        InvioTokenC : Banca -> Cliente;
45        InvioTokenA : Cliente -> ACMEat ;
46        RichiestaValiditá : ACMEat -> Banca ;
47        ValiditáToken : Banca -> ACMEat ;
48        // Token valido ?
49        (
50          // No
51          ErrorePagamento: ACMEat -> Cliente   |
52          AnnullamentoR : ACMEat -> Ristorante |
53          AnnullamentoS : ACMEat -> SDS        |
54        ) + (
55          // Sí
56          AttivazioneOrdineR : ACMEat -> Ristorante |
57          AttivazioneOrdineS : ACMEat -> SDS        |
58          ConfermaOrdine: ACMEat -> Cliente ;
59
60          // Manca meno di un'ora ?
61          (
62            // No
63            // Annullare ?
64            (
65              // Sí
66              AnnullamentoOrdine: Cliente -> ACMEat ;
67              (
68                AnnullamentoPagamento: ACMEat -> Banca;
69                Rimborso: Banca -> Cliente;
70              ) |
71              (
72                AnnullamentoR : ACMEat -> Ristorante;
73                RicevutoAnnullamento: Ristorante -> ACMEat;
74              ) |
75              (
76                AnnullamentoS : ACMEat -> SDS;
77                RicevutoAnnullamento: SDS -> ACMEat;
78              )
79            ) + (
80              // No
81              1
82            )
83          ) + (
84            // Sí
85            (
86              PagamentoR : ACMEat -> Banca;
87              RicezionePagamentoR : Banca -> Ristorante;
88            ) |
89            (
90              PagamentoS : ACMEat -> Banca;
91              RicezionePagamentoS : Banca -> SDS ;
```

```
92            ) |
93            (
94              ConsegnaMerceF : Ristorante -> Fattorino ;
95              ConcegnaMerceC : Fattorino -> Cliente;
96              ConfermaRicevutaSpedizione: Fattorino -> ACMEat;
97            )
98          )
99        ) + (
100         //Scadenza timer
101         ErrorePagamento: ACMEat -> Cliente |
102         (
103           Annullamento: ACMEat -> Ristorante;
104           RicevutoAnnullamento: Ristorante -> ACMEat;
105         ) |
106         (
107           Annullamento: ACMEat -> Corriere ;
108           RicevutoAnnullamento: Corriere -> ACMEat;
109         )
110       )
111     )
112   )
113 )
114
115 ModificaInformazioniLocali :==  (
116   RichiestaAggiornamento : Ristorante -> ACMEat ;
117   // Prima delle 10 ?
118   (
119     // No
120     RichiestaRifiutata : ACMEat -> Ristorante
121   ) + (
122     // Sí
123     RichiestaAccettata : ACMEat -> Ristorante
124   )
125 )
```

Listing 1: Coreografia dello scenario di utilizzo di ACMEat

## 3.2 Correctedness

...

## 3.3 Sistema di Ruoli

### 3.3.1 ACMEat

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :==  (
  SelezioneComune@Cliente;
  InvioListaLocali@Cliente
)

CoreografiaOrdine :==   (
  Ordine@Cliente ;
  Richiesta disponibilitá ristorante@Ristorante ;
  DisponibilitáR@Ristorante ;
  // Disponibile ?
  (
    // No
    Annullamento procedura@Cliente
  ) + (
    // Sí
    (
      Richiesta disponibilitá SDS@SDS ;
      (
        // In 15 secondi
        PreventivoDisponibilitá@SDS +
        1
      )
  )* ;
  // Lista vuota ?
  (
    // Sí
    AnnullamentoC@Cliente |
    AnnullamentoR@Ristorante ;
  ) + (
    // No
    Contatta SDS costo minore@SDS ;
    ACK@SDS ;
    CreazioneRichiestaPagamento@Banca ;
    ConfermaCreazioneTransazione@Banca ;
    RedirezionePagamento@Cliente ;
    // Modellazione timeout
    (
      1;
      1;
      InvioTokenA@Cliente ;
      RichiestaValiditá@Banca ;
      ValiditáToken@Banca ;
      // Token valido ?
```

```
50      (
51        // No
52        ErrorePagamento@Cliente |
53        AnnullamentoR@Ristorante |
54        AnnullamentoS@SDS |
55      ) + (
56        // Sí
57        AttivazioneOrdineR@Ristorante |
58        AttivazioneOrdineS@SDS |
59        ConfermaOrdine@Cliente ;
60        // Manca meno di un'ora ?
61        (
62          // No
63          // Annullare ?
64          (
65            // Sí
66            AnnullamentoOrdine@Cliente;
67            (
68              AnnullamentoPagamento@Banca ;
69              1 ;
70            ) |
71            (
72              AnnullamentoR@Ristorante ;
73              RicevutoAnnullamento@RistorantE ;
74            ) |
75            (
76              AnnullamentoS@SDS ;
77              RicevutoAnnullamento@SDS ;
78            )
79          ) +
80          (
81            // No
82            1
83          )
84        ) +
85        (
86          // Sí
87          (
88            PagamentoR@Banca ;
89            1 ;
90          ) |
91          (
92            PagamentoS@Banca ;
93            1 ;
94          ) |
95          (
96            1 ;
97            1 ;
98            ConfermaRicevutaSpedizione@Fattorino ;
99          )
100       )
101     ) +
102     (
```

```
        //Scadenza timer
        ErrorePagamento@Cliente |
        (
          Annullamento@Ristorante ;
          RicevutoAnnullamento@Ristorante ;
        ) |
        (
          Annullamento@Corriere ;
          RicevutoAnnullamento@Corriere ;
        )
      )
    )
  )
)

ModificaInformazioniLocali :==  (
  RichiestaAggiornamento@Ristorante ;
  // Prima delle 10 ?
  (
    // No
    RichiestaRifiutata@Ristorante
  ) +
  (
    // Sí
    RichiestaAccettata@Ristorante
  )
)
```

Listing 2: Proiezione della coreografia relativamente ad ACMEat

### 3.3.2 Banca

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  1 ;
  1
)

CoreografiaOrdine :== (
  1 ;
  1 ;
  1 ;
  // Disponibile ?
  (
    // No
    1
  ) +
  (
    // Sí
    (
      1 ;
      (
         // In 15 secondi
        1 +
        1
      )
    )* ;
    // Lista vuota ?
    (
      // Sí
      1 |
      1 ;
    ) +
    (
      // No
      1 ;
      1 ;
      CreazioneRichiestaPagamento@ACMEat ;
      ConfermaCreazioneTransazione@ACMEat ;
      1 ;
      // Modellazione timeout
      (
        Pagamento@Cliente ;
        InvioTokenC@Cliente ;
        1 ;
        RichiestaValiditá@ACMEat ;
        ValiditáToken@ACMEat ;
        // Token valido ?
```

```
52          (
53            // No
54            1 |
55            1 |
56            1 |
57          ) +
58          (
59            // Sí
60            1 |
61            1 |
62            1 ;
63            // Manca meno di un'ora ?
64            (
65              // No
66              // Annullare ?
67              (
68                // Sí
69                1 ;
70                (
71                  AnnullamentoPagamento@ACMEat ;
72                  Rimborso@Cliente ;
73                ) |
74              (
75                1 ;
76                1 ;
77              ) |
78              (
79                1 ;
80                1 ;
81              )
82            ) +
83            (
84              // No
85              1
86            )
87          ) +
88          (
89            // Sí
90            (
91              PagamentoR@ACMEat ;
92              RicezionePagamentoR@Ristorante ;
93            ) |
94            (
95              PagamentoS@ACMEat ;
96              RicezionePagamentoS@SDS ;
97            ) |
98            (
99              1 ;
100             1 ;
101             1 ;
102           )
103         )
104       ) +
```

