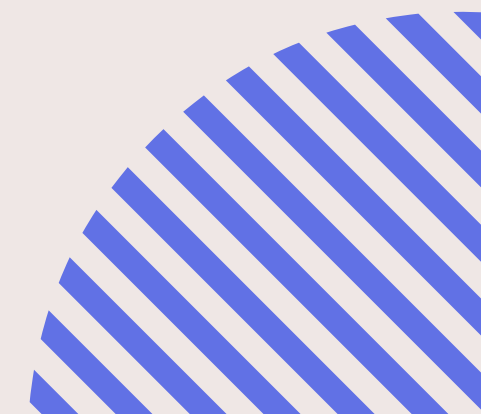
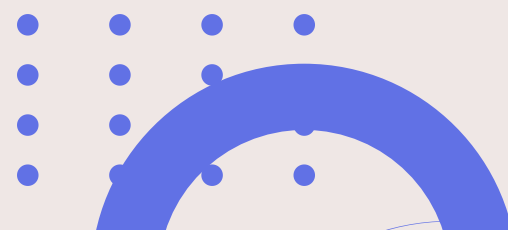
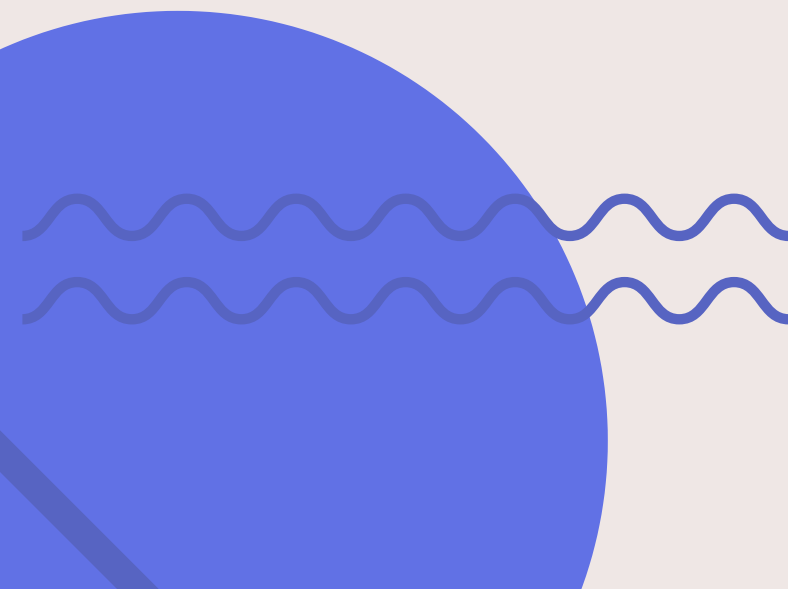
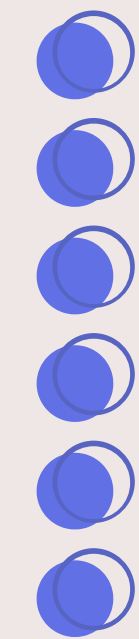
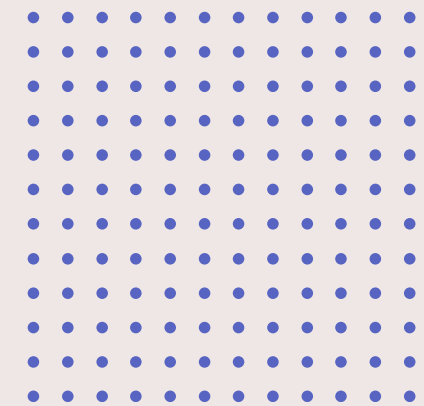
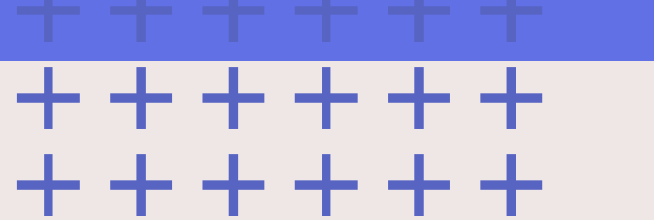
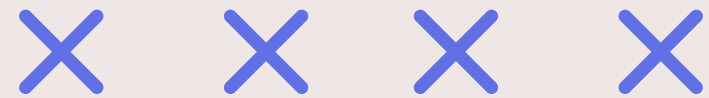


NEXTFIT



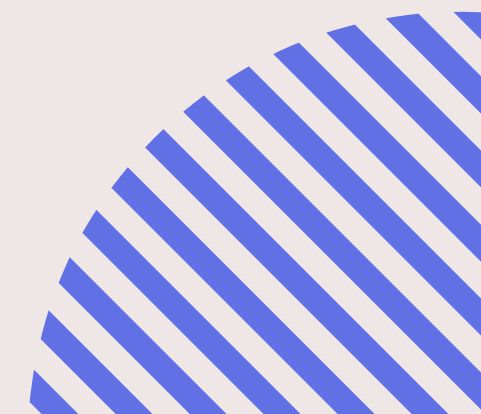
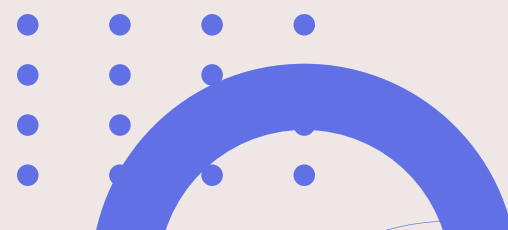
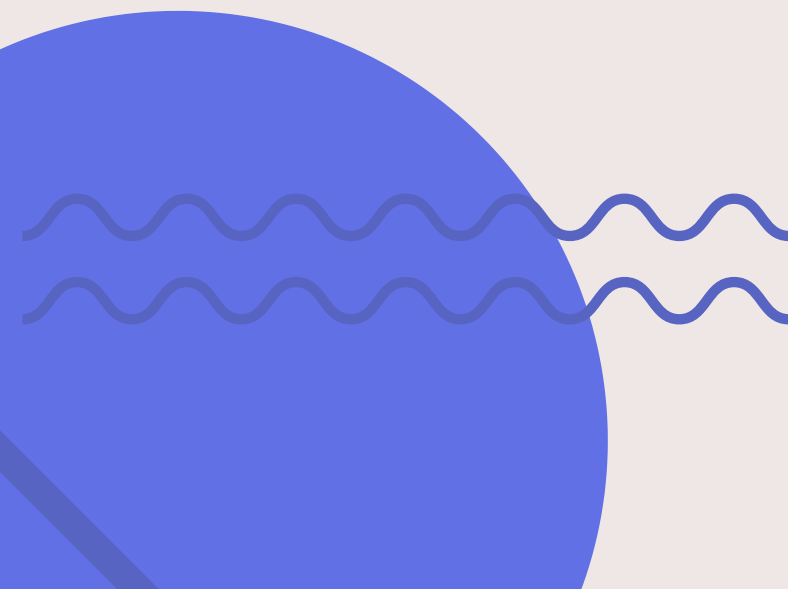
Programmazione ad oggetti
anno 2023/2024

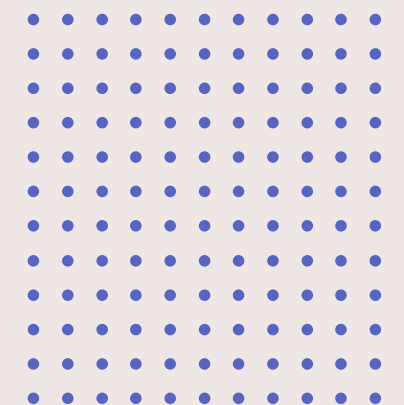
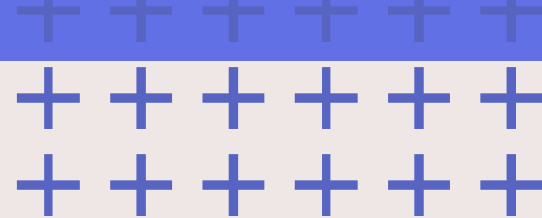
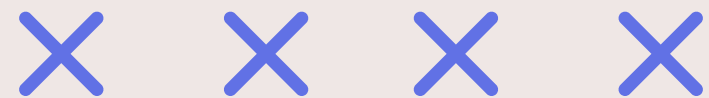




