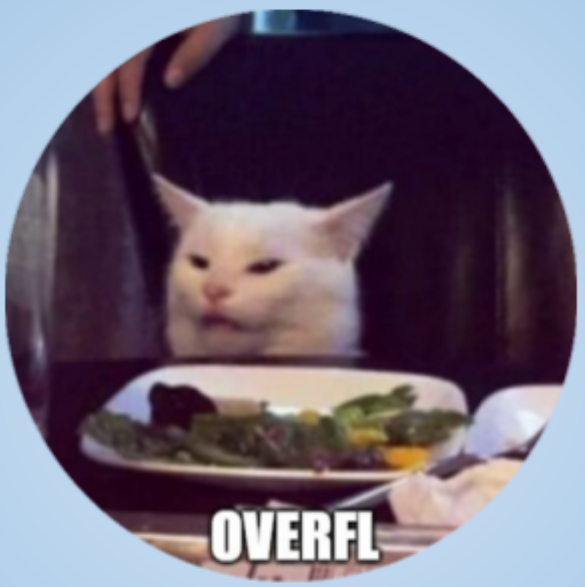
Progetto Palestra

Analisi e Progettazione del Software
Preappello del 08/06/2023



Gruppo Overfl

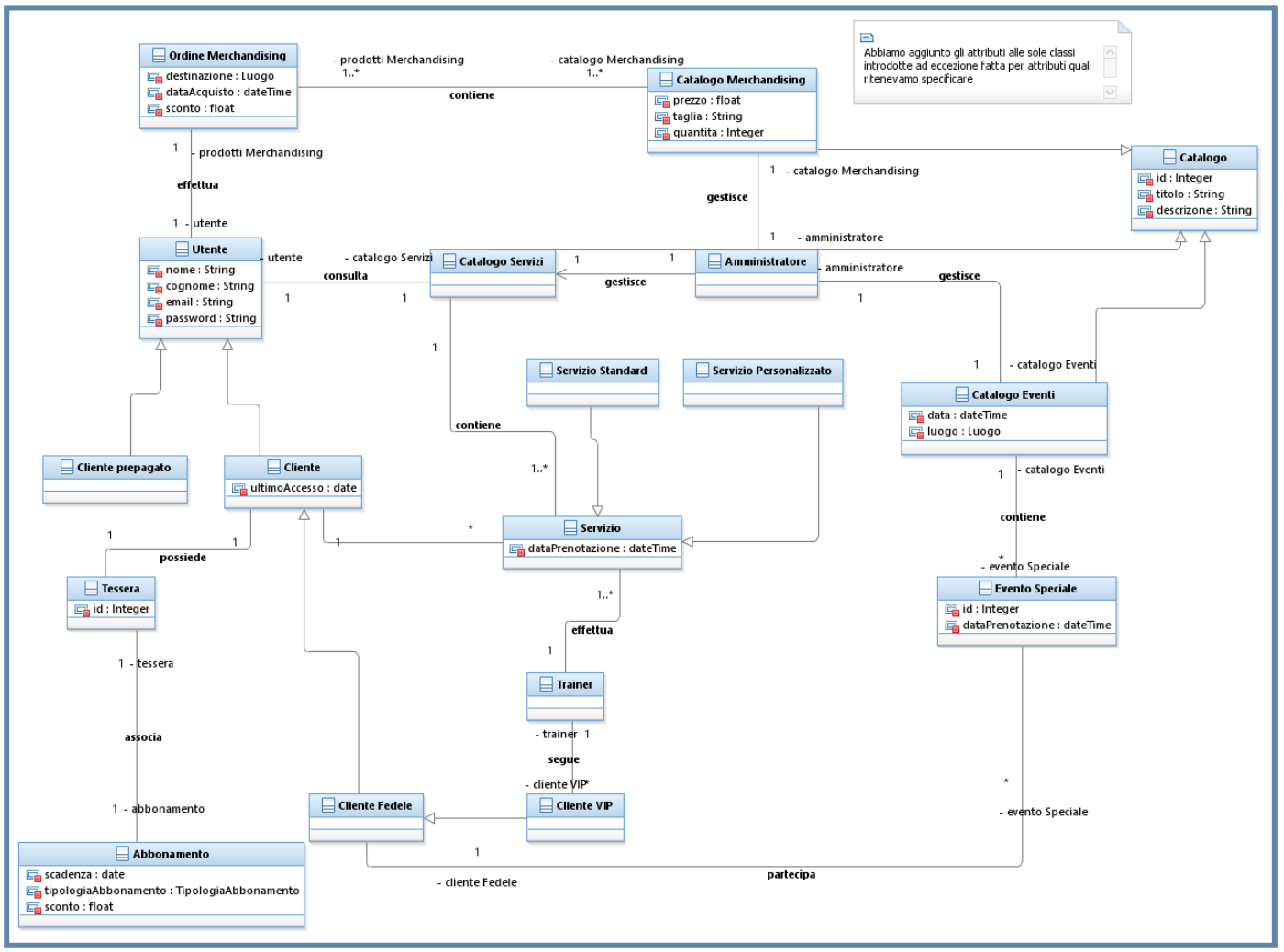
Brini Luca - mat. 879459
Broccoletti Andrea - mat. 886155
Falbo Andrea - mat. 887525
Karzal Youness - mat. 879430
Lesinigo Simone - mat. 899540

Sommario

Link

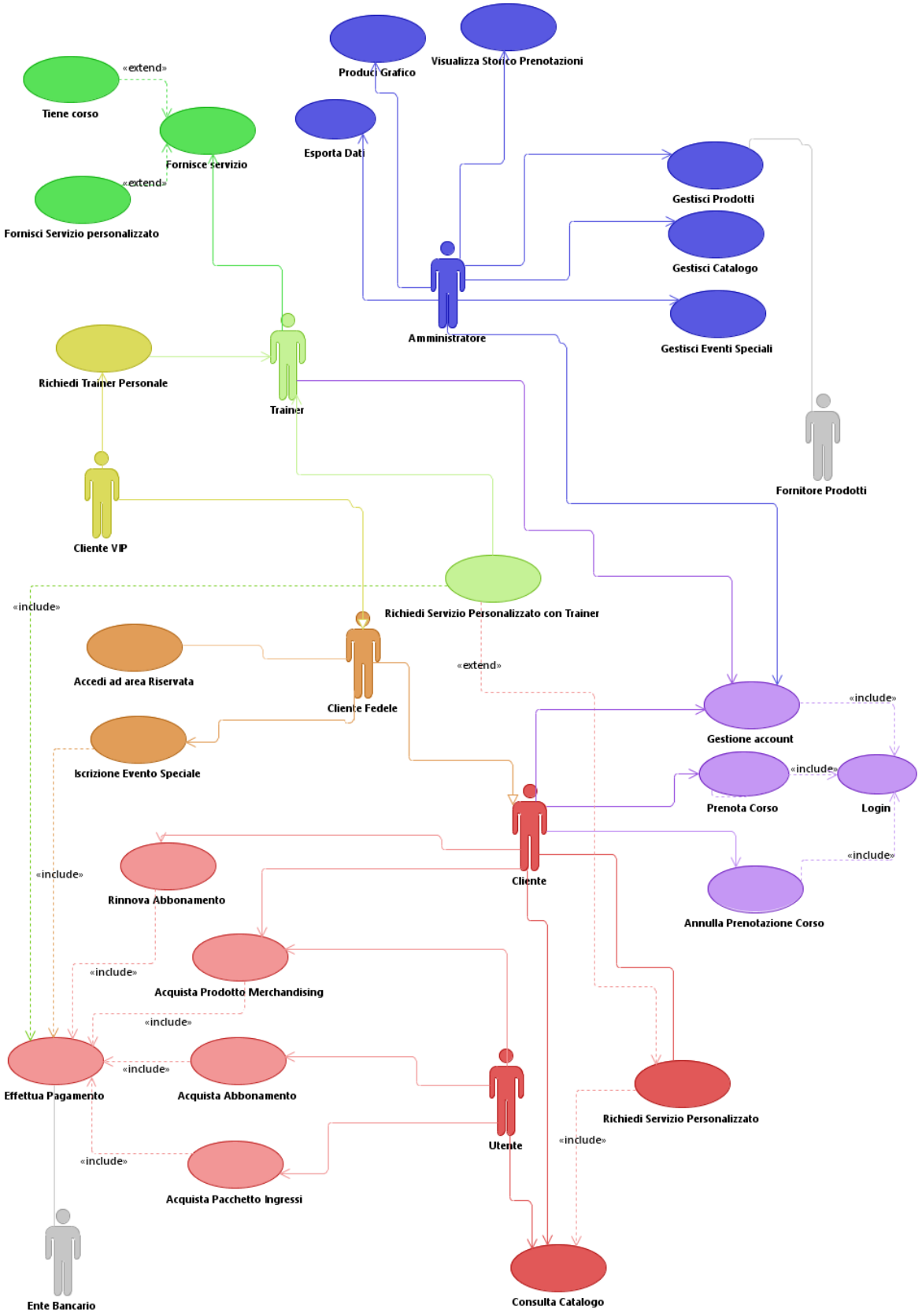
Link alla cartella Drive per visualizzare le foto in alta risoluzione

Modello di Dominio



Casi d'uso

Diagramma dei casi d'uso



Casi d'uso dettagliati

Falbo Andrea

Richiedi Trainer Personale

Selezione del Caso d'Uso	Descrizione
Nome del Caso d'Uso	Richiedi Trainer Personale
Portata	Palestra
Livello	Obiettivo Utente
Attore Primario	Cliente VIP
Parti Interessate e Interessi	-Cliente VIP: Vuole essere seguito da un Trainer Personale -Trainer: Vuole seguire un Cliente VIP
Pre-condizioni	Nessuna
Garanzia di successo	Un Trainer viene assegnato al Cliente VIP
Scenario Principale di Successo	1. Il caso d'uso inizia quando un Cliente VIP seleziona "Richiedi Trainer Personale" 2. Il Sistema mostra i Trainer disponibili 3. Il Cliente VIP seleziona il Trainer personale
Estensioni	2a. Il Sistema non ha Trainer disponibili al momento 1. Il Sistema comunica che non sono stati trovati Trainer disponibili e di riprovare più tardi
Requisiti speciali	Il sistema dovrà fornire la possibilità di filtrare i Trainer in base alle attività in cui sono specializzati (Usabilità)
Elenco delle variabili tecnologiche e dei dati	Il Sistema fornirà uno strumento di messaggistica online tra il Cliente e il Trainer per questioni organizzative
Frequenza di ripetizione	Potenzialmente uno per ogni cliente VIP
Varie	Il Sistema dovrebbe permettere ad un Cliente VIP di poter essere seguito da più Trainer, uno per ciascuna attività svolta