```
105       (
106         // Scadenza timer
107         1 |
108         (
109           1 ;
110           1 ;
111         ) |
112         (
113           1 ;
114           1 ;
115         )
116       )
117     )
118   )
119 )
120
121 ModificaInformazioniLocali :== (
122   1 ;
123   // Prima delle 10 ?
124   (
125     // No
126     1
127   ) +
128   (
129     // Sí
130     1
131   )
132 )
```

Listing 3: Proiezione della coerografia relativamente alla banca

### 3.3.3 Cliente

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  1 ;
  InvioListaLocali@ACMEat
)

CoreografiaOrdine :== (
  Ordine@ACMEat ;
  1 ;
  1 ;
  // Disponibile ?
  (
    // No
    Annullamento procedura@ACMEat
  ) +
  (
    // Sí
    (
      1 ;
      (
        // In 15 secondi
        1 +
        1
      )
    )* ;
    // Lista vuota ?
    (
      // Sí
      AnnullamentoC@ACMEat |
      1;
    ) +
    (
      // No
      1 ;
      1 ;
      1 ;
      1 ;
      RedirezionePagamento@ACMEat ;
      // Modellazione timeout
      (
        Pagamento@Banca ;
        InvioTokenC@Banca ;
        InvioTokenA@ACMEat ;
        1 ;
        1 ;
        // Token valido ?
```

```
        (
          // No
          ErrorePagamento@ACMEat |
          1 |
          1
        ) +
        (
          // Sí
          1 |
          1 |
          ConfermaOrdine@ACMEat ;
          // Manca meno di un'ora ?
          (
            // No
            // Annullare ?
            (
              // Sí
              AnnullamentoOrdine@ACMEat ;
              (
                1 ;
                Rimborso@Banca ;
              ) |
              (
                1 ;
                1 ;
              ) |
              (
                1 ;
                1
              )
            ) +
            (
              // No
              1
            )
          ) +
          (
            // Sí
            (
              1 ;
              1
            ) |
            (
              1 ;
              1
            ) |
            (
              1 ;
              ConcegnaMerceC@Fattorino ;
              1
            )
          )
        ) +
```

```
        (
          //Scadenza timer
          ErrorePagamento@ACMEat |
          (
            1 ;
            1
          ) |
          (
            1 ;
            1
          )
        )
      )
    )
  )
)


ModificaInformazioniLocali :== (
  1 ;
  // Prima delle 10 ?
  (
    // No
    1
  ) +
  (
    // Sí
    1
  )
)
```

Listing 4: Proiezione della coreografia relativamente al cliente

### 3.3.4 Fattorino

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  1 ;
  1
)

CoreografiaOrdine :== (
  1 ;
  1 ;
  1 ;
  // Disponibile ?
  (
    // No
    1
  ) +
  (
    // Sí
    (
      1 ;
      (
        // In 15 secondi
        1 +
        1
      )
    )* ;
    // Lista vuota ?
    (
      // Sí
      1 |
      1
    ) +
    (
      // No
      1 ;
      1 ;
      1 ;
      1 ;
      1 ;
      // Modellazione timeout
      (
        1 ;
        1 ;
        1 ;
        1 ;
        1 ;
        // Token valido ?
```

```
        (
          // No
          1 |
          1 |
          1
        ) +
        (
          // Sí
          1 |
          1 |
          1 ;
                                              // Manca meno di un'ora ?
          (
            // No
            // Annullare ?
            (
              // Sí
              1 ;
              (
                1 ;
                1
              ) |
              (
                1 ;
                1
              ) |
              (
                1 ;
                1
              )
            ) +
            (
              // No
              1
            )
          ) +
          (
            // Sí
            (
              1 ;
              1
            ) |
            (
              1 ;
              1
            ) |
            (
              ConsegnaMerceF@Ristorante ;
              ConcegnaMerceC@Cliente ;
              ConfermaRicevutaSpedizione@ACMEat
            )
          )
        ) +
```

```
105         (
106           // Scadenza timer
107           1 |
108           (
109             1 ;
110             1
111           ) |
112           (
113             Annullamento@ACMEat ;
114             RicevutoAnnullamento@ACMEat ;
115           )
116         )
117       )
118     )
119   )
120 )
121
122 ModificaInformazioniLocali :== (
123   1 ;
124   // Prima delle 10 ?
125   (
126     // No
127     1
128   ) + (
129     // Sí
130     1
131   )
132 )
```

Listing 5: Proiezione della coreografia relativamente al fattorino

### 3.3.5 Ristorante

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  1 ;
  1
)

CoreografiaOrdine :== (
  1 ;
  Richiesta disponibilitá ristorante@ACMEat ;
  DisponibilitáR@ACMEat ;
  // Disponibile ?
  (
    // No
    1
  ) +
  (
    // Sí
    (
      1 ;
      (
        // In 15 secondi
        1 +
        1
      )
    )* ;
    // Lista vuota ?
    (
      // Sí
      1 |
      AnnullamentoR@ACMEat
    ) +
    (
      // No
      1 ;
      1 ;
      1 ;
      1 ;
      1 ;
      // Modellazione timeout
      (
        1 ;
        1 ;
        1 ;
        1 ;
        1 ;
        // Token valido ?
```

```
52          (
53            // No
54            1 |
55            AnnullamentoR@ACMEat |
56            1
57          ) +
58          (
59            // Sí
60            AttivazioneOrdineR@ACMEat |
61            1 |
62            1 ;
63            // Manca meno di un'ora ?
64            (
65              // No
66              // Annullare ?
67              (
68                // Sí
69                1 ;
70                (
71                  1 ;
72                  1
73                ) |
74                (
75                  AnnullamentoR @ACMEat ;
76                  RicevutoAnnullamento@ACMEat
77                ) |
78                (
79                  1 ;
80                  1
81                )
82              ) +
83              (
84                // No
85                1
86              )
87            ) +
88            (
89              // Sí
90              (
91                1 ;
92                RicezionePagamentoR@Banca ;
93              ) |
94              (
95                1 ;
96                1
97              ) |
98              (
99                1 ;
100               1;
101               1
102             )
103           )
104         ) +
```

```
105         (
106           // Scadenza timer
107           1 |
108           (
109             Annullamento@ACMEat ;
110             RicevutoAnnullamento@ACMEat
111           ) |
112           (
113             1 ;
114             1
115           )
116         )
117       )
118     )
119   )
120 )
121
122
123 ModificaInformazioniLocali :== (
124   RichiestaAggiornamento@ACMEat ;
125   // Prima delle 10 ?
126   (
127     // No
128     RichiestaRifiutata@ACMEat
129   ) +
130   (
131     // Sí
132     RichiestaAccettata@ACMEat
133   )
134 )
```

Listing 6: Proiezione della coreografia relativamente al ristorante

### 3.3.6 Servizio di spedizione

```
Coreografia :== (
  CoreografiaRichiestaMenu |
  CoreografiaOrdine |
  ModificaInformazioniLocali
)

CoreografiaRichiestaMenu :== (
  1 ;
  1
)

CoreografiaOrdine :==   (
  1 ;
  1 ;
  1 ;
  // Disponibile ?
  (
    // No
    1
  ) +
  (
    // Sí
    (
      Richiesta disponibilitá SDS@ACMEat ;
      (
        // In 15 secondi
        PreventivoDisponibilitá@ACMEat +
        1
      )
    )* ;
    // Lista vuota ?
    (
      // Sí
      1 |
      1
    ) +
    (
      // No
      Contatta SDS costo minore@ACMEat ;
      ACK@ACMEat ;
      1 ;
      1 ;
      1 ;
      // Modellazione timeout
      (
        1 ;
        1 ;
        1 ;
        1 ;
        1 ;
        // Token valido ?
```

```
52          (
53            // No
54            1 |
55            1 |
56            AnnullamentoS@ACMEat |
57          ) +
58          (
59            // Sí
60            1 |
61            AttivazioneOrdineS@ACMEat |
62            1 ;
63            // Manca meno di un'ora ?
64            (
65              // No
66              // Annullare ?
67              (
68                // Sí
69                1 ;
70                (
71                  1 ;
72                  1
73                ) |
74                (
75                  1 ;
76                  1
77                ) |
78                (
79                  AnnullamentoS@ACMEat ;
80                  RicevutoAnnullamento@ACMEat
81                )
82              ) +
83              (
84                // No
85                1
86              )
87            ) +
88            (
89              // Sí
90              (
91                1 ;
92                1
93              ) |
94              (
95                PagamentoS : ACMEat -> Banca ;
96                RicezionePagamentoS : Banca -> SDS ;
97            ) |
98            (
99              1 ;
100             1 ;
101             1
102           )
103         )
104       ) +
```

```
105       (
106         // Scadenza timer
107         1 |
108         (
109           1 ;
110           1
111         ) |
112         (
113           1 ;
114           1
115         )
116       )
117     )
118   )
119 )
120
121 ModificaInformazioniLocali :== (
122   1 ;
123   // Prima delle 10 ?
124   (
125     // No
126     1
127   ) +
128   (
129     // Sí
130     1
131   )
132 )
```