NAMENOT FOUND

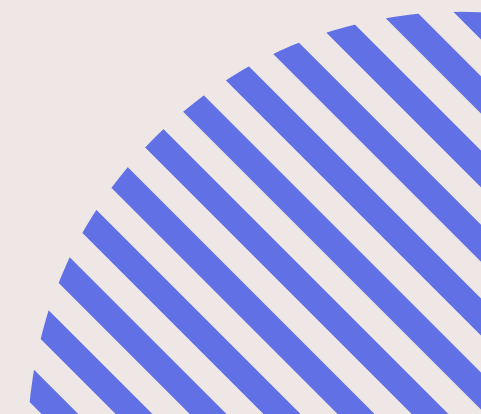
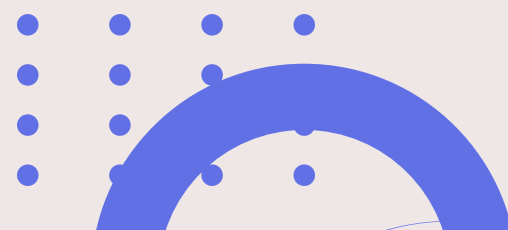
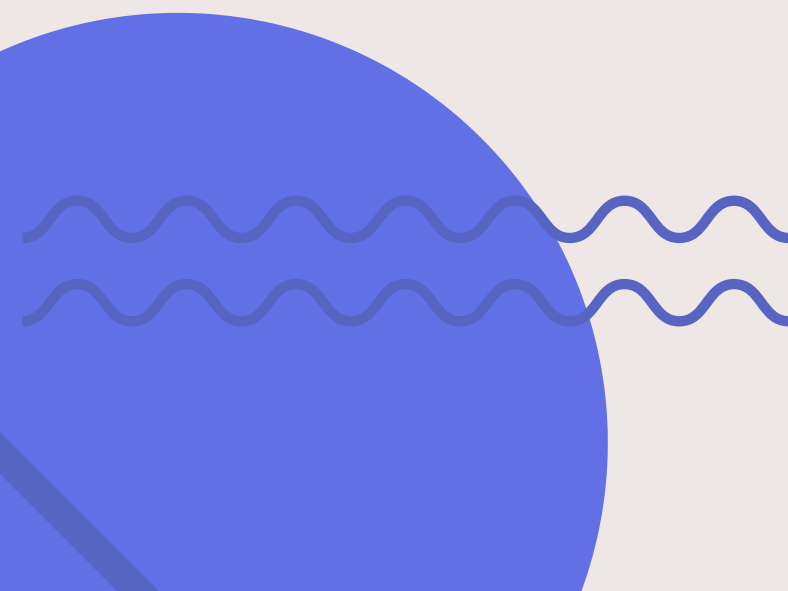
- Riccardo Corsaro
- Giovanni Mari





INTRODUZIONE

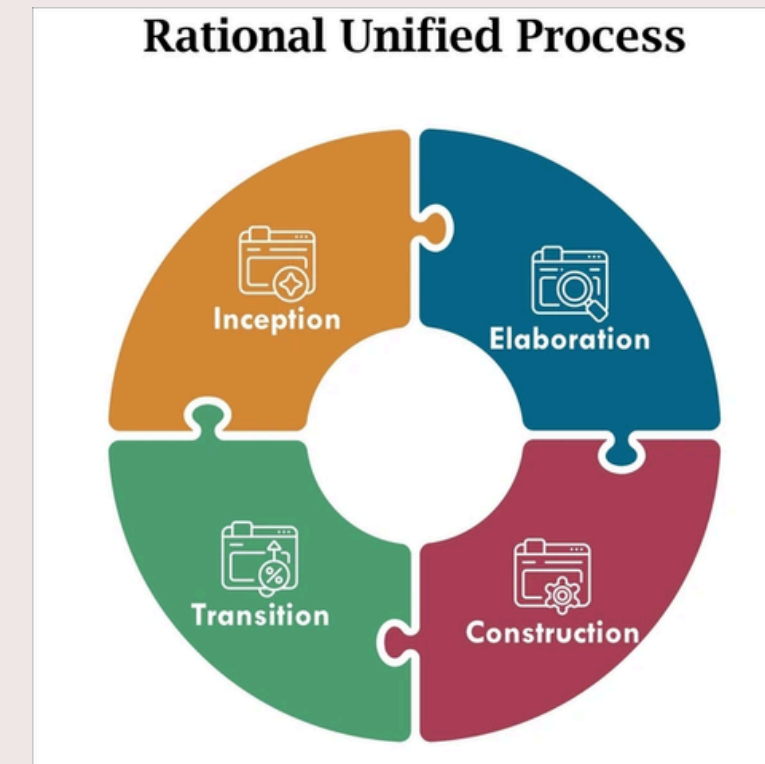
Il progetto NextFit si basa sulla creazione di un'interfaccia per la gestione di clienti e dipendenti di una ipotetica palestra. Ogni cliente avrà a disposizione un'area dove poter svolgere azioni.

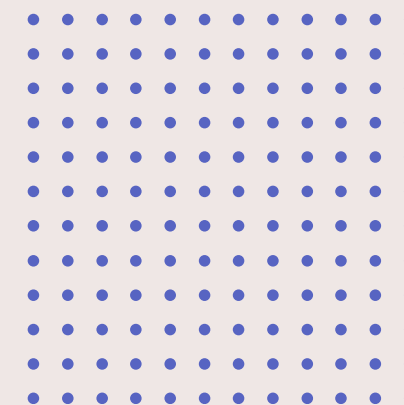
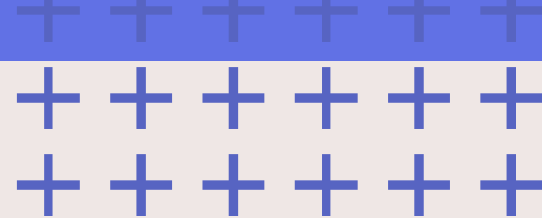
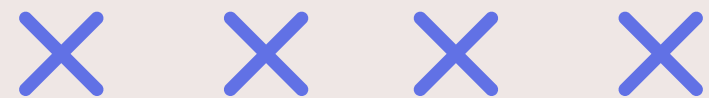


PROCESSO SOFTWARE: UP

Come processo software per la progettazione ci siamo basati su Unified Process (Up) che si compone di diverse fasi:

- Requisiti
- Analisi
- Progettazione
- Implementazione
- Test





REQUISITI FUNZIONALI

1 - CLIENTE

- Registrazione/autenticazione
- Disiscrizione
- Rinnovo abbonamento
- Scelta PT
- Scelta Corso
- Visualizzazione e scheda

2 - PERSONAL TRAINER

- Autenticazione
- Visualizzazione clienti
- Compilazione e invio scheda

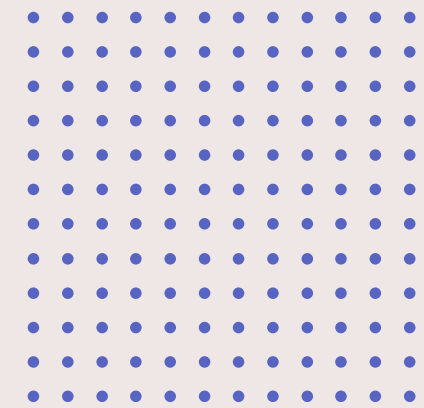
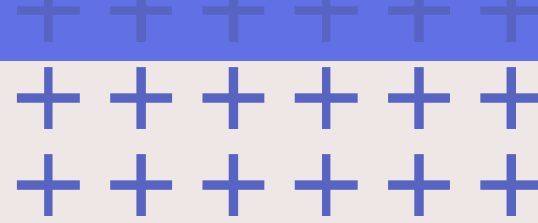
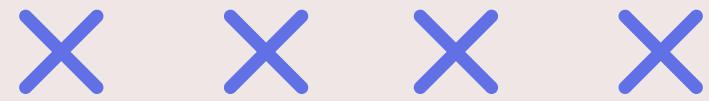
3 - CORSISTA

- Autenticazione
- Visualizzazione dei propri corsi
- Visualizzazione iscritti ai corsi

4 - PROPRIETARIO

- Autenticazione
- Registrazione dipendenti
- Eliminazione dipendenti
- Registrazione corsi
- Eliminazione corsi





REQUISITI NON FUNZIONALI

ORGANIZZATIVI

- Il sistema deve essere ultimato entro il 27/09/2024.
- La documentazione deve essere scritta in italiano.

