



Università degli Studi di Firenze

Esame Ingegneria del Software

Mattia Pardeo

17/10/2022

Contents

1	Descrizione del progetto	2
2	Progettazione	4
2.1	Software utilizzati	4
2.2	Class diagram	5
2.3	Use case diagram	6
2.4	Templates	7
2.4.1	Modifica dati	7
2.4.2	Iscrizione evento	8
2.4.3	Modifica nome palestra	9
2.5	Mockups	10
2.6	Descrizione classi	13
2.6.1	Package Federazione	13
2.6.2	Package DAOs	15
2.6.3	Package Controllers	16
2.7	Design patterns	17
2.7.1	Factory method Pattern	17
2.7.2	Front Controller Pattern	17
2.7.3	Data Access Object Pattern	18
3	Testing	19
3.1	Codici test	20
3.1.1	Test ControllerAllievo	20
3.1.2	Test ControllerMaestro	21

Chapter 1

Descrizione del progetto

Il progetto consiste nella creazione di un applicazione Java che funga da backend di un sito web di una federazione di arti marziali che permetta la gestione dei propri utenti e di eventi creati dagli istruttori e maestri presenti all'interno di quest'ultima. In particolare sarà permesso:

- Per ogni utente l'iscrizione alla federazione tramite contatto telefonico o in presenza, dopodiché gli saranno fornite le credenziali di accesso al sito web. Dopo averle ricevute potrà accedere al sito e visualizzare:
 - ogni evento disponibile a cui potrà iscriversi, la descrizione, dove si terrà e chi è l'organizzatore,
 - le discipline insegnate con la loro descrizione, gli insegnanti di quest'ultima e in quali palestre si tengono i rispettivi corsi,
 - gli insegnanti con i rispettivi dati, le discipline che insegnano e in quali palestre,
 - le palestre in cui si tengono i corsi di ogni disciplina e quale insegnante tiene il corso. Potrà iscriversi ad una palestra e a qualsiasi disciplina al suo interno. Non potrà iscriversi ad un'altra palestra fintanto che l'abbonamento non sarà scaduto.

Potrà inoltre modificare i propri dati, come ad esempio i propri dati anagrafici (nome, cognome, data di nascita, codice fiscale), i contatti (telefono ed email) e la password precedentemente fornita.

- Iscrivere nuovi maestri ed istruttori da parte di altri maestri già appartenenti alla federazione e nuovi allievi da parte sia di istruttori che maestri. Ogni istruttore e maestro non necessariamente deve tenere dei corsi in una o più palestre.

- Far creare agli istruttori e maestri appartenenti alla federazione eventi come stage o corsi istruttore. I corsi istruttore saranno creabili solo dai maestri. Gli stage saranno creabili sia da maestri che da istruttori. Dopo la creazione potranno decidere anche di modificarne i dati o eliminarli e visualizzarne gli iscritti.
- Ogni maestro, oltre alle attività già citate, potrà svolgere operazioni più gestionali, come ad esempio inserire nuove discipline e palestre, modificarne i dati ed eliminare quelle già esistenti nella federazione. Potrà anche ricercare gli utenti presenti nella federazione per modificarne i dati o rimuoverlo dal database.

Al primo avvio dell'applicazione web sarà necessario fornire un maestro, in quanto è l'utente che possiede tutti i privilegi.