Listing 7: Proiezione della coreografia relativamente al servizio di spedizione

# 4 Documentazione

Durante la seconda fase di lavoro è stato realizzanto un diagramma di collaborazione BPMN (Fig. 1) con l'obiettivo di modellare l'intera realtà descritta a scopo documentativo, compresi i dettagli di ogni partecipante. Tale diagramma e la relativa export `.png` sono forniti in allegato a questa relazione.

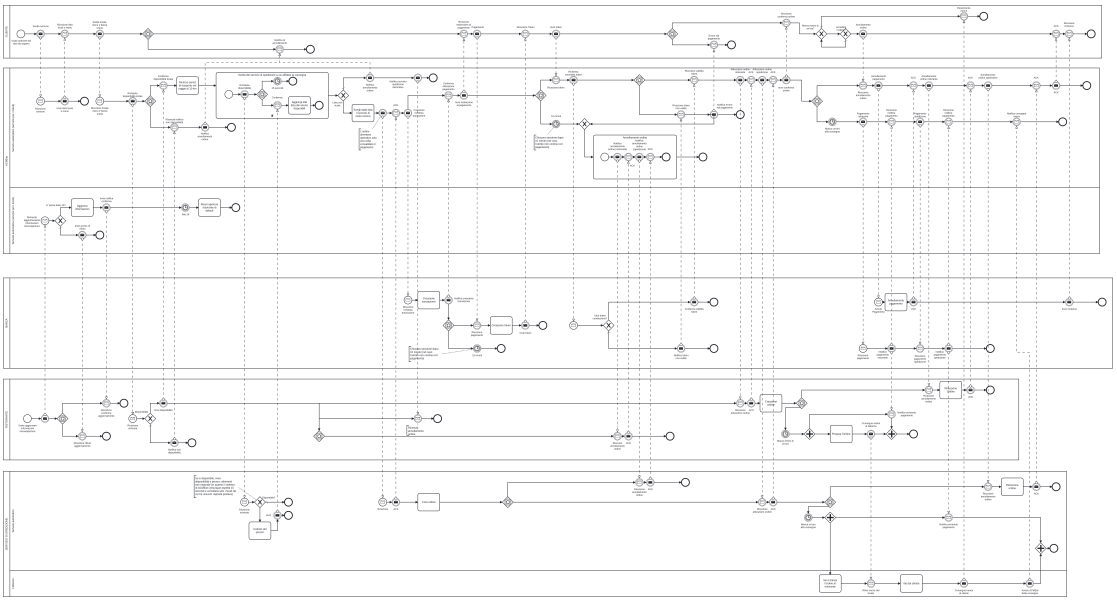

Figura 1: Diagramma di collaborazione BPMN

Di seguito, riportiamo alcuni estratti rilevanti di questo diagramma.

