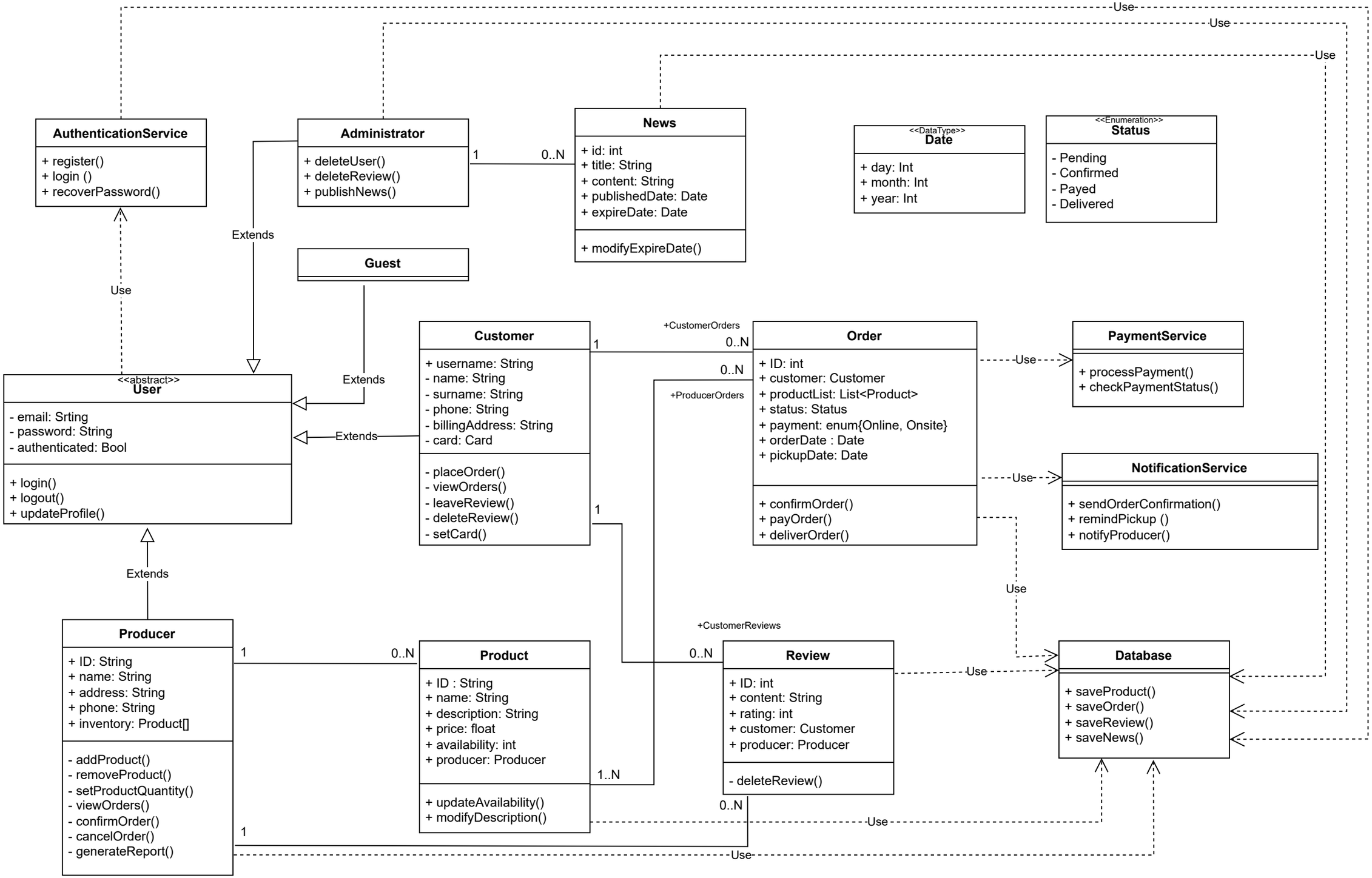
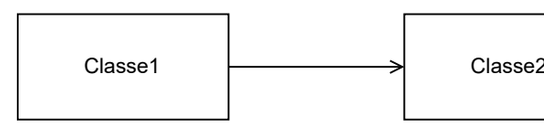
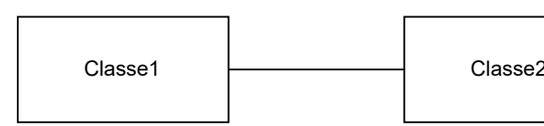
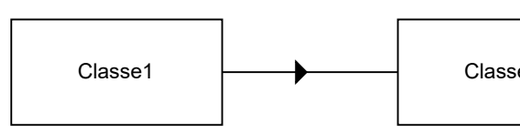