ESTERNI

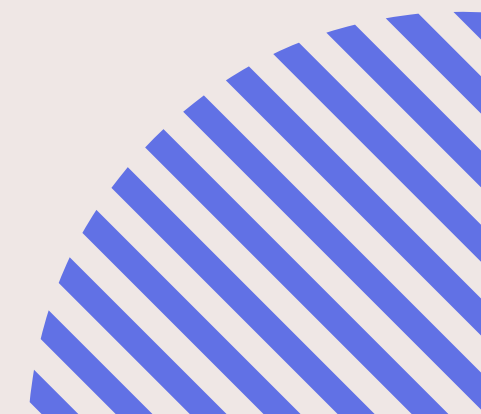
- L'utente prima dell'utilizzo del sistema accetta l'utilizzo ed il salvataggio dei propri dati da parte del sistema.

SICUREZZA

- L'accesso al sistema deve essere protetto da meccanismi di autenticazione, attraverso mail e password.

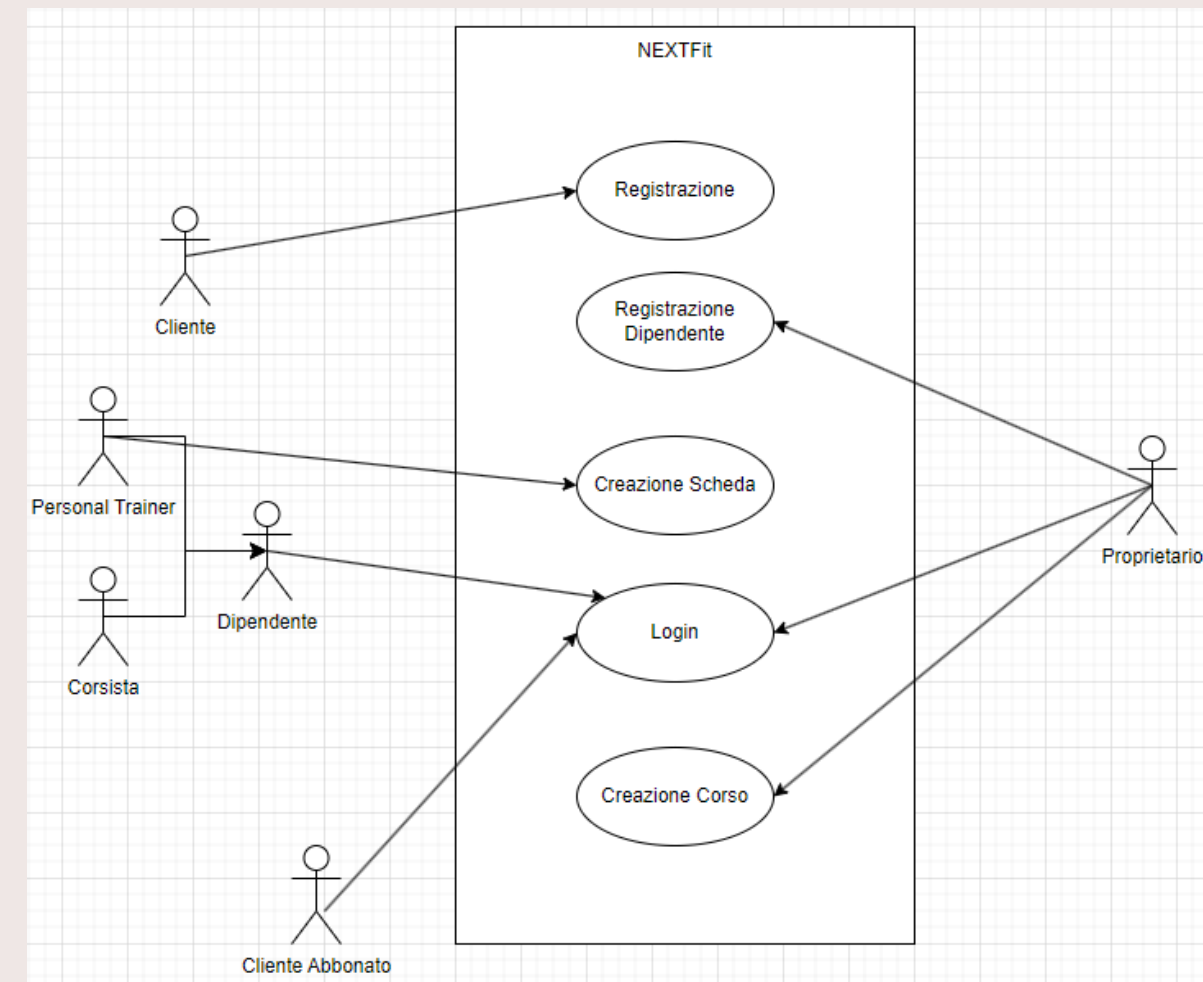
TECNICI

- Il sistema deve essere sviluppato utilizzando i seguenti linguaggi di programmazione: JAVA e MYSQL.
- L'interfaccia grafica deve essere sviluppata con il framework "SWING" di Java.

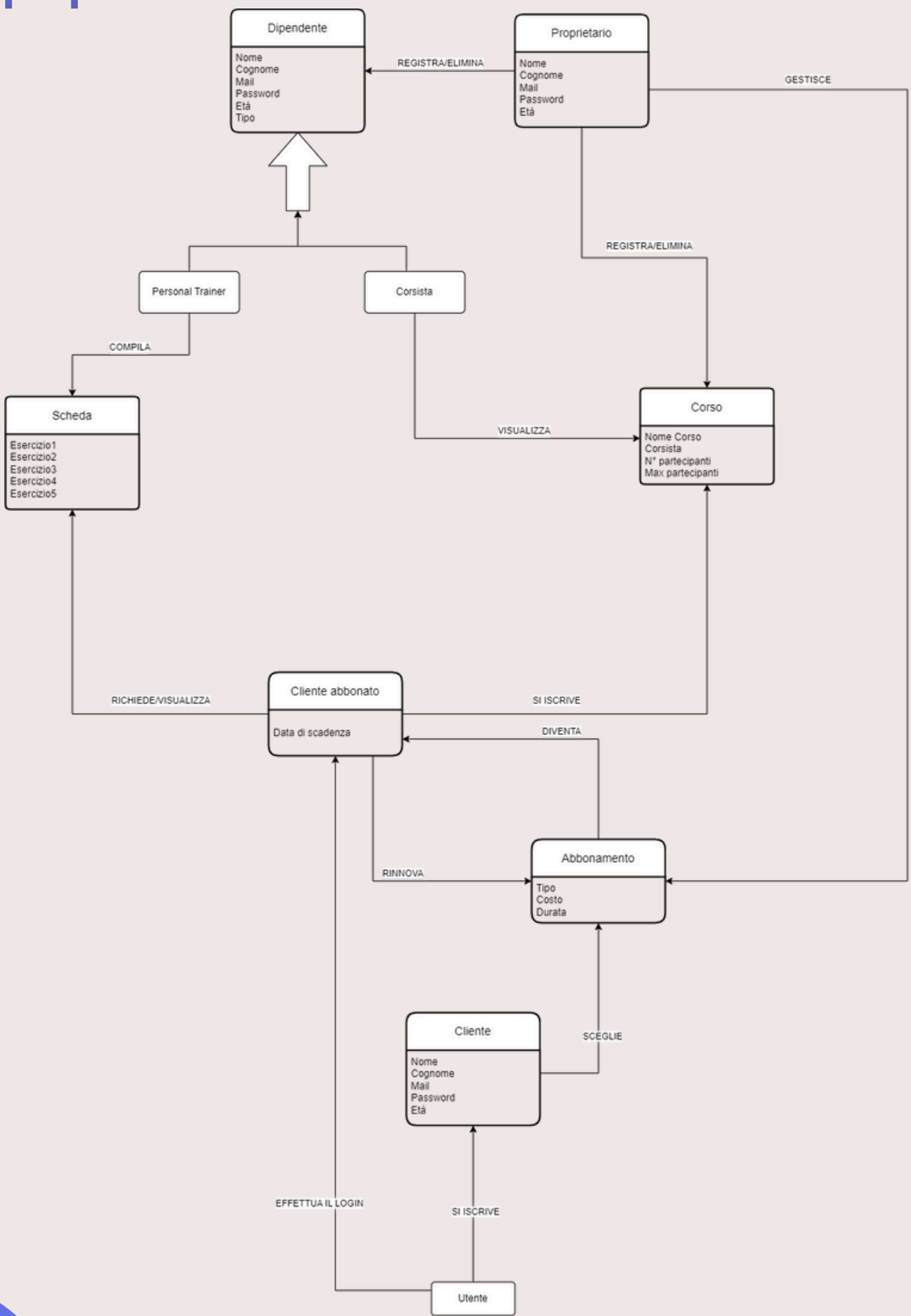


CASI D'USO

UC1: Registrazione Cliente
UC2: Registrazione dipendenti
UC3: Creazione scheda
UC4: Login
UC5: Creazione Corso