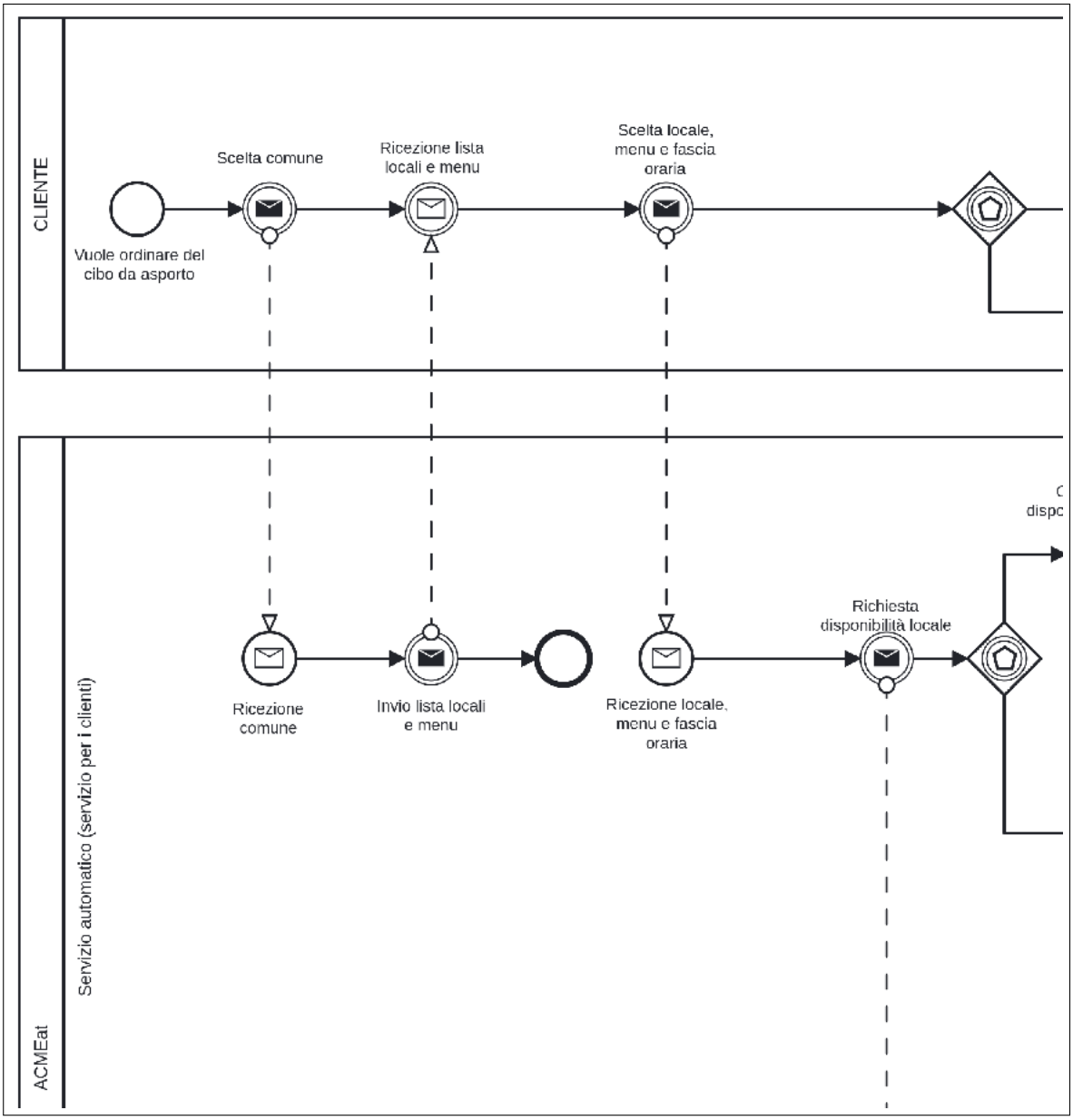


Figura 2: Scambio di messaggi iniziale fra un cliente e ACMEat

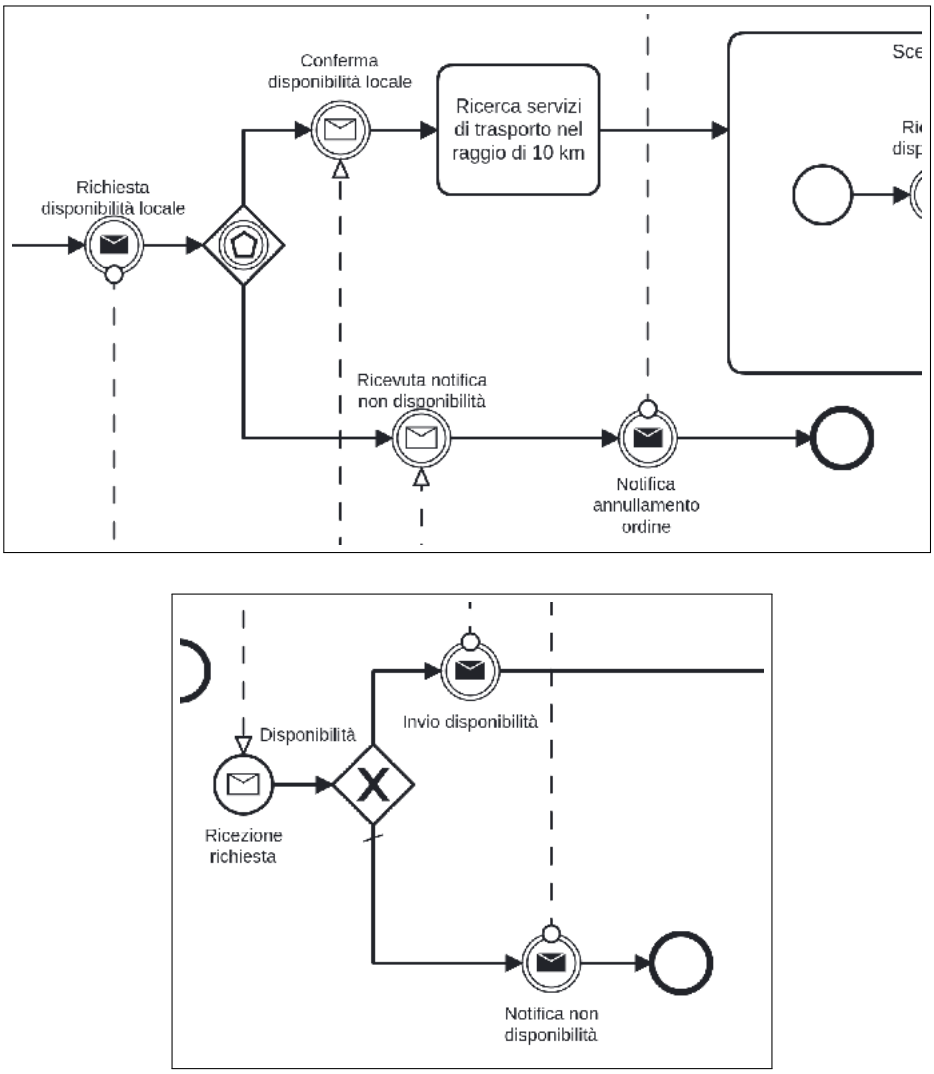


Figura 3: Scambio di messaggi fra ACMEat e un ristorante per verificarne la disponibilità