Use
Use
Use
AuthenticationService
+ register()
+ login ()
+ recoverPassword()
Administrator
+ deleteUser()
+ deleteReview()
+ publishNews()
1
0..N
News
+ id: int
+ title: String
+ content: String
+ publishedDate: Date
+ expireDate: Date
+ modifyExpireDate()
<<DataType>>
Date
+ day: Int
+ month: Int
+ year: Int
<<Enumeration>>
Status
- Pending
- Confirmed
- Payed
- Delivered
Extends
Guest
Use
<<abstract>>
User
- email: Srting
- password: String
- authenticated: Bool
+ login()
+ logout()
+ updateProfile()
Extends
Extends
Customer
+ username: String
- name: String
- surname: String
- phone: String
- billingAddress: String
- card: Card
- placeOrder()
- viewOrders()
- leaveReview()
- deleteReview()
- setCard()
+CustomerOrders
1
0..N
0..N
+ProducerOrders
1
Order
+ ID: int
+ customer: Customer
+ productList: List<Product>
+ status: Status
+ payment: enum{Online, Onsite}
+ orderDate : Date
+ pickupDate: Date
+ confirmOrder()
+ payOrder()
+ deliverOrder()
Use
PaymentService
+ processPayment()
+ checkPaymentStatus()
Use
NotificationService
+ sendOrderConfirmation()
+ remindPickup ()
+ notifyProducer()
Use
Extends
Producer
+ ID: String
+ name: String
+ address: String
+ phone: String
+ inventory: Product[]
- addProduct()
- removeProduct()
- setProductQuantity()
- viewOrders()
- confirmOrder()
- cancelOrder()
- generateReport()
1
0..N
Product
+ ID : String
+ name: String
+ description: String
+ price: float
+ availability: int
+ producer: Producer
+ updateAvailability()
+ modifyDescription()
+CustomerReviews
0..N
1..N
Review
+ ID: int
+ content: String
+ rating: int
+ customer: Customer
+ producer: Producer
- deleteReview()
0..N
Use
Database
+ saveProduct()
+ saveOrder()
+ saveReview()
+ saveNews()
Use
1
Use



