

Progetto: Fitness Ready

Sommario

[**1 – Ideazione ed analisi dei requisiti** 6](#_Toc49640463)

[1.1 – Requisiti 6](#_Toc49640464)

[1.2 – Obiettivi e casi d’uso 7](#_Toc49640465)

[1.3 – Modello dei casi d’uso 8](#_Toc49640466)

[**UC1: Registrazione Cliente** 8](#_Toc49640467)

[**UC2: Login Utente** 9](#_Toc49640468)

[**UC3: Effettua Acquisto** 10](#_Toc49640469)

[**UC4: Visualizza Acquisti** 11](#_Toc49640470)

[**UC5: Modifica Ordine** 12](#_Toc49640471)

[**UC6: Modifica Dati** 12](#_Toc49640472)

[**UC7: Visualizza Scheda** 13](#_Toc49640473)

[**UC8: Esegui Esercizi** 14](#_Toc49640474)

[**UC9: Gestisci Prenotazioni, CRUD** 15](#_Toc49640475)

[**UC10: Gestisci Prodotti, CRUD** 16](#_Toc49640476)

[**UC11: Gestisci Schede, CRUD** 17](#_Toc49640477)

[**UC12: Gestisci Promozioni, CRUD** 18](#_Toc49640478)

[1.4 – Regole di business 19](#_Toc49640479)

[**2 – Documento di Visione** 19](#_Toc49640480)

[2.1 – Introduzione 19](#_Toc49640481)

[2.2 – Scopo 19](#_Toc49640482)

[2.3 – Portata 20](#_Toc49640483)

[2.4 – Posizionamento 20](#_Toc49640484)

[2.4.1 – Opportunità di business 20](#_Toc49640485)

[2.4.2 – Formulazione del problema 20](#_Toc49640486)

[2.4.3 – Formulazione della posizione del prodotto 21](#_Toc49640487)

[2.5 – Parti interessate e descrizione utenti 21](#_Toc49640488)

[2.5.1 – Riepilogo delle parti interessate 21](#_Toc49640489)

[2.5.2 – Riepilogo degli utenti 21](#_Toc49640490)

[2.6 – Descrizione generale del prodotto 22](#_Toc49640491)

[2.6.1 – Introduzione 22](#_Toc49640492)

[2.6.2 – Descrizione breve funzioni cliente 22](#_Toc49640493)

[2.6.3 – Descrizione breve funzioni amministratore 23](#_Toc49640494)

[**3 – Dettagli sviluppo applicazione** 23](#_Toc49640495)

[3.1 – Riepilogo iterazioni 23](#_Toc49640496)

[3.1.1 – Iterazione 1 24](#_Toc49640497)

[3.1.2 – Iterazione 2 24](#_Toc49640498)

[3.1.3 – Iterazione 3 25](#_Toc49640499)

[3.2 – Modello di Dominio 26](#_Toc49640500)

[**4 – SSD e Contratti delle Operazioni** 27](#_Toc49640501)

[4.1 – SSD e Contratti UC1: Registrazione Cliente 28](#_Toc49640502)

[**4.1.1 – CO1: registraCliente** 28](#_Toc49640503)

[**4.1.2 – CO2: assegnaScheda** 28](#_Toc49640504)

[4.2 – SSD e Contratti UC2: Login Utente 29](#_Toc49640505)

[**4.2.1 – CO1: effettuaLogin** 29](#_Toc49640506)

[4.3 – SSD e Contratti UC3: Effettua Acquisto 30](#_Toc49640507)

[**4.3.1 – CO1: aggiungiAlCarrello** 30](#_Toc49640508)

[**4.3.2 – CO2: effettuaAcquisto** 30](#_Toc49640509)

[**4.3.3 – CO3: aggiungiAcquisto** 31](#_Toc49640510)

[**4.3.4 – CO4: effettuaOrdine** 31](#_Toc49640511)

[**4.3.5 – CO5: aggiungiOrdine** 32](#_Toc49640512)

[**4.3.6 – CO6: aggiornaCatalogo** 32](#_Toc49640513)

[**4.3.7 – CO7: buyAndUpdate** 32](#_Toc49640514)

[4.4 – SSD e Contratti UC4: Visualizza Acquisti 33](#_Toc49640515)

[**4.4.1 – CO1: visualizzaAcquisti** 33](#_Toc49640516)

[4.5 – SSD e Contratti UC5: Modifica Ordine 34](#_Toc49640517)

[**4.5.1 – CO1: modificaOrdine** 34](#_Toc49640518)

[4.6 – SSD e Contratti UC6: Modifica Dati 35](#_Toc49640519)

[**4.6.1 – CO1: modificaDati** 35](#_Toc49640520)

[**4.6.2 – CO2: assegnaScheda** 36](#_Toc49640521)

[4.7 – SSD e Contratti UC7: Visualizza Scheda 36](#_Toc49640522)

[**4.7.1 – CO1: getSchedaCliente** 36](#_Toc49640523)

[4.8 – SSD e Contratti UC8: Esegui Esercizi 37](#_Toc49640524)

[**4.8.1 – CO1: eseguiEsercizio** 37](#_Toc49640525)

[4.9 – SSD e Contratti UC9: Gestisci Prenotazioni, CRUD 38](#_Toc49640526)

[**4.9.1 – CO1: effettuaPrenotazione** 38](#_Toc49640527)

[**4.9.2 – CO1.1: getPrenotazione** 38](#_Toc49640528)

[**4.9.3 – CO1.2: modificaPrenotazione** 39](#_Toc49640529)

[**4.9.3 – CO1.3: eliminaPrenotazione** 39](#_Toc49640530)

[4.10 – SSD e Contratti UC10: Gestisci Prodotti, CRUD 40](#_Toc49640531)

[**4.10.1 – CO1: aggiungiProdotto** 40](#_Toc49640532)

[**4.10.2 – CO1.1: modificaProdotto** 40](#_Toc49640533)

[**4.10.3 – CO1.2: eliminaProdotto** 41](#_Toc49640534)

[4.11 – SSD e Contratti UC11: Gestisci Schede, CRUD 41](#_Toc49640535)

[**4.11.1 – CO1: aggiungiScheda** 41](#_Toc49640536)

[**4.11.2 – CO1.1: getScheda** 42](#_Toc49640537)

[**4.11.3 – CO1.2: modificaScheda** 42](#_Toc49640538)

[**4.11.4 – CO1.3: eliminaScheda** 43](#_Toc49640539)

[4.12 – SSD e Contratti UC12: Gestisci Promozioni, CRUD 43](#_Toc49640540)

[**4.12.1 – CO1: aggiungiPromozione** 43](#_Toc49640541)

[**4.12.2 – CO1.1: modificaDatiPromozione** 44](#_Toc49640542)

[**4.12.3 – CO1.2: cambiaStatoPromozione** 44](#_Toc49640543)

[**4.12.4 – CO1.3: eliminaPromozione** 44](#_Toc49640544)

[**5 – Progettazione Fitness Ready** 45](#_Toc49640545)

[5.1 – Diagramma delle classi di progetto 45](#_Toc49640546)

[5.2 – Diagrammi di sequenza di progetto 46](#_Toc49640547)

[5.2.1 – UC1 46](#_Toc49640548)

[5.2.2 – UC2 48](#_Toc49640549)

[5.2.3 – UC3 48](#_Toc49640550)

[5.2.4 – UC4 53](#_Toc49640551)

[5.2.5 – UC5 54](#_Toc49640552)

[5.2.6 – UC6 54](#_Toc49640553)

[5.2.7 – UC7 55](#_Toc49640554)

[5.2.8 – UC8 55](#_Toc49640555)

[5.2.9 – UC9 56](#_Toc49640556)

[5.2.10 – UC10 58](#_Toc49640557)

[5.2.11 – UC11 60](#_Toc49640558)

[5.2.12 – UC12 62](#_Toc49640559)

[5.3 – Design Pattern utilizzati 65](#_Toc49640560)

[**6 – Testing** 66](#_Toc49640561)

[6.1 – File AuthenticationTest 66](#_Toc49640562)

[**6.1.1 – Login Test** 67](#_Toc49640563)

[**6.1.2 – Sing-In Test** 67](#_Toc49640564)

[6.2 – File PurchaseTest 68](#_Toc49640565)

[**6.2.1 – Add-To-Cart Test** 68](#_Toc49640566)

[**6.2.2 – Execute-Purchase Test** 68](#_Toc49640567)

[6.3 – File BookingTest 69](#_Toc49640568)

[**6.3.1 – Execute-Booking Test** 69](#_Toc49640569)

[**7 – Glossario** 69](#_Toc49640570)

# **1 – Ideazione ed analisi dei requisiti**

## – Requisiti

Una società che si occupa della vendita di alimenti ipocalorici/proteici ed attrezzature per il fitness, per invogliare i potenziali clienti ad acquistare i propri prodotti, richiede la realizzazione di un’applicazione software che, riferendosi ai dati inseriti in fase di registrazione, oltre a fornire uno shop online per la vendita dei suddetti prodotti, assegni automaticamente ai clienti un piano d’esercizi ed una dieta alimentare sotto forma di una cosiddetta “scheda fitness” da seguire, in base all’obiettivo che essi intendano raggiungere (definizione muscolare o aumento della massa muscolare).

Tale applicazione dovrà essere gestita da un apposito addetto della suddetta società, avente le adeguate conoscenze fisico-alimentari, a cui saranno date delle credenziali che gli garantiranno i privilegi d’amministratore con cui gestire i servizi offerti ai clienti. Inoltre, l’applicazione richiederà periodicamente al cliente registrato il reinserimento dei dati riguardanti le sue misure fisiche in modo tale che il sistema effettui tutti i controlli necessari e fornisca un eventuale nuova scheda fitness da seguire per l’obiettivo prescelto.

Nello specifico, tra le funzioni dell’applicazione, si ha che:

* L’amministratore deve poter inserire, controllare, ed eventualmente modificare (es. in caso d’errore) i prodotti venduti nello shop.
* L’amministratore deve poter inserire nuove schede fitness per i clienti oltre che, eventualmente, modificarle od eliminarle.
* L’amministratore deve poter creare e gestire promozioni sui prodotti presenti a suo piacimento.
* Il cliente deve poter visualizzare l’elenco di tutti prodotti presenti nello shop e ricercare quelli di suo interesse per l’acquisto.
* Il cliente deve poter prenotare un prodotto desiderato qualora non fosse momentaneamente disponibile.
* Il cliente può decidere, in qualsiasi momento, di modificare i suoi dati personali con conseguente assegnazione di una nuova scheda di esercizi e nuova dieta.
* Il cliente deve poter visualizzare la propria scheda fitness in qualsiasi momento.
* Il cliente, una volta effettuato un acquisto, deve poter modificare (es. in caso di errato inserimento) i dati relativi all’ordine d’acquisto effettuato.
* Il cliente può ottenere dei gettoni virtuali utilizzabili all’interno della stessa applicazione che verranno attribuiti dal sistema in diversi casi (*vedi paragrafo 1.4 – Regole di business*).

## – Obiettivi e casi d’uso

Tramite i requisiti precedentemente elencati è possibile individuare l’attore principale che eseguirà le varie operazioni attraverso il sistema e gli obiettivi che egli intende portare a termine tramite il suo utilizzo, da cui si ricavano i seguenti relativi casi d’uso principali:

|  |  |  |
| --- | --- | --- |
| Attore | Obiettivo | Caso d’uso |
| Cliente | Registrazione del cliente all’interno del sistema | UC1: Registrazione Cliente |
| Utente (Cliente / Amministratore) | Accesso del generico utente al sistema per usufruire dei servizi forniti dal sistema stesso | UC2: Login Utente |
| Cliente | Acquisto di un prodotto disponibile nell’elenco visualizzabile nella sezione apposita con annesso calcolo di sconti e promozioni (qualora ci siano) | UC3: Effettua Acquisto |
| Cliente | Visualizzazione degli acquisti effettuati contenti il dettaglio su data ed ora di esecuzione, sui prodotti presi e sul totale pagato | UC4: Visualizza Acquisti |
| Cliente | Modifica dei dati di spedizione relativi ad uno specifico ordine tra quelli effettuati | UC5: Modifica Ordine |
| Cliente | Modifica dei dati inseriti dal cliente per ottenere una nuova scheda fitness più consona alle sue esigenze | UC6: Modifica Dati |
| Cliente | Visualizzare le informazioni relative alla propria scheda fitness in qualsiasi momento per evitare il dimenticarsi dei suggerimenti dati dal sistema | UC7: Visualizza Scheda |
| Cliente | Verificare l’esecuzione degli esercizi presenti nella scheda fitness da parte del cliente che la possiede | UC8: Esegui Esercizi |
| Cliente | Gestione delle prenotazioni del cliente (creazione, visione, aggiornamento, rimozione di una determinata prenotazione) | UC9: Gestisci Prenotazioni  (CRUD) |
| Amministratore | Inserimento, rimozione, ricerca o modifica di un prodotto nell’elenco | UC10: Gestisci Prodotti (CRUD) |
| Amministratore | Inserimento, rimozione, ricerca o modifica di una scheda fitness | UC11: Gestisci Schede (CRUD) |
| Amministratore | Creare, ricercare, modificare o rimuovere una promozione | UC12: Gestisci Promozioni (CRUD) |