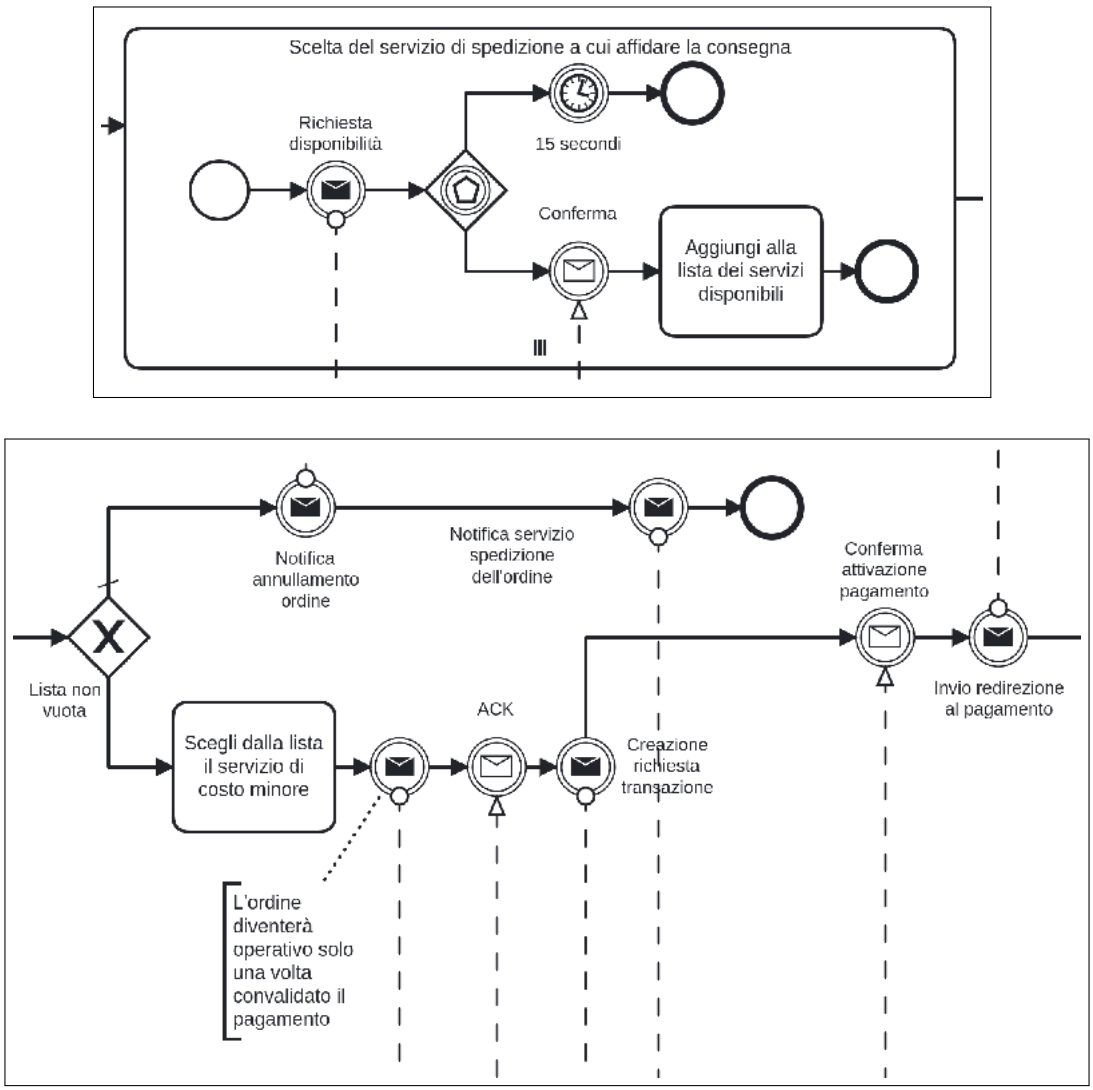


Figura 4: Scelta del servizio di spedizione all'interno di ACMEat

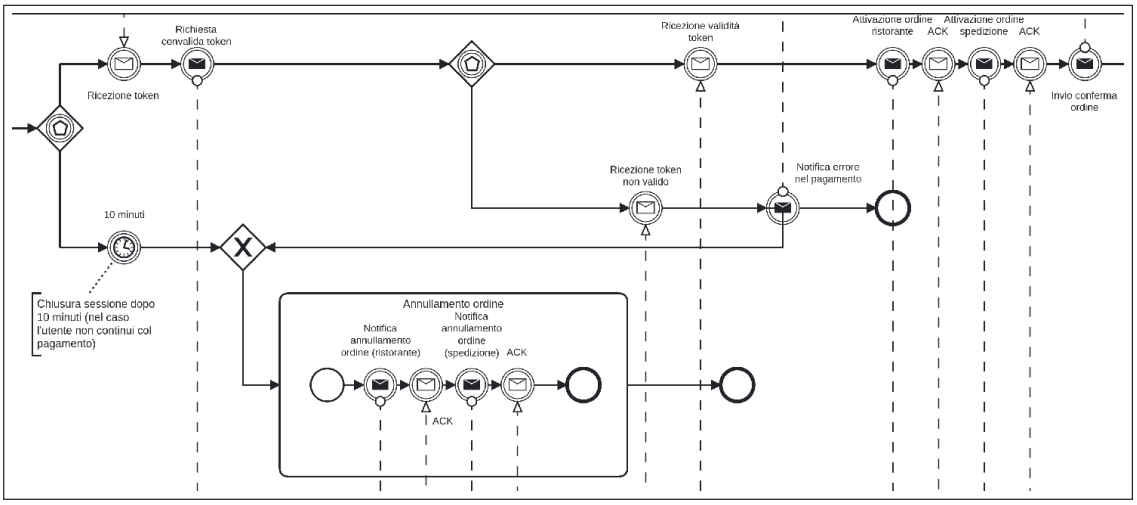


Figura 5: Gestione pagamento dopo redirezione cliente da parte di ACMEat

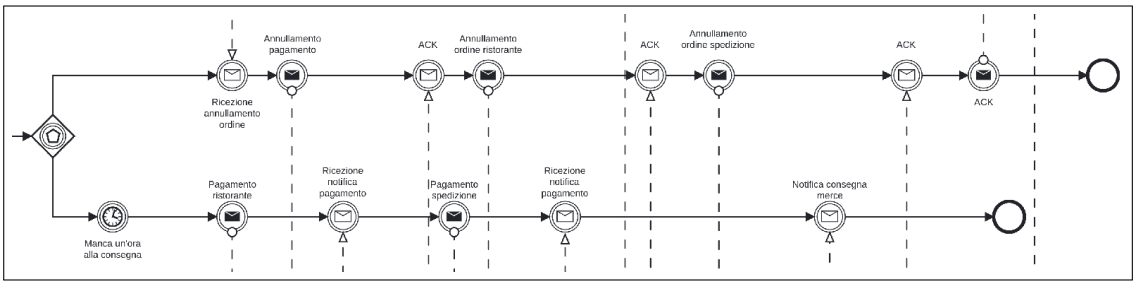


Figura 6: Gestione (annullamento) ordine da parte di ACMEat

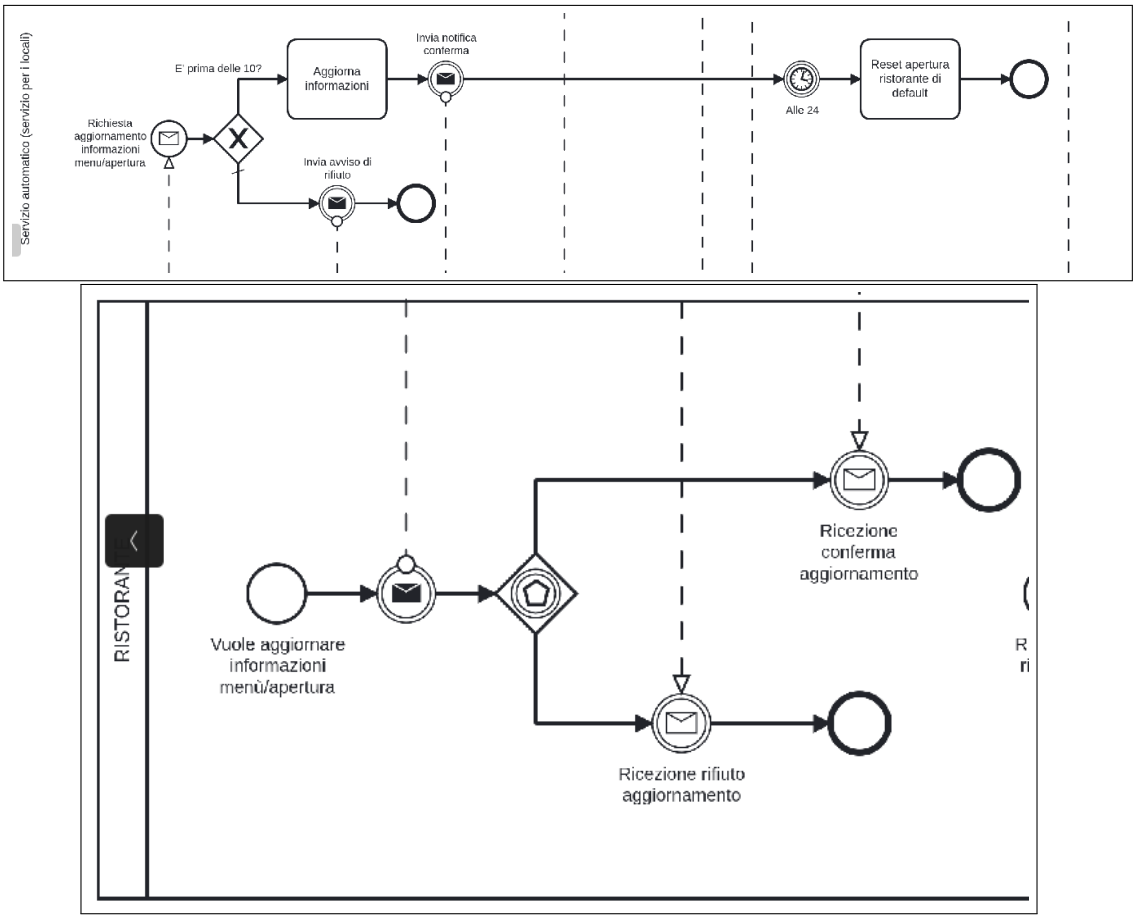


Figura 7: Gestione richiesta aggiornamento informazioni ristorante da parte di ACMEat (sopra) e del ristorante (sotto)