Broccoletti Andrea

Iscrizione Evento Speciale

Selezione del Caso d'Uso	Descrizione
Nome del Caso d'Uso	Iscrizione Evento Speciale
Portata	Palestra
Livello	Obiettivo Utente
Attore Primario	Cliente Fedele
Parti Interessate e Interessi	-Cliente Fedele: Il Cliente Fedele vuole iscriversi a un Evento Speciale
Pre-condizioni	Nessuna
Garanzia di successo	La prenotazione del Cliente Fedele viene registrata correttamente
Scenario Principale di Successo	1. Il caso d'uso inizia quando un Cliente Fedele seleziona "Iscrizione Evento Speciale" 2. Il Sistema fornisce la lista di eventi speciali disponibili 3. Il Cliente Fedele seleziona un evento a cui desidera partecipare 4. Il Sistema calcola il prezzo dell'evento 5. Il Cliente Fedele effettua il pagamento 6. Il Sistema notifica l'avvenuto successo della prenotazione
Estensioni	4a. Se il Cliente Fedele è VIP allora il costo dell'evento è pari a 0€ 5b. Il pagamento non va a buon fine 1. Il Sistema chiede al Cliente Fedele se vuole tentare nuovamente il pagamento o tornare alla lista degli eventi speciali disponibili
Requisiti speciali	La sessione di prenotazione scade dopo 10 minuti (Prestazionali)
Elenco delle variabili tecnologiche e dei dati	Il Sistema invia una email al Cliente Fedele con luogo, data e ora dell'evento speciale
Frequenza di ripetizione	Potenzialmente una per ogni Evento Speciale
Varie	Il Sistema dovrebbe notificare il Cliente Fedele il giorno prima dell'Evento Speciale

Lesinigo Simone

Visualizza Storico Prenotazioni

Selezione del Caso d'Uso	Descrizione
Nome del Caso d'Uso	Visualizza Storico Prenotazioni
Portata	Palestra
Livello	Obiettivo Utente
Attore Primario	Amministratore
Parti Interessate e Interessi	-Amministratore: vuole visualizzare lo storico delle prenotazioni
Pre-condizioni	Nessuna
Garanzia di successo	Nessuna
Scenario Principale di Successo	1. Il caso d'uso inizia quando l'Amministratore seleziona "Visualizza Storico Prenotazioni" 2. Il Sistema chiede all'Amministratore se vuole inserire filtri per data, tipologia di Cliente, tipo di Abbonamento e/o carnet di ingressi per la visualizzazione dello storico delle prenotazioni 3. L'Amministratore inserisce i filtri desiderati 4. Il Sistema mostra lo storico delle prenotazioni in base ai filtri scelti
Estensioni	3a. L'Amministratore decide di produrre i grafici e/o di esportare i dati
Requisiti speciali	Il Sistema dovrà fornire la possibilità di filtrare i dati (Usabilità)
Elenco delle variabili tecnologiche e dei dati	• Ogni grafico può essere scaricato come PDF o PNG o XLS • I dati vengono esportati in un file XLS
Frequenza di ripetizione	Potenzialmente infinito
Varie	Nessuna

Brini Luca

Rinnova Abbonamento

Selezione del Caso d'Uso	Descrizione
Nome del Caso d'Uso	Rinnovo Abbonamento
Portata	Palestra
Livello	Obiettivo Utente
Attore Primario	Cliente
Parti Interessate e Interessi	-Cliente: vuole rinnovare il proprio l'abbonamento per la palestra
Pre-condizioni	-Il Cliente deve avere l'ultimo abbonamento attivo oppure scaduto da meno di 7 giorni
Garanzia di successo	Il cliente rinnova l'abbonamento
Scenario Principale di Successo	1. Il caso d'uso inizia quando il Cliente seleziona "Rinnova Abbonamento" 2. Il Sistema chiede il numero della tessera associata al Cliente 3. Il Cliente inserisce il proprio numero di tessera 4. Il Sistema chiede la tipologia di abbonamento da rinnovare 5. Il Cliente seleziona la tipologia di abbonamento da rinnovare 6. Il Sistema calcola il prezzo del rinnovo dell'abbonamento 7. Il Cliente effettua il pagamento 8. Il Sistema carica il nuovo abbonamento sulla tessera del Cliente 9. Il Cliente viene notificato dell'avvenuto acquisto
Estensioni	3a. Il Cliente inserisce il numero di tessera non valido 1. Il sistema ritorna errore e chiede di riprovare 6a. L'abbonamento selezionato è annuale 1. Viene applicato uno sconto del 5% 6b. L'abbonamento selezionato è biennale 1. Viene applicato uno sconto del 10% 6c. L'abbonamento selezionato ha limite di utilizzo fino alle ore 17.00 1. Viene applicato uno sconto del 30% 6d. L'abbonamento è rinnovato nel periodo Giugno-Agosto 1. Viene applicato uno sconto del 20%
Requisiti speciali	Nessuno
Elenco delle variabili tecnologiche e dei dati	Il Sistema offre la possibilità di rinnovo automatico dell'abbonamento
Frequenza di ripetizione	Potenzialmente infinito
Varie	Il Sistema potrebbe fornire un servizio di link affiliati per ottenere vantaggi

Karzal Youness

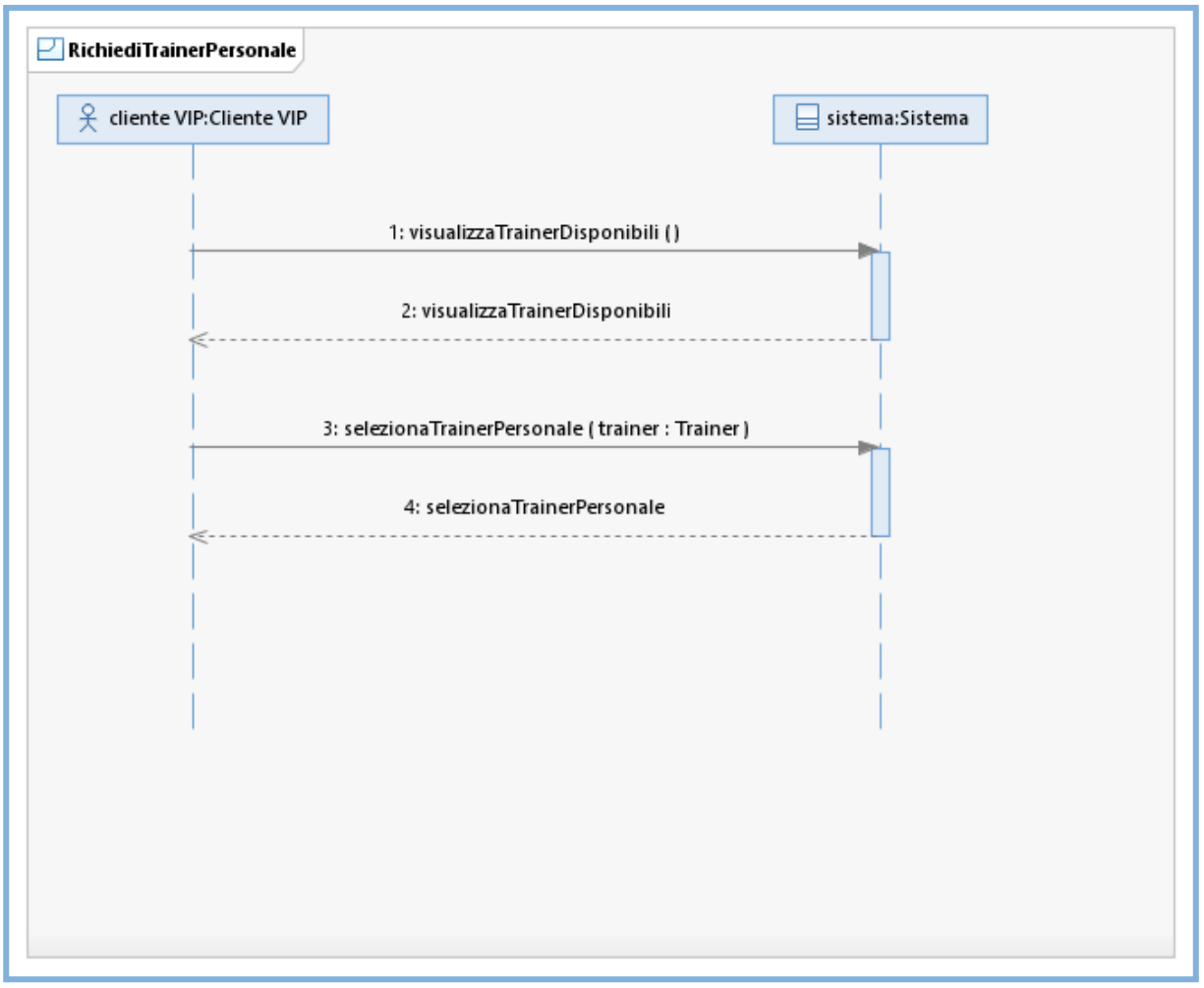
Acquista Merchandising (Cliente)

Selezione del Caso d'Uso	Descrizione
Nome del Caso d'Uso	Acquista Merchandising (cliente)
Portata	Palestra
Livello	Obiettivo Utente
Attore Primario	Cliente
Parti Interessate e Interessi	-Cliente: il Cliente vuole acquistare dei prodotti
Pre-condizioni	Nessuna
Garanzia di successo	Viene creato un nuovo ordine dei prodotti acquistati
Scenario Principale di Successo	1. Il caso d'uso inizia quando un Cliente seleziona "Acquista Merchandising" 2. Il Sistema mostra la lista dei prodotti disponibili 3. Il Cliente aggiunge i prodotti che vuole acquistare al carrello 4. Il Cliente procede all'acquisto 5. Il Sistema chiede l'indirizzo di spedizione del cliente 6. Il Cliente inserisce i dettagli di spedizione 7. Il Sistema calcola il prezzo dei prodotti 8. Il Cliente effettua il pagamento 9. Il Sistema notifica il gestore dei prodotti del nuovo ordine 10. Il Sistema aggiorna il numero di prodotti disponibili 11. Il Sistema crea l'ordine dei prodotti acquistati e genera un resoconto dell'ordine
Estensioni	7a. Il Cliente è Cliente Fedele 1. Viene applicato uno sconto del 30% 7b. Il Cliente è Cliente VIP 1. Viene applicato uno sconto del 45% 8a. Il pagamento non va a buon fine 1. Il Sistema chiede al Cliente se vuole tentare nuovamente il pagamento o tornare alla lista dei prodotti disponibili 2. Il Cliente effettua la scelta
Requisiti speciali	Il Sistema dovrà fornire la possibilità di filtrare i prodotti per prezzo e per tipo (Usabilità)
Elenco delle variabili tecnologiche e dei dati	Il Cliente potrà visualizzare lo stato del proprio ordine tramite un sistema di tracciamento nella propria area personale
Frequenza di ripetizione	Potenzialmente infinito
Varie	• Il Sistema dovrebbe fornire una chat per il supporto sui propri ordini • Il Sistema dovrebbe offrire la presenza di appositi locker all'interno della palestra come punti di spedizione alternativi