MODELLO DI DOMINIO



UML CLASSI

STRUTTURA PROGETTO

Model

- > Abbonamenti.java
- > Cliente.java
- > ClienteAbbonato.java
- > Corsi.java
- > Corsista.java
- > Corso.java
- > Dietista.java
- > Dipendente.java
- > Esercizio.java
- > Fisioterapista.java
- > IscrittoalCorso.java
- > IstruttorediSala.java
- > Palestra.java
- > PersonalTrainer.java
- > Proprietario.java
- > RichiestaAIPT.java
- > Richieste.java
- > Scheda.java
- > Tester.java

View

- > AbbonamentiView.java
- > CreaSchedaView.java
- > ElimCorsoView.java
- > ElimDipView.java
- > LatoClienteView.java
- > LatoCorsistaView.java
- > LatoPTView.java
- > ListaCLView.java
- > ListaCorsiPropView.java
- > ListaCorsiView.java
- > ListalsCorsiView.java
- > ListaPTView.java
- > LoginView.java
- > PrimaPaginaView.java
- > ProprietarioView.java
- > RegistraCorsoView.java
- > RegistraDipView.java
- > RegView.java
- > RinAbbView.java
- > SerAggView.java
- > VisuSchedaView.java

Controller

- > AbbonamentiController.java
- > CreaSchedaController.java
- > ElimClienteController.java
- > ElimCorsoController.java
- > ElimDipController.java
- > LogController.java
- > NavigationController.java
- > RegController.java
- > RegistraCorsoController.java
- > RegistraDipController.java
- > RinAbbController.java
- > SceltaCorsoController.java
- > SceltaPTController.java

DIAGRAMMA DI SEQUENZA AUTENTICAZIONE

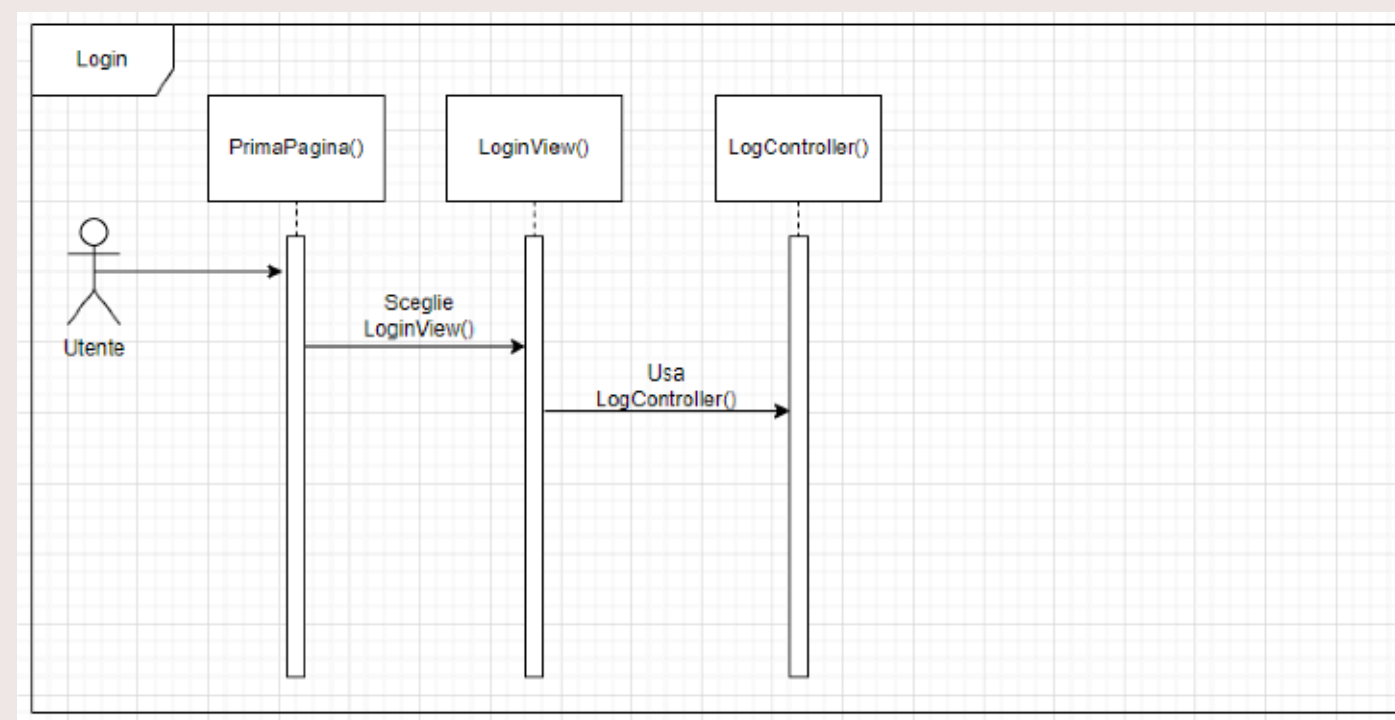
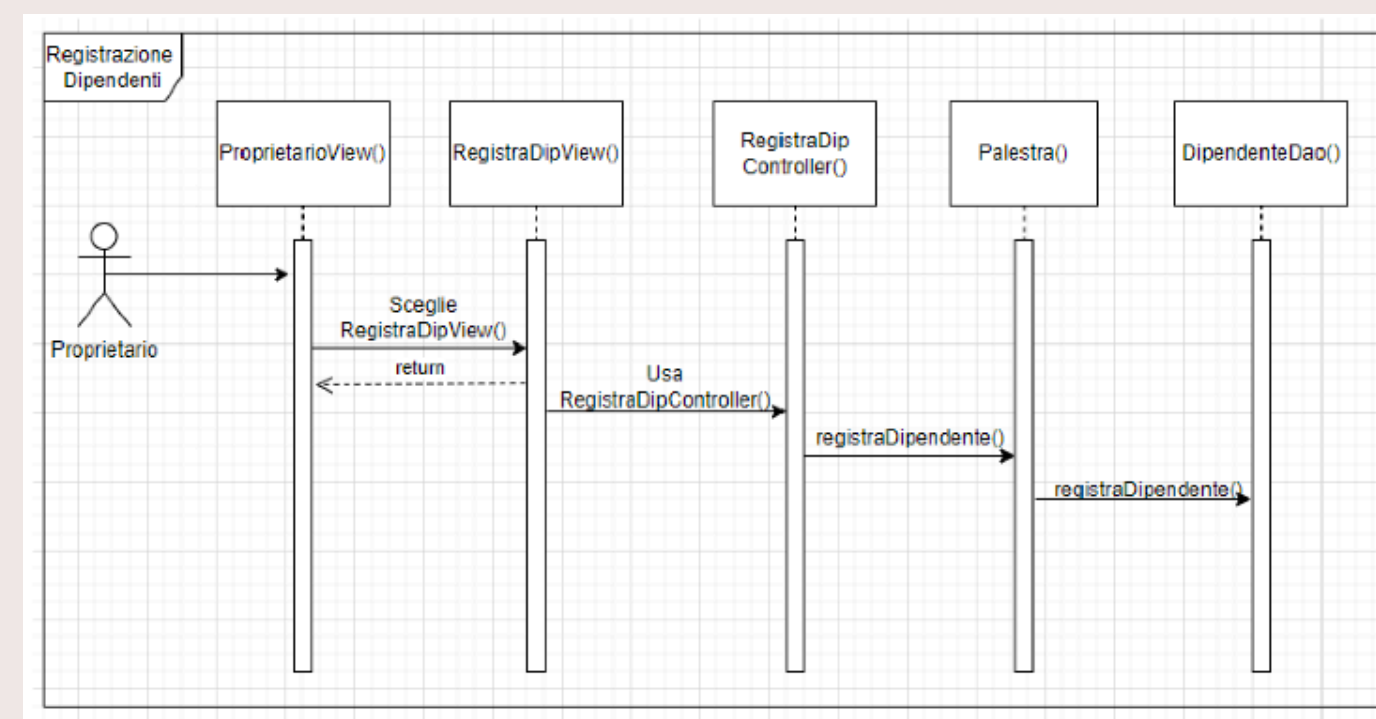
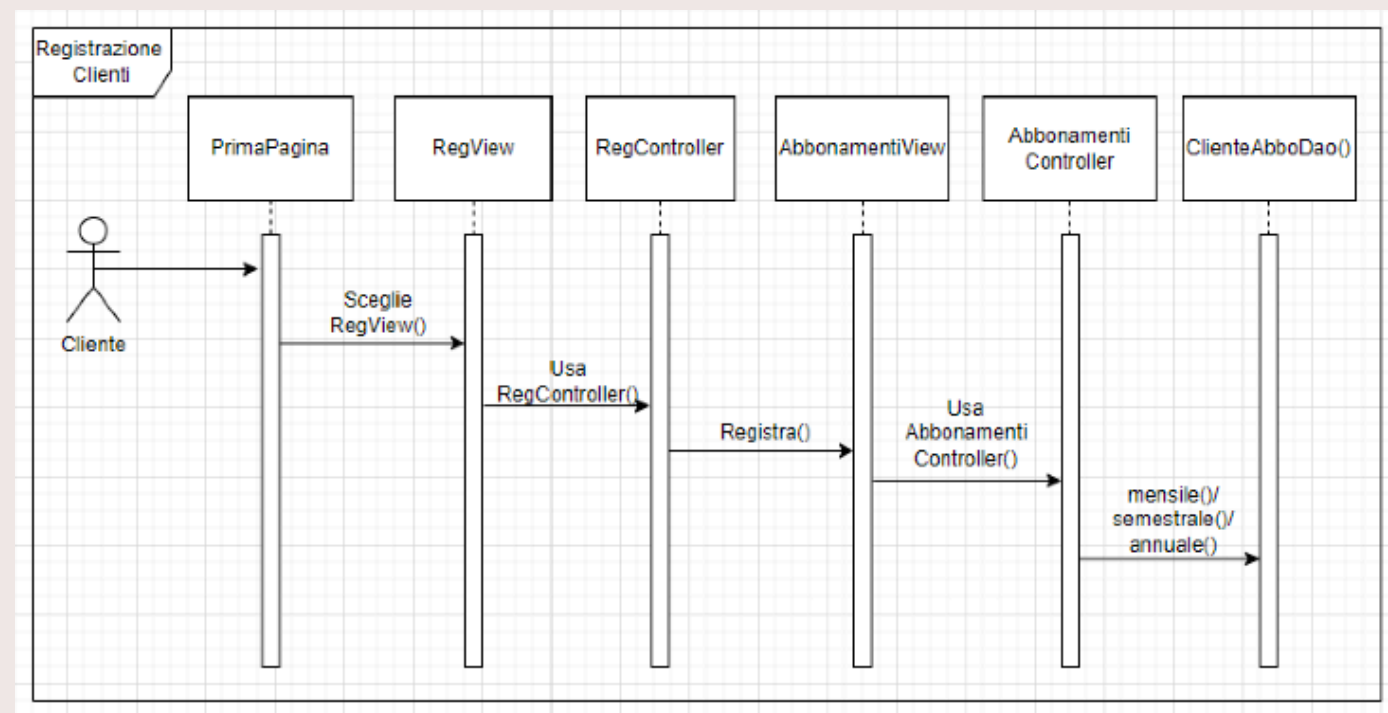