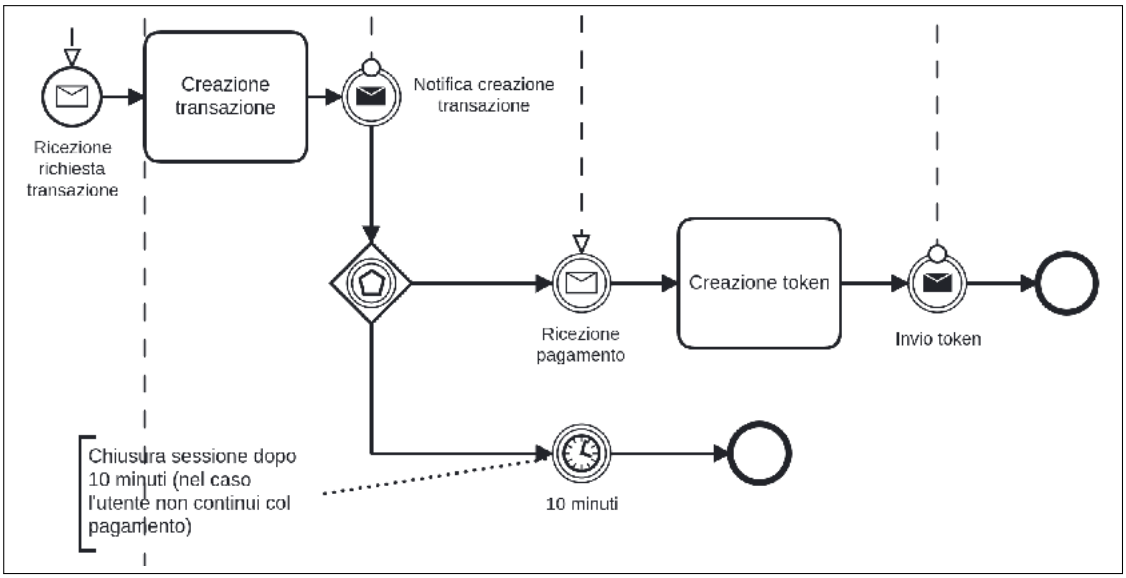


Figura 8: Gestione pagamento da parte della banca

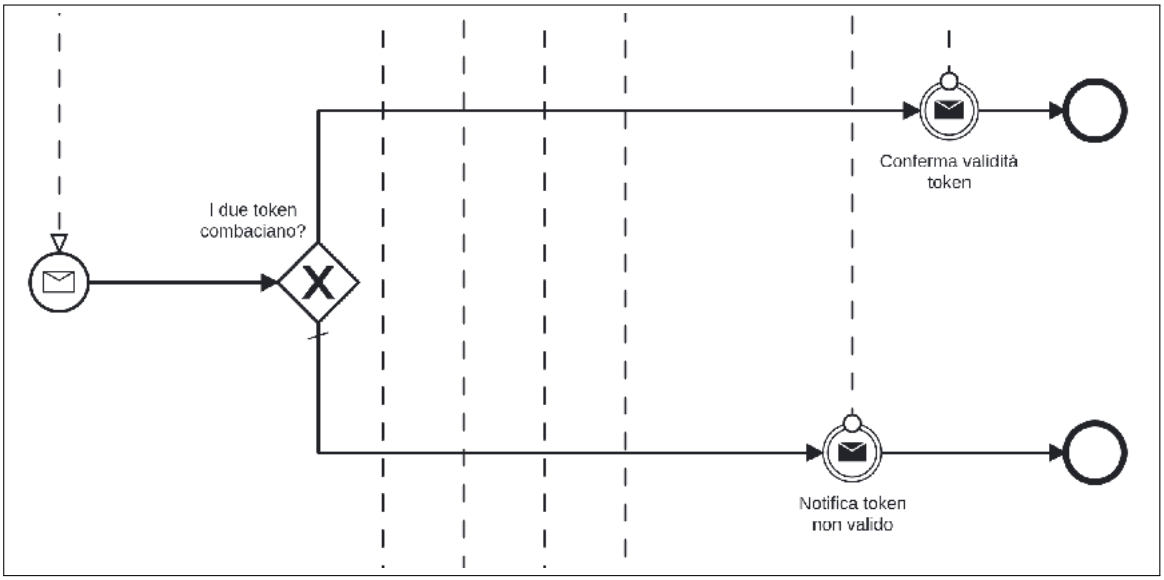


Figura 9: Verifica token da parte di ACMEat

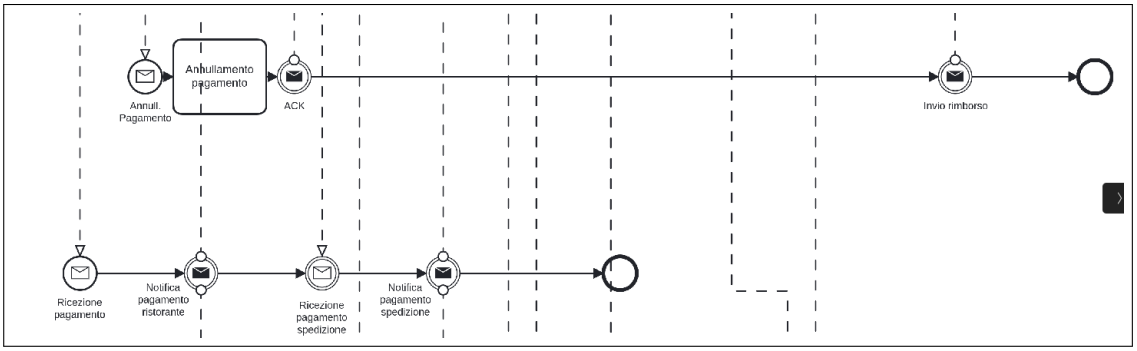


Figura 10: Gestione rimborsi e pagamenti da parte della banca

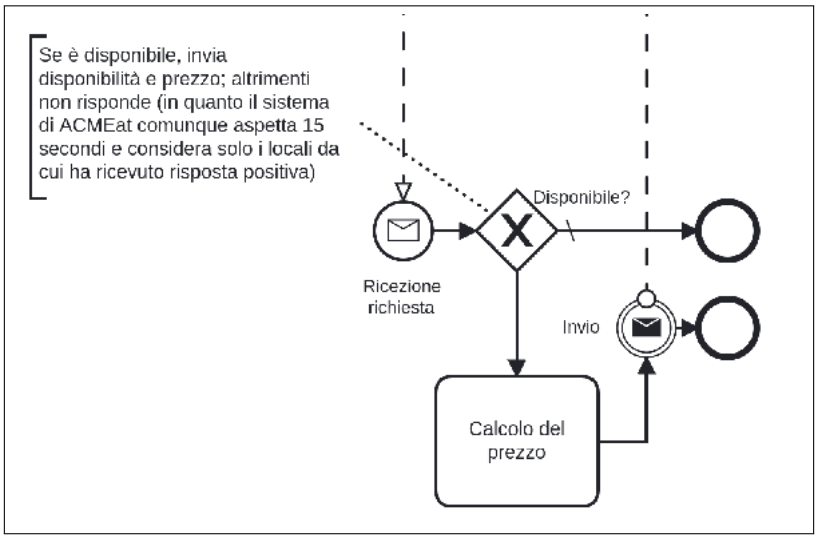


Figura 11: Notifica disponibilità servizio di spedizione

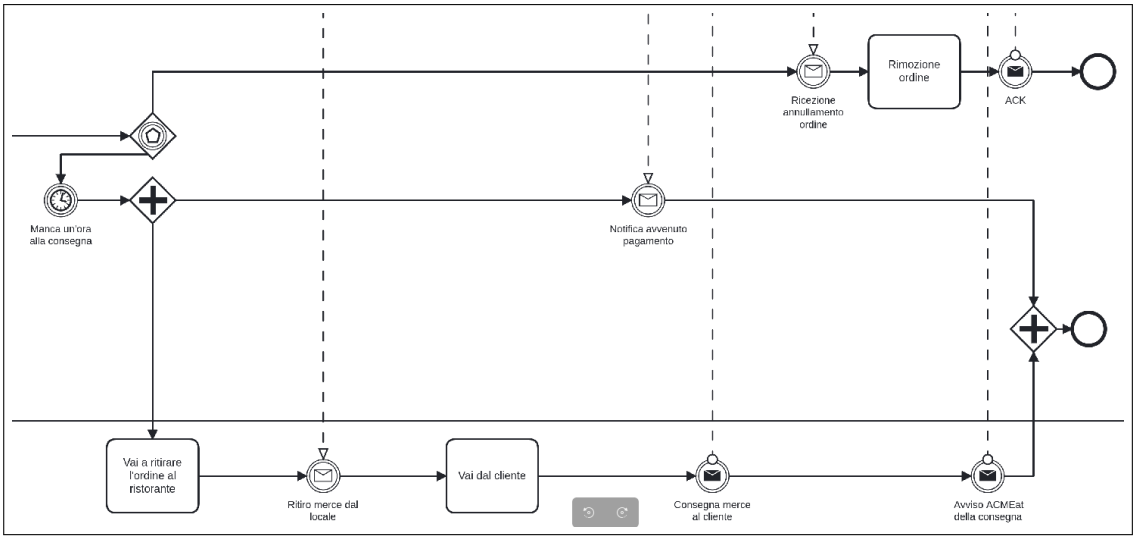


Figura 12: Gestione consegna da parte di servizio di spedizione e fattorino

# 5 Progettazione

La terza fase di lavoro ha visto la progettazione di una SOA per la realizzazione del sistema, documentata utilizzando UML.

# 6 Sviluppo

La quarta fase di lavoro ha visto la realizzazione del sistema. Come da specifica, sono stati realizzati i seguenti servizi:

- Il servizio centrale ACMEat, il quale rende accessibili capabilities realizzate attraverso il BPMS Camunda;
- Il servizio bancario, realizzato in Jolie;
- Il servizio delle società di consegna ACMEDeliver;
- Il servizio delle società di ristorazione ACMERestaurant;
- Il servizio di geo-localizzazione ACMEGeolocate.