Diagramma di Sequenza di Sistema

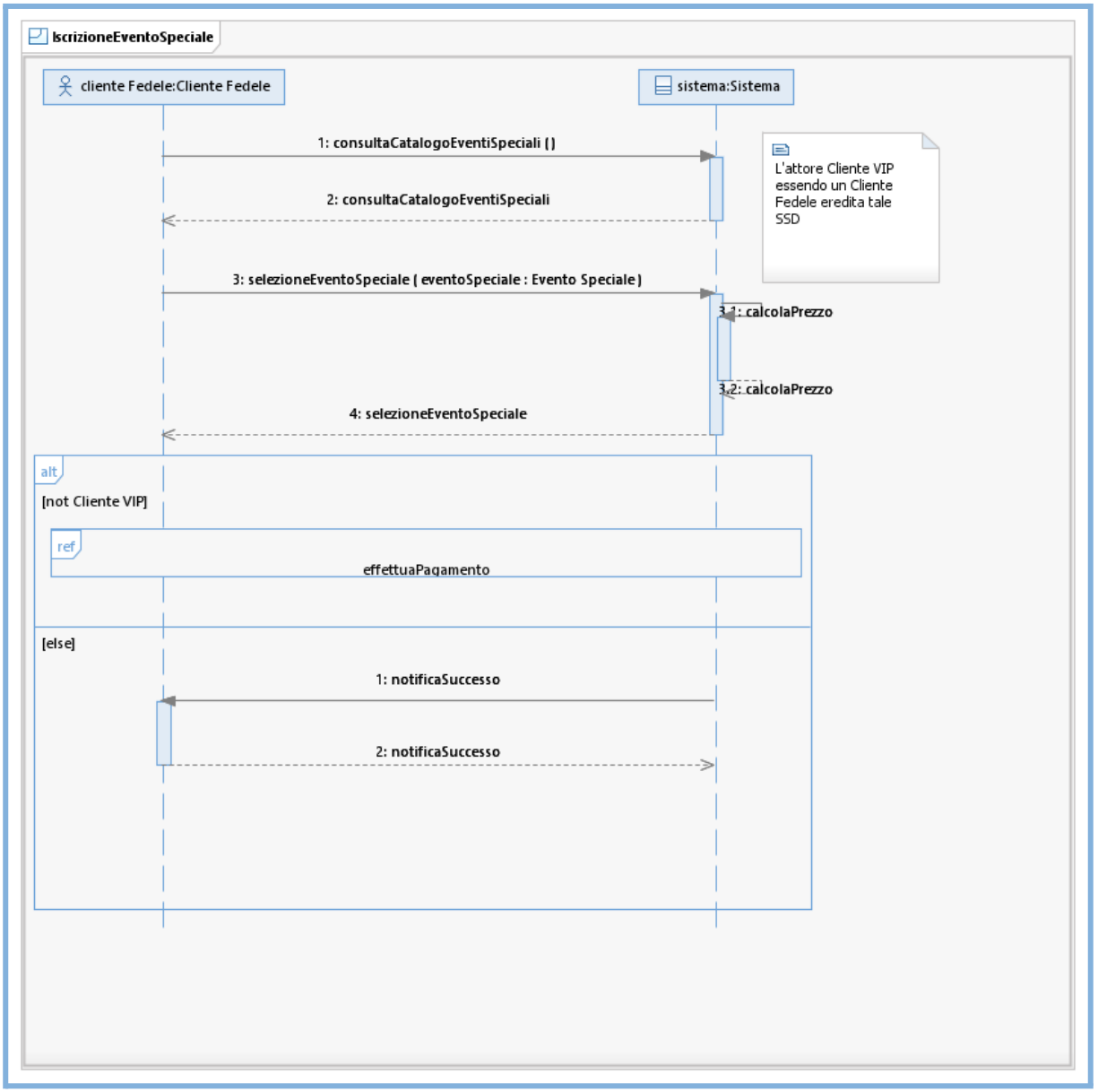
Falbo Andrea

Richiedi Trainer Personale



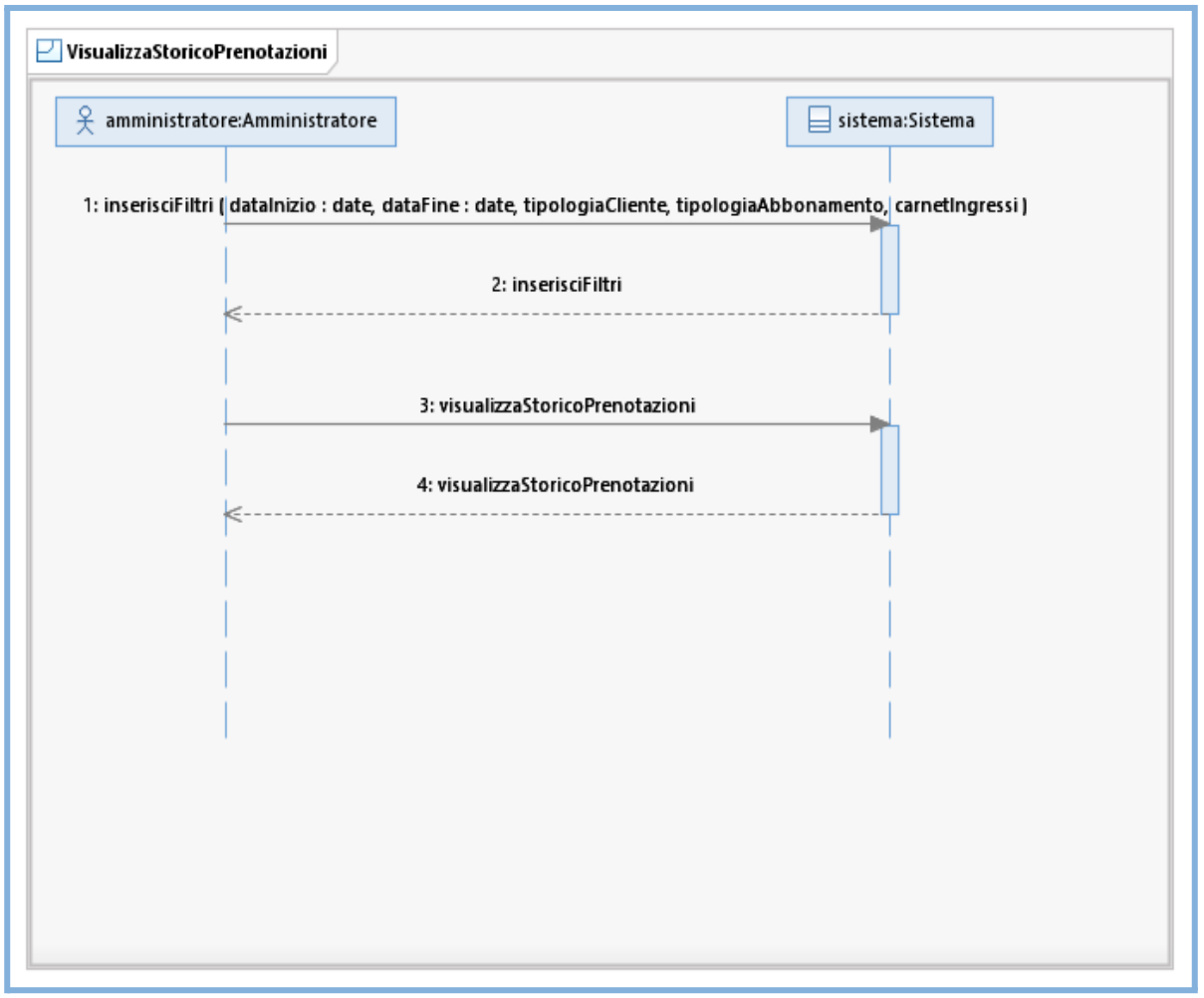
Broccoletti Andrea

Iscrizione Evento Speciale



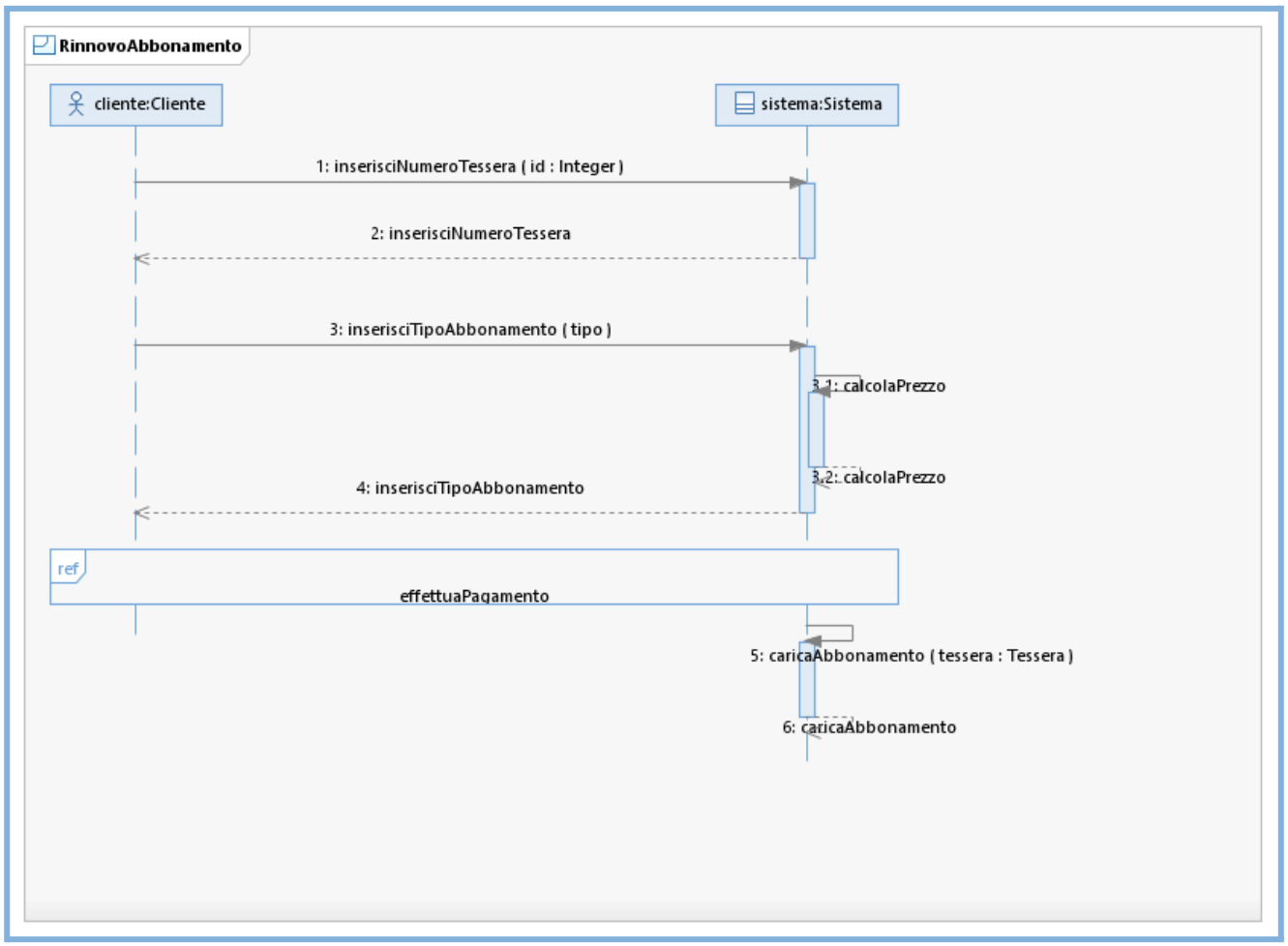
Lesinigo Simone

Visualizza Storico Prenotazioni



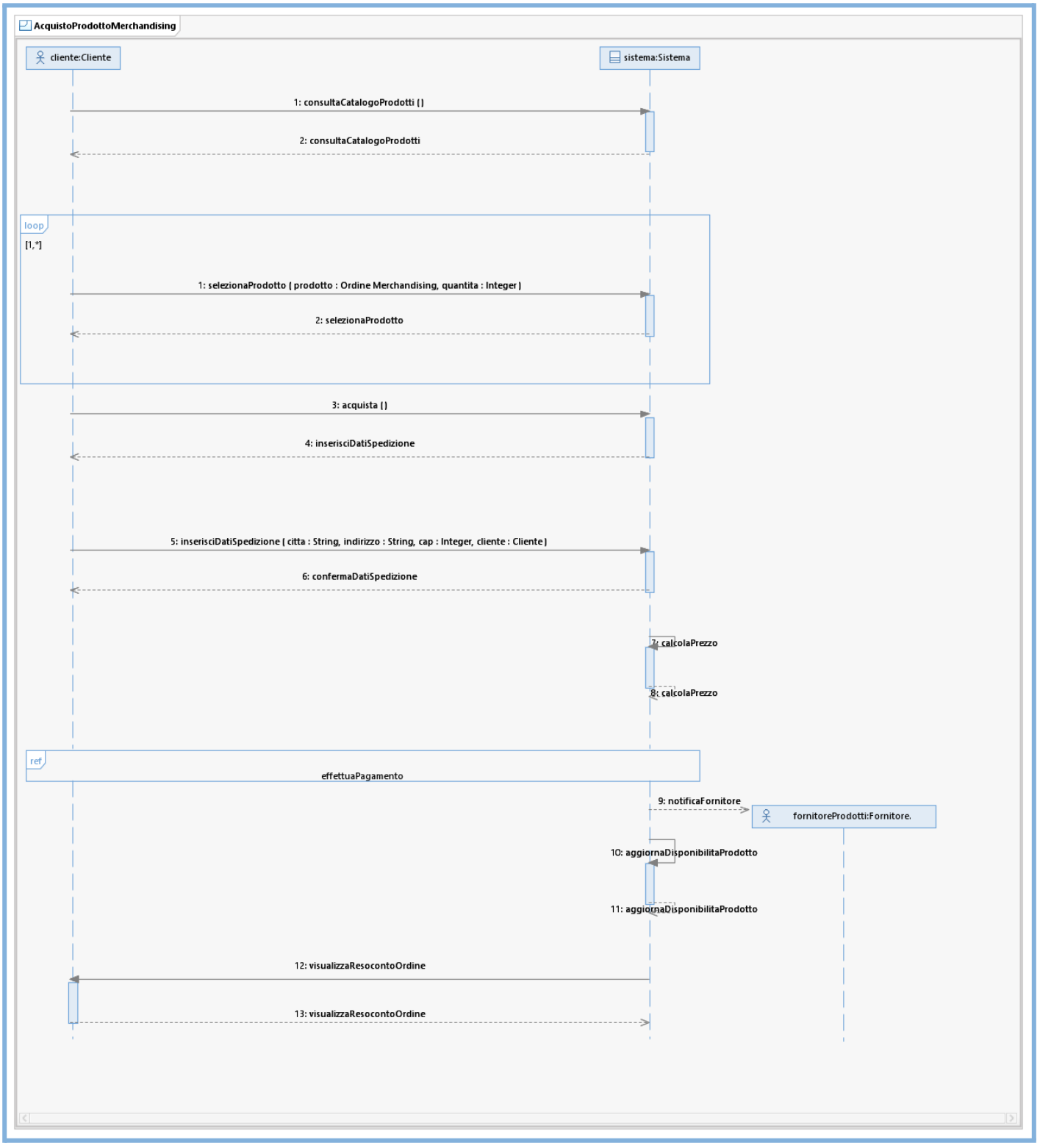
Brini Luca

Rinnovo Abbonamento



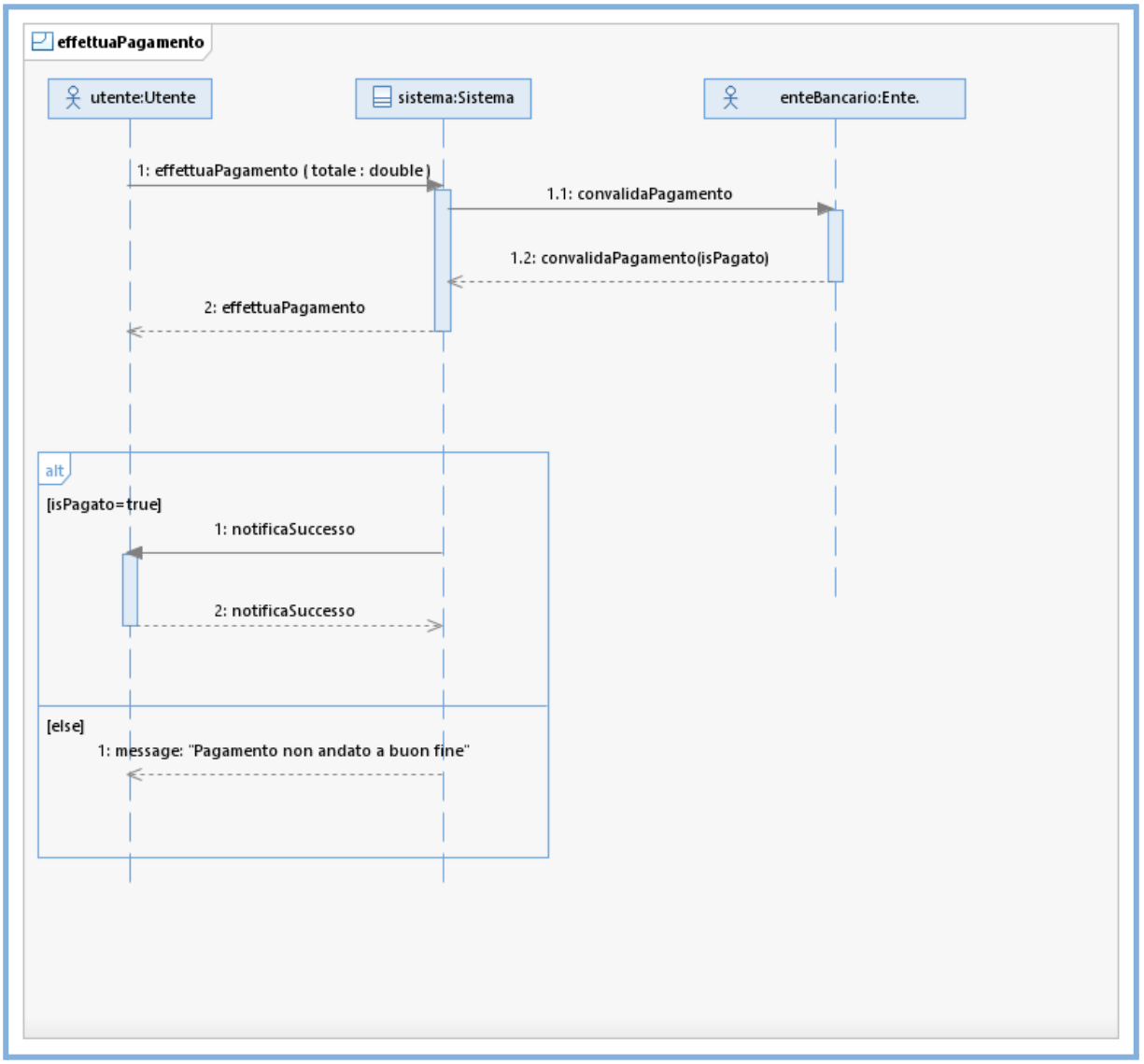
Karzal Youness

Acquista Merchandising (Cliente)



Gruppo Overfl

Effettua Pagamento



Contratti

Falbo Andrea

SelezionaTrainerPersonale

Contratto	
Operazione	SelezionaTrainerPersonaler(Trainer trainer)
Riferimento caso d'Uso	Richiedi Trainer Personale
Precondizioni	È in corso la selezione di un Trainer trainer da parte di un Cliente cliente durante la richiesta di trainer personale
Postcondizioni	• Viene associata l'istanza trainer di Trainer al Cliente cliente

Broccoletti Andrea

SelezionaEventoSpeciale

Contratto	
Operazione	SelezionaEventoSpeciale(CatalogoEventi eventoSpeciale)
Riferimento caso d'Uso	Iscrizione Evento Speciale
Precondizioni	È in corso la selezione di un CatalogoEventi eventoSpeciale durante l'iscrizione ad un evento speciale
Postcondizioni	• Viene creata l'istanza prenotazioneEvento di EventoSpeciale • L'attributo prenotazioneEvento.evento viene inizializzato a eventoSpeciale

Lesinigo Simone

VisualizzaStoricoPrenotazioni

Contratto	