Classe2 NON può vedere/accedere agli attributi/metodi d



Classe2 e classe1 possono entrambe vedere tutti gli attribu



Se vuoi dare la direzionalità di lettura devi usare ques

2

di classe1

2

uti e metodi

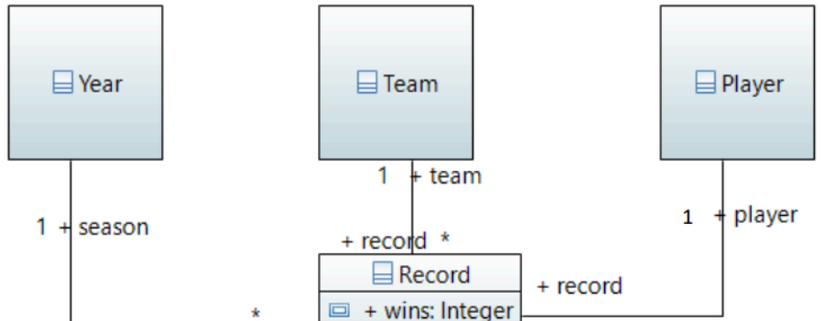
e2

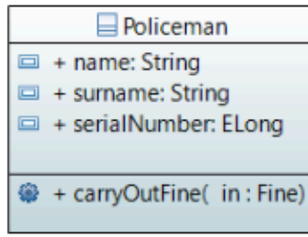
ta scrittura

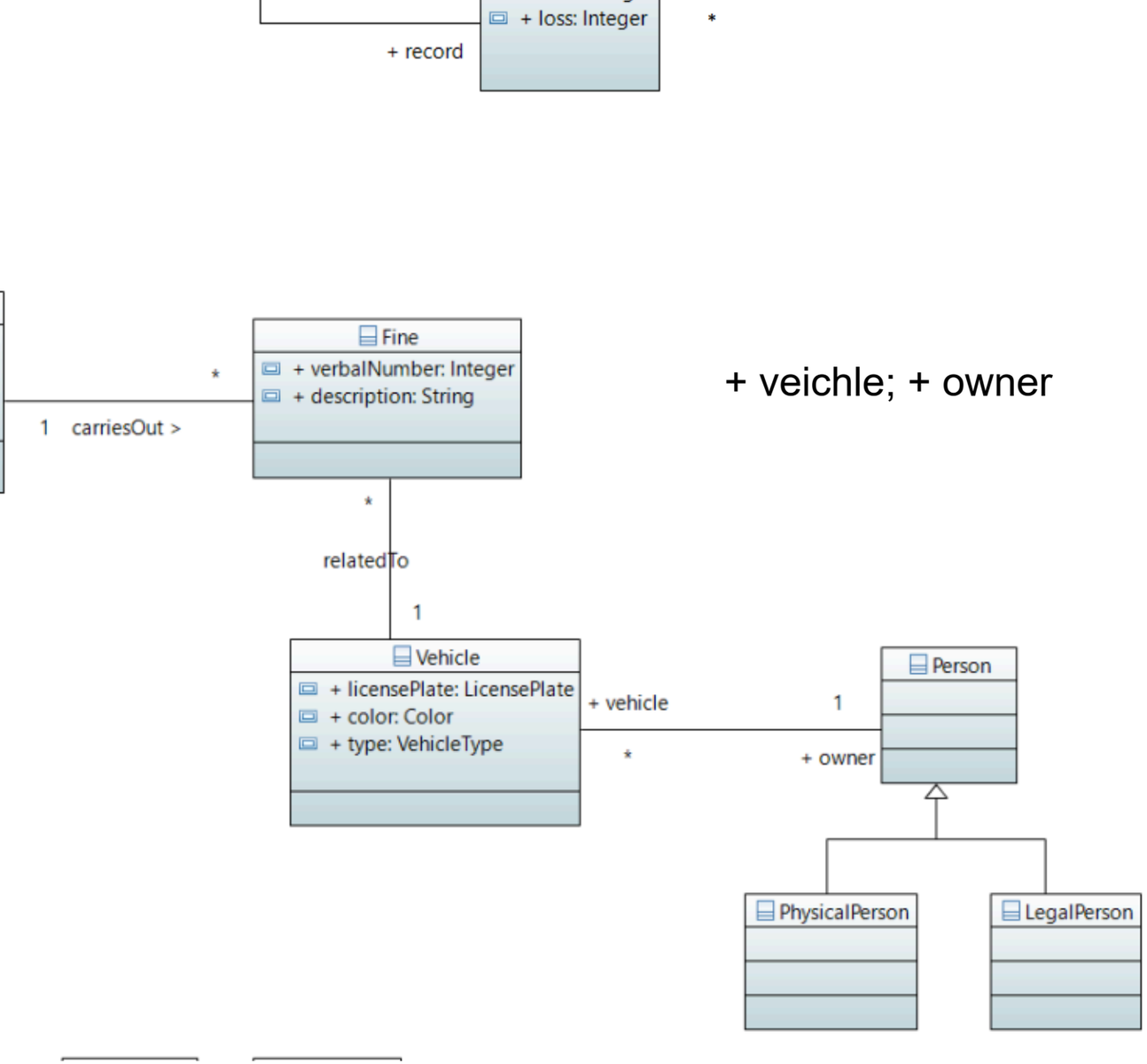
La freccia seve proprio per "nascondere" l'accesso alla classe padre, dunque
puoi usarla anche con il rombo pieno/vuoto (aggregazione, composizione) come in questo caso.

Altra osservazione:
preferisci utilizare questa scrittura. Serve per poi creare OCL

Vedi quei: + season, +Team, + player ecc che sono fuori

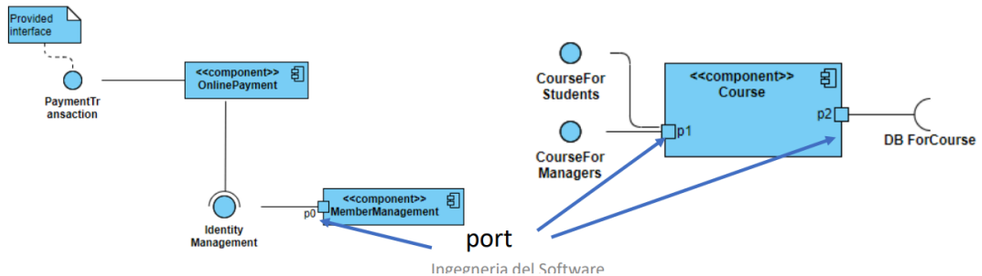



Policeman
+ name: String
+ surname: String
+ serialNumber: ELong
+ carryOutFine(in : Fine)



+ veichle; + owner

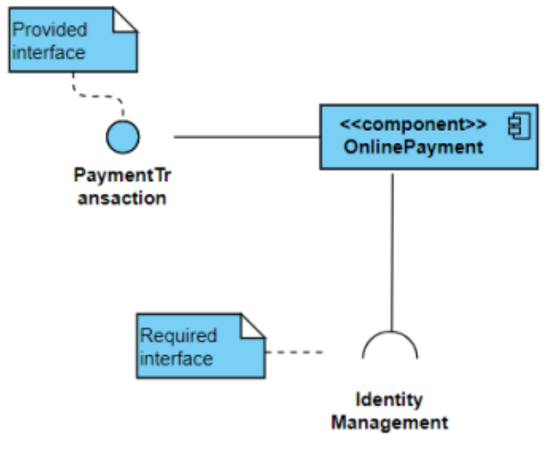
A port groups together a “semantically cohesive” set of interfaces
• It is a specific point of interaction between external component and internal component
• It can have a name (e.g. p0, p1, p2)