## – Modello dei casi d’uso

### **UC1: Registrazione Cliente**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC1: Registrazione Cliente |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole effettuare la registrazione al sistema software Fitness Ready. * **SISTEMA**: vuole tutte le informazioni necessarie relative al nuovo cliente da registrare. |
| **Precondizioni** | Il cliente non deve essere presente nella memoria del sistema. |
| **Garanzia di successo** | Corretta memorizzazione del cliente all’interno del sistema. |
| **Scenario principale di successo** | 1. Il cliente richiede al sistema di poter effettuare la registrazione tramite l’opzione appositamente predisposta. 2. Il sistema richiede al cliente la compilazione degli appositi campi (nome, cognome, e-mail, password, altezza, peso, ecc.). 3. Il cliente inserisce i dati e da conferma. 4. Il sistema effettua un controllo sul corretto inserimento dei dati utente, sull’e-mail inserita, per vedere se esiste già una registrazione a suo carico, ed un confronto tra i campi “Password” e “Conferma Password” appena compilati. 5. Il sistema notifica al cliente l’avvenuta registrazione mostrando un messaggio a schermo ed assegna automaticamente una scheda fitness e 5 gettoni omaggio al suo profilo. |
| **Estensioni** | 1. Il sistema rileva che l’e-mail inserita è già presente in memoria.    1. Il sistema notifica al cliente che l’e-mail inserita è già in uso e si predispone per una nuova registrazione.    2. Il cliente inserisce una nuova e-mail per effettuare la registrazione.    3. Il sistema effettua nuovamente gli appositi controlli. 2. Il sistema rileva che i campi “Password” e “Conferma Password” non combaciano.    1. Il sistema notifica al cliente l’incongruenza tra i due campi e si predispone per una nuova registrazione.    2. Il cliente reinserisce i dati per la registrazione.    3. Il sistema effettua nuovamente gli appositi controlli. 3. Il sistema rileva che alcuni campi non sono stati compilati.    1. Il sistema notifica al cliente la presenza di campi vuoti e permette di riempirli.    2. Il cliente inserisce i dati mancanti.    3. Il sistema effettua nuovamente gli appositi controlli. |
| **Frequenza di ripetizioni** | L’operazione dev’essere eseguita una sola volta correttamente per ogni nuovo utente. |

### **UC2: Login Utente**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC2: Login Utente |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Utente (Cliente / Amministratore) |
| **Parti interessate** | * **UTENTE:** vuole effettuare l’accesso per usufruire dei servizi offerti dall’applicazione Fitness Ready. * **SISTEMA**: vuole tutti i dati necessari per riconoscere ed autenticare l’utente. |
| **Precondizioni** | L’utente deve essere già presente all’interno della memoria del sistema. |
| **Garanzia di successo** | Accesso completo ai servizi offerti dall’applicazione secondo i privilegi in possesso dell’utente che ha effettuato il login. |
| **Scenario principale di successo** | 1. L’utente richiede l’accesso al sistema ed i suoi servizi tramite il form di login predisposto dall’applicazione. 2. Il sistema richiede i dati dell’utente da lui inseriti in fase di registrazione (e-mail e password). 3. L’utente inserisce i propri dati. 4. Il sistema notifica al cliente l’avvenuto accesso tramite un messaggio di benvenuto, suggerendo di visualizzare per prima cosa la propria scheda [*vedi caso d’uso Visualizza Scheda*], e successivamente mostra la schermata principale. |
| **Estensioni** | 1. Il sistema rileva che i dati inseriti dall’utente sono errati.    1. Il sistema mostra all’utente un messaggio d’avvertimento e consente la correzione dei dati.    2. L’utente corregge i dati errati.    3. Il sistema effettua nuovamente gli appositi controlli. |
| **Frequenza di ripetizioni** | Ogni qual volta l’utente ha bisogno di accedere all’applicazione Fitness Ready. |

### **UC3: Effettua Acquisto**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC3: Effettua Acquisto |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole acquistare un prodotto presente nello shop. * **SISTEMA**: vuole rendere l’acquisto da parte del cliente semplice e veloce oltre che tenere traccia degli acquisti da lui effettuati. |
| **Precondizioni** | Il cliente ha effettuato il login al sistema ed in caso di sua momentanea indisponibilità avere la possibilità di prenotarlo. |
| **Garanzia di successo** | Il cliente è riuscito ad acquistare il/i prodotto/i desiderato mentre il sistema ha aggiornato le informazioni relative ai prodotti stessi. |
| **Scenario principale di successo** | 1. Il cliente sceglie l’opzione “Acquista Prodotto” nel menù “Acquista”. 2. Il sistema mostra l’elenco dei prodotti disponibili con le relative informazioni d’interesse per il cliente (nome, prezzo, disponibilità, ecc.). 3. Il cliente ricerca il prodotto desiderato, lo seleziona e preme il pulsante di aggiunta al carrello. 4. Il sistema chiede la quantità desiderata del prodotto scelto, effettua un confronto con la disponibilità del prodotto e, se disponibile, lo inserisce nel carrello. 5. I passi dal 2 al 4 vengono ripetuti finché il cliente non decide di procedere all’acquisto spostandosi sul carrello. 6. Il sistema riepiloga le informazioni dell’ordine, ne calcola il totale applicando in automatico eventuali promozioni, mostra il risultato al cliente e attende la conferma dell’acquisto. 7. Il cliente conferma l’acquisto tramite il pulsante “Acquista”. 8. Il sistema richiede i dati per il pagamento e la consegna del/dei prodotto/i (tra cui il metodo di pagamento, se con carta o con gettoni). 9. Il cliente inserisce i dati necessari a completare l’acquisto. 10. Il sistema effettua l’acquisto e crea un nuovo ordine con i dati inseriti, dopo di che aggiorna la lista acquisti del cliente e le disponibilità dei prodotti ed infine notifica che l’acquisto è andato a buon fine. |
| **Estensioni** | 1. In qualsiasi momento il cliente decide di annullare l’acquisto tramite il tasto “Annulla Acquisto” e ritorna alla schermata principale. 2. Il sistema rileva la momentanea indisponibilità del prodotto selezionato dal cliente.    1. Il sistema notifica al cliente l’indisponibilità del prodotto e ne impedisce l’inserimento nel carrello.    2. Il sistema consente al cliente di prenotare il prodotto [*vedi caso d’uso Gestisci Prenotazioni*]. 3. Il cliente decide di rimuovere un prodotto presente nel carrello selezionandolo nella tabella di riepilogo ed usando il tasto “Rimuovi Prodotto”. 4. Il sistema rimuove il prodotto selezionato e ricalcola il totale dovuto per l’acquisto. 5. Il cliente decide di modificare la quantità di un prodotto nel carrello selezionandolo nella tabella di riepilogo ed usando il tasto “Modifica Quantità”. 6. Il sistema richiede la nuova quantità desiderata per il prodotto selezionato, effettua un confronto con la sua disponibilità e, se disponibile, effettua la modifica per poi ricalcolare il totale dovuto per l’acquisto. 7. Il sistema rileva un errore nei dati inseriti, annulla l’acquisto e notifica l’errore al cliente |
| **Frequenza di ripetizioni** | Ogni qual volta il cliente desidera acquistare uno o più prodotti. |

### **UC4: Visualizza Acquisti**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC4: Visualizza Acquisti |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole visualizzare il proprio storico acquisti con i vari dettagli ad essi legati. * **SISTEMA**: vuole mostrare gli acquisti effettuati relativi al cliente che ha effettuato il login. |
| **Precondizioni** | Il cliente deve aver effettuato almeno un acquisto [*vedi caso d’uso Effettua Acquisto*]. |
| **Garanzia di successo** | Stampa di tutti gli acquisti eseguiti dal cliente. |
| **Scenario principale di successo** | 1. Il cliente sceglie l’opzione “Visualizza Acquisti” nel menù “Acquista”. 2. Il sistema recupera e stampa a video l’elenco di tutti gli acquisti effettuati dal cliente comprensivi di dettagli sui singoli prodotti acquistati, totale pagato e data ed ora di esecuzione. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente vuole visualizzare i suoi acquisti. |

### **UC5: Modifica Ordine**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC5: Modifica Ordine |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole modificare i dati di spedizione di un ordine effettuato in precedenza. * **SISTEMA**: vuole tutti i dati necessari per poter effettuare la modifica dell’ordine indicato dal cliente. |
| **Precondizioni** | Il cliente deve aver effettuato almeno un acquisto o una prenotazione [*vedi caso d’uso Effettua Acquisto/Gestisci Prenotazioni*]. |
| **Garanzia di successo** | Modifica dei dati di spedizione dell’ordine scelto dal cliente. |
| **Scenario principale di successo** | 1. Il cliente utilizza il tasto “Modifica Ordine” presente nella sezione “Ordini”. 2. Il sistema richiede l’inserimento del numero dell’ordine che il cliente vuole modificare e dei nuovi dati di spedizione. 3. Il cliente inserisce i dati richiesti dal sistema. 4. Il sistema effettua il cambio di dati e notifica al cliente che la modifica è andata a buon fine. |
| **Estensioni** | 1. Il sistema rileva un errore nei dati inseriti dal cliente.    1. Il sistema mostra al cliente un messaggio d’avvertimento ed annulla la modifica.    2. Il cliente utilizza nuovamente il tasto “Modifica Ordine” presente nella sezione “Ordini”.    3. Il sistema richiede nuovamente i dati da modificare.    4. Il cliente inserisce i dati corretti. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente vuole modificare i dati di uno specifico ordine. |

### **UC6: Modifica Dati**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC6: Modifica Dati |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole modificare i propri dati personali per eventualmente farsi assegnare una nuova scheda fitness dal sistema. * **SISTEMA**: vuole tutti i dati necessari per poter effettuare la modifica dei dati specificati dal cliente. |
| **Precondizioni** | Il cliente dev’essere registrato al sistema Fitness Ready ed aver effettuato l’accesso [*vedi casi d’uso Registrazione Cliente e Login Utente*]. |
| **Garanzia di successo** | Modifica dei dati profilo del cliente attualmente loggato. |
| **Scenario principale di successo** | 1. Il cliente seleziona l’opzione “Dati Profilo” del menù “Profilo”. 2. Il sistema mostra tutti i dati inseriti dal cliente al momento della registrazione [*vedi caso d’uso Registra Cliente*]. 3. Il cliente modifica i campi dei dati che intende aggiornare e preme il tasto “Modifica Dati”. 4. Il sistema effettua i dovuti controlli sui dati inseriti, effettua la modifica, assegna eventualmente una nuova scheda fitness al cliente e lo notifica al cliente. |
| **Estensioni** | 1. Il sistema rileva un errore nei dati inseriti dal cliente.    1. Il sistema mostra al cliente un messaggio d’avvertimento ed annulla la modifica.    2. Il cliente corregge gli errori ed utilizza nuovamente il tasto “Modifica Dati”.    3. Il sistema effettua nuovamente i dovuti controlli ed effettua la modifica. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente intende modificare i propri dati e farsi assegnare una nuova scheda fitness. |

### **UC7: Visualizza Scheda**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC7: Visualizza Scheda |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole visualizzare i dettagli relativi alla scheda fitness che gli è stata assegnata dal sistema. * **SISTEMA**: vuole mostrare tutti i dettagli relativi alla scheda fitness in possesso del cliente. |
| **Precondizioni** | Il cliente dev’essere registrato al sistema Fitness Ready ed aver effettuato l’accesso [*vedi casi d’uso Registrazione Cliente e Login Utente*]. |
| **Garanzia di successo** | Visualizzazione di dieta e piano esercizi della scheda fitness assegnata al cliente. |
| **Scenario principale di successo** | 1. Il cliente seleziona l’opzione “Visualizza Scheda” del menù “Profilo”. 2. Il sistema mostra i dettagli relativi alla scheda fitness, assegnata al cliente in fase di registrazione [*vedi caso d’uso Registra Cliente*] o in seguito alla modifica dei propri dati personali [*vedi caso d’uso Modifica Dati*], divisi per dieta ed esercizi da svolgere. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente desidera vedere la propria scheda fitness. |