Operazione	VisualizzaStoricoPrenotazioni(FiltroPrenotazioni filtro)
Riferimento caso d'Uso	Visualizza Storico Prenotazioni
Precondizioni	È in corso la visualizzazione dello storico delle prenotazioni
Postcondizioni	• L'Amministratore visualizza le istanze dello storico delle prenotazioni in base al FiltroPrenotazioni filtro selezionato in precedenza

Brini Luca

CaricaAbbonamento

Contratto	
Operazione	CaricaAbbonamento(Tessera tessera, TipologiaAbbonamento tipoAbbonamento)
Riferimento caso d'Uso	Rinnovo Abbonamento
Precondizioni	È in corso il rinnovo di un abbonamento per il Cliente cliente
Postcondizioni	• Viene creata una nuova istanza abbonamento di Abbonamento • L'attributo abbonamento.scadenza viene inizializzato in base alla TipologiaAbbonamento tipo scelta • L'attributo abbonamento.sconto viene inizializzato • Viene associata l'istanza abbonamento di Abbonamento alla Tessera tessera del Cliente cliente

Karzal Youness

InserisciDatiSpedizione

Contratto	
Operazione	InserisciDatiSpedizione (Luogo destinazione, Cliente cliente)
Riferimento caso d'Uso	Acquista Merchandising
Precondizioni	Si sta effettuando un OrdineMerchandising ordine
Postcondizioni	• Viene creata l'istanza ordine di OrdineMerchandising • L'attributo ordine.destinazione viene inizializzato a destinazione • L'attributo ordine.isPagato viene inizializzato a false • Viene associata l'istanza ordine di OrdineMerchandising al Cliente cliente