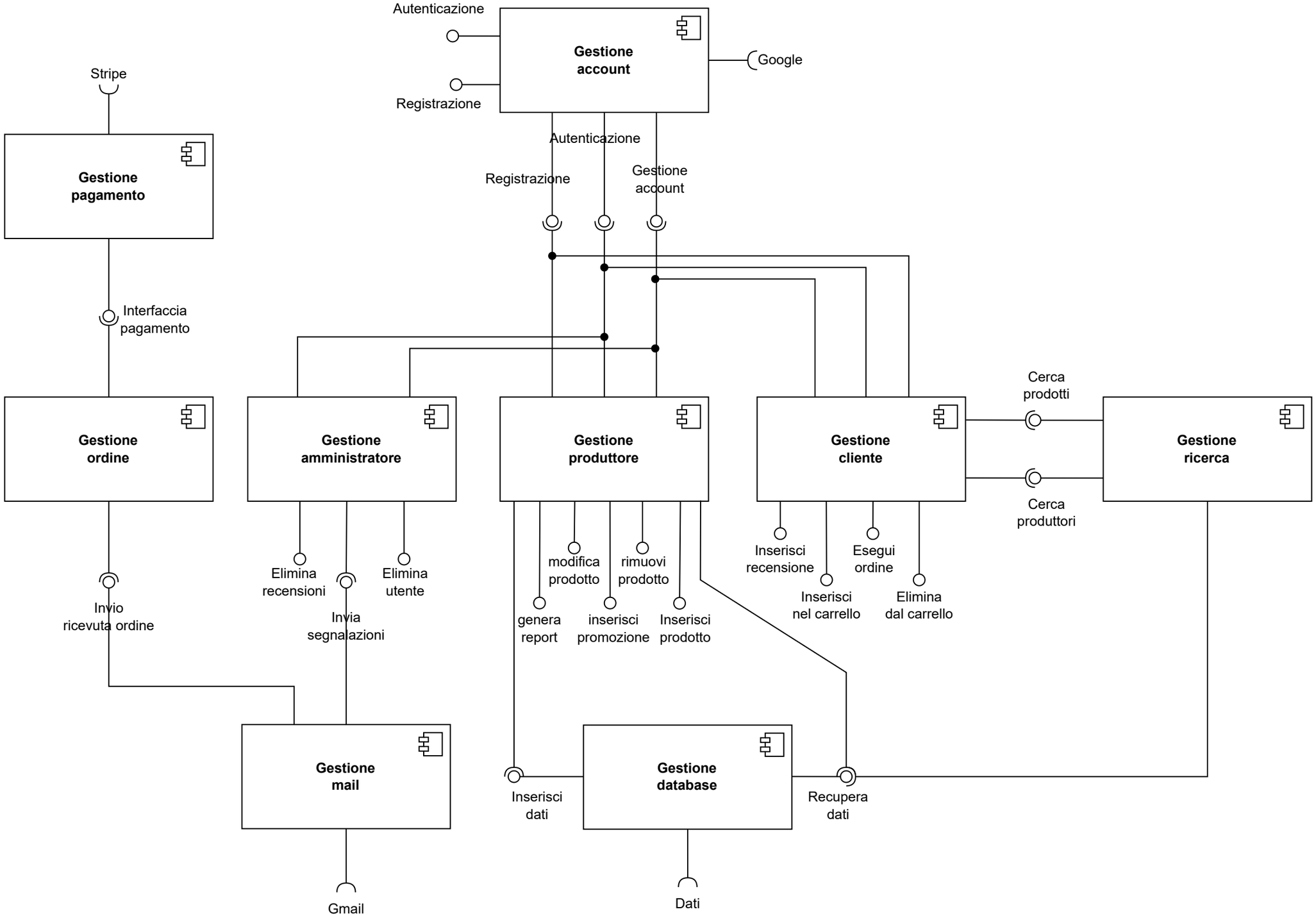


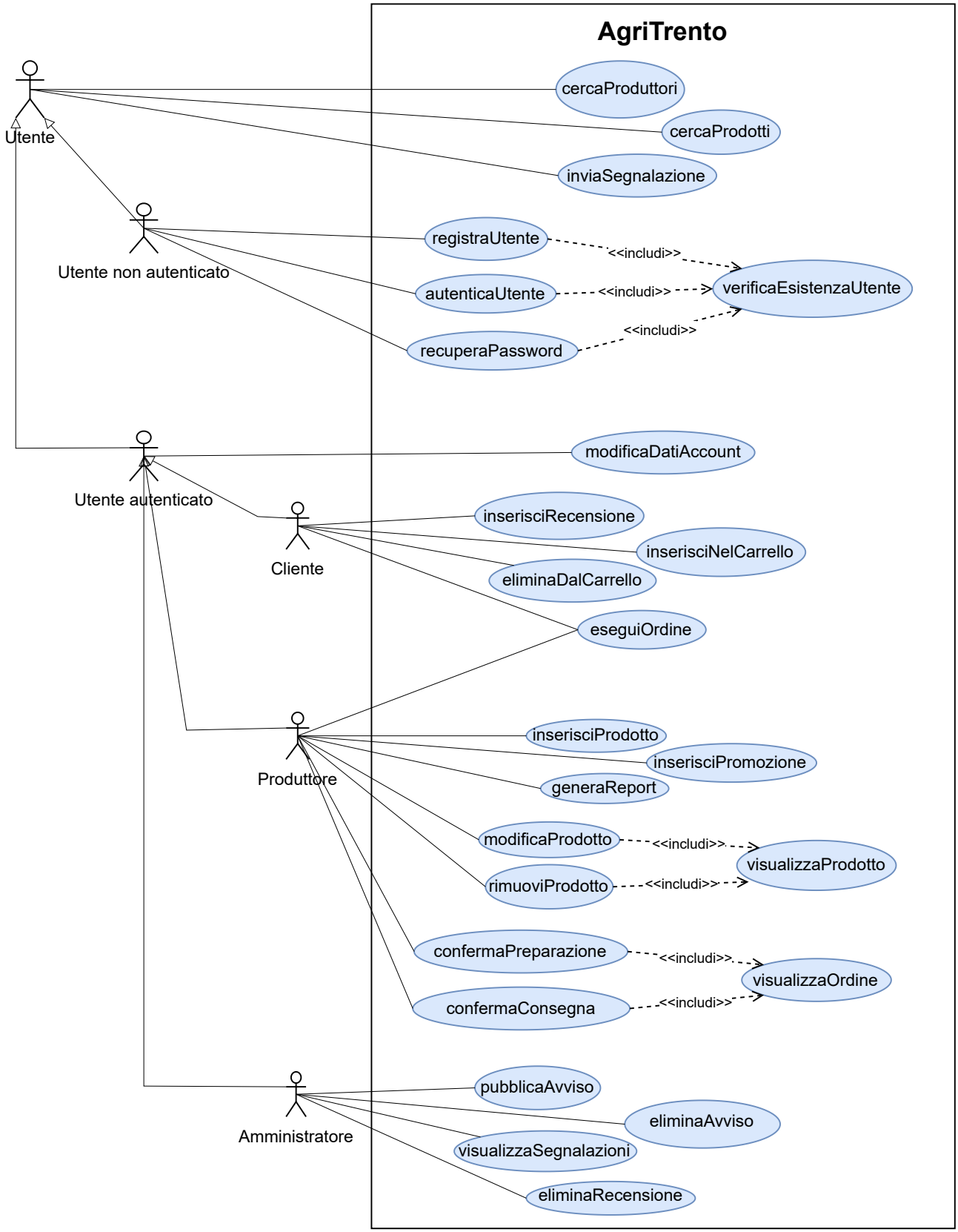
• «executable»: a component that runs on a processor.
• «library»: a set of resources referenced by an executable during runtime.
• «table»: a database component accessed by an executable.
• «file»: typically represents data or source code.
• «document»: a document such as a Web page.

A component can have
▪ Interfaces: a declaration of a set of operations and obligations

A **provided** interface (lollipop) of a component is an interface that the component realises
• A **required** interface (socket) of a component is an interface that the component needs to function
---
▪ Usage dependencies: dependencies among elements so that one element requires another element for its full implementation




Autenticazione
Gestione account
Google
Registrazione
Stripe
Gestione pagamento
Autenticazione
Registrazione
Gestione account
Interfaccia pagamento
Cerca prodotti
Gestione ordine
Gestione amministratore
Gestione produttore
Gestione cliente
Gestione ricerca
Cerca produttori
Elimina recensioni