DIAGRAMMA DI SEQUENZA CREAZIONE SCHEDA

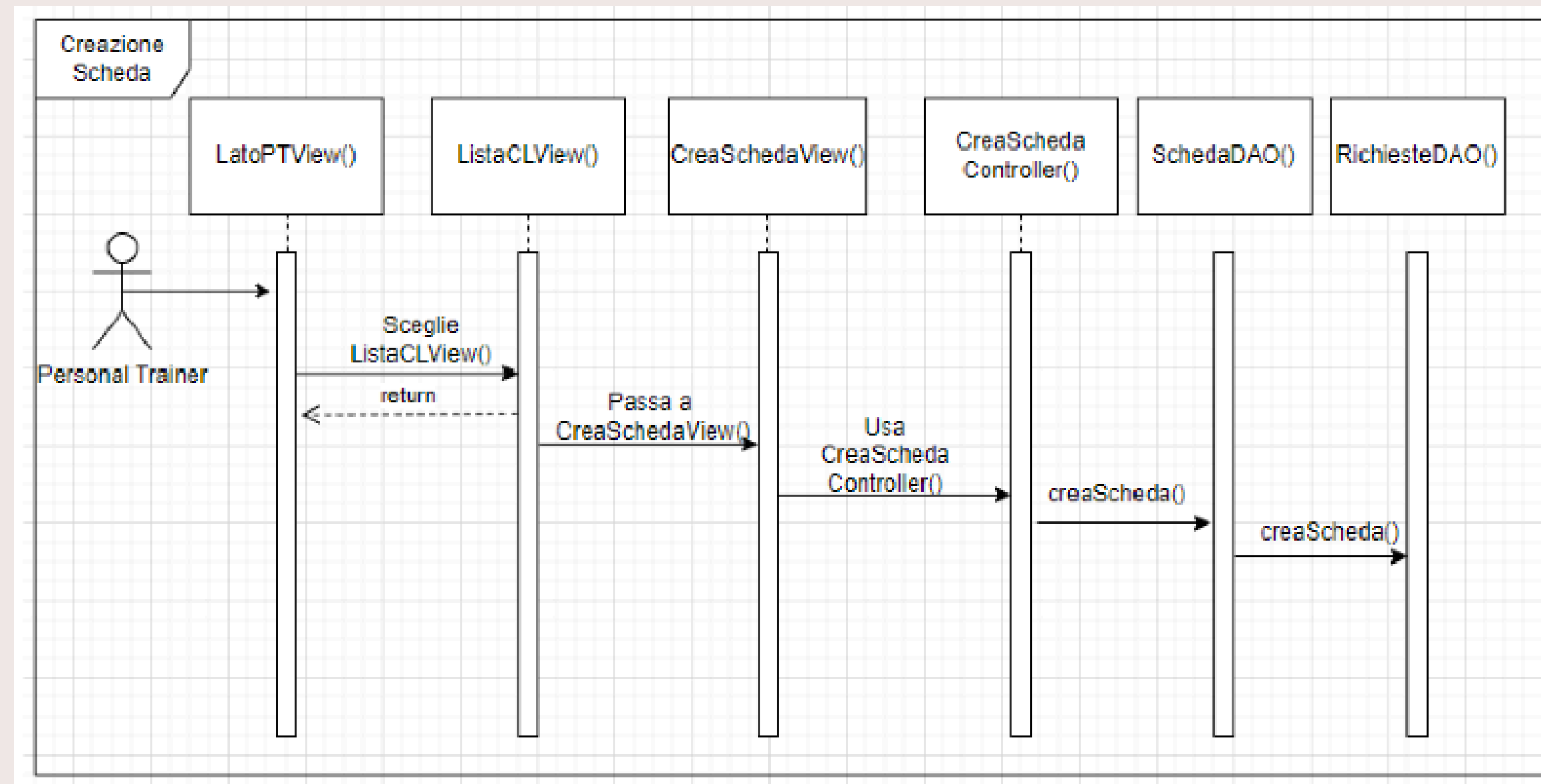


DIAGRAMMA DI SEQUENZA CREAZIONE CORSO

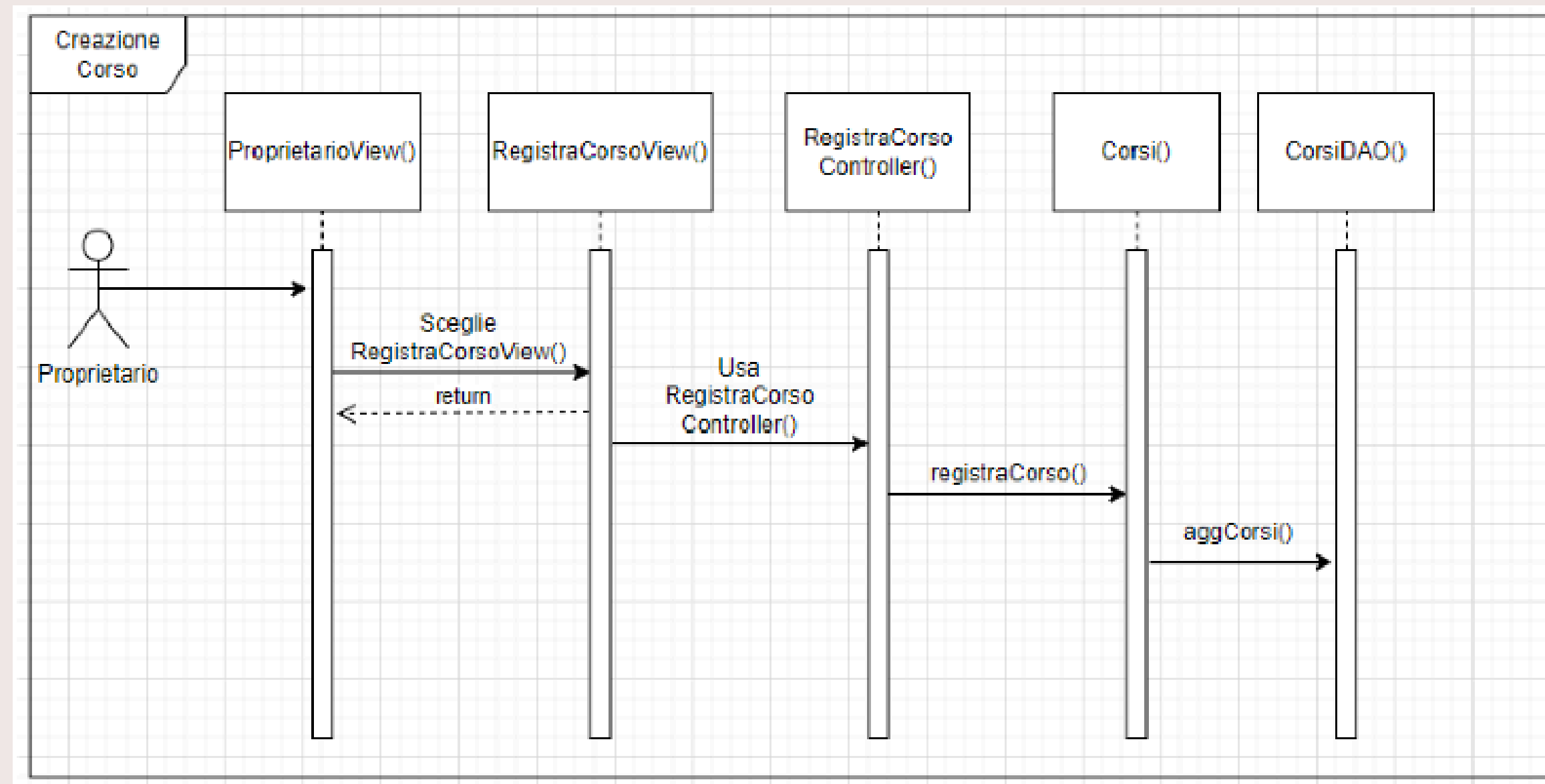
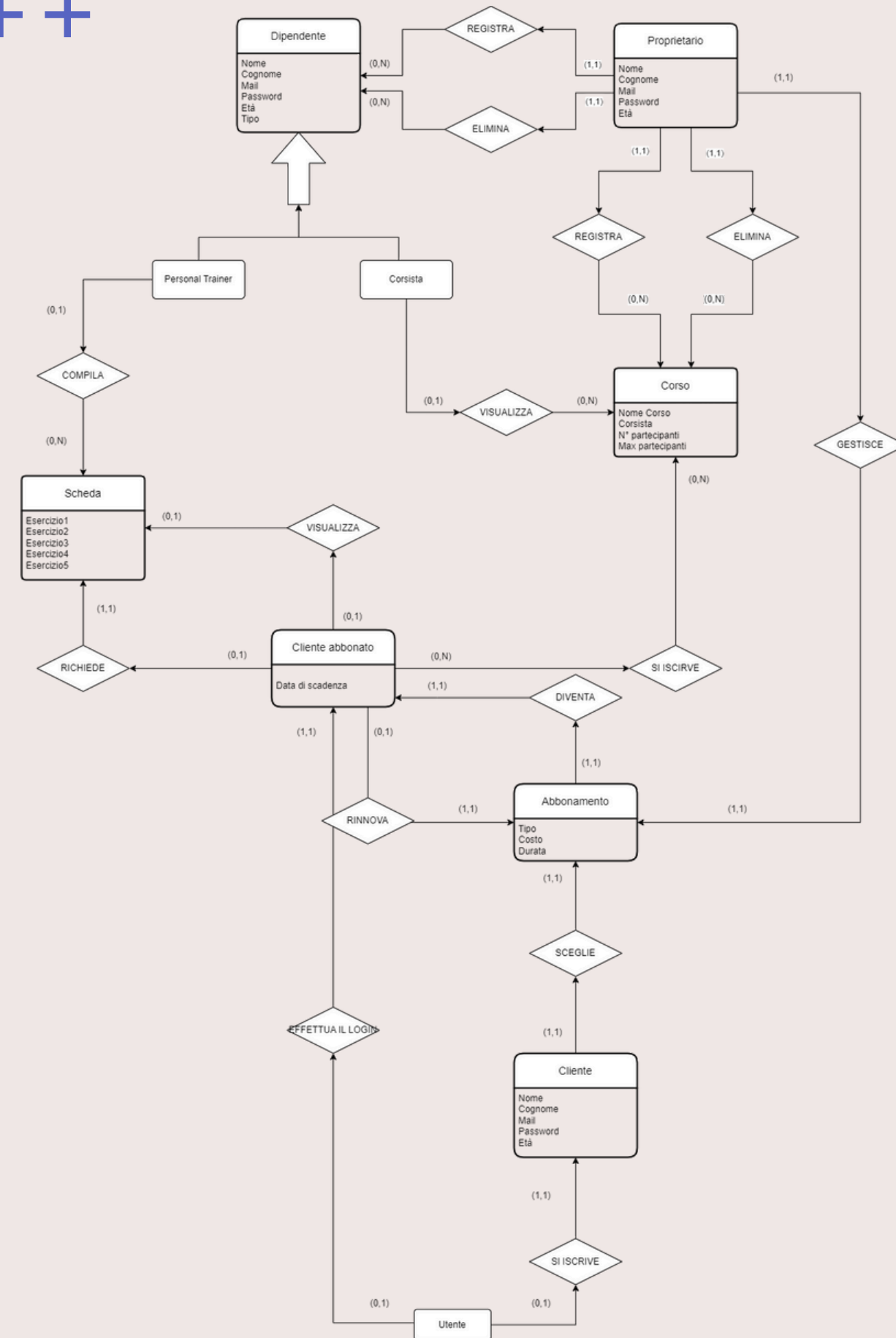