### **UC8: Esegui Esercizi**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC8: Esegui Esercizi |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole eseguire il piano esercizi assegnatogli dal sistema Fitness Ready. * **SISTEMA**: vuole verificare l’esecuzione del piano esercizi in possesso del cliente. |
| **Precondizioni** | Il cliente dev’essere registrato al sistema Fitness Ready ed aver effettuato l’accesso [*vedi casi d’uso Registrazione Cliente e Login Utente*]. |
| **Garanzia di successo** | Gli esercizi assegnati al cliente vengono contrassegnati come eseguiti. |
| **Scenario principale di successo** | 1. Il cliente preme il tasto “Esegui Esercizi” appositamente predisposto nella pagina di visualizzazione della propria scheda fitness raggiunta tramite l’opzione “Visualizza Scheda” del menù “Profilo”. 2. Il sistema mostra una schermata avente un timer ed i relativi tasti per il suo corretto funzionamento, ovvero “Start”, “Stop”, “Reset”, “Next”, “Close”, assieme ad una piccola spiegazione degli stessi. 3. Il cliente preme il tasto “Start” per avviare il timer ed eseguire l’esercizio. 4. Il sistema avvia il timer che si fermerà una volta raggiunto il tempo pari alla durata dell’esercizio corrente. 5. Il cliente, allo scadere del timer, preme sul tasto “Next” per proseguire con il prossimo esercizio della lista. 6. I passi dal 3 al 5 vengono ripetuti fino all’esecuzione di tutti gli esercizi assegnati al cliente. 7. Il sistema notifica al cliente il completamento del piano esercizi, aggiorna lo stato degli esercizi della lista come “ESEGUITO” e chiude la schermata contente il timer. |
| **Estensioni** | 1. Il cliente in qualsiasi momento decide d’interrompere l’esecuzione degli esercizi tramite il tasto “Close”. 2. Il cliente decide di fare una pausa durante l’esercizio premendo il tasto “Stop”. 3. Il cliente decide di ricominciare l’esercizio tramite il tasto “Reset”. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente desidera eseguire il piano esercizi assegnatogli. |

### **UC9: Gestisci Prenotazioni, CRUD**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC8: Gestisci Prenotazioni, CRUD |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Cliente |
| **Parti interessate** | * **CLIENTE**: vuole gestire (creare, visualizzare, aggiornare e/o eliminare) le proprie prenotazioni. * **SISTEMA**: vuole rendere la procedura di gestione delle prenotazioni semplice e veloce per il cliente. |
| **Precondizioni** | Il cliente dev’essere registrato al sistema Fitness Ready ed aver effettuato l’accesso [*vedi casi d’uso Registrazione Cliente e Login Utente*]. |
| **Garanzia di successo** | Corretta gestione delle prenotazioni relative al cliente che ha effettuato il login al sistema Fitness Ready. |
| **Scenario principale di successo** | 1. Il cliente seleziona un prodotto al momento non disponibile (disponibilità pari a 0) nella sezione “Acquista Prodotto” del menù “Acquista” per poi premere sul tasto “Prenota” appositamente predisposto. 2. Il sistema richiede l’inserimento dei dati necessari per la prenotazione (quantità, indirizzo, metodo di pagamento, ecc.…) e, una volta inseriti, recupera i dati del prodotto selezionato ed effettua la prenotazione notificandola al cliente. |
| **Estensioni** | 1. Il cliente vuole visualizzare i dettagli di una prenotazione da lui effettuata, quindi seleziona la prenotazione desiderata all’interno della tabella presente nella sezione “Prenotazioni” e preme il tasto “Dettagli Prenotazione”.    1. Il sistema mostra i dettagli relativi alla prenotazione selezionata dal cliente. 2. Il cliente vuole modificare i dati di una prenotazione da lui effettuata, quindi seleziona la prenotazione desiderata all’interno della tabella presente nella sezione “Prenotazioni” e preme il tasto “Modifica Prenotazione”.    1. Il sistema richiede l’inserimento dei dati da modificare, una volta inseriti esegue l’eventuale rimborso crediti ed effettua la modifica notificandola al cliente. 3. Il cliente vuole eliminare una prenotazione da lui effettuata, quindi seleziona la prenotazione desiderata all’interno della tabella presente nella sezione “Prenotazioni” e preme il tasto “Elimina Prenotazione”.    1. Il sistema chiede al cliente conferma, una volta ottenuta esegue il rimborso crediti ed effettua l’eliminazione notificandola al cliente. |
| **Frequenza di ripetizioni** | Ogni volta che il cliente desidera gestire le sue prenotazioni. |

### **UC10: Gestisci Prodotti, CRUD**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC10: Gestisci Prodotti, CRUD |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Amministratore |
| **Parti interessate** | * **AMMINISTRATORE**: vuole gestire (creare, visualizzare, aggiornare e/o eliminare) i prodotti presenti nel catalogo. * **SISTEMA**: vuole rendere la procedura di gestione dei prodotti semplice e veloce per l’amministratore. |
| **Precondizioni** | L’amministratore deve aver effettuato l’accesso al sistema Fitness Ready [*vedi casi d’uso Login Utente*]. |
| **Garanzia di successo** | Corretta gestione dei prodotti presenti nel catalogo del sistema. |
| **Scenario principale di successo** | 1. L’amministratore vuole aggiungere un nuovo prodotto al catalogo e preme sul tasto “Aggiungi Prodotto” presente nella sezione “Prodotti”. 2. Il sistema richiede l’inserimento dei dati necessari alla creazione di un nuovo prodotto (nome, categoria, prezzo, ecc…), verifica la correttezza dei dati inseriti ed effettua l’aggiunta notificandola all’amministratore. |
| **Estensioni** | 1. L’amministratore vuole modificare un prodotto del catalogo, quindi lo seleziona dalla tabella presente nella sezione “Prodotti” e preme il tasto “Modifica Prodotto”.    1. Il sistema richiede l’inserimento dei dati da modificare, verifica la correttezza dei dati inseriti ed effettua la modifica notificandola all’amministratore. 2. L’amministratore vuole eliminare un prodotto del catalogo, quindi lo seleziona dalla tabella presente nella sezione “Prodotti” e preme il tasto “Elimina Prodotto”.    1. Il sistema chiede conferma dell’eliminazione del prodotto ed una volta ottenuta elimina lo stesso dal catalogo notificandolo all’amministratore. |
| **Frequenza di ripetizioni** | Ogni volta che l’amministratore desidera gestire i prodotti del catalogo. |

### **UC11: Gestisci Schede, CRUD**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC11: Gestisci Schede, CRUD |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Amministratore |
| **Parti interessate** | * **AMMINISTRATORE**: vuole gestire (creare, visualizzare, aggiornare e/o eliminare) le schede fitness presenti in memoria. * **SISTEMA**: vuole rendere la procedura di gestione delle schede fitness semplice e veloce per l’amministratore. |
| **Precondizioni** | L’amministratore deve aver effettuato l’accesso al sistema Fitness Ready [*vedi casi d’uso Login Utente*]. |
| **Garanzia di successo** | Corretta gestione delle schede fitness presenti in memoria. |
| **Scenario principale di successo** | 1. L’amministratore vuole aggiungere una nuova scheda fitness all’elenco e preme sul tasto “Aggiungi Scheda” presente nella sezione “Schede Fitness”. 2. Il sistema richiede l’inserimento dei dati necessari alla creazione di un nuovo prodotto (nome, categoria, prezzo, ecc…), verifica la correttezza dei dati inseriti ed effettua l’aggiunta notificandola all’amministratore. |
| **Estensioni** | 1. L’amministratore vuole visualizzare dieta e piano esercizi di una scheda presente in memoria, quindi la seleziona dalla tabella presente nella sezione “Schede Fitness” e preme il tasto “Visualizza Dieta/Esercizi Scheda”.    1. Il sistema mostra dieta e piano esercizi della scheda selezionata. 2. L’amministratore vuole modificare i dati relativi ad una scheda fitness presente in memoria, quindi la seleziona dalla tabella presente nella sezione “Schede Fitness” e preme il tasto “Modifica Scheda”.    1. Il sistema richiede l’inserimento dei dati da modificare, verifica la correttezza dei dati inseriti ed effettua la modifica notificandola all’amministratore. 3. L’amministratore vuole eliminare una scheda fitness tra quelle presenti in memoria, quindi la seleziona dalla tabella presente nella sezione “Schede Fitness” e preme il tasto “Elimina Scheda”.    1. Il sistema chiede conferma dell’eliminazione della scheda ed una volta ottenuta elimina la stessa dall’elenco delle schede in memoria e lo notifica all’amministratore. |
| **Frequenza di ripetizioni** | Ogni volta che l’amministratore desidera gestire le schede fitness presenti in memoria. |

### **UC12: Gestisci Promozioni, CRUD**

|  |  |
| --- | --- |
| **Nome del caso d’uso** | UC12: Gestisci Promozioni, CRUD |
| **Portata** | Applicazione Fitness Ready |
| **Livello** | Obiettivo Utente |
| **Attore primario** | Amministratore |
| **Parti interessate** | * **AMMINISTRATORE**: vuole gestire (creare, visualizzare, aggiornare e/o eliminare) le promozioni presenti in memoria. * **SISTEMA**: vuole rendere la procedura di gestione delle promozioni semplice e veloce per l’amministratore. |
| **Precondizioni** | L’amministratore deve aver effettuato l’accesso al sistema Fitness Ready [*vedi casi d’uso Login Utente*]. |
| **Garanzia di successo** | Corretta gestione delle promozioni presenti in memoria. |
| **Scenario principale di successo** | 1. L’amministratore vuole aggiungere una nuova promozione all’elenco e preme sul tasto “Aggiungi Promozione” presente nella sezione “Promozioni”. 2. Il sistema richiede l’inserimento dei dati necessari alla creazione di una nuova promozione (nome, categoria prodotti in sconto, percentuale di sconto), verifica la correttezza dei dati inseriti ed effettua l’aggiunta notificandola all’amministratore. |
| **Estensioni** | 1. L’amministratore vuole modificare i dati relativi ad una promozione presente in elenco, quindi la seleziona dalla tabella presente nella sezione “Promozioni” e preme il tasto “Modifica Dati Promozione”.    1. Il sistema richiede l’inserimento dei dati da modificare, verifica la correttezza dei dati inseriti ed effettua la modifica notificandola all’amministratore. 2. L’amministratore vuole attivare o disattivare una promozione presente in elenco, quindi la seleziona dalla tabella presente nella sezione “Promozioni” e preme il tasto “Attiva/Disattiva Promozione”.    1. Il sistema chiede conferma per l’attivazione o disattivazione di una promozione ed una volta ottenuta procede all’attivazione o disattivazione della stessa notificando l’esito all’amministratore. 3. L’amministratore vuole eliminare una promozione tra quelle presenti in elenco, quindi la seleziona dalla tabella presente nella sezione “Promozioni” e preme il tasto “Elimina Promozioni”.    1. Il sistema chiede conferma dell’eliminazione della promozione ed una volta ottenuta elimina la stessa dall’elenco delle promozioni notificandolo all’amministratore. |
| **Frequenza di ripetizioni** | Ogni volta che l’amministratore desidera gestire le promozioni presenti in elenco. |

## 1.4 – Regole di business

Di seguito viene presentato l’elenco delle regole utilizzate per gestire gli acquisti dei prodotti da parte dei clienti tramite l’uso dell’applicazione Fitness Ready:

* **R1**: al cliente verranno attribuiti dei gettoni spendibili all’interno dell’applicazione Fitness Ready nei seguenti casi:
  + 5 gettoni al momento della propria registrazione all’applicazione Fitness Ready.
  + 1 gettone per ogni ordine con totale spesa articoli superiore ai 5€.
  + 10 gettoni per ogni ordine con totale spesa articoli superiore ai 30€.
* **R2**: ad ogni acquisto di uno o più prodotti vengono aggiunti 10€ all’importo totale per coprire le spese di spedizione.
* **R3**: se il prezzo totale dei prodotti acquistati raggiunge o supera i 35€ la spedizione è gratuita.
* **R4**: il pagamento tramite gettoni comporta sempre spedizione gratuita.

# **2 – Documento di Visione**

## 2.1 – Introduzione

L’obiettivo di questo progetto è la creazione di un’applicazione, chiamata “Fitness Ready”, per promuovere la vendita degli articoli di una società producente integratori alimentari ed attrezzature per il fitness invogliando i potenziali clienti a migliorare il loro stato fisico tramite l’attribuzione automatica di una scheda fitness da parte della suddetta applicazione, fornendo loro uno shop online in cui acquistare l’occorrente ed allo stesso tempo che consenta ad uno o più amministratori autorizzati e competenti la gestione completa di tutti gli aspetti legati sia alle schede che ai prodotti in vendita.

## 2.2 – Scopo

Lo scopo del presente documento di visione è quello di raccogliere, analizzare e definire le caratteristiche, le esigenze ed i servizi offerti dell’applicazione “Fitness Ready”. Nel documento è possibile osservare le parti interessate all’applicazione e le loro necessità, tenendo presente che maggiori dettagli riguardo all’esaudimento di quest’ultime sono presenti nel Modello dei casi d’uso.

## 2.3 – Portata

Questo documento di visione è relativo al progetto “Fitness ready”, sviluppato e revisionato dallo studente Nuñez Pereira Mario Roberto. Il suddetto studente si occuperà di sviluppare una soluzione di tipo software per corrispondere alle richieste ed esigenze del committente tramite l’utilizzo di tecnologie multipiattaforma Java-oriented compatibili con i sistemi operativi maggiormente diffusi.

## 2.4 – Posizionamento

### 2.4.1 – Opportunità di business

L’applicazione che sarà sviluppata permetterà, alla società commissionante, lo sviluppo di aumentare i propri guadagni facendo pubblicità ai suoi prodotti ed ampliando il proprio numero di clienti, fornendo a questi ultimi un servizio di shop online con cui, seguendo i suggerimenti forniti dalla stessa applicazione, ottenere ciò che necessitano per migliorare il proprio fisico ordinandolo comodamente a casa ed a prezzi più convenienti. Tale applicazione, inoltre, consentirà ad un addetto, scelto dalla società come amministratore di sistema, di applicare promozioni sui prodotti inseriti.