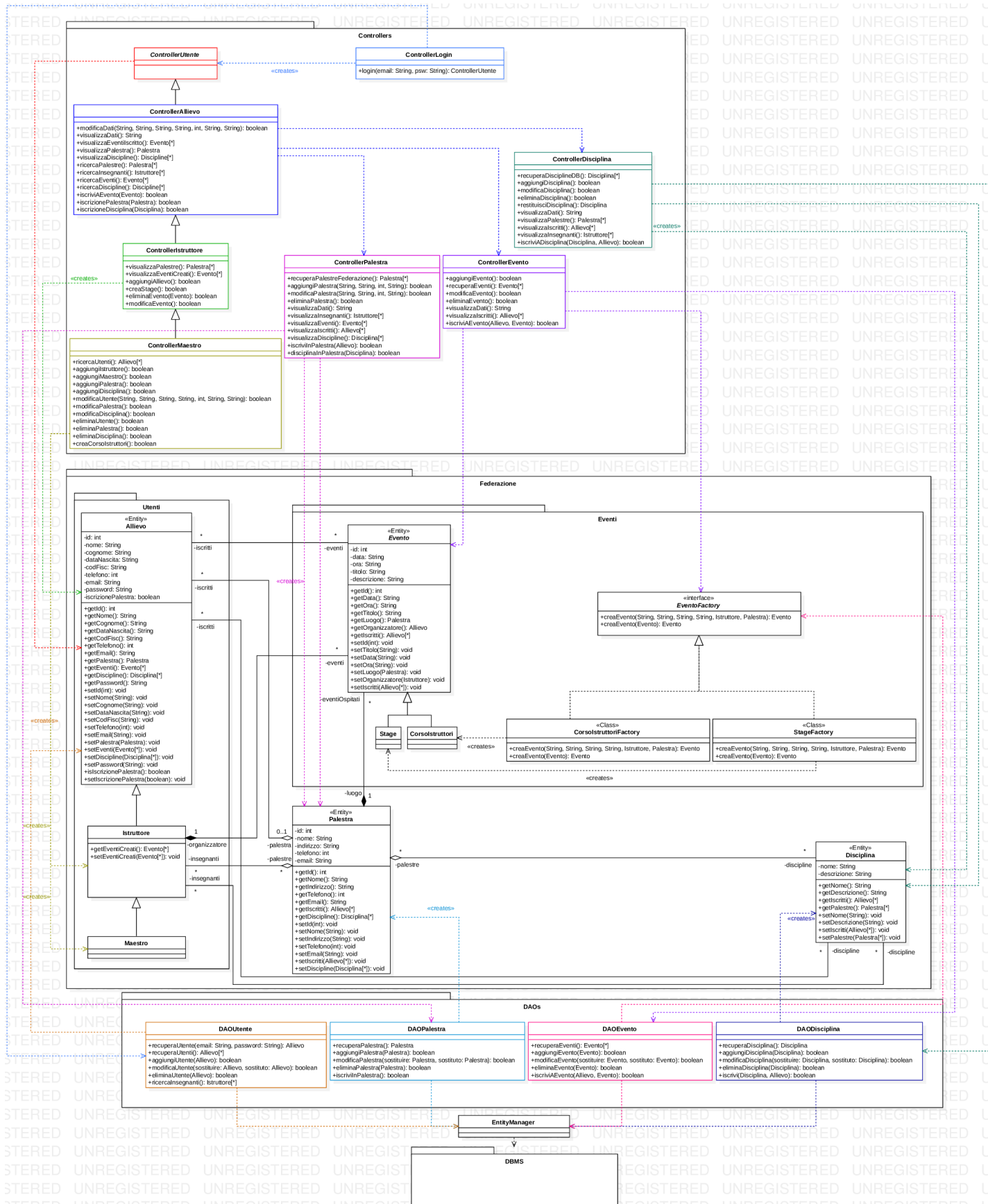
Chapter 2

Progettazione

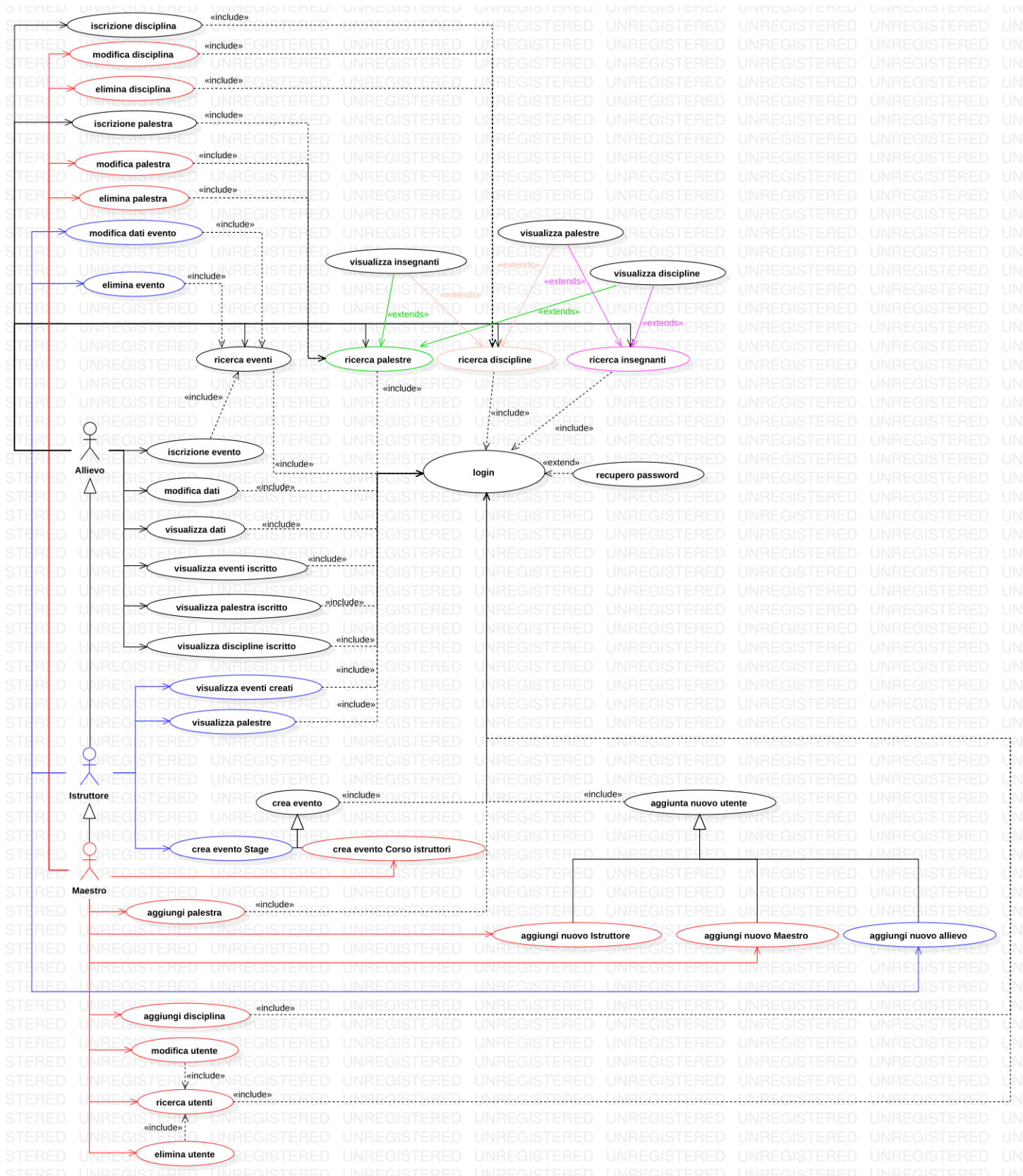
2.1 Software utilizzati

Per progettare il class diagram implementativo e lo use case diagram è stato utilizzato StarUML. Per i mockups è stato utilizzato Balsamiq. Come IDE per scrivere codice Java è stato usato NetBeans.

2.2 Class diagram



2.3 Use case diagram



2.4 Templates

2.4.1 Modifica dati

UC#1	Modifica dati
Scope	
Level	User goal
Actor	Allievo / Istruttore / Maestro
Basic course	<ol style="list-style-type: none"> 1. L'Utente effettua il login (mockup Pagina Login) 2. Una volta effettuato il login preme sul pulsante "Visualizza dati" (mockup Homepage) 3. Preme sul pulsante in basso a sinistra "Modifica dati" (mockup Pagina Visualizza dati) 4. Riempie il form con i nuovi dati che vuole modificare e preme il pulsante "Conferma" (mockup Pagina Modifica dati) 5. Verrà visualizzata una pagina di conferma di avvenuta modifica (mockup Pagina Conferma cambiamento dati)
Alternative course 1	1a. L'utente preme su "Recupera password", sarà inviata una mail con la password
Alternative course 4	4a. Se i dati inseriti non rispettano le convenzioni rispettivi ad ogni campo verrà creato un messaggio di errore e rimarrà sulla stessa pagina per rieseguire l'immissione dei dati

Figure 2.1: Modifica dati Template

2.4.2 Iscrizione evento

UC#2	Iscrizione evento
Scope	
Level	User goal