I backend realizzati, di conseguenza, sono i seguenti:

- **acmeat**, backend REST per interagire con il database di ACMEat e con il BPMS;
- **acmedeliver**, backend REST che rappresenta una società di consegna e ne consente la gestione e utilizzo;
- **acmerestaurant**, backend REST minimale per la gestione utenti di un ristorante e che consente l'autenticazione presso ACMEat per la gestione degli ordini;
- **acmegeolocate**, backend REST per la geo-localizzazione (implementata appoggiandosi ad openstreetmap);
- **bank**, backend Jolie per la gestione dei fondi dei clienti, fattorini, ristoranti e di acmeat;
- **bank_intermediary**, backend REST per superare le restrizioni CORS di Jolie, usato solo ed esclusivamente dal frontend della banca.

Tutti i backend (fatta eccezione per bank) sono stati realizzati con le seguenti tecnologie:

- Python 3.8+;
  - fastapi - Framework per la creazione di REST API;
  - bcrypt - Modulo crittografico;
  - beautifulsoup4 - Parsing risposte SOAP;
  - psycopg2-binary - Driver per Postgres;
  - pycamunda - Framework per la comunicazione con Camunda;
  - requests - Framework per la gestione di richieste HTTP;
  - SQLAlchemy - ORM
  - Eventuali dipendenze dei moduli sopra indicati.

- Poetry (Package Manager Python);
- Postgres.

Il diaologo fra Jolie e BPMS avviene via SOAP.

## 6.1 Backend Python

In questa sezione verranno approfonditi i backend sviluppati in Python, ovvero tutti meno **bank**. La struttura utilizzata all'interno dei vari backend è identica (in alcuni, determinate cartelle potrebbero mancare, in quanto un certo componente potrebbe non venire utilizzato), ed è così costituita:

- /
  - **database**: contiene la definizione delle tabelle che vengono create all'interno del DB, la definizione di enum e l'oggetto Session tramite il quale si interroga la base di dati;
  - **deps**: contiene le dipendenze di Fastapi, un meccanismo che consente di ottenere informazioni prima di entrare nel corpo della funzione che gestisce un certo endpoint - ad esempio, sapere in anticipo se un utente è autenticato o meno.
  - **errors**: contiene la definizione degli errori personalizzati;
  - **routers**: contiene le funzioni che gestiscono le richieste ai vari endpoint del backend, nella struttura gerarchica di cartelle /api/[soggetto]/v1/[soggetto].py;
  - **schemas**: contiene gli schemi di risposta e richiesta accettati dall'applicazione; services: contiene il server, il worker e le funzioni necessarie per eseguire le task della coreografia di ACMEat;
  - __main__.py: runner del server;
  - authentication.py: modulo per l'autenticazione tramite JWT;
  - configuration.py: modulo per la configurazione dell'applicazione;
  - crud.py: modulo che contiene funzioni di utility per la creazione, l'aggiornamento e la ricerca di dati all' interno del database;
  - dependencies.py: modulo che contiene funzioni da cui dipendono altri moduli dell'applicazione;
  - handlers.py: gestori di eccezioni specifici per l'applicazione;
  - responses.py: risposte personalizzate non-json.

### 6.1.1 Parametri di configurazione

Al fine di poter operare, è necessario che ai backend vengano fornite le corrette variabili d'ambiente.

- acmeat
  - JWT_KEY=pippo: la password con cui i JWT vengono cifrati;
  - DB_URI=postgresql://postgres:password@localhost/acmeat: l'uri del database di acmeat;
  - BIND_IP=127.0.0.1: l'indirizzo ip su cui eseguire il binding del socket;
  - BIND_PORT=8004: la porta su cui eseguire il binding del socket;
  - BANK_URI=http://127.0.0.1:2000: l'indirizzo a cui contattare la banca;
  - BANK_USERNAME=acmeat: l'username per l'accesso alla banca;
  - BANK_PASSWORD=password: la password per l'accesso alla banca.
- acmedeliver
  - JWT_KEY=pippo: la password con cui i JWT vengono cifrati;
  - DB_URI=postgresql://postgres:password@localhost/acmedeliver: l'uri del database di acmedeliver;
  - BIND_IP=127.0.0.1: l'indirizzo ip su cui eseguire il binding del socket;
  - BIND_PORT=8003: la porta su cui eseguire il binding del socket;
  - PRICE_PER_KM=2: il costo per chilometro della società di spedizioni;
  - GEOLOCATE_URL=http://127.0.0.1:8001: l'indirizzo a cui contattare il servizio di geo-localizzazione.
- acmegeolocatee
  - BIND_IP=127.0.0.1: l'indirizzo ip su cui eseguire il binding del socket;
  - BIND_PORT=8001: la porta su cui eseguire il binding del socket.
- acmerestaurant
  - JWT_KEY=pippo: la password con cui i JWT vengono cifrati;
  - DB_URI=postgresql://postgres:password@localhost/acmerestaurant: l'uri del database di acmerestaurant;
  - BIND_IP=127.0.0.1: l'indirizzo ip su cui eseguire il binding del socket;
  - BIND_PORT=8007: la porta su cui eseguire il binding del socket;
  - ACME_EMAIL=owner1@gmail.com: l'email del proprietario dell'attività tramite la quale accede ad ACMEat;
  - ACME_PASSWORD=password: la password del proprietario dell'attività tramite la quale accede ad ACMEat;
  - ACME_RESTAURANT_ID=59294bd6-f61b-48a2-9f53-f76d378b95d9: l'id del ristorante su ACMEat;
  - ACME_URL=http://127.0.0.1:8004: l'indirizzo a cui contattare ACMEat.

Immagine qui

- bank_intermediary
  - BIND_IP=127.0.0.1: l'indirizzo ip su cui eseguire il binding del socket;
  - BIND_PORT=8006: la porta su cui eseguire il binding del socket;
  - BANK_URI=http://127.0.0.1:2000: l'indirizzo a cui contattare la banca.

### 6.1.2 Dettagli sui backend

In questa sezione, verranno specificate le caratteristiche dei vari backend. Per informazioni sulle route disponibili per ogni backend, visitare la pagina /docs dei backend una volta avviati. Per la documentazione del codice, visionare il sorgente e relativi commenti.

#### acmeat

Acmeat è il backend principale, e consente di:

- Permettere a nuovi utenti di iscriversi al servizio, in veste di cliente o ristoratore;
- Registrare un ristorante e gestirne le caratteristiche, menu inclusi;
- Permettere agli utenti di eseguire ordinazioni presso un locale, e tramite la coreografia Camunda gestirne il ciclo di vita;
- Permettere agli amministratori di registrare città in cui il serizio è attivo, e gestire la lista dei fattorini affiliati ad ACMEat.

Tutte le informazioni necessarie vengono immagazzinate all'interno di una base di dati così strutturata: Il ciclo di vita di un ordine passa attraverso le seguenti fasi:

- **Created**: l'ordine è stato appena creato;
- **w_restaurant_ok**: l'ordine è in attesa di conferma da parte del ristorante;
- **w_deliverer_ok**: l'ordine è in attesa di conferma da parte di un fattorino;
- **confirmed_by_thirds**: l'ordine è stato confermato dalle terze parti (ristorante e fattorino);
- **cancelled**: l'ordine è stato cancellato, dall'utente oppure dal processo camunda per problemi riscontrati;
- **w_payment**: l'ordine è in attesa di essere pagato. Se non pagato entro 5 minuti, o pagato in modo errato, verrà cancellato;
- **w_cancellation**: l'ordine può venire cancellato dall'utente fino ad un'ora prima dall'orario indicato;

- **w_kitchen**: l'ordine sta venendo preparato in cucina;
- **w_transport**: l'ordine è in attesa del fattorino;
- **delivering**: l'ordine è in consegna;
- **delivered**: l'ordine è stato consegnato.

Gli utenti di acmeat possono appartenere ad una di tre categorie:

- **Cliente**: Possono solo creare ordini e gestire i propri, può creare un ristorante (a quel punto il tipo di utente verrà modificato);
- **Ristoratore**: Può gestire il proprio locale e crearne di nuovi, oltre ai privilegi del cliente;
- **Amministratore**: Può gestire la lista delle città e dei fattorini.