### 2.4.2 – Formulazione del problema

|  |  |
| --- | --- |
| **Descrizione del problema** | In seguito alle indagini di mercato degli ultimi anni, si è potuto notare un notevole incremento della propensione dei clienti agli acquisti online a discapito della vendita al dettaglio all’interno dei negozi fisici di vari articoli, tra cui quelli venduti dalla società. |
| **Attori coinvolti** | Amministratore “Fitness Ready” |
| **Impatto** | Minore affluenza di clienti all’interno dei negozi con conseguente riduzione del numero di vendite e del guadagno complessivo della società |
| **Benefici in caso di una soluzione di successo** | Notevole ampliamento della clientela con relativo aumento dei guadagni ricavati dalle vendite, oltre ad una migliore e più diretta gestione delle vendite dei prodotti con conseguente riduzione degli errori, dei tempi di elaborazione e della complessità generale del processo di vendita |

### 2.4.3 – Formulazione della posizione del prodotto

|  |  |
| --- | --- |
| **Destinatari** | Il prodotto è destinato ad una società specializzata nella produzione e vendita di prodotti per il fitness |
| **Obiettivi** | La società committente ha la necessità di un’applicazione che incrementi le vendite dei suoi prodotti permettendo la gestione degli stessi ed invogli i clienti al loro acquisto in modo da massimizzare i guadagni |
| **Tipologia** | Prodotto software personalizzato |
| **Funzione** | “Fitness Ready” offre una soluzione software per la gestione delle vendite, prenotazioni, consegne e informazioni dei prodotti della società integrata ad un servizio di assistenza fisico-alimentare per i clienti |
| **Soluzioni alternative attuali** | Le attuali soluzioni alternative disponibili sul mercato non offrono uno spazio di vendita adeguato (poiché presenti numerosi altri venditori con conseguente elevata concorrenza) e dei servizi consoni alle richieste del committente |
| **Caratteristiche prodotto** | L’applicazione “Fitness Ready” garantisce la prontezza d’utilizzo ai clienti mentre offre alla società committente uno spazio di vendita interamente riservato ed aggiunge funzionalità di controllo che agevolano il lavoro di quest’ultimo |

## 2.5 – Parti interessate e descrizione utenti

### 2.5.1 – Riepilogo delle parti interessate

|  |  |  |
| --- | --- | --- |
| **Nome** | **Descrizione** | **Responsabilità** |
| Società | L’azienda che ha commissionato lo sviluppo dell’applicazione | Fornire quante più informazioni possibili sul prodotto desiderato per agevolare lo sviluppo dello stesso |
| Sviluppatore Software | Principale responsabile del corretto sviluppo dell’applicazione richiesta dal committente | Identificare, analizzare ed implementare i requisiti ottenuti dalle informazioni date dal committente |

### 2.5.2 – Riepilogo degli utenti

|  |  |  |
| --- | --- | --- |
| **Nome** | **Descrizione** | **Responsabilità** |
| Cliente | Utilizzatore dell’applicazione software | Corretto inserimento dei dati richiesti dall’applicazione |
| Amministratore | Utilizzatore dell’applicazione software con privilegi speciali | Gestione delle schede fitness e delle informazioni relative ai prodotti della società per la corretta vendita |

## 2.6 – Descrizione generale del prodotto

### 2.6.1 – Introduzione

L’applicazione Fitness Ready è pensata per poter essere trasposta ed utilizzata su qualsiasi dispositivo grazie alla portabilità derivante dall’utilizzo del linguaggio ad oggetti Java. Essa propone funzionalità diverse a seconda dell’utente che effettua il login. Tali funzioni sono rese di facile utilizzo grazie ad un’interfaccia grafica intuitiva, pensata per velocizzare le operazioni effettuate dall’utente che usa l’applicazione ed evitare così procedure rigide come quelle che si avrebbero con un’interfaccia esclusivamente testuale.

### 2.6.2 – Descrizione breve funzioni cliente

Il cliente che si registra all’applicazione Fitness Ready riceve, al momento stesso della registrazione, una scheda fitness avente una dieta ed un piano esercizi da seguire diversi a seconda dei dati da lui inseriti.

**Esempio Scheda Fitness**

**Nome Scheda**

|  |  |
| --- | --- |
| **DIETA** | **ESERCIZI** |
| COLAZIONE:  Portata 1  …  Portata N  PRANZO:  Portata 1  …  Portata N  CENA:  Portata 1  …  Portata N | CATEGORIA ESERCIZI:  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato  CATEGORIA ESERCIZI:  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato  CATEGORIA ESERCIZI:  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato  Esercizio, n° serie x n° ripetizioni --- stato |

Una volta effettuato il login, il cliente può:

* Visualizzare, ed eventualmente modificare, i propri dati e la scheda fitness che gli è stata assegnata dal sistema.
* Eseguire gli esercizi presenti nel suo piano esercizi, tramite l’apposita funzione che avvia un timer adibito al controllo dell’esecuzione degli stessi.
* Acquistare e/o prenotare prodotti con due diversi metodi di pagamento (carta di credito o gettoni), ricevendo inoltre notifiche sull’eventuale presenza di promozioni attualmente attive su di una specifica categoria di prodotti.
* Modificare i dati di un ordine nel caso di errato inserimento dei dati di spedizione.
* Gestire liberamente le proprie prenotazioni.

### 2.6.3 – Descrizione breve funzioni amministratore

L’amministratore presente nella memoria dell’applicazione Fitness Ready, una volta effettuato il login tramite le credenziali concessigli, ha accesso a tutte le funzioni per la gestione (ovvero creazione, visualizzazione, modifica e/o cancellazione) dei vari oggetti presi in considerazione dall’applicazione stessa. Nello specifico l’amministratore può:

* Gestire i prodotti venduti all’interno dell’applicazione.
* Gestire le schede fitness da assegnare e/o assegnate ai clienti.
* Gestire le promozioni riguardanti una specifica categoria di prodotti.

# **3 – Dettagli sviluppo applicazione**

## 3.1 – Riepilogo iterazioni

Lo sviluppo dell’applicazione Fitness Ready è stato svolto in modo iterativo, in modo da apportare delle modifiche graduali e ben controllate, ridurre gli errori di sviluppo, ed eventualmente permettere una più facile correzione di quest’ultimi. La realizzazione dell’applicazione è stata articolata in tre iterazioni principali che hanno permesso di coprire la totalità dei casi d’uso mostrati nel capitolo “*1 – Ideazione ed analisi dei requisiti*”. Di seguito vengono riportati i dettagli più significativi delle operazioni svolte durante le iterazioni.

### 3.1.1 – Iterazione 1

In questa iterazione vengono implementati, aggiornati e descritti dettagliatamente i casi d’uso *Registrazione Cliente*, *Login Utente*, *Effettua Acquisto* e *Visualizza Acquisti* (presenti nel modello dei casi d’uso del file “*Iterazione 1.docx*”).

Per quanto riguarda il caso d’uso *UC3: Effettua Acquisto*, alcune delle funzionalità descritte nel suo dettaglio, come l’inserimento dei dati necessari per la creazione dell’ordine (cioè indirizzo di spedizione, metodo di pagamento, ecc.) o l’aggiunta di gettoni al proprio profilo una volta effettuato l’acquisto, non sono ancora state implementate poiché ci si è concentrati sulla creazione ed il salvataggio del solo acquisto (con le relative informazioni) e sull’aggiornamento dei dati relativi ai prodotti una volta concluso quest’ultimo. Per questa stessa ragione non è stata ancora implementata nemmeno la funzione di assegnazione di una scheda fitness al cliente registrato al sistema. Tali funzionalità, considerate momentaneamente di secondaria importanza, verranno aggiunte in seguito nelle successive iterazioni.

Grazie ai test effettuati con il supporto delle librerie JUnit sui principali metodi elaborati in questa iterazione, per l’implementazione dei casi d’uso precedentemente indicati, è stato possibile correggere e migliorare il codice sorgente dell’applicazione ancora in via di sviluppo. Per poter effettuare i test sopracitati è stata introdotta un’apposita classe StartUp, non facente parte del dominio, che all’avvio dell’applicazione fornisce alcuni dati con cui lavorare (ulteriori dettagli riguardanti il testing effettuato in questa iterazione sono reperibili nel file “*Testing.docx*” all’interno della cartella “*2 - Iterazione 1*” di tale progetto).

### 3.1.2 – Iterazione 2

In questa iterazione è stata innanzitutto rivista e migliorata l’implementazione legata al caso d’uso *Effettua Acquisto* (oltre che il suo dettaglio nel modello dei casi d’uso) al fine di aumentare la coesione e ridurre l’accoppiamento tra le classi di progetto introducendo, inoltre, tutte le funzionalità legate al suddetto caso d’uso tralasciate durante l’iterazione 1 (il dettaglio sul funzionamento di tale implementazione è visionabile nei diagrammi di sequenza di progetto UC3 del file “*Iterazione 2.asta*”). Si è inoltre migliorata la correttezza logica del progetto spostando accuratamente il metodo aggiungiAcquisto(...) dalla classe “FitnessReady” a quella di “GestoreAcquisto” senza però alterare la logica del metodo stesso.

In aggiunta, in questa iterazione vengono implementati, aggiornati e descritti dettagliatamente i casi d’uso *Modifica Ordine*, *Modifica Dati*, *Visualizza Scheda* e *Gestisci Prenotazioni* oltre ad inserire, implementare e descrivere nel dettaglio un nuovo caso d’uso *Esegui Esercizio.*

Per quanto riguarda l’implementazione dei casi d’uso *UC5: Modifica Ordine*, *UC7: Visualizza Scheda*, *UC8: Esegui Esercizio*, *UC9: Gestisci Prenotazioni* è stata necessaria la creazione di nuove entità (tra cui Ordine, Prenotazione, Scheda, Dieta, PianoEsercizi, eccetera) e strutture per la loro gestione (ottenute tramite l’applicazione di determinati pattern) che durante l’iterazione 1 sono state temporaneamente tralasciate per arrivare ad un’applicazione parzialmente funzionante da mostrare alle parti interessate dalla stessa.

I test effettuati in questa iterazione sono stati per lo più (ma non interamente) eseguiti provando l’applicazione stessa praticamente anziché usufruire del supporto della libreria JUnit con apposite classi di test. Grazie al testing effettuato è stato possibile verificare e migliorare la correttezza delle funzioni implementati per i vari casi d’uso (ulteriori dettagli relativi ad i casi d’uso sopracitati, alla loro implementazione ed al testing sono reperibili nel modello dei casi d’uso del file “*Iterazione 2.docx*” e nei file “*Pattern utilizzati.docx*”, “*Contratti delle Operazioni.docx*”, “*Testing.docx*”, “*Iterazione 2.asta*” della cartella “*3 – Iterazione 2*” di tale progetto).