Elimina utente
modifica prodotto
rimuovi prodotto
Inserisci recensione
Esegui ordine
Invio ricevuta ordine
Invia segnalazioni
genera report
inserisci promozione
Inserisci prodotto
Inserisci nel carrello
Elimina dal carrello
Gestione mail
Gestione database
Inserisci dati
Recupera dati
Gmail
Dati


AgriTrento
Utente
Utente non autenticato
Utente autenticato
Cliente
Produttore
Amministratore
cercaProduttori
cercaProdotti
inviaSegnalazione
registraUtente
autenticaUtente
recuperaPassword
verificaEsistenzaUtente
<<includi>>
<<includi>>
<<includi>>
modificaDatiAccount
inserisciRecensione
inserisciNelCarrello
eliminaDalCarrello
eseguiOrdine
inserisciProdotto
inserisciPromozione
generaReport
modificaProdotto
rimuoviProdotto
visualizzaProdotto
<<includi>>
<<includi>>
confermaPreparazione
confermaConsegna
visualizzaOrdine
<<includi>>
<<includi>>
pubblicaAvviso
eliminaAvviso
visualizzaSegnalazioni
eliminaRecensione

cNaming conventions

• Event: noun + past-participle verb (e.g. insurance claim lodged)

• Activity: verb + noun (e.g. assess credit risk)

When to use sub-processes?