Architettura Logica

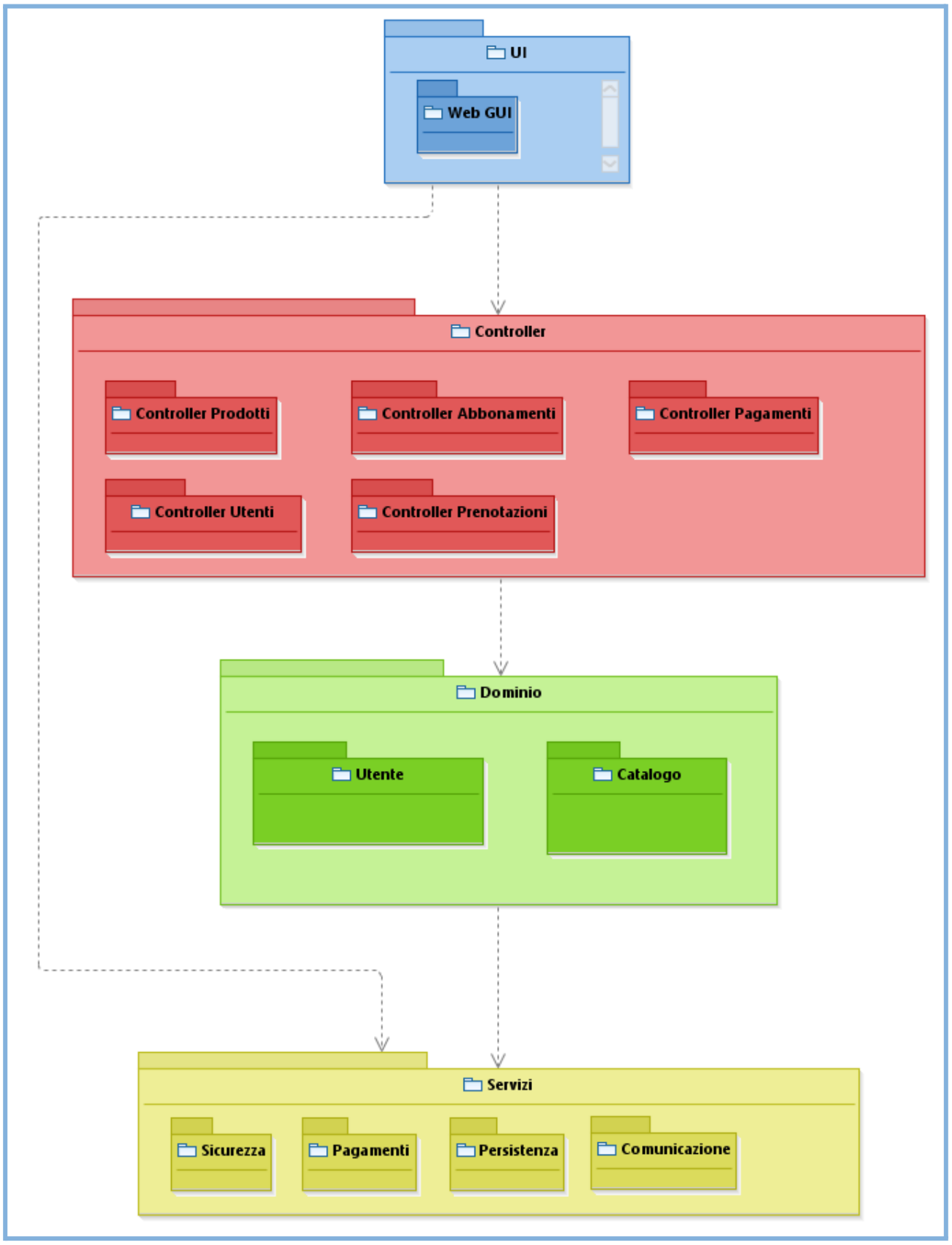
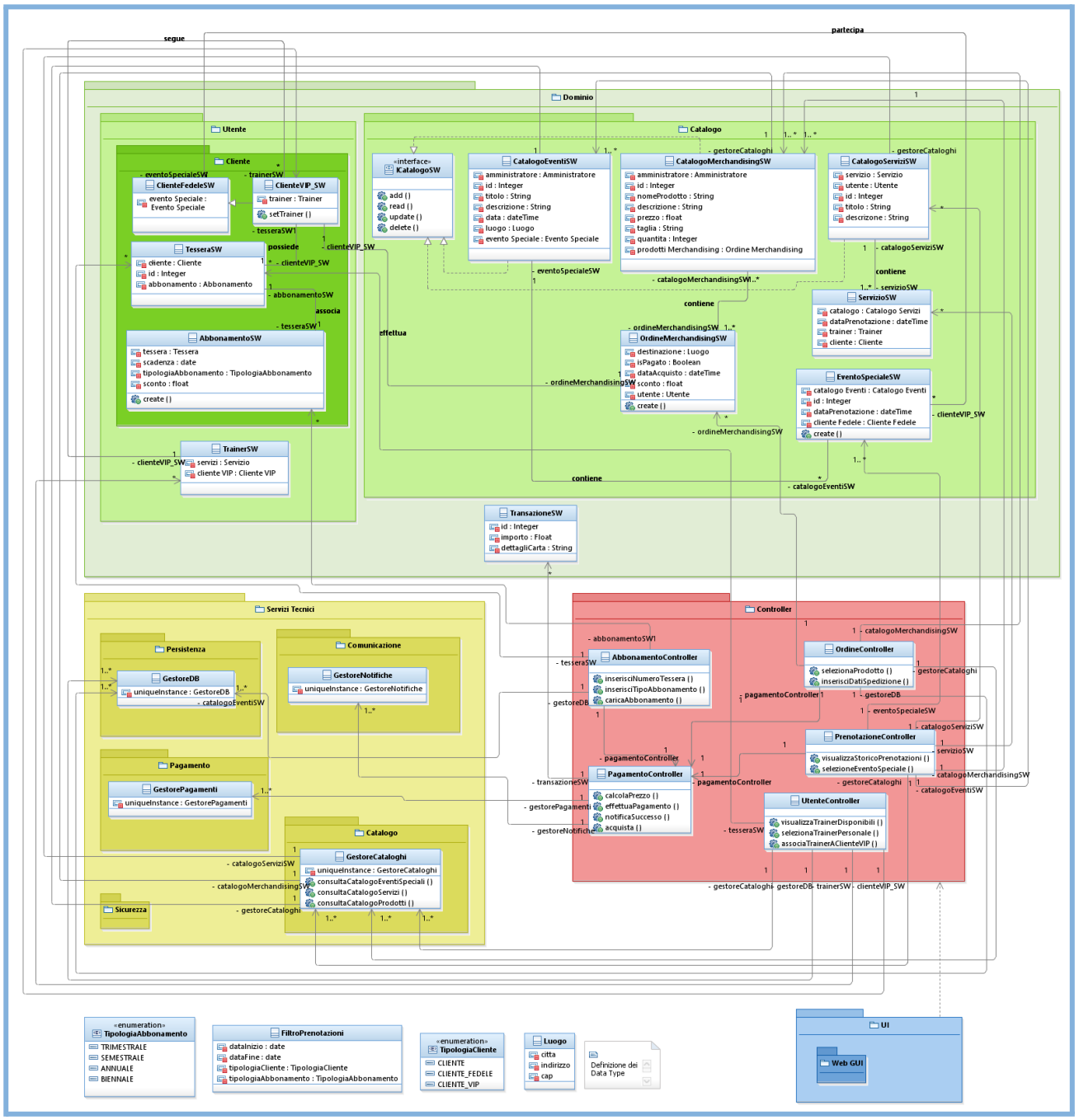
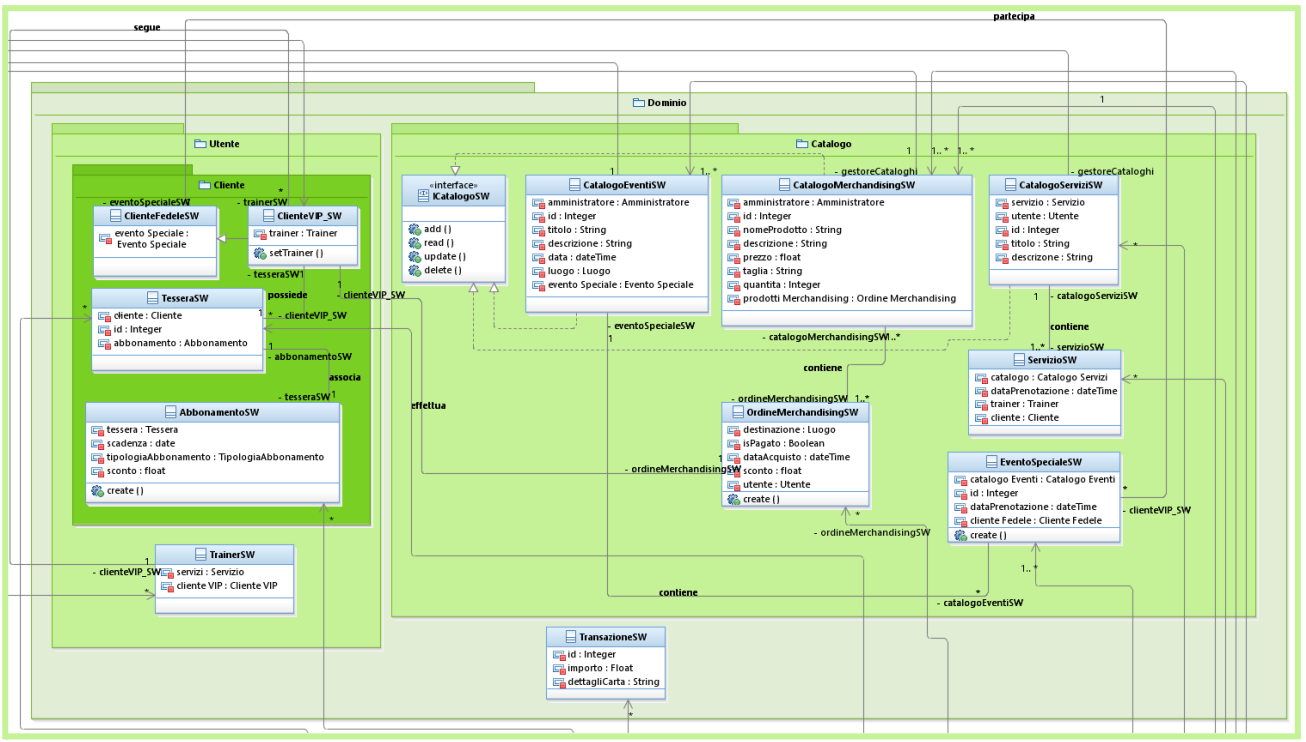