### 3.1.3 – Iterazione 3

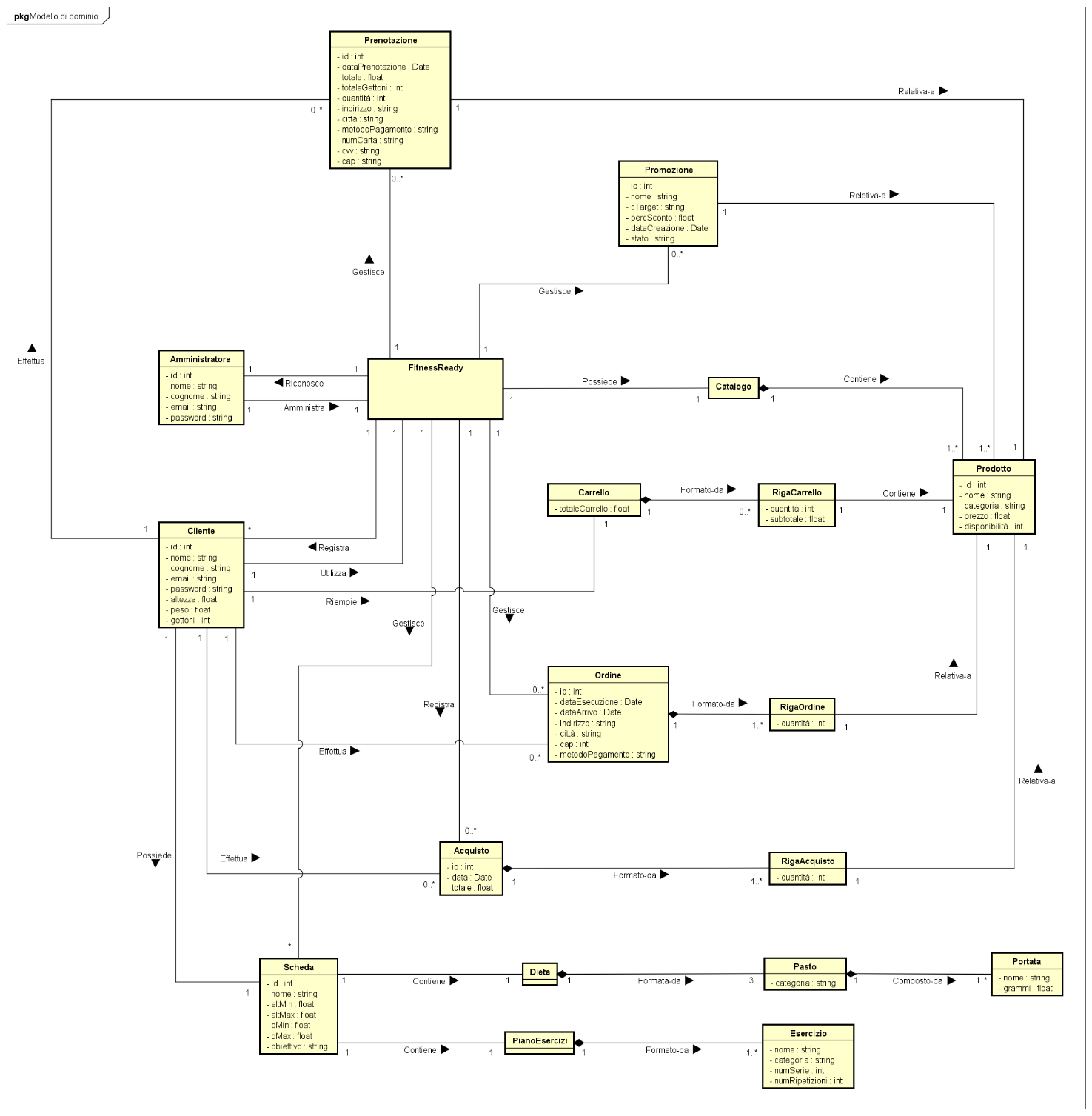
In questa iterazione vengono implementati, aggiornati e descritti dettagliatamente i casi d’uso *UC10: Gestisci Prodotti*, *UC11: Gestisci Schede* e *UC12: Gestisci Promozioni*; viene contemporaneamente eliminato il caso d’uso *UC13: Altera Stato Promozione* le cui caratteristiche sono state incorporate nel caso d’uso UC12 citato in precedenza.

L’implementazione del caso d’uso UC12 ha comportato l’introduzione dell’entità Promozione, tramite la quale è possibile calcolare gli eventuali sconti da effettuare in caso di acquisto di uno più prodotti da essa previsti. Per questo motivo il metodo effettuaAcquisto(…) del GestoreAcquisto viene aggiornato con l’aggiunta del calcolo dell’eventuale totale scontato da applicare in caso di acquisto di prodotti coperti da un’eventuale promozione attiva al momento del pagamento.

Il testing dell’applicazione in questa iterazione è stato completamente eseguito tramite l’effettiva esecuzione della stessa applicazione, poiché più rapido grazie all’interfaccia grafica predisposta con cui è stato possibile vedere nell’immediato i risultati delle varie operazioni.

## 3.2 – Modello di Dominio

Il modello di dominio è un diagramma che consente di visualizzare tutte le classi concettuali individuate durante l’analisi orientata agli oggetti (OOA) e lo sviluppo del progetto. Mediante il suddetto diagramma, che risulta essere un elaborato fondamentale per una buona OOA, vengono messi in evidenza tutti gli attributi delle classi concettuali, e le relazioni di maggior rilievo che si instaurano tra quest’ultime. Esattamente come l’applicazione ed i vari documenti, il modello di dominio è stato sviluppato in modo iterativo, motivo per cui di volta in volta sono state applicate ad esso tutte le novità o le modifiche previste dall’iterazione corrente. A seguire viene mostrato il modello di dominio ottenuto dopo alla conclusione dell’ultima iterazione.



# **4 – SSD e Contratti delle Operazioni**

In questo capitolo vengono mostrati tutti i diagrammi di sequenza di sistema (SSD) ed i contratti delle operazioni (i quali esplicitano, con un maggiore grado di dettaglio, i risultati ottenuti tramite le operazioni presenti nei diagrammi) ottenuti durante l’analisi orientata agli oggetti. Per facilitare la lettura di tali diagrammi e dei contratti si suddivide tale capitolo per i casi d’uso relativi ad essi.

## 4.1 – SSD e Contratti UC1: Registrazione Cliente

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.1.1 – CO1: registraCliente**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: registraCliente |
| **Operazione** | registraCliente(nome : String, cognome : String, email : String, password : String, confermaPassword : String, altezza : String, peso : String) |
| **Riferimenti** | Caso d’uso UC1: Registrazione Cliente |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata creata un’istanza *newCliente* di *Cliente* * Gli attributi di *newCliente* sono stati inizializzati tramite i dati inseriti dall’utente * *newCliente* è stato associato alla lista *listaUtenti* di *FitnessReady* |

### **4.1.2 – CO2: assegnaScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO2: assegnaScheda |
| **Operazione** | assegnaScheda(cliente : Cliente) |
| **Riferimenti** | Caso d’uso UC1: Registrazione Cliente |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata assegnata un’istanza *scheda* di *Scheda* all’istanza *cliente* di *Cliente* |

## 4.2 – SSD e Contratti UC2: Login Utente

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.2.1 – CO1: effettuaLogin**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: effettuaLogin |
| **Operazione** | effettuaLogin(email : String, password : String) |
| **Riferimenti** | Caso d’uso UC2: Login Utente |
| **Pre-condizioni** | * È presente in memoria un’istanza *Cliente* o *Amministratore* che richiede l’accesso al sistema |
| **Post-condizioni** | * È stata inizializzata un’istanza *utenteAttuale* di *Utente* con i relativi privilegi in base alle credenziali inserite dall’utente |

## 4.3 – SSD e Contratti UC3: Effettua Acquisto

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.3.1 – CO1: aggiungiAlCarrello**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: aggiungiAlCarrello |
| **Operazione** | aggiungiAlCarrello(quantity : int, price : float) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Il cliente deve essere autenticato * È presente un’istanza *carrello* di *Carrello* |
| **Post-condizioni** | * È stata creata un’istanza *riga* di *RigaCarrello* * Gli attributi di *riga* sono stati inizializzati tramite i dati sia ricavati dal prodotto selezionato che inseriti dal cliente * *riga* è stata associata alla lista *carrello.righeCarrello* |

### **4.3.2 – CO2: effettuaAcquisto**

|  |  |
| --- | --- |
| **Nome del contratto** | CO2: effettuaAcquisto |
| **Operazione** | effettuaAcquisto(metodoPagamento : String) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * È presente un’istanza *carrello* di *Carrello* avente una lista *righeCarrello* non vuota |
| **Post-condizioni** | * È stata creata un’istanza *newAcquisto* di *Acquisto* * È stata creata un’istanza *rigaAcquisto* di *RigaAcquisto* per ogni elemento di *carrello.righeCarrello* * Una *rigaAcquisto* è stata associata alla lista *newAcquisto.righeAcquisto* per ogni elemento della lista *carrello.righeCarrello* |

### **4.3.3 – CO3: aggiungiAcquisto**

|  |  |
| --- | --- |
| **Nome del contratto** | CO3: aggiungiAcquisto |
| **Operazione** | aggiungiAcquisto(newAcquisto : Acquisto, emailClienteAttuale : String, mappaAcquistiClienti : Map<String, List<Acquisto>>) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Dev’essere presente un’istanza *newAcquisto* di *Acquisto* |
| **Post-condizioni** | * *newAcquisto* è stato associato alla lista *mappaAcquistiClienti.get(emailClienteAttuale).listaAcquistiCliente* |

### **4.3.4 – CO4: effettuaOrdine**

|  |  |
| --- | --- |
| **Nome del contratto** | CO4: effettuaOrdine |
| **Operazione** | effettuaOrdine(newAcquisto : Acquisto, dataEsecuzione : Date, dataArrivo : Date, indirizzo : String, città : String, cap : String, metodoPagamento : String, numCarta : String, cvv : String) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Dev’essere presente un’istanza *newAcquisto* di *Acquisto* da cui derivare l’ordine |
| **Post-condizioni** | * È stata creata un’istanza *newOrdine* di *Ordine* * È stata creata un’istanza *rigaOrdine* di *RigaOrdine* per ogni elemento della lista *newAcquisto.righeAcquisto* * Una *rigaOrdine* è stata associata alla lista *newOrdine.righeOrdine* per ogni elemento della lista *newAcquisto.righeAcquisto* |

### **4.3.5 – CO5: aggiungiOrdine**

|  |  |
| --- | --- |
| **Nome del contratto** | CO5: aggiungiOrdine |
| **Operazione** | aggiungiOrdine(emailClienteAttuale : String, newOrdine : Ordine) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Dev’essere presente un’istanza n*ewOrdine* di *Ordine* |
| **Post-condizioni** | * *newOrdine* è stato associato alla lista *mappaOrdiniClienti.get(emailClienteAttuale).listaOrdiniCliente* |

### **4.3.6 – CO6: aggiornaCatalogo**

|  |  |
| --- | --- |
| **Nome del contratto** | CO6: aggiornaCatalogo |
| **Operazione** | aggiornaCatalogo(newAcquisto : Acquisto) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Dev’essere stata creata un’istanza *newAcquisto* di *Acquisto* |
| **Post-condizioni** | * Il parametro *disp* dei prodotti presenti nella lista *newAcquisto.righeAcquisto* viene aggiornato |

### **4.3.7 – CO7: buyAndUpdate**

|  |  |
| --- | --- |
| **Nome del contratto** | CO7: buyAndUpdate |
| **Operazione** | buyAndUpdate(dataEsecuzione : Date, dataArrivo : Date, indirizzo : String, città : String, metodoPagamento : String, numCarta : String, cvv : String) |
| **Riferimenti** | Caso d’uso UC3: Effettua Acquisto |
| **Pre-condizioni** | * Dev’essere stato effettuato un corretto inserimento dei dati richiesti per l’acquisto |
| **Post-condizioni** | * *clienteAttuale.gettoni* è diventato *clienteAttuale.gettoni –* *carrello.totaleGettoni* (se *metodoPagamento = “Gettoni”*) * *clienteAttuale.gettoni* è diventato *clienteAttuale.gettoni + 1* (se *metodoPagamento = “Carta di credito”* e *carrello.totaleCarrello* è compreso tra 5€ e 30€) * *clienteAttuale.gettoni* è diventato *clienteAttuale.gettoni + 10* (*se metodoPagamento = “Carta di credito”* e *carrello.totaleCarrello > 30*) * *newAcquisto.totale* è diventato *carrello.totaleCarrello + 10* (se *metodoPagamento = “Carta di credito”* e *carrello.totaleCarrello < 35*) * *mappaAcquistiClienti* è stata aggiornata * l’istanza *gestoreAcquisto* (relativo all’acquisto in corso) è stata cancellata |

## 4.4 – SSD e Contratti UC4: Visualizza Acquisti

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.4.1 – CO1: visualizzaAcquisti**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: visualizzaAcquisti |
| **Operazione** | visualizzaAcquisti() |
| **Riferimenti** | Caso d’uso UC4: Visualizza Acquisti |
| **Pre-condizioni** |  |
| **Post-condizioni** | * Viene creata un’istanza *stringa* di *String*, contente tutti gli acquisti del cliente |

## 4.5 – SSD e Contratti UC5: Modifica Ordine

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.5.1 – CO1: modificaOrdine**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: modificaOrdine |
| **Operazione** | modificaOrdine(idOrdine : int, emailClienteAttuale : String, indirizzo : String, città : String, cap : String) |
| **Riferimenti** | Caso d’uso UC5: Modifica Ordine |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *ordine* di *Ordine* in *mappaOrdiniClienti.get(emailClienteAttuale).listaOrdiniCliente* * Dev’essere stato effettuato un corretto inserimento dei dati richiesti |
| **Post-condizioni** | * *ordine.indirizzo* è diventato *indirizzo* * *ordine.città* è diventato *città* * *ordine.cap* è diventato *cap* |

## 4.6 – SSD e Contratti UC6: Modifica Dati

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.6.1 – CO1: modificaDati**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: modificaDati |
| **Operazione** | modificaDati(nome : String, cognome : String, email : String, password : String, altezza : String, peso : String, obiettivo : String) |
| **Riferimenti** | Caso d’uso UC6: Modifica Dati |
| **Pre-condizioni** | * Dev’essere presente in memoria un’istanza *clienteAttuale* di *Cliente* * Dev’essere stato effettuato un corretto inserimento dei dati richiesti |
| **Post-condizioni** | * *clienteAttuale.nome* è diventato *nome* * *clienteAttuale.cognome* è diventato *cognome* * *clienteAttuale.email* è diventato *email* * *clienteAttuale.password* è diventato *password* * *clienteAttuale.altezza* è diventato *altezza* * *clienteAttuale.peso* è diventato *peso* * *clienteAttuale.obiettivo* è diventato *obiettivo* |

### **4.6.2 – CO2: assegnaScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO2: assegnaScheda |
| **Operazione** | assegnaScheda(cliente : Cliente) |
| **Riferimenti** | Caso d’uso UC6: Modifica Dati |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata assegnata un’istanza *scheda* di *Scheda* all’istanza *cliente* di *Cliente* |