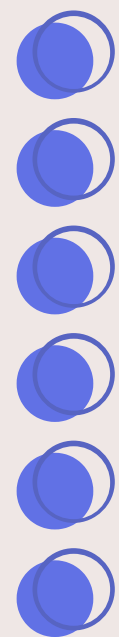
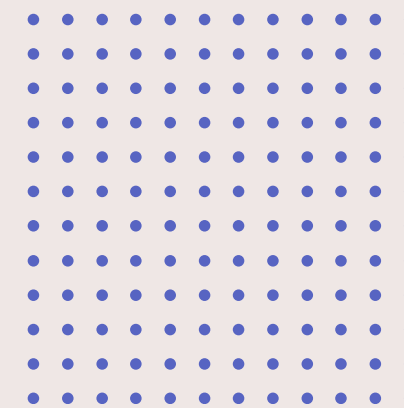
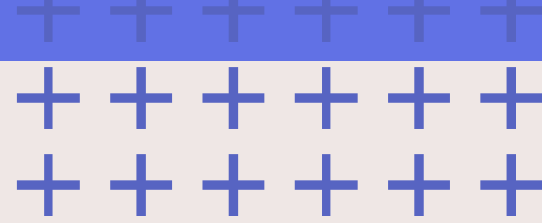
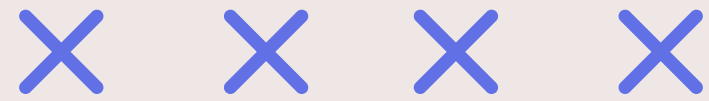