1. Decompose large models into smaller ones, making them easier to understand and maintain
2. Share common fragments across multiple processes
3. Delimit parts of a process that can be:
▪ Repeated
▪ Interrupted

Start modeling with one single “white-box” pool

• Initially, put the events and tasks in only one pool – the pool of the party who is running the process

• Leave all other pools “black-boxed”

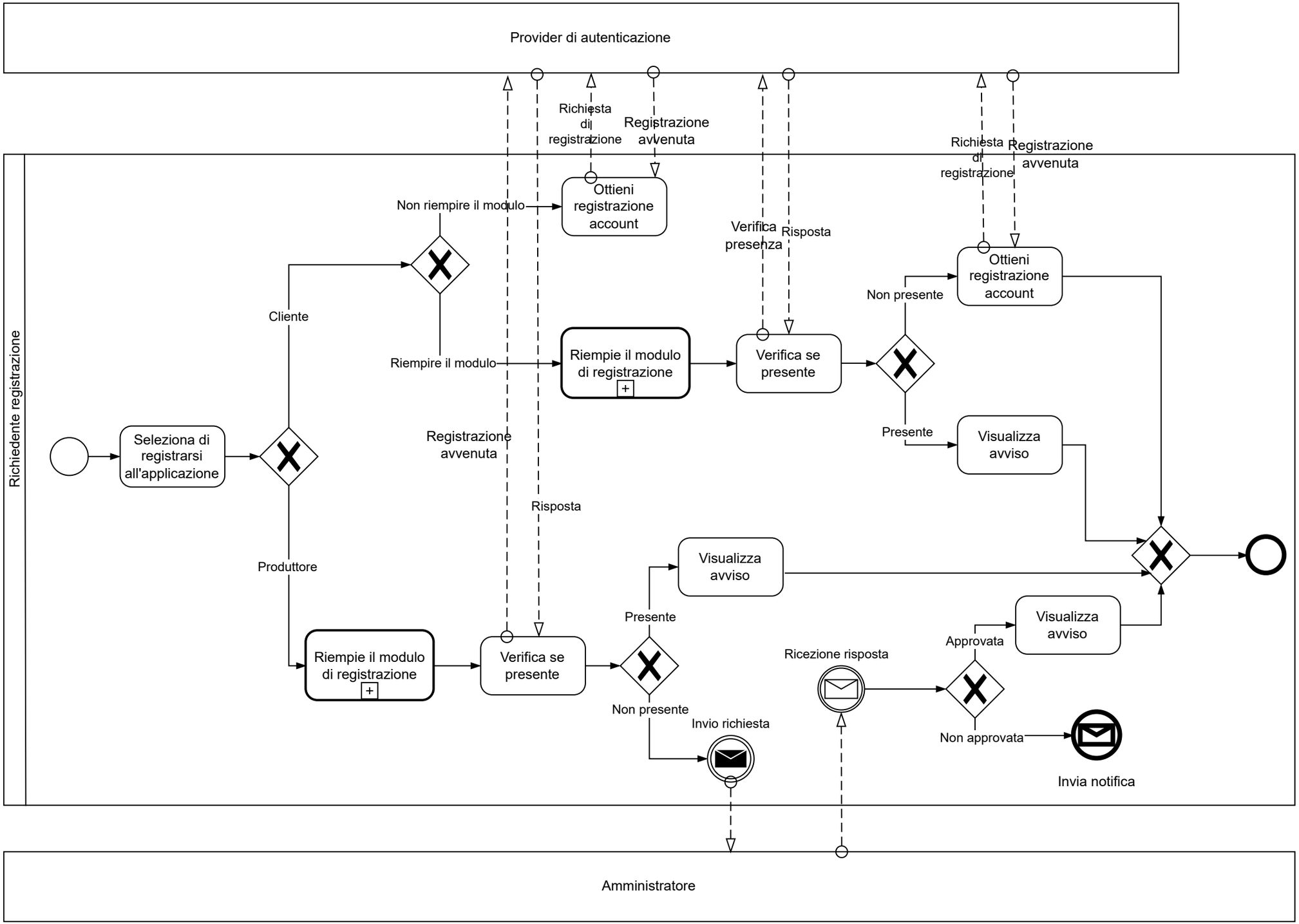
• Once you have modeled this way, and once the process diagram inside the white-box pool is complete, you can model the details (events and tasks) in the other pools if that is useful.