## 4.7 – SSD e Contratti UC7: Visualizza Scheda

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.7.1 – CO1: getSchedaCliente**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: getSchedaCliente |
| **Operazione** | getSchedaCliente(emailCliente : String) |
| **Riferimenti** | Caso d’uso UC7: Visualizza Scheda |
| **Pre-condizioni** | * Dev’essere presente in memoria un’istanza *clienteAttuale* di *Cliente* |
| **Post-condizioni** | * È stata recuperata un’istanza *scheda* di *Scheda* dalla mappa *mappaSchedeClienti* |

## 4.8 – SSD e Contratti UC8: Esegui Esercizi

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.8.1 – CO1: eseguiEsercizio**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: eseguiEsercizio |
| **Operazione** | eseguiEsercizio(i : int) |
| **Riferimenti** | Caso d’uso UC8: Esegui Esercizi |
| **Pre-condizioni** | * Dev’essere presente in memoria un’istanza *clienteAttuale* di *Cliente* |
| **Post-condizioni** | * È stata recuperata un’istanza *esercizio* di *Esercizio* * *esercizio.stato* è diventato *“ESEGUITO”* |

## 4.9 – SSD e Contratti UC9: Gestisci Prenotazioni, CRUD

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.9.1 – CO1: effettuaPrenotazione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: effettuaPrenotazione |
| **Operazione** | effettuaPrenotazione(cliente : Cliente, dataPrenotazione : Date, prod : Prodotto, quantità : int, indirizzo : String, città : String, cap : String, metodoPagamento : String, numCarta : String, cvv : String) |
| **Riferimenti** | Caso d’uso UC9: Gestisci Prenotazioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente in memoria un’istanza *clienteAttuale* di *Cliente* * Dev’essere presente un’istanza *prodotto* di *Prodotto* con *prodotto.disp* pari 0 |
| **Post-condizioni** | * È stata creata un’istanza *newPrenotazione* di *Prenotazione* * *newPrenotazione* è stata associata alla lista *mappaPrenotazioniClienti.listaPrenotazioniCliente* |

### **4.9.2 – CO1.1: getPrenotazione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.1: getPrenotazione |
| **Operazione** | getPrenotazione(emailCliente : String, idPrenotazione : int) |
| **Riferimenti** | Caso d’uso UC9: Gestisci Prenotazioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *prenotazione* di *Prenotazione* nella lista *mappaPrenotazioniClienti.listaPrenotazioniCliente* |
| **Post-condizioni** | * È stata recuperata un’istanza *prenotazione* di *Prenotazione*. |

### **4.9.3 – CO1.2: modificaPrenotazione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.2: modificaPrenotazione |
| **Operazione** | modificaPrenotazione(oldPrenotazione : Prenotazione, cliente : Cliente, quantità : int, indirizzo : String, città : String, cap : String, metodoPagamento : String, numCarta : String, cvv : String) |
| **Riferimenti** | Caso d’uso UC9: Gestisci Prenotazioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *prenotazione* di *Prenotazione* nella lista *mappaPrenotazioniClienti.listaPrenotazioniCliente* |
| **Post-condizioni** | * *prenotazione.quantità* è diventato *quantità* * *prenotazione.indirizzo* è diventato *indirizzo* * *prenotazione.città* è diventato *città* * *prenotazione.cap* è diventato *cap* * *prenotazione.metodoPagamento* è diventato *metodoPagamento* * *prenotazione.totale* è diventato *totale* * *prenotazione.totalGettoni* è diventato *totaleGettoni* * *prenotazione.numCarta* è diventato *numCarta* * *prenotazione.cvv* è diventiato *cvv* |

### **4.9.3 – CO1.3: eliminaPrenotazione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.3: eliminaPrenotazione |
| **Operazione** | eliminaPrenotazione(cliente : Cliente, idPrenotazione : int) |
| **Riferimenti** | Caso d’uso UC9: Gestisci Prenotazioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *prenotazione* di *Prenotazione* nella lista *mappaPrenotazioniClienti.listaPrenotazioniCliente* |
| **Post-condizioni** | * È stata cancellata un’istanza *prenotazione* di *Prenotazione* dalla lista *mappaPrenotazioniClienti.listaPrenotazioniCliente* |

## 4.10 – SSD e Contratti UC10: Gestisci Prodotti, CRUD

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.10.1 – CO1: aggiungiProdotto**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: aggiungiProdotto |
| **Operazione** | aggiungiProdotto(nome : String, categoria : String, prezzo : float, prezzoGettoni : int, disp : int) |
| **Riferimenti** | Caso d’uso UC10: Gestisci Prodotti, CRUD |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata creata un’istanza *newProdotto* di *Prodotto* * *newProdotto* è stata associata alla lista *catalogo.listaProdotti* |

### **4.10.2 – CO1.1: modificaProdotto**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.1: modificaProdotto |
| **Operazione** | modificaProdotto(idProdotto : String, nome : String, categoria : String, prezzo : float, prezzoGettoni : int, disp : int) |
| **Riferimenti** | Caso d’uso UC10: Gestisci Prodotti, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *prodotto* di *Prodotto* nella lista *catalogo.listaProdotti* |
| **Post-condizioni** | * *prodotto.nome* è diventato *nome* * *prodotto.categoria* è diventato *categoria* * *prodotto.prezzo* è diventato *prezzo* * *prodotto.prezzoGettoni* è diventato *prezzoGettoni* * *prodotto.disp* è diventato *disp* |

### **4.10.3 – CO1.2: eliminaProdotto**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.2: eliminaProdotto |
| **Operazione** | eliminaProdotto(idProd : int) |
| **Riferimenti** | Caso d’uso UC10: Gestisci Prodotti, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *prodotto* di *Prodotto* nella lista *catalogo.listaProdotti* |
| **Post-condizioni** | * È stata cancellata un’istanza *prenotazione* di *Prenotazione* dalla lista *catalogo.listaProdotti* |

## 4.11 – SSD e Contratti UC11: Gestisci Schede, CRUD

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.11.1 – CO1: aggiungiScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: aggiungiScheda |
| **Operazione** | aggiungiScheda(nome : String, altMin : float, altMax : float, pMin : float, pMax : float, obiettivo : String, dieta : Dieta, pianoEsercizi : PianoEsercizi) |
| **Riferimenti** | Caso d’uso UC11: Gestisci Schede, CRUD |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata creata un’istanza *newScheda* di *Scheda* * *newScheda* è stata associata alla lista *listaSchede* |

### **4.11.2 – CO1.1: getScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.1: getScheda |
| **Operazione** | getScheda(idScheda : int) |
| **Riferimenti** | Caso d’uso UC11: Gestisci Schede, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *scheda* di *Scheda* nella lista *listaSchede* |
| **Post-condizioni** | * È stata recuperata un’istanza *scheda* di *Scheda*. |

### **4.11.3 – CO1.2: modificaScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.2: modificaScheda |
| **Operazione** | modificaScheda(idScheda : int, nome : String, altMin : float, altMax : float, pMin : float, pMax : float, obiettivo : String, dieta : Dieta, pianoEsercizi : PianoEsercizi) |
| **Riferimenti** | Caso d’uso UC11: Gestisci Schede, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *scheda* di *Scheda* nella lista *listaSchede* |
| **Post-condizioni** | * *scheda.nome* è diventato *nome* * *scheda.altMin* è diventato *altMin* * *scheda.altMax* è diventato *altMax* * *scheda.pMin* è diventato *pMin* * *scheda.pMax* è diventato *pMax* * *scheda.obiettivo* è diventato *obiettivo* * *scheda.dieta* è diventato *dieta* * *scheda.pianoEsercizi* è diventato *pianoEsercizi* |

### **4.11.4 – CO1.3: eliminaScheda**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.3: eliminaScheda |
| **Operazione** | eliminaScheda(idScheda : int) |
| **Riferimenti** | Caso d’uso UC11: Gestisci Schede, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *scheda* di *Scheda* nella lista *listaSchede* |
| **Post-condizioni** | * È stata cancellata un’istanza *scheda* di *Scheda* dalla lista *listaSchede* |

## 4.12 – SSD e Contratti UC12: Gestisci Promozioni, CRUD

Immagine che contiene screenshot

Descrizione generata automaticamente

### **4.12.1 – CO1: aggiungiPromozione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1: aggiungiPromozione |
| **Operazione** | aggiungiPromozione (nome : String, cTarget : String, percSconto : float) |
| **Riferimenti** | Caso d’uso UC12: Gestisci Promozioni, CRUD |
| **Pre-condizioni** |  |
| **Post-condizioni** | * È stata creata un’istanza *newPromozione* di *Promozione* * *newPromozione* è stata associata alla lista *listaPromozioni* |

### **4.12.2 – CO1.1: modificaDatiPromozione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.1: modificaDatiPromozione |
| **Operazione** | modificaPromozione (idPromo : int, nome : String, cTarget : String, percSconto : float) |
| **Riferimenti** | Caso d’uso UC12: Gestisci Promozioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *promozione* di *Promozioni* nella lista *listaPromozioni* |
| **Post-condizioni** | * *promozione.nome* è diventato *nome* * *promozione.cTarget* è diventato *cTarget* * *promozione.percSconto* è diventato *percSconto* |

### **4.12.3 – CO1.2: cambiaStatoPromozione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.2: cambiaStatoPromozione |
| **Operazione** | cambiaStatoPromozione (idPromo : int) |
| **Riferimenti** | Caso d’uso UC12: Gestisci Promozioni, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *promozione* di *Promozioni* nella lista *listaPromozioni* |
| **Post-condizioni** | * *promozione.stato* è diventato *“ATTIVA”* o *“NON ATTIVA”* |

### **4.12.4 – CO1.3: eliminaPromozione**

|  |  |
| --- | --- |
| **Nome del contratto** | CO1.3: eliminaPromozione |
| **Operazione** | eliminaPromozione(idPromozione : int) |
| **Riferimenti** | Caso d’uso UC12: Gestisci Promozione, CRUD |
| **Pre-condizioni** | * Dev’essere presente almeno un’istanza *promozione* di *Promozione* nella lista *listaPromozioni* |
| **Post-condizioni** | * È stata cancellata un’istanza *promozione* di *Promozione* dalla lista *listaPromozioni* |

# **5 – Progettazione Fitness Ready**

La progettazione orientata agli oggetti (OOD) consente di definire e caratterizzare degli oggetti software, enfatizzando le interazioni fra essi per soddisfare tutti i requisiti individuati. La progettazione è di fondamentale importanza, in quanto tutte le classi individuate verranno poi implementate durante la fase di OOD. Proprio seguendo questa logica, al fine di mostrare i risultati ottenuti tramite questa fase, in questo capito vengono di seguito mostrati il “*diagramma delle classi di progetto*”, che permette di descrivere il tipo di oggetti facenti parte del sistema e le relazioni che vi sono tra essi, ed i “*diagrammi di sequenza di progetto*” (suddivisi per caso d’uso), che mostrano le interazioni tra oggetti facenti parte di uno stesso insieme ed al contempo il dettaglio del loro comportamento nel cosiddetto “formato a steccato”, ed infine una descrizione di quali “*Design Pattern*” e come essi siano stati utilizzati per conseguire il loro raggiungimento.

## 5.1 – Diagramma delle classi di progetto

Immagine che contiene screenshot

Descrizione generata automaticamente

Si invita a visionare il file Astah in allegato alla documentazione per apprezzare maggiormente i dettagli di tale diagramma.

## 5.2 – Diagrammi di sequenza di progetto

### 5.2.1 – UC1

#### 5.2.1.1 – registraCliente

Immagine che contiene screenshot

Descrizione generata automaticamente

**registraCliente**: tale operazione permette ad un cliente di registrarsi presso il sistema Fitness Ready, fornendo a quest’ultimo le informazioni da esso richieste tra cui un’e-mail che non sia già in uso da un’eventuale altro cliente.

#### 5.2.1.2 – assegnaScheda

Immagine che contiene screenshot

Descrizione generata automaticamente

**assegnaScheda**: questa funzione permette di assegnare una scheda fitness al cliente che si registra presso il sistema Fitness Ready a seconda dei dati da esso inseriti al momento della registrazione.

### 5.2.2 – UC2

#### 5.2.2.1 – effettuaLogin

Immagine che contiene screenshot

Descrizione generata automaticamente

**effettuaLogin**: La funzione in questione permette ad utente (sia esso cliente o amministratore) di avere accesso ai servizi a lui offerti dal sistema Fitness Ready, servizi che variano a seconda dei privilegi di cui gode l’utente stesso che effettua il login.

### 5.2.3 – UC3

#### 5.2.3.1 – aggiungiAlCarrello

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiAlCarrello**: questa funzione permette di aggiungere al proprio carrello un prodotto selezionato dalla tabella prodotti indicandone la quantità desiderata.

#### 5.2.3.2 – buyAndUpdate

Immagine che contiene testo, mappa

Descrizione generata automaticamente

**buyAndUpdate**: questa funzione esegue tutte le operazioni necessarie per eseguire un acquisto, creare un ordine ed aggiornare i dati sia del cliente che ha effettuato l’acquisto, sia dei prodotti che sono stati acquistati (tali operazioni sono illustrate e descritte di seguito all’interno di questa sezione).