DIAGRAMMA ERA

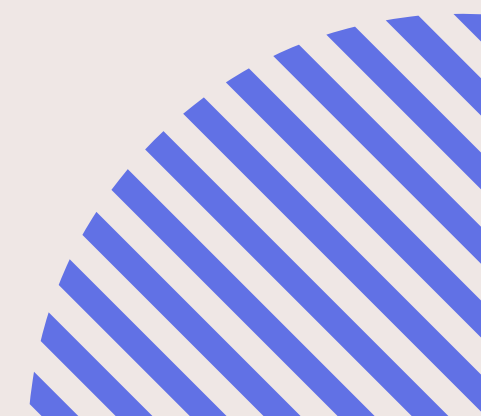
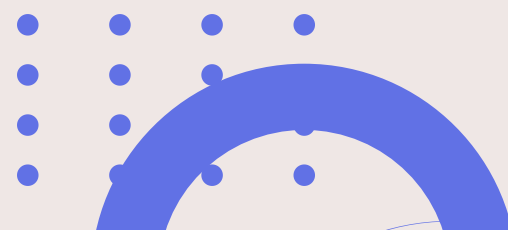
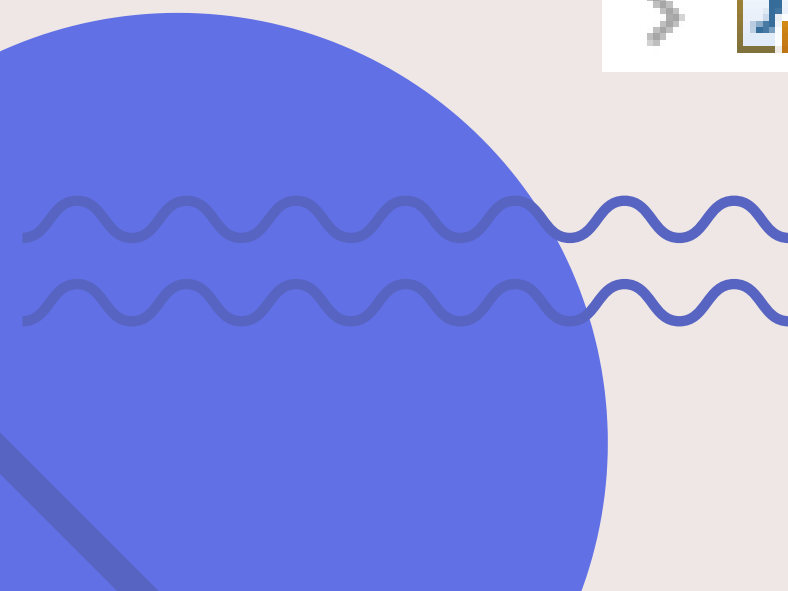




PATTERN DAO

- >  ClienteAbboDAO.java
- >  CorsiDAO.java
- >  DBConnection.java
- >  DipendenteDAO.java
- >  IscrittoalcorsoDAO.java
- >  RichiesteDAO.java
- >  SchedaDAO.java

Utilizzato per mantenere il
flusso dei dati



PATTERN SINGLETON

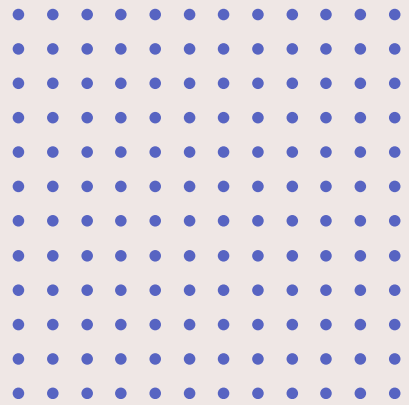
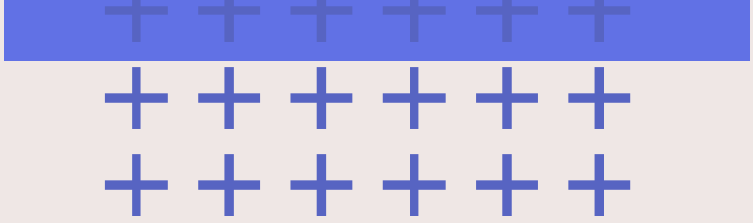
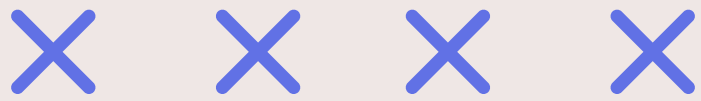
```
private Proprietario() {
    this.abbonamentoMensile = new Abbonamenti("Mensile", 50.0);
    this.abbonamentoSemestrale = new Abbonamenti("Semestrale", 250.0);
    this.abbonamentoAnnuale = new Abbonamenti("Annuale", 450.0);
    this.mail = "proprietario@example.com";
    this.password = "IlProp";
}

public static Proprietario getInstance() {
    if (instance == null) {
        instance = new Proprietario();
    }
    return instance;
}
```

```
private Palestra(int maxC, int maxD) {
    this.maxC = maxC;
    this.maxCA = maxC;
    this.maxD = maxD;
    clienti = new Cliente[maxC];
    dipendenti = new Dipendente[maxD];
    clientiAbbo = new ClienteAbbonato[maxCA];
    contatoreC = 0;
    contatoreCA = 0;
    contatoreD = 0;
}



public static Palestra getInstance() {
    if (instance == null) {
        instance = new Palestra(100, 100);
    }
    return instance;
}
```

Utilizzato per avere una sola istanza delle classi palestra e proprietario

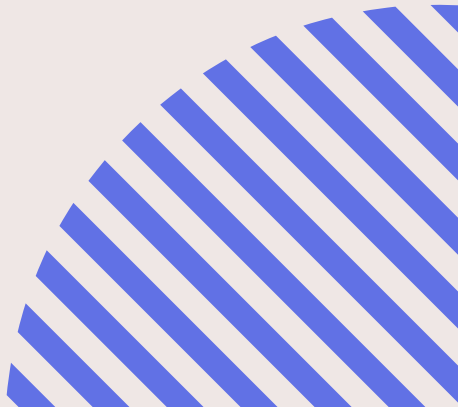
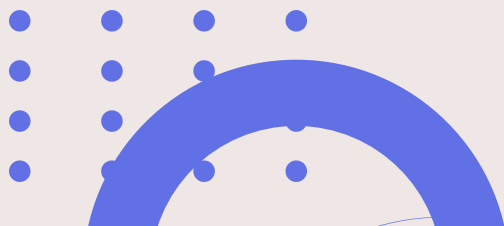
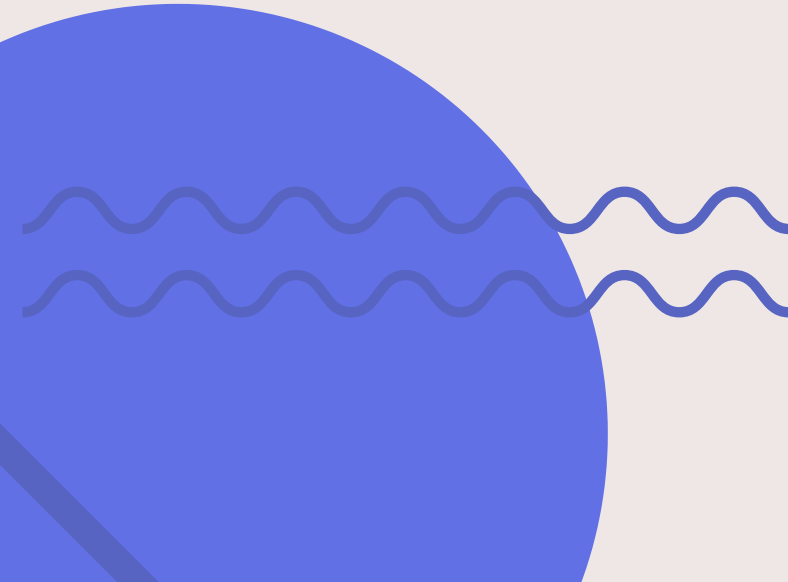