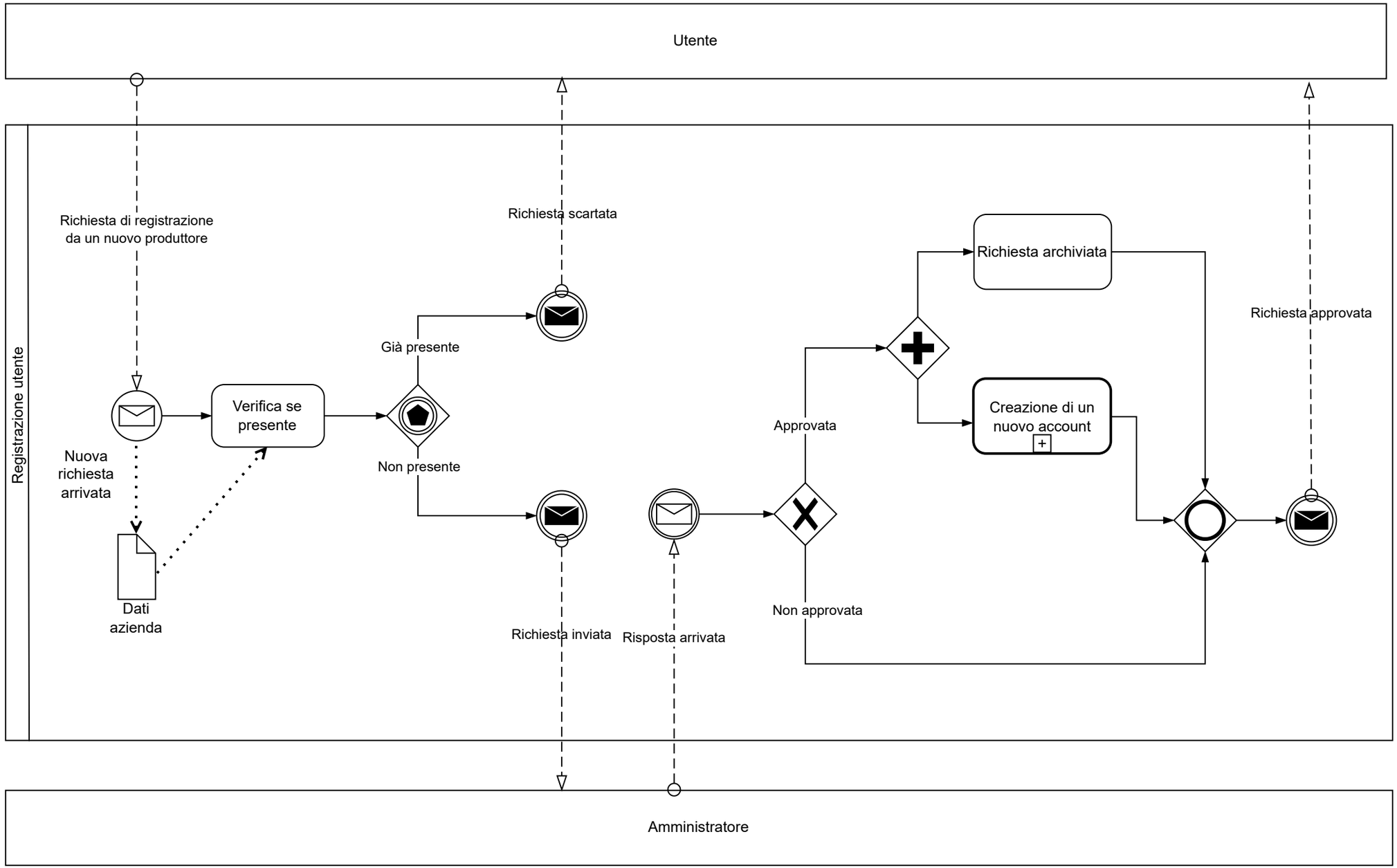
1. Give a name to every event and task
2. For tasks: verb followed by business object name and possibly complement
o Issue Driver Licence, Renew Licence via Agency
3. For events: object + past participle
o Invoice received, Claim settled
4. Label each XOR-split with a condition
o Policy is invalid, Claim is inadmissible