#### 5.2.3.3 – effettuaAcquisto

Immagine che contiene testo, mappa

Descrizione generata automaticamente

**effettuaAcquisto**: questa funzione crea e restituisce effettivamente il nuovo acquisto una volta eseguito il pagamento.

#### 5.2.3.4 – aggiungiAcquisto

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiAcquisto**: tale funzione si occupa di aggiungere l’acquisto appena effettuato con i relativi dettagli alla lista di acquisti allegata al cliente che lo ha eseguito.

#### 5.2.3.5 – effettuaOrdine

Immagine che contiene screenshot

Descrizione generata automaticamente

**effettuaOrdine**: questa funzione crea e restituisce effettivamente il nuovo ordine relativo all’acquisto appena eseguito.

#### 5.2.3.6 – aggiungiOrdine

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiOrdine**: tale funzione si occupa di aggiungere l’ordine appena creato con i relativi dettagli alla lista degli ordini allegata al cliente che ha eseguito l’acquisto.

#### 5.2.3.7 – aggiornaCatalogo

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiornaCatalogo**: la funzione in questione si occupa di aggiornare le disponibilità dei prodotti che sono stati acquistati dal cliente al momento dell’esecuzione dell’acquisto stesso.

### 5.2.4 – UC4

#### 5.2.4.1 – visualizzaAcquisti

Immagine che contiene screenshot

Descrizione generata automaticamente

**visualizzaAcquisti**: tale operazione permette di ottenere la visualizzazione dell’intera lista acquisti del cliente che la richiede.

### 5.2.5 – UC5

#### 5.2.5.1 – modificaOrdine

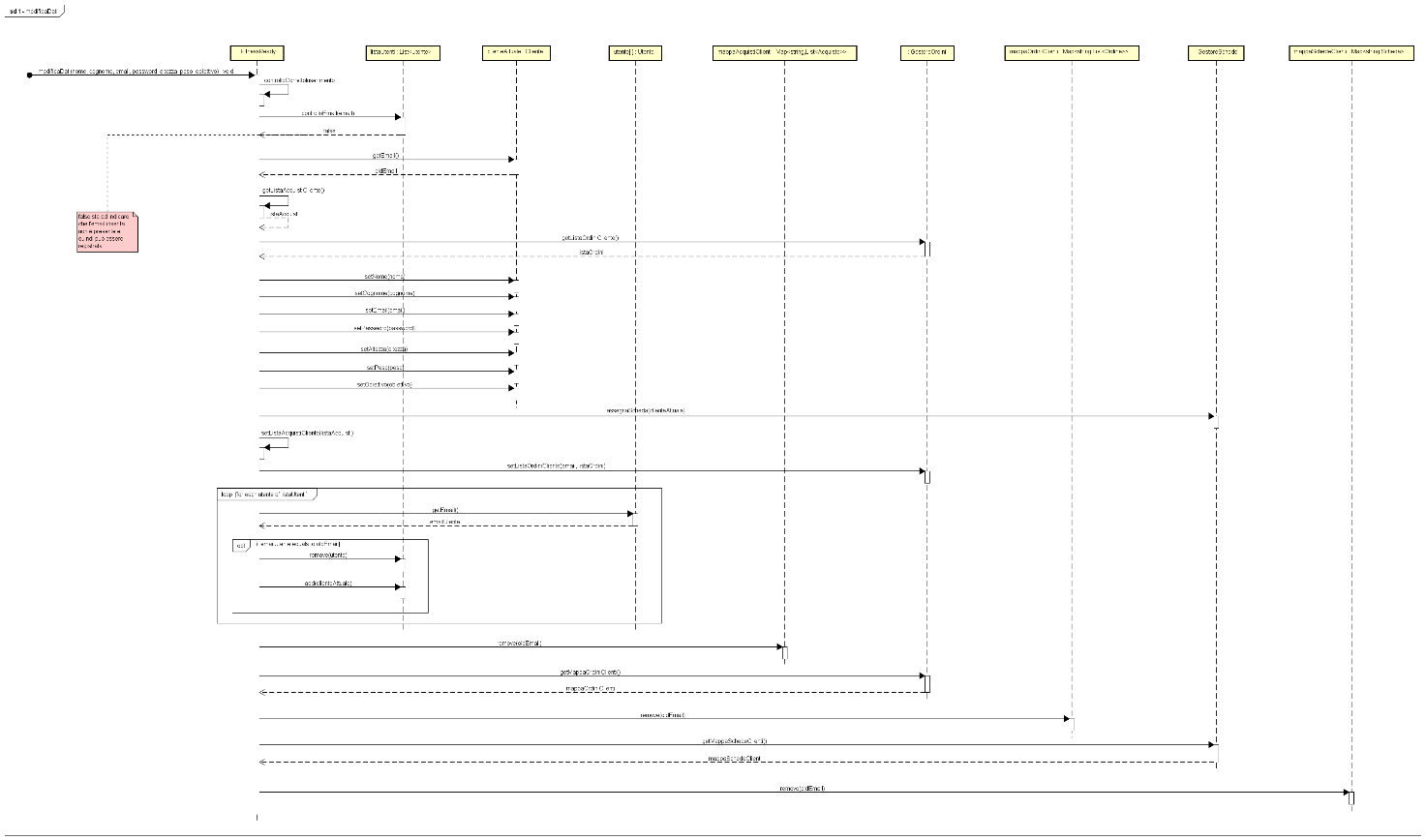
Immagine che contiene screenshot

Descrizione generata automaticamente

**modificaOrdine**: questa funzione permette di modificare gli eventuali errori d’inserimento dei dati di spedizione per uno specifico ordine selezionato.

### 5.2.6 – UC6

#### 5.2.6.1 – modificaDati



**modificaDati**: questa funzione permette al cliente di modificare i propri dati personali e, di conseguenza, farsi assegnare un’eventuale altra scheda fitness relativa ai nuovi dati inseriti.

#### 5.2.6.2 – assegnaScheda

In questo caso d’uso viene riutilizzata la funzione **assegnaScheda**,già descritta nella sezione 5.2.1 – UC1, per via del cambio di dati operato dalla funzione **modificaDati** precedentemente illustrata.

### 5.2.7 – UC7

#### 5.2.7.1 – getSchedaCliente

Immagine che contiene screenshot

Descrizione generata automaticamente

**getSchedaCliente**: tale funzione permette di recuperare la scheda fitness assegnata al cliente che ha effettuato il login per poterne visualizzare le informazioni associate.

### 5.2.8 – UC8

#### 5.2.8.1 – eseguiEsercizio

Immagine che contiene screenshot

Descrizione generata automaticamente

**eseguiEsercizio**: questa operazione consente di cambiare lo stato di un esercizio della scheda fitness del cliente loggato da “NON ESEGUITO” ad “ESEGUITO” e verificare così l’esecuzione degli esercizi da parte del cliente stesso.

### 5.2.9 – UC9

#### 5.2.9.1 – effettuaPrenotazione

Immagine che contiene screenshot

Descrizione generata automaticamente

**effettuaPrenotazione**: tale funzione si occupa di creare una prenotazione per un prodotto avente disponibilità pari a 0 ed allegarla al cliente che l’ha effettuata.

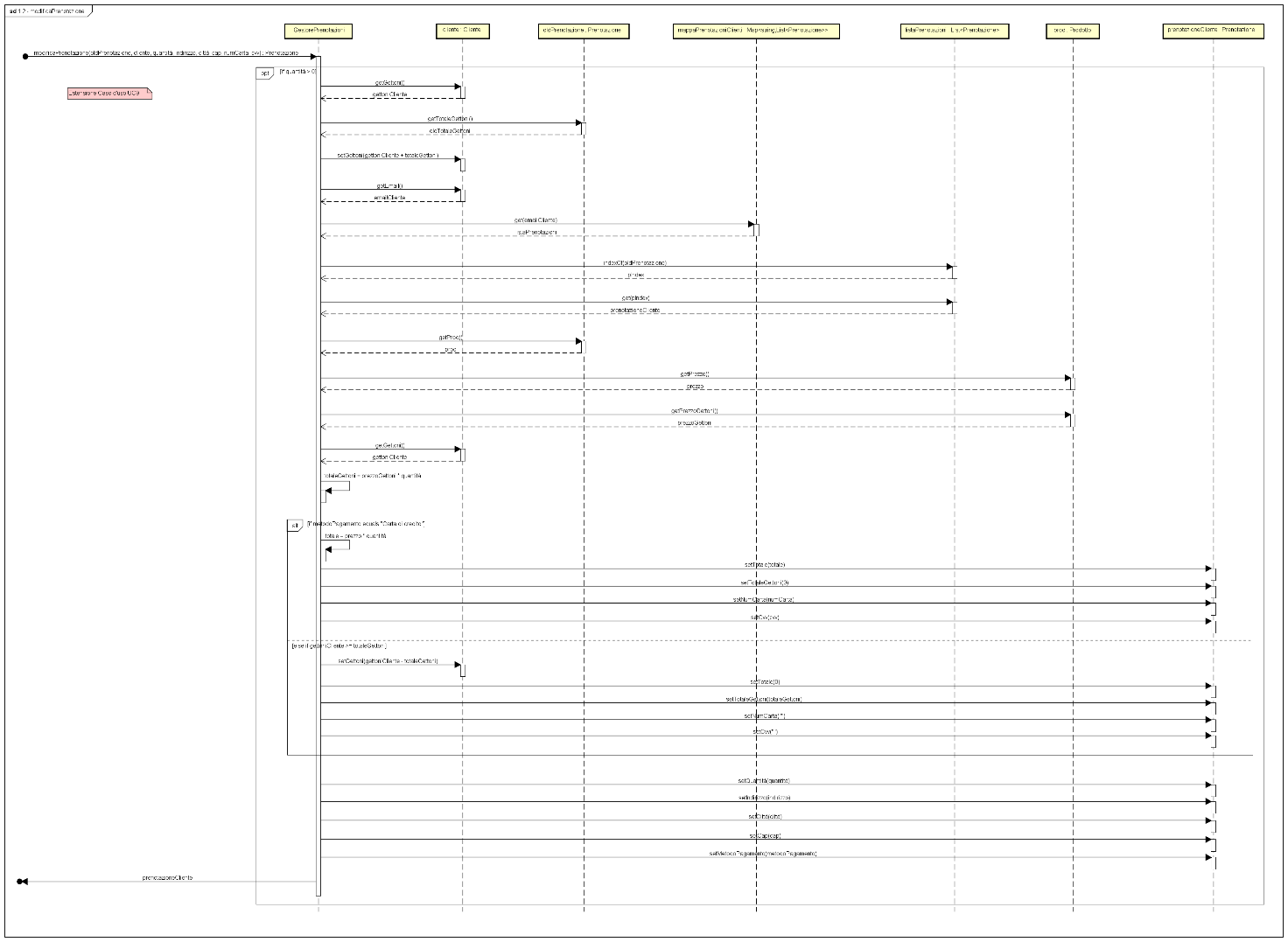
#### 5.2.9.2 – getPrenotazione

Immagine che contiene screenshot

Descrizione generata automaticamente

**getPrenotazione**: questa funzione permette di recuperare una prenotazione specificata dal cliente e visualizzarne i relativi dettagli.

#### 5.2.9.3 – modificaPrenotazione



**modificaPrenotazione**: quest’operazione consente di modificare tutti i dati relativi ad una prenotazione effettuata dal cliente, eccezion fatta per il prodotto prenotato dalla stessa.

#### 5.2.9.4 – eliminaPrenotazione

Immagine che contiene screenshot

Descrizione generata automaticamente

**eliminaPrenotazione**: tale funzione permette al cliente di eliminare una prenotazione dall’elenco prenotazioni da lui effettuate.

### 5.2.10 – UC10

#### 5.2.10.1 – aggiungiProdotto

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiProdotto**: la funzione qui mostrata consente all’amministratore la creazione ed aggiunta di un nuovo prodotto all’elenco in memoria al sistema Fitness Ready.

#### 5.2.10.2 – modificaProdotto

Immagine che contiene screenshot

Descrizione generata automaticamente

**modificaProdotto**: tale funzione consente all’amministratore la modifica dei dati relativi ad uno specifico prodotto tra quelli presenti in elenco.

#### 5.2.10.3 – eliminaProdotto

Immagine che contiene screenshot

Descrizione generata automaticamente

**eliminaProdotto**: questa operazione consente all’amministratore l’eliminazione di uno specifico prodotto selezionato dall’elenco dei prodotti presenti in memoria.

### 5.2.11 – UC11

#### 5.2.11.1 – aggiungiScheda

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiScheda**: tale funzione permette all’amministratore la creazione e l’aggiunta in elenco di una nuova scheda fitness tramite le informazioni richieste dal sistema.

#### 5.2.11.2 – getScheda

Immagine che contiene screenshot

Descrizione generata automaticamente

**getScheda**: questa funzione permette di recuperare una scheda fitness dall’elenco schede per poterne visualizzare i dettagli associati.