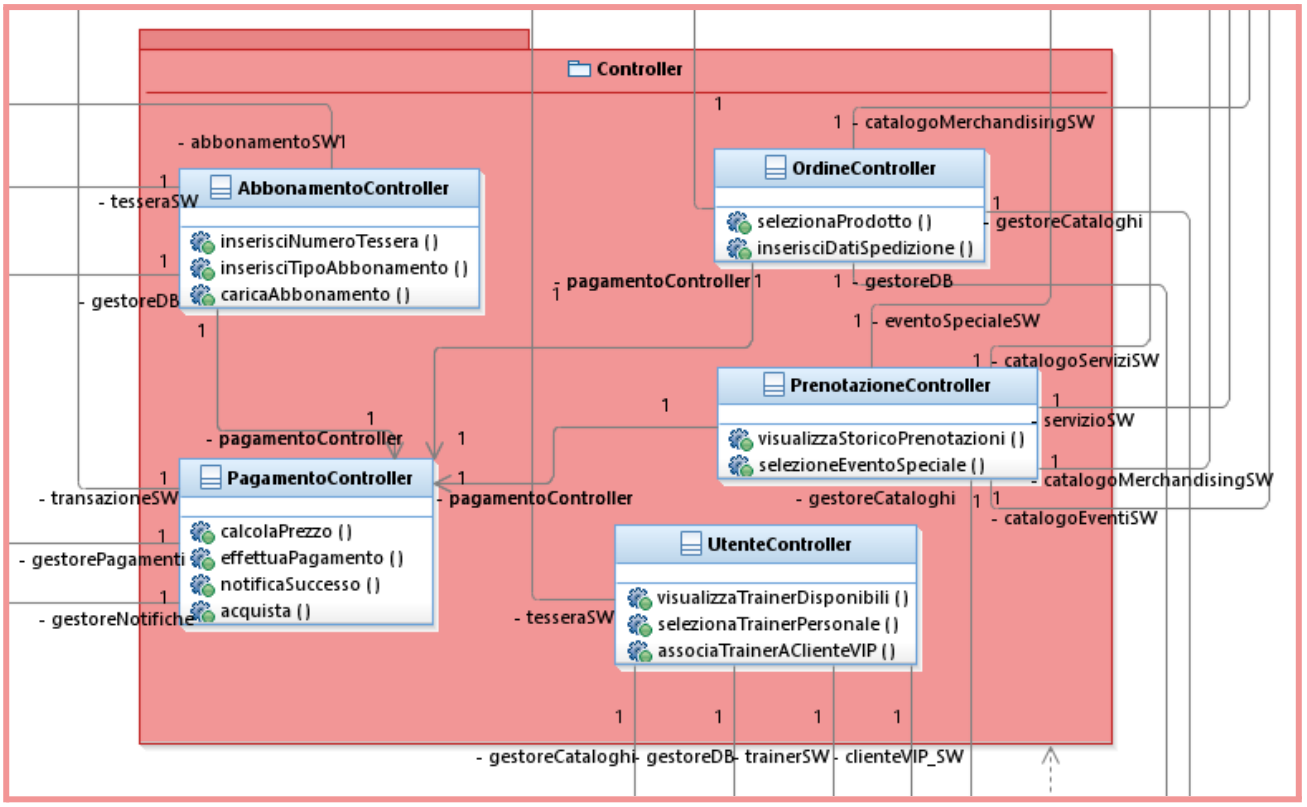
Diagramma delle Classi Software



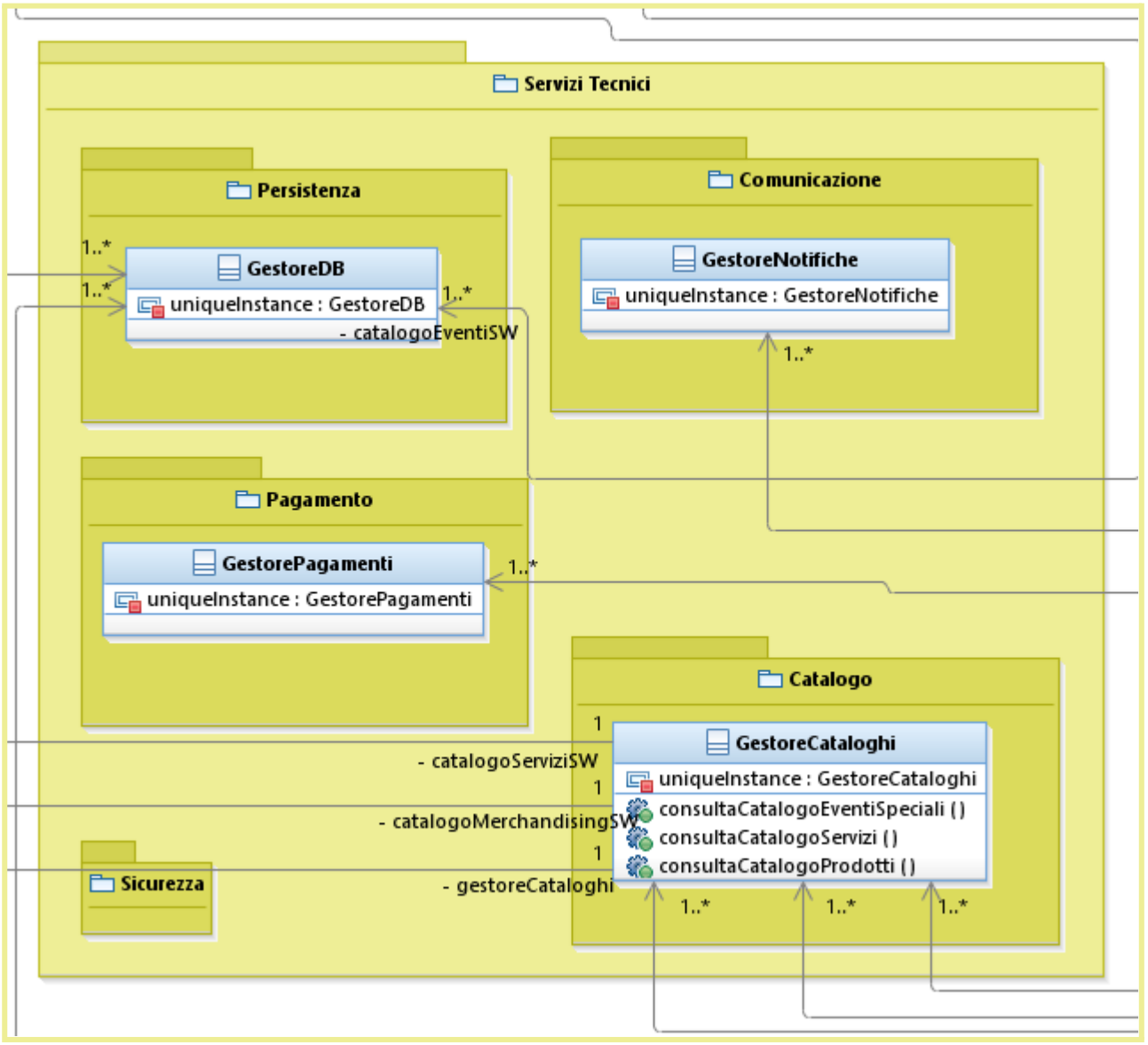
Package Dominio



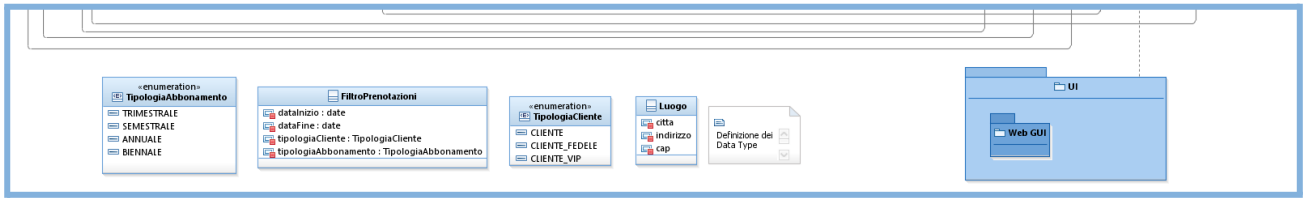
Package Controller



Package Servizi Tecnici



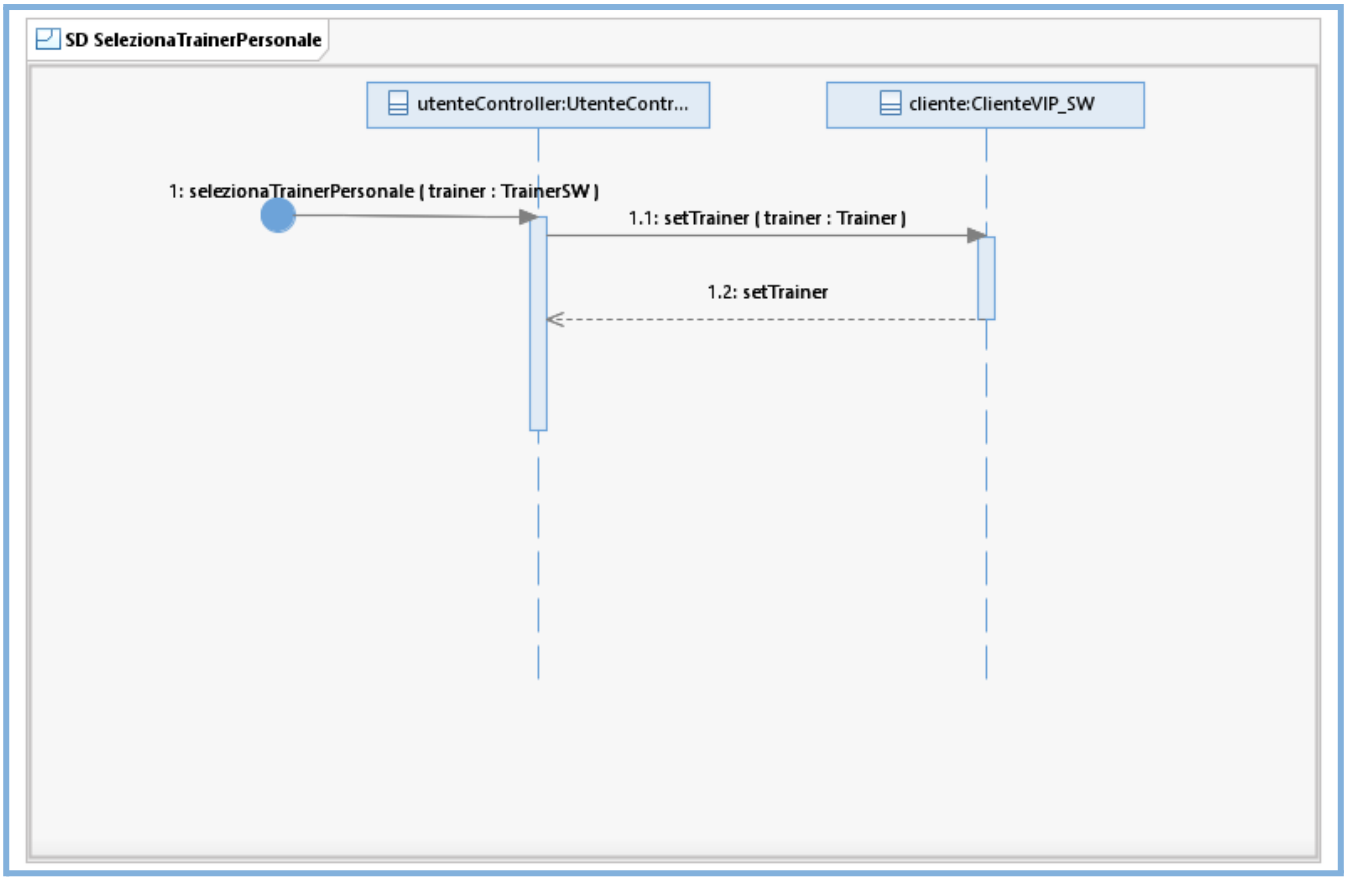
Package UI e Data Type



Diagrammi di Sequenza

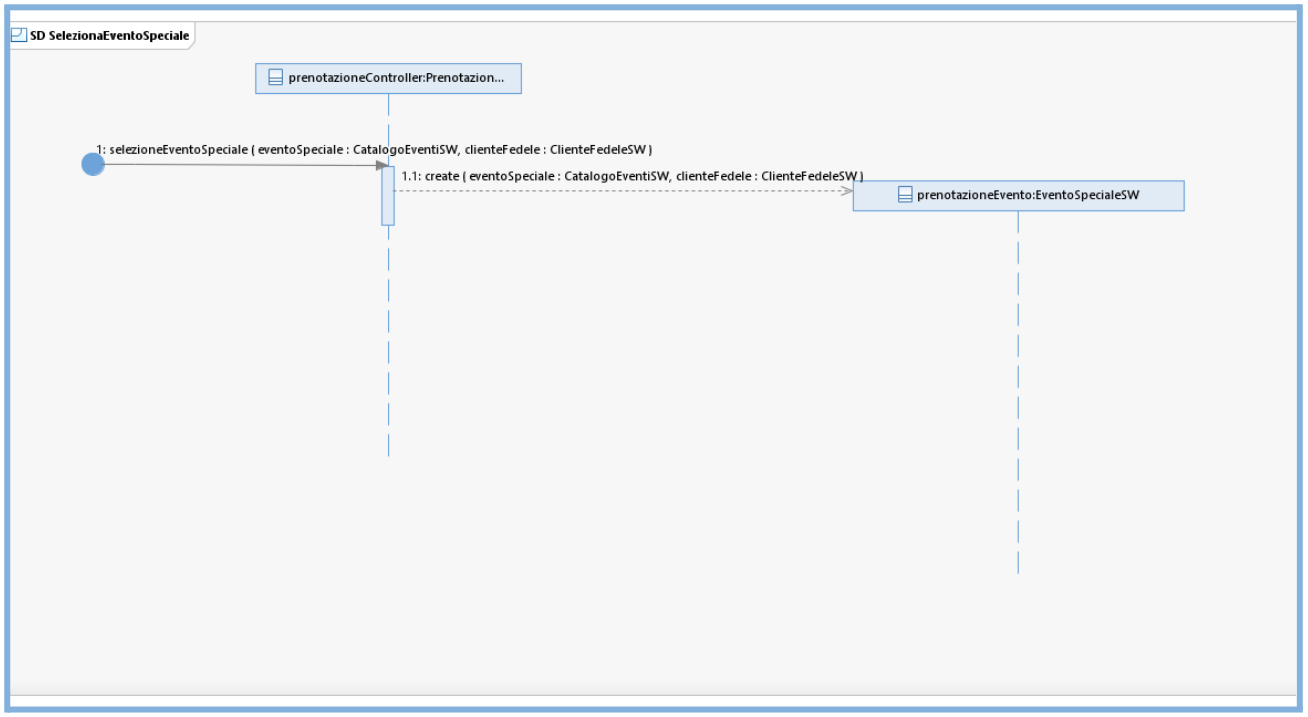
Falbo Andrea

SelezionaTrainerPersonale



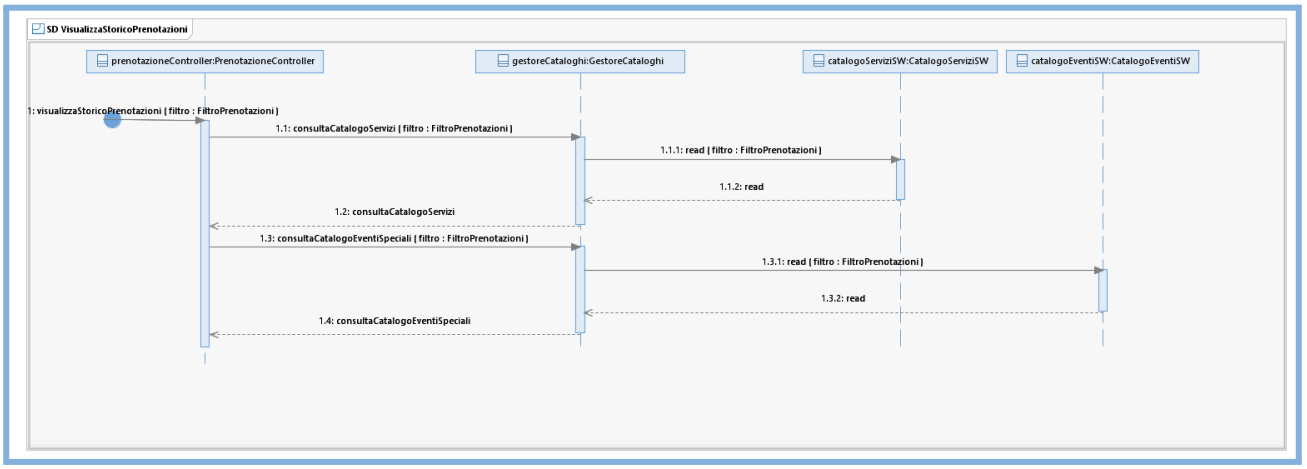
Broccoletti Andrea

SelezionaEventoSpeciale



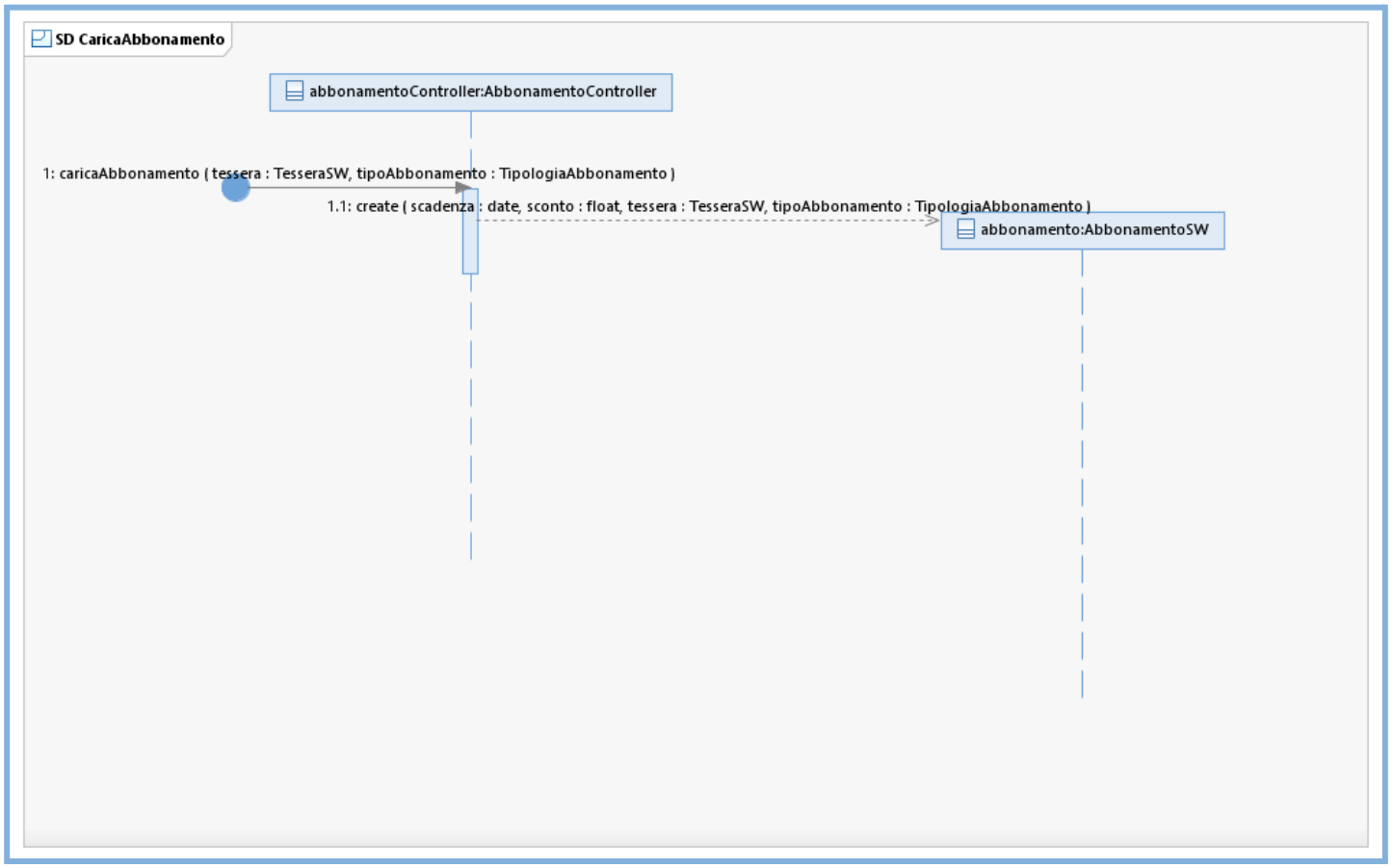
Lesinigo Simone

VisualizzaStoricoPrenotazioni