Model in blocks

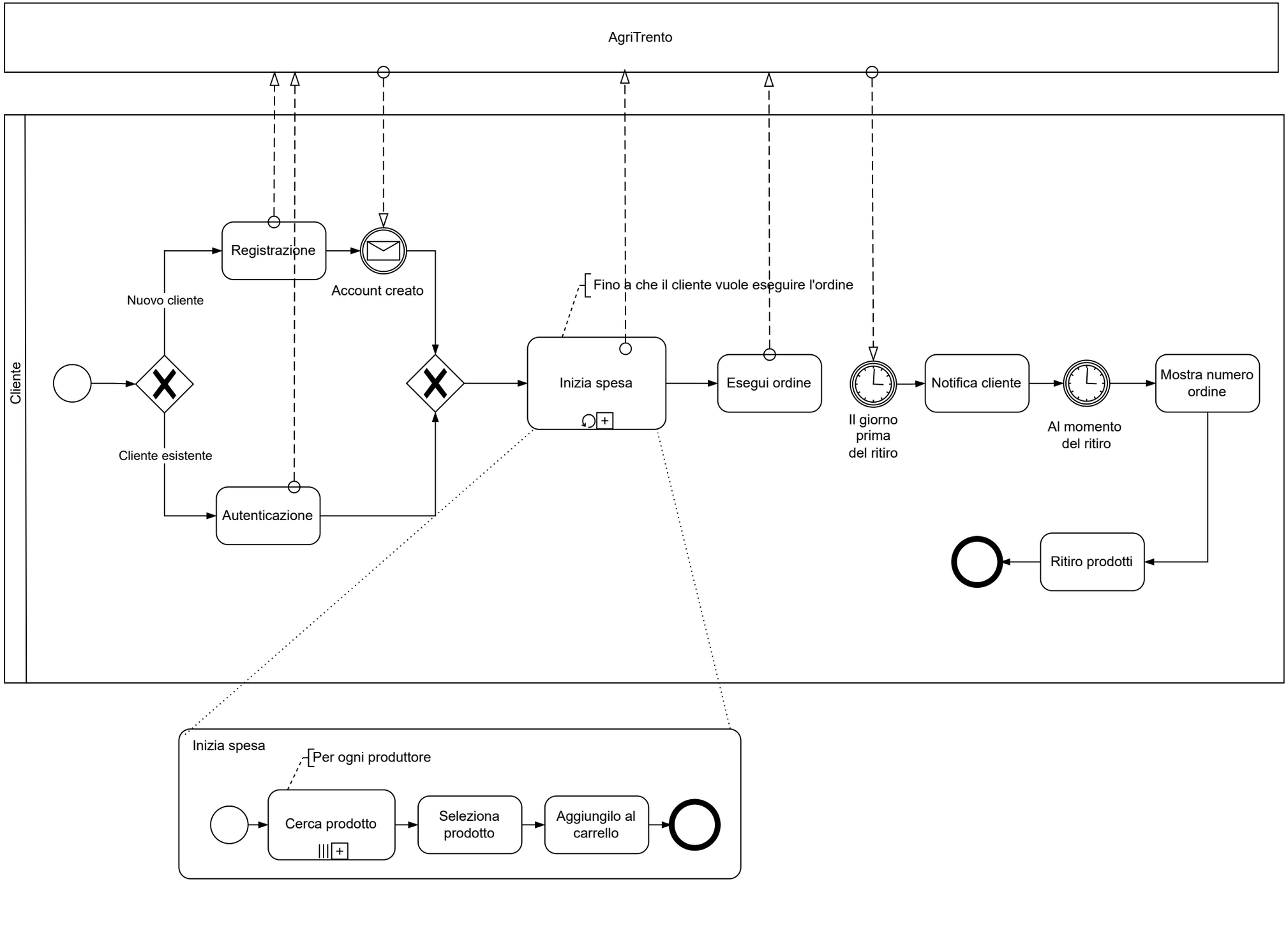
▪ Pair up each AND-split with an AND-join and each XORsplit with a XOR-join, whenever possible

▪ Exception: sometimes a XOR-split leads to two end events – different outcomes (cf. order management example)


Provider di autenticazione
Richiedente registrazione
Richiesta di registrazione
Registrazione avvenuta
Ottieni registrazione account
Non riempire il modulo
Riempire il modulo
Cliente
Produttore
Seleziona di registrarsi all'applicazione
Riempie il modulo di registrazione
Verifica se presente
Verifica presenza
Risposta
Non presente
Presente
Visualizza avviso
Registrazione avvenuta
Risposta
Visualizza avviso
Presente
Non presente
Invio richiesta
Ricezione risposta
Approvata
Non approvata
Visualizza avviso
Invia notifica
Amministratore



DA SCRIVERE CHE SI TRATTA DI UN BPMN PER GESTIRE UN UNTENTE CHE VUOLE DIVENTARE UN PRODUTTORE
Low Level details


AgriTrento
Cliente
Registrazione
Account creato
Nuovo cliente
Cliente esistente
Autenticazione
Fino a che il cliente vuole eseguire l'ordine
Inizia spesa
Esegui ordine
Il giorno
prima
del ritiro
Notifica cliente
Al momento
del ritiro
Mostra numero
ordine
Ritiro prodotti
Inizia spesa
Per ogni produttore
Cerca prodotto
Seleziona
prodotto
Aggiungilo al
carrello