#### 5.2.11.3 - modificaScheda

Immagine che contiene screenshot

Descrizione generata automaticamente

**modificaScheda**: questa funzione consente la modifica dei dati relativi ad una scheda fitness selezionata dall’elenco delle schede.

#### 5.2.11.4 - eliminaScheda

Immagine che contiene screenshot

Descrizione generata automaticamente

**eliminaScheda**: tale funzione permette all’amministratore la cancellazione di una specifica scheda fitness selezionata dall’elenco schede in maniera permanente.

### 5.2.12 – UC12

#### 5.2.12.1 – aggiungiPromozione

Immagine che contiene screenshot

Descrizione generata automaticamente

**aggiungiPromozione**: questa operazione consente all’amministratore la creazione ed aggiunta di una nuova promozione in elenco tramite l’inserimento dei dati richiesti.

#### 5.2.12.2 – modificaDatiPromozione

Immagine che contiene screenshot

Descrizione generata automaticamente

**modificaDatiPromozione**: la seguente operazione permette all’amministratore la modifica dei generici dati relativi ad una specifica promozione selezionata.

#### 5.2.12.3 – cambiaStatoPromozione

Immagine che contiene screenshot

Descrizione generata automaticamente

**cambiaStatoPromozione**: questa operazione consente all’amministratore di attivare o disattivare una specifica promozione selezionata, in modo che un cliente in fase di acquisto goda di uno sconto relativo alla percentuale di sconto della promozione stessa se essa copre gli articoli da lui scelti.

#### 5.2.12.4 – eliminaPromozione

Immagine che contiene screenshot

Descrizione generata automaticamente

**eliminaPromozione**: questa funzione permette all’amministratore di cancellare permanentemente una specifica promozione selezionata dall’elenco promozioni.

## 5.3 – Design Pattern utilizzati

I Design Pattern possono essere definiti come una soluzione progettuale ad un problema ricorrente. Si tratta di modelli logici, ideati da ingegneri con maggiore esperienza nell’ambito progettuale, che descrivono un generico problema che capita spesso assieme ad una soluzione consigliata e comprovata dello stesso. Di seguito vengono elencati i pattern principalmente utilizzati per la progettazione dell’applicazione Fitness Ready e l’utilizzo che se n’è fatto:

* **Creator**: è stato utilizzato questo “*Pattern GRASP di base*” nell’operazione registraCliente(…) dell’UC1 per indicare che lo stesso sistema Fitness Ready crea un’istanza di Cliente.
* **Information Expert**: questo “*Pattern GRASP di base*” è stato utilizzato in tutti i casi d’uso per individuare le classi aventi le informazioni necessarie a svolgere le differenti operazioni e di conseguenza assegnargli le dovute responsabilità, come ad esempio nell’operazione registraCliente(…) dell’UC1 in cui il sistema Fitness Ready è in possesso delle informazioni per registrare un nuovo cliente ed aggiungerlo alla lista degli utenti.
* **Controller**: questo “*Pattern GRASP di base*” è stato utilizzato per specificare che il sistema Fitness Ready, ovvero il *Facade Controller*, è l’entità avente l’incarico di coordinare in maniera opportuna tutte le informazioni provenienti dalla UI.
* **Polymorphism**: questo “*Pattern GRASP*” è stato utilizzato per ottenere l’accesso il corretto accesso alle funzioni del sistema Fitness Ready in base al tipo di utente (Cliente o Amministratore) che effettua il login tramite l’operazione effettuaLogin(…).
* **Singleton**: questo “*Pattern GoF Creazionale*” è stato utilizzato per creare un'unica istanza del sistema Fitness Ready e fornire un accesso globale a tale unica istanza.
* **Composite:** questo “*Pattern GoF Strutturale*” è stato utilizzato per creare la composizione di una dieta ed un piano esercizi da associare alle schede fitness in modo da consentire il trattamento dei suddetti come composti da: diversi pasti (colazione, pranzo, cena) avente ognuno le proprie portate per quanto riguarda la dieta; diversi tipi di esercizi (cardio, gambe, braccia, torso, ecc…) per quanto riguarda il piano esercizi.
* **Observer:** questo “*Pattern GoF Comportamentale*” è stato utilizzato per fare in modo che, alla creazione di una prenotazione (ovvero il nostro “Osservatore”), il sistema crei un nuovo acquisto ed un nuovo ordine, quando la disponibilità del prodotto relativo alla prenotazione effettuata (ovvero il nostro “Osservato”) venga aggiornato e sia pari o superiore alla quantità prenotata dal cliente, e contemporaneamente cancelli la prenotazione stessa.
* **Iterator**: questo “*Pattern GoF Comportamentale*” è stato utilizzato per ottenere un accesso sequenziale controllato alle varie liste utilizzate nell’implementazione dei casi d’uso UC9, UC10, UC11 ed UC12 per l’eliminazione sicura di uno specifico elemento dalle stesse, evitando così l’insorgere di errori riguardanti l’accesso multiplo alle suddette da parte di più entità che le coinvolgono.

# **6 – Testing**

La fase di test, nel ciclo di sviluppo, riveste un’importanza rilevante ai fini della correttezza del prodotto finale, consentendo di valutare il livello di qualità raggiunto, evidenziando gli errori e permettendone la loro correzione. A tale scopo infatti sono stati effettuati dei test su alcune delle funzioni più rilevanti per lo sviluppo dell’applicazione Fitness Ready, i cui dettagli sono riportati qui di seguito.

## 6.1 – File AuthenticationTest

In tale file si è testato il corretto funzionamento delle funzioni di login e registrazione presenti nella classe “FitnessReady”. Prima di eseguire i test di seguito descritti viene eseguita una funzione di setUp che genera una lista di utenti di prova necessari per la verifica dei test e con cui si recupera l’unica istanza presente del sistema FitnessReady.

### **6.1.1 – Login Test**

In questo test è stato visionato il corretto funzionamento del metodo effettuaLogin(…) del sistema FitnessReady, la quale prende in ingresso l’email e la password dell’utente, sia esso Cliente o Amministratore, e restituisce i privilegi dell’utente loggatosi da cui derivano le funzioni che esso potrà effettuare grazie al suddetto sistema. Poiché il metodo effettuaLogin(…) esegue dei controlli per determinare che i parametri passatigli siano corretti essa lancia diversi tipi di eccezioni (appositamente create per le diverse situazioni), si è quindi deciso di basare il test sia sul lancio di tali eccezioni verificandole tramite la funzione assertThrows(), con cui si confrontano l’eccezione attesa con quella effettivamente ricevuta, sia sul corretto login dei due utenti (un Cliente ed un Amministratore) tramite la funzione assertEquals(), con cui si confrontano i privilegi attesi con quelli recuperati.

### **6.1.2 – Sing-In Test**

In questo test è stato visionato il corretto funzionamento del metodo registraCliente(…) del sistema FitnessReady, il quale prende in ingresso nome, cognome, e-mail e password del cliente oltre a confermaPassword (necessario al cliente per il corretto inserimento della sua password). Dopo aver specificato queste informazioni, il metodo generà ed aggiungerà un nuovo cliente con tali attributi alla lista di utenti posseduta dal sistema. Poiché anche nel metodo sopra descritto vengono lanciate diverse eccezioni in caso d’inserimento dati scorretto, per effettuare il test sono stati forniti dati intenzionalmente errati o già presenti per provocare il lancio delle relative eccezioni, per la cui verifica si è ricorso nuovamente alla funzione assertThrows(), per poi creare un nuovo cliente con cui confrontare quello inserito nella lista utenti usando il metodo registraCliente() tramite la funzione assertEquals().

## 6.2 – File PurchaseTest

In tale file si è testato il corretto funzionamento della classe “GestoreAcquisto”, che gestisce tutte le operazioni relative all’acquisto di uno o più prodotti. Poiché uno stesso gestore acquisto permane solo per la durata (o l’annullamento) di un singolo acquisto di uno o più prodotti, per una corretta esecuzione dei test di seguito descritti viene innanzitutto recuperata l’unica istanza del sistema FitnessReady ed in più, prima di ogni test, il gestore acquisto viene re inizializzato.

### **6.2.1 – Add-To-Cart Test**

In questo test è stato visionato il corretto funzionamento del metodo aggiungiAlCarrello(…) del GestoreAcquisto, il quale ha il compito di inserire il prodotto selezionato nell’apposito carrello d’acquisto e per questo tale metodo prende in ingresso il suddetto prodotto e la quantità desiderata dal cliente. Il metodo in questione si occupa sia di effettuare dei controlli sui parametri d’ingresso i quali se quest’ultimi non sono corretti, lanciano diversi tipi di eccezioni a seconda dell’errore riscontrato, sia di ritornare un parametro booleano per determinare se il prodotto che si cerca di aggiungere sia o no già presente nel carrello. Per questa ragione il test si concentra prima sulla corretta ricezione di tali eccezioni in caso di errato inserimento o selezione del prodotto tramite la funzione assertThrows(), e poi sulla verifica del corretto inserimento di un prodotto appositamente creato, cercando d’inserire quest’ultimo due volte e verificandone l’esito tramite le funzioni asserTrue(), per il primo tentativo, e assertFalse(), per il secondo tentativo.

### **6.2.2 – Execute-Purchase Test**

Questo test ci permette di controllare il corretto funzionamento dell’effettiva esecuzione di un nuovo acquisto relativo ai prodotti inseriti nel carrello durante la fase di selezione. Il metodo della classe GestoreAcquisto interessato in questo test è effettuaAcquisto(), il quale restituisce l’acquisto che viene concluso in modo tale che venga aggiunto alla lista acquisti relativa del cliente che lo effettua. Nel test viene prima verificata l’impossibilità di effettuare un acquisto nel caso in cui il carrello sia vuoto tramite la funzione assertThrows(), poiché in tali condizioni il metodo lancia un’eccezione, e poi viene controllata la corretta esecuzione dell’acquisto confrontando il risultato ottenuto tramite il metodo qui trattato con un acquisto appositamente creato per il confronto grazie alla funzione assertEquals().

## 6.3 – File BookingTest

In tale file è stato testato il corretto funzionamento della classe “GestorePrenotazioni”, che gestisce tutte le operazioni relative alla prenotazione di un prodotto momentaneamente non disponibile. Per la corretta esecuzione dei test di seguito descritti viene prima di tutto eseguita una funzione di setUp tramite la quale si recupera l’unica istanza del sistema FitnessReady e vengono inizializzati un cliente di prova che esegue la prenotazione ed un prodotto con disponibilità 0 da prenotare.

### **6.3.1 – Execute-Booking Test**

Questo test viene eseguito per poter controllare la corretta creazione di una nuova prenotazione relativa ad un prodotto con disponibilità nulla. Il metodo della classe GestorePrenotazioni interessato in questo test è effettuaPrenotazione(), il quale genera una nuova prenotazione a seconda dei dati inseriti dal cliente (necessari per effettuare la prenotazione) per poi aggiungerla alla lista di prenotazione per il relativo cliente che la esegue. Per prima cosa viene testata l’impossibilità di effettuare la prenotazione nel caso in cui la quantità inserita dal cliente sia minore o uguale a 0 oppure nel caso il cliente intenda effettuare la prenotazione utilizzando i propri gettoni ma essi siano insufficienti tramite la funzione assertThrows(), che permette di verificare il lancio da parte del sistema delle relative eccezioni nei casi sopra descritti. Dopo di che viene testata l’effettiva creazione di una nuova prenotazione, inizializzando un’apposita prenotazione da confrontare con quella ottenuta tramite il metodo effettuaPrenotazione(), usando la funzione assertEquals().

# **7 – Glossario**

Vengono qui di seguito riportati i termini più significativi ai fini del progetto e le relative definizioni.

* **Fitness Ready**: nominativo usato per indicare l’applicazione software ideata per l’attribuzione di schede fitness e la vendita dei prodotti della società che ha richiesto il suo sviluppo.
* **Sistema**: termine generico utilizzato per indicare l’applicazione Fitness Ready in maniera rapida e sintetica.
* **Utente**: generico utilizzatore dell’applicazione Fitness Ready.
* **Cliente**: persona fisica registrata in memoria dall’applicazione Fitness Ready con l’intendo di usufruire dei servizi offerti dalla suddetta applicazione.
* **Amministratore**: utente con privilegi speciali, già presente in memoria, scelto dalla società richiedente lo sviluppo dell’applicazione con le adeguate competenze fisico-alimentari per la gestione dell’applicazione Fitness Ready.
* **Scheda fitness**: scheda tecnica redatta dall’amministratore ed attribuita automaticamente dal sistema ai clienti correttamente registrati all’applicazione Fitness Ready, contente un piano di esercizi fisici ed una dieta da seguire per raggiungere l’obiettivo prefissatosi.
* **Prodotto**: termine generico utilizzato per indicare gli articoli, come integratori alimentari, cibi proteici ed attrezzature sportive, prodotti dalla società commissionante lo sviluppo dell’applicazione Fitness Ready e venduti all’interno della stessa.
* **Notifica**: messaggio generato dall’applicazione Fitness Ready per confermare il successo o il fallimento di una generica operazione eseguita tramite il suo utilizzo.