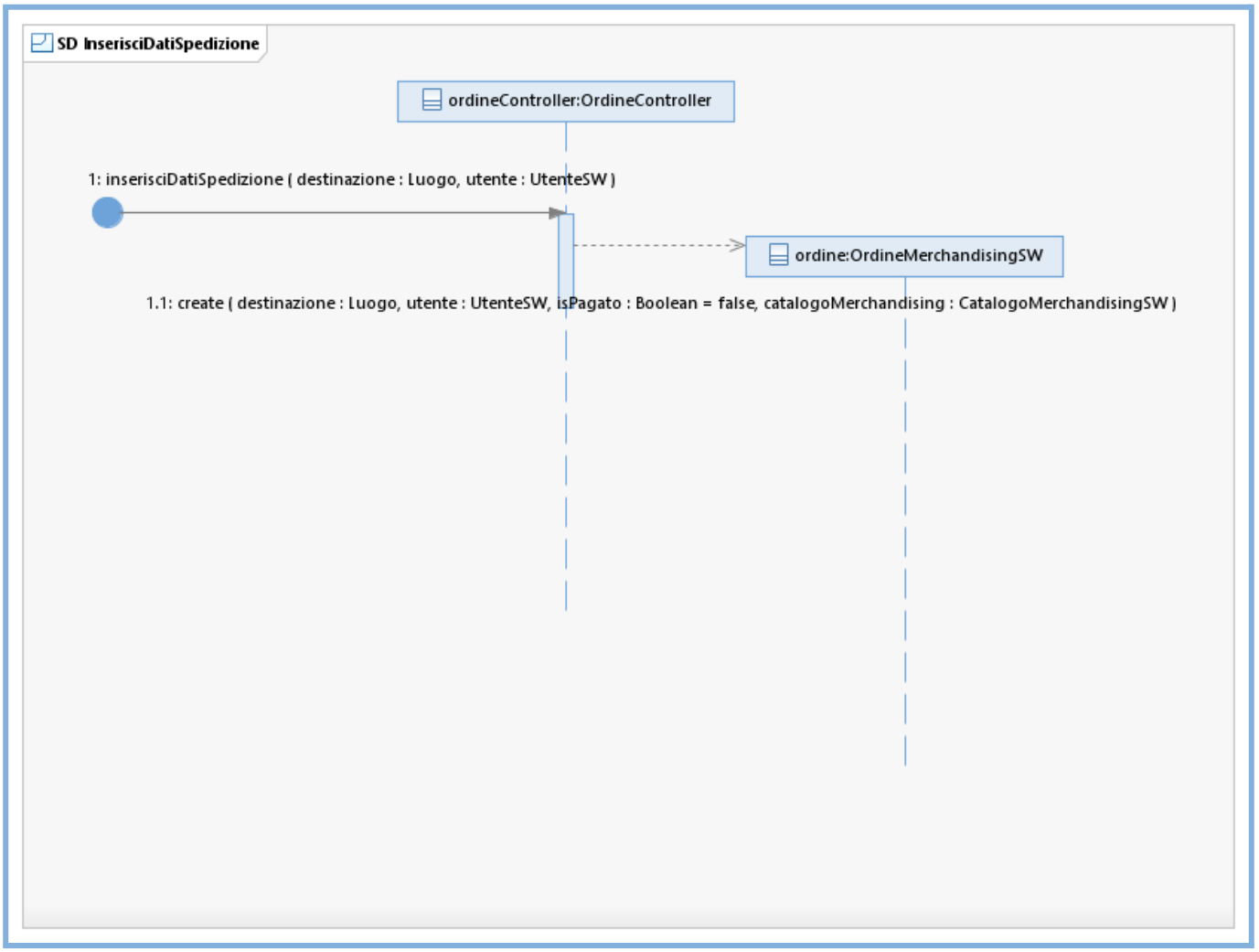
Brini Luca

CaricaAbbonamento



Karzal Youness

InserisciDatiSpedizione



Pattern

Grasp

Protected Variation	
Classi	**Applicazione**
TransazioneSW, GestionePagamenti	Sono le classi in cui sono previste delle variazioni o instabilità, sono quindi state progettate per correggere tali instabilità, evitando di costruire percorsi troppo complessi

Pure Fabrication	
Classi	**Applicazione**
ICatalogoSW, GestoreDB	Sono le classi introdotte per sostenere un'alta coesione e un basso accoppiamento, sono quindi classi artificiali che non sono riferite a concetti nel dominio.