PATTERN CREATOR



- >  `Palestra.java`
- >  `Richieste.java`
- >  `Corsi.java`

Utilizzato per avere una classe che crea altri oggetti



TEST REGISTRAZIONE CLIENTE

Controllo del corretto
funzionamento della
registrazione da parte del
cliente

```
@Test
public void testRegistrationTrue() {
    Cliente c1 = new Cliente("Matteo", "Grigio", "Paolo@gmail.com", "password", 22);
    boolean result = true;
    if (p.registraCliente(c1) == true) {
        Abbonamenti a1 = new Abbonamenti("Mensile", 50.0);
        ClienteAbbonato c2 = new ClienteAbbonato(c1, a1);

        cdao.selectAll(p);

        result = cdao.insertCliente(c1, c2);
        assertTrue(result);
    }
}

@Test
public void testRegistrationFalse() {
    Cliente c1 = new Cliente("Matteo", "Grigio", "Matteo@gmail.com", "password", 15);
    boolean result1 = true;
    if (p.registraCliente(c1) == true) {

        Abbonamenti a1 = new Abbonamenti("Mensile", 50.0);
        ClienteAbbonato c2 = new ClienteAbbonato(c1, a1);

        cdao.selectAll(p);

        result1 = cdao.insertCliente(c1, c2);
        assertFalse(result1);
    }
}
```

TEST REGISTRAZIONE E ELIMINAZIONE DIPENDENTI

Controllo del corretto
funzionamento della
registrazione e
eliminazione dipendenti

```
@Test
void testDipRegistration() {

    ddao.selectAll(p);

    Dipendente d1 = new Corsista("Carlo", "Gialli", "cg@cg.com", "cgcg8", 23, 2500, "Corsista");

    boolean result = true;

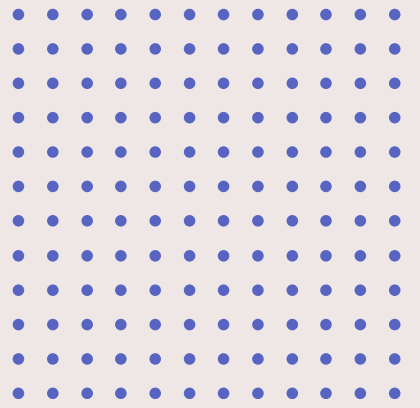
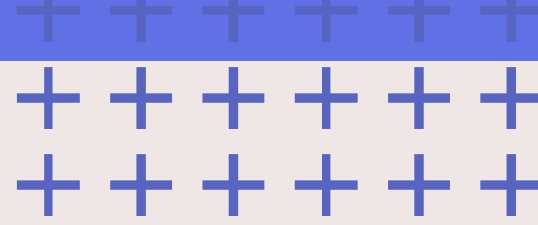
    result = p.registraDipendente(d1);
    assertTrue(result);
}

@Test
void testDipElim() {

    ddao.selectAll(p);

    boolean result = true;

    result = ddao.eliminaDip("Giovanni", "Bianchi", "giovanni.bianchi@example.com", p);
    assertTrue(result);
}
```



TEST SCELTA PERSONAL TRAINER



Controllo della scelta
di un PT da parte di un
cliente

```
@Test
void testscelta() {

    cdao.selectAll(p);
    ddao.selectAll(p);
    rdao.selectAll(p, r);

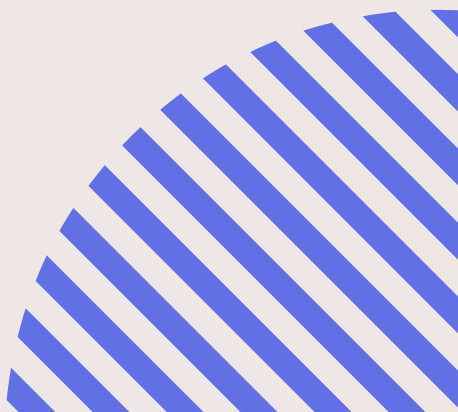
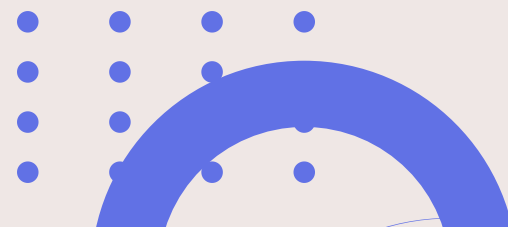
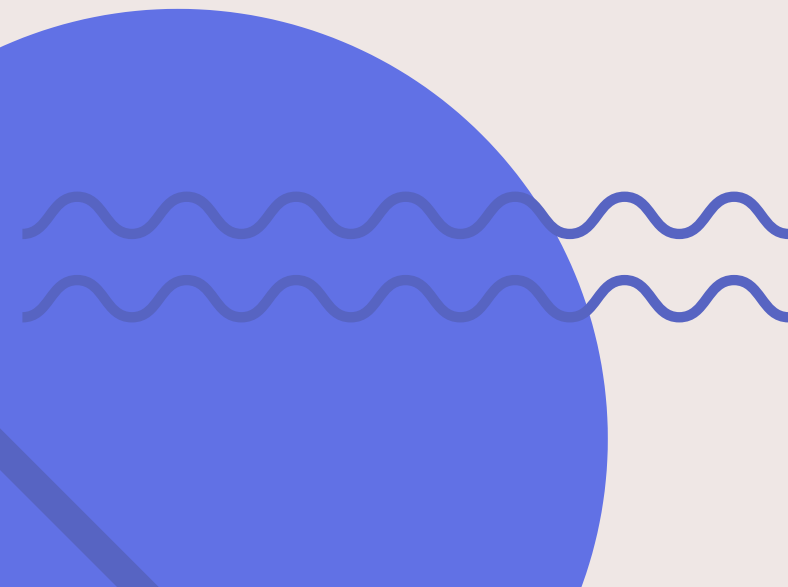
    ClienteAbbonato c1 = p.accessoCli("marco.blu@example.com", "password789");
    PersonalTrainer pt = (PersonalTrainer) p.accessoDip("antonio.marroni@example.com", "topsecret");

    r.aggRichiesta(pt, c1, 0);

    boolean result = true;

    result = rdao.insertRichiesta(r.ricaricaRichiesta(c1, pt));

    assertTrue(result);
}
```



TEST SCELTA CORSO

Controllo della scelta
di un corso da parte di
un cliente

```
@Test
void test() {

    cdao.selectAll(p);
    ddao.selectAll(p);
    ccdao.selectAll(p, c);

    ClienteAbbonato c1 = p.accessoCli("marco.blu@example.com", "password789");

    Corso cc = c.getCorso(0);
    boolean result=true;

    result=idao.insertIscrizione(c1, cc);

    ccdao.upIscritti(cc.getNome());

    assertTrue(result);
}
```

TEST CREAZIONE E ELIMINAZIONE CORSI

Controllo del corretto funzionamento della creazione ed eliminazione di corsi

```
@Test
void testCreaCorso() {

    ddao.selectAll(p);
    cdao.selectAll(p, c);

    Corsista cc = (Corsista) p.ricercaCorsista("Chiara", "Neri", "chiara.neri@example.com");

    Corso c1 = new Corso("Atletica", cc, 10, 0);
    boolean result = true;

    result = c.aggCorsi(c1);
    assertTrue(result);
}

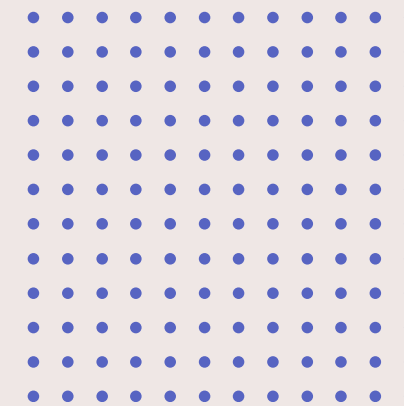
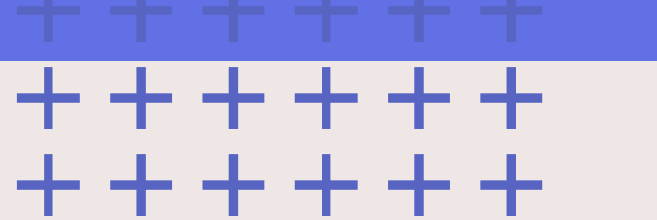
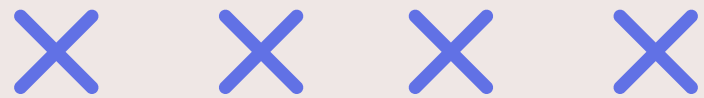
@Test
void testElimCorso() {

    ddao.selectAll(p);
    cdao.selectAll(p, c);

    Corsista cc = (Corsista) p.ricercaCorsista("Maria", "Rossi", "maria.rossi@example.com");

    boolean result = true;

    result = cdao.deleteCorso("Yoga", cc.getNome(), cc.getCognome(), cc.getMail(), c, p);
    assertTrue(result);
}
```



GRAZIE PER
L'ATTENZIONE