Le società di consegna accedono ai sistemi di acmeat tramite un token, e questo consente loro di aggiornare lo stato dell'ordine (da "in consegna" a "consegnato").

ACMEmanager

ACMEmanager è il server a cui la coreografia camunda delega l'esecuzione dei job. Le task vengono svolte da un worker, il quale è in ascolto per i seguenti topic:

- **restaurant_confirmation**: ricezione di una conferma (o meno) da parte del ristorante;
- **deliverer_preview**: ottenimento dei preventivi dei fattorini nel raggio di 10km dal locale;
- **deliverer_confirmation**: conferma con il fattorino con il prezzo minore dell'ordine;
- **payment_request**: attesa della ricezione di un pagamento da parte dell'utente;
- **payment_received**: verifica del pagamento ricevuto con la banca;
- **confirm_order**: conferma dell'ordine;
- **restaurant_abort**: notifica annullamento ordine dal lato del ristoratore;
- **deliverer_abort**: notifica annullamento ordine dal lato del fattorino;
- **user_refund**: se pagato, l'ordine viene rimborsato all'utente;
- **order_delete**: l'ordine viene indicato come cancellato;
- **pay_restaurant**: acmeat paga il ristorante;

Immagine qui

- **pay_deliverer**: acmeat paga il fattorino.

A questi topic corrispondono funzioni omonime, le quali utilizzano le seguenti variabili di processo:

- **order_id**: l'id dell'ordine interno ad acmeat;
- **success**: flag che indica se l'ultima operazione ha avuto successo o meno;
- **paid**: flag che indica se l'ordine è stato pagato;
- **payment_success**: flag che indica se l'ordine è stato pagato correttamente;
- **TTW**: TimeToWait, durata in secondi alla fine del periodo di cancellazione;
- **found_deliverer**: flag che indica se è stato trovato un fattorino;
- **restaurant_accepted**: flag che indica se il ristorante ha accettato la richiesta.

Il worker è basato su Pycamunda, è multithreaded ed è in grado di interagire con il database tramite SQLAlchemy. Le richieste fatte alla banca vengono trasmesse tramite soap.

#### acmedeliver

Acmedeliver è il backend che rappresenta un'azienda di consegne. Permette di:

- Gestire il personale;
- Gestire e ricevere richieste di consegna;
- Gestire la lista dei propri clienti, a cui viene dato accesso;
- Aggiornare il cliente sullo stato della consegna.

Tutte le informazioni necessarie vengono immagazzinate all'interno di una base di dati così strutturata: Gli utenti di acmedeliver possono essere fattorini oppure amministratori, dove gli amministratori sono in grado di aggiungere fattorini all'azienda.

#### acmegeolocate

Acmegeolocate è il backend che fornisce il servizio di geo-localizzazione. Dato lo stato, la città, la via e il numero civico utilizza openstreetmap per ricavarne le coordinate, e in base alla richiesta fornire la distanza tra i due punti in km.

Immagine qui

acmerestaurant
Acmerestaurant è il backend che fornisce autenticazione alle richieste del ristorante, che comunica poi con acmeat. Gli utenti contenuti all'interno del database non sono utenti di acmeat, ma del ristorante (ad esempio, ogni cameriere può avere un account all'interno del ristorante), e le richieste vengono fatte a nome del titolare del ristorante (che ha un account su acmeat). Il backend consente di eseguire un numero limitato di operazioni, ovvero la lettura degli ordini e accettare/rifiutare/consegnare (ad un fattorino) un ordine. La gestione dei menu, così come degli orari di apertura e delle altre caratteristiche del ristorante è da eseguire su acmeat. La struttura del database è la seguente:

bank_intermediary
Intermediario per superare il blocco CORS di Jolie, si limita a inoltrare le richieste che gli arrivano in " simil-soap" (soap dentro un oggetto json) al backend della banca. Viene usato solo ed esclusivamente dal frontend di acmebank.

### 6.1.3 Istruzioni per l'avvio in ambiente di testing

1. Installare postgresql, python3 e poetry;
2. Creare un database per l'applicazione che si vuole avviare;
3. Clonare il repository github;
4. Entrare nella cartella "Applications" del repository, eseguire il comando `poetry install`;
5. Eseguire il comando `poetry shell`;
6. Impostare le variabili d'ambiente richieste dal servizio desiderato;
7. Eseguire il comando `python -m ${service_name}`.

### 6.1.4 Istruzioni per il deployment in produzione

In questa sezione, vengono indicati i passaggi necessari per il deployment dell'applicazione in produzione. Si suppone l'utilizzo del sistema operativo Ubuntu, e vengono omessi i passaggi per la realizzazione del reverse poxy e dei certificati per l'https.

Setup iniziale

1. Da root, inserire il comando useradd `${nome_servizio}`;
2. Da root, creare inserire il comando `adduser user` per creare un utente con cui proseguire la configurazione;

3. Da root, inserire il comando `usermod -aG sudo user` per inserire l'utente user nel gruppo sudoers;
4. Eseguire l'accesso con l'utente user.

### Installazione dipendenze software

1. Inserire il comando `sudo apt-get update`;
2. Inserire il comando `sudo apt-get install postgresql python3`;
3. Eseguire il comando `curl -sSL https://install.python-poetry.org` python3— - per installare poetry.

### Setup del singolo backend

1. Spostarsi nella cartella "/srv" e scaricare il repository git;
2. Spostarsi nella sottocartella del repository "Applications";
3. Eseguire l'accesso come l'utente `${nome_servizio}`
4. Installare le dipendenze tramite poetry install. Sarà necessario capire quale sia il percorso dell'ambiente creato, il quale dovrebbe essere sotto la cartella `/home/${nome_servizio}/.ca` e a cui ci si riferirà come `${poetry_path}`;
5. Eseguire l'accesso con l'utente postgres, eseguire il comando `psql`;
6. Creare il database `${nome_database}`;
7. Creare l'utente `{nome_servizio}` con `CREATEUSER'{nome_servizio}' WITH ENCRYPTED PASSWORD`
8. Fornire all'utente appena creato i privilegi sul database con da `GRANT ALL PRIVILEGES ON DATABASE`
9. Inserire il comando exit 2 volte;
10. Trasferire il possesso della cartella del servizio interessato all'utente `${nome_servizio}` con il comando `sudo chown ${nome_servizio} ${cartella_servizio}`.

### Configurazione del server come servizio systemd

1. Creare il file `${nome_servizio}.service` nella cartella `/etc/systemd/system` tramite il comando `sudo touch /etc/systemd/system/${nome_servizio}.service`;
2. Creare la cartella `${nome_servizio}.service.d` nella cartella `/etc/systemd/system` tramite il comando `sudo mkdir /etc/systemd/system/${nome_servizio}.service.d`;

3. Inserire nel file `${nome_servizio}.service le seguenti righe`:

```
[Unit]
Name=${nome_servizio}
Description=${nome_servizio} fastapi server
Wants=network-online.target
After=network-online.target nss-lookup.target

[Service]
Type=exec
User=${nome_servizio}
Group=${nome_servizio}
# Replace with the directory where you cloned the repository
WorkingDirectory=/srv/ACMEat/Applications/${nome_servizio}/
# Replace with the directory where you cloned the repository and the
    poetry path
ExecStart=/home/${nome_servizio}/.cache/pypoetry/virtualenvs/${
   poetry_path}/bin/python3 __main__.py

[Install]
WantedBy=multi-user.target
```

4. Creare un file nella cartella appena creata chiamato `override.conf` e popolarlo in questo modo, tenendo conto delle variabili d'ambiente necessarie per quel particolare servizio:

```
[Service]
Environment=KEY=value
```

5. Ricaricare i file di configurazione dei servizi con `sudo systemctl daemon-reload`, per poi avviarlo con il comando `sudo systemd start ${nome_servizio}`.

### 6.1.5 Post-Installazione

Per poter testare l'applicazione senza dover riempire a mano il database, eseguire lo script `post_install.py` nella cartella "Applications".

# 7 Conclusioni