Controller	
Classi	**Applicazione**
AbbonamentoController, OrdineController, PrenotazioneController, UtenteController, PagamentoController	Viene definito un controller specifico per ogni componente del sistema, per separare le responsabilità. Sono quindi degli oggetti che creano e mediano tra i vari componenti.

GoF

Singleton	
Classi	**Applicazione**
GestoreDB CatalogoEventiSW CatalogoMerchandising SW CatalogoServiziSW	Sono le classi per le quali è presente una sola istanza nel sistema.

In particolare, le classi in esame sono singleton perché:

Singleton	
Classi	**Applicazione**
GestoreDB	Non ha senso avere più gestori per un singolo database
CatalogoEventiSW	Esiste un solo catalogo degli eventi e una sola istanza relativa
CatalogoServiziSW	Esiste un solo catalogo dei servizi e una sola istanza relativa
CatalogoMerchandising SW	Esiste un solo catalogo del merchandising e una sola istanza relativa
GestorePagamenti	Esiste un solo gestore dei pagamenti, non ha senso avere più gestori anche per questioni di sicurezza (*vedi Protected Variation*)
GestoreCataloghi	Esiste un solo gestore dei cataloghi

Observer	
Classi	**Applicazione**
GestoreNotifiche	Il pattern Publish Subscribe viene considerato nella gestione delle notifiche agli eventi: solo se un utente si registra ad un evento, ne ottiene le notifiche.

Implementazione Java

```java
public interface ICatalogo<T> {

    public List<T> read(FiltroPrenotazioni filtro){

    }

    public void add(T record){

    }

    public void update (int id, T newRecord){

    }

    public void delete (int id){

    }
}
```

```java
public class CatalogoEventi implements ICatalogo<EventoSpeciale> {

    @Override
    public void add(EventoSpeciale record) {
        // Aggiungi un nuovo record nel catalogo
    }

    @Override
    public void delete(int id) {
        // Rimuovi un record dal catalogo

    }

    @Override
    public List<EventoSpeciale> read(FiltroPrenotazioni filtro) {
        // Ritorna
        return null;
    }

    @Override
    public void update(int id, EventoSpeciale newRecord) {
        // Aggiorna un record nel catalogo

    }

}
```

```java
public class GestoreCataloghi {
    GestoreCataloghi uniqueInstance;

    public GestoreCataloghi(){
        this.uniqueInstance = this;
    }

    public static void consultaCatalogoEventiSpeciali(FiltroPrenotazioni filtro){
        // Prendo l'instanza del catalogoEventi
        CatalogoEventi catalogoEventi = new CatalogoEventi();
        catalogoEventi.read(filtro);

        // Altre cose...
    }

    public static void consultaCatalogoServizi(FiltroPrenotazioni filtro){
        // Uguale a sopra
    }

    public static void consultaCatalogoProdotti(FiltroPrenotazioni filtro){
        // Uguale a sopra
    }

}
```

```java
public class EventoSpeciale {
    final private int id;
    final private CatalogoEventi evento;
    final private LocalDateTime dataPrenotazione;Ï
    final private ClienteFedele clienteFedele;

    public EventoSpeciale(int id, CatalogoEventi evento, LocalDateTime dataPrenotazione, ClienteFedele clienteFedele) {
        this.id = id;
        this.evento = evento;
        this.dataPrenotazione = dataPrenotazione;
        this.clienteFedele = clienteFedele;
    }

    public int getId() {
        return id;
    }

    public CatalogoEventi getEvento() {
        return evento;
    }

    public LocalDateTime getDataPrenotazione() {
        return dataPrenotazione;
    }

}
```

```java
public class ClienteVIP extends ClienteFedele {
    private Trainer trainer;

    public ClienteVIP(Trainer trainer){
        this.trainer = trainer;
    }

    public setTrainer(Trainer trainer){
        this.trainer = trainer;
    }

}
```

```java
public class Utente {
    private String nome;
    private String cognome;
    private int id;

    public Utente(String nome, String cognome, int id) {
        this.nome = nome;
        this.cognome = cognome;
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public String getCognome() {
        return cognome;
    }

    public int getId() {
        return id;
    }

}
```

```java
public class Trainer extends Utente{

}
```

```java
public enum TipologiaAbbonamento {
    TRIMESTRALE,
    SEMESTRALE,
    ANNUALE,
    BIENNALE
}
```

```java
public enum TipologiaCliente {
    CLIENTE,
    CLIENTE_FEDELE,
    CLIENTE_VIP
}
```

```java
import java.time.LocalDateTime;

public class FiltroPrenotazioni {
    final public LocalDateTime dataInizio;
    final public LocalDateTime dataFine;
    final public TipologiaAbbonamento tipologiaAbbonamento;
    final public TipologiaCliente tipologiaCliente;

    public FiltroPrenotazioni(
            LocalDateTime dataInizio,
            LocalDateTime dataFine,
            TipologiaAbbonamento tipologiaAbbonamento,
            TipologiaCliente tipologiaCliente) {
        this.dataInizio = dataInizio;
        this.dataFine = dataFine;
        this.tipologiaAbbonamento = tipologiaAbbonamento;
        this.tipologiaCliente = tipologiaCliente;
    }

}
```

Modifiche e Aggiunte

Diagramma dei Casi d'Uso

Aggiunti i seguenti casi d'uso:
- Visualizza Storico Prenotazioni
- Produci Grafico
- Esporta Dati
- Richiedi Trainer Personale
- Rinnova Abbonamento
- Iscrizione Evento Speciale
- Acquista Prodotto Merchandising
- Gestisci Prodotti
- Gestisci Eventi Speciali

Aggiunti i seguenti attori:
- Cliente VIP
- Ente Bancario
- Fornitore Prodotti

Note:
- Cliente VIP eredita da Cliente Fedele il caso d'uso "**Iscrizione Evento Speciale**", il quale include il caso d'uso "**Effettua Pagamento**". In questo specifico caso, il pagamento sarà di 0 euro

Modello di Dominio

Aggiunte le seguenti classi:
- Cliente VIP
- Catalogo
- Catalogo Eventi
- Catalogo Merchandising
- Evento Speciale
- Ordine Merchandising
- Abbonamento

Rinominata la seguente classe:
- Catalogo in **Catalogo Servizi**

Associazioni:
- Aggiunto collegamento tra **Trainer** e **Cliente VIP**
- Aggiunto collegamento tra **Cliente Fedele** ed **Evento Speciale**
- Aggiunto collegamento tra **Abbonamento** e **Tessera**
- Aggiunti collegamenti tra **Catalogo Merchandising**, **Catalogo Eventi** e **Catalogo Servizi** rispettivamente a **Ordine Merchandising**, **Ordine Evento Speciale** e **Servizio**

Generalizzazioni:
- La classe **Cliente Fedele** generalizza la classe **Cliente VIP**
- La classe **Catalogo** generalizza le classi **Catalogo Merchandising**, **Catalogo Eventi** e **Catalogo Servizi**

Attributi:
- Aggiunti attributi sconto, tipologiaAbbonamento, scadenza ad **Abbonamento**
- Aggiunto attributi nome, cognome, mail, password ad **Utente**
- Aggiunto attributo ultimoAccesso a Cliente, per gestire il passaggio tra le varie tipologie di clienti
- Aggiunti attributi prezzo, taglia, quantità a **Catalogo Merchandising**
- Aggiunti attributi data e luogo a **Catalogo Eventi**
- Aggiunti attributi id, titolo e descrizione a **Catalogo**
- Aggiunti attributi destinazione, dataAcquisto e sconto a **Ordini Merchandising**
- Aggiunto attributo dataPrenotazione a **Servizio**
- Aggiunto attributo id a **Tessera**
- Aggiunti attributi id e dataPrenotazione a **Evento Speciale**

Note:
- Si suppone che un Cliente VIP faccia un salto da Cliente a Cliente VIP passando per Cliente Fedele, quindi ne eredita tutti i vantaggi e caratteristiche. Se un Cliente VIP non rinnova l'abbonamento VIP, torna ad essere Cliente se non soddisfa i requisiti per essere Fedele
- Aggiunto **Abbonamento Controller** per gestire gli abbonamenti
- il **Pagamento** è considerato come una classe che gestisce le informazioni del pagamento, il controller gestisce la transazione in sé. Il pagamento è inserito nel package di dominio. La transazione è quella effettuata per il pagamento
- il sistema del fornitore viene notificato dell'ordine dei prodotti

Diagramma di sequenza di sistema

Note:

- SSD **EffettuaPagamento** utilizzato per la gestione dei pagamenti negli altri SSD che prevedono un pagamento
- l'attore Cliente VIP, essendo anche un Cliente Fedele, eredita l'SSD **IscrizioneEventoSpeciale**
- nell'SSD **EffettuaPagamento**, il flag **isPagato**, che indica un pagamento avvenuto con successo, viene impostato a True

Architettura Logica

Note:

- Correzione del bug di importazione della consegna

Diagramma delle Classi di Progetto

Riportiamo unicamente delle osservazioni principali, propedeutiche alla comprensione del nostro progetto.

Note:

- le classi **TransazioneSW** del package **Dominio**, **GestorePagamenti** del package **Servizio**, **PagamentoController** del package **Controller** hanno funzionalità complementari: un'istanza di TransazioneSW rappresenta i dati e lo stato di una transazione, GestorePagamenti è una classe software che si appoggia ad un ente esterno che fornisce i servizi di pagamento tramite Secure Rest API. Infine, PagamentoController funge da controller e governa l'interfacciamento con il GestorePagamenti
- **ICatalogoSW** funge da interfaccia CRUD alle classi **CatalogoServiziSW**, **CatalogoMerchandisingSW** e **CatalogoEventiSW**
- **TipologiaAbbonamento** e **TipologiaCliente** sono due enumerazioni
- **FiltroPrenotazioni** e **Luogo** sono due data type
- **EventoSpecialeSW** rappresenta l'istanza di una prenotazione per un evento speciale, disponibile nel **CatalogoEventiSW**. Analogo il ragionamento per **ServiziSW** e **OrdineMerchandisingSW**