Actor	Allievo / Istruttore / Maestro
Basic course	<ol style="list-style-type: none">1. L'Utente effettua il login (mockup Pagina Login)2. Una volta effettuato il login preme sul pulsante "Ricerca eventi" (mockup Homepage)3. Seleziona col mouse l'evento di interesse e preme il pulsante in basso a sinistra "Seleziona evento" (mockup Pagina Ricerca eventi)4. Verranno mostrati i dati dell'evento e un pulsante in basso a sinistra "Iscriviti" con cui iscriversi (mockup Pagina Descrizione evento)5. Verrà visualizzata una pagina di conferma di avvenuta iscrizione (mockup Pagina Iscrizione confermata)
Alternative course 1	<ol style="list-style-type: none">1a. L'utente preme su "Recupera password", sarà inviata una mail con la password con cui accedere

Figure 2.2: Iscrizione evento Template

2.4.3 Modifica nome palestra

UC#3	Modifica nome palestra
Scope	
Level	User goal
Actor	Maestro
Basic course	<ol style="list-style-type: none"> 1. L'Utente effettua il login (mockup Pagina Login) 2. Una volta effettuato il login preme sul pulsante "Gestione palestre" (mockup Homepage Maestro) 3. Preme sul pulsante "Modifica palestra" (mockup Pagina Gestione Palestre) 4. Seleziona col mouse la palestra da modificare dalla lista e preme il pulsante in basso a sinistra "Seleziona palestra" (mockup Pagina Ricerca palestra) 5. Verranno visualizzati gli attuali dati della palestra, preme quindi il pulsante in basso a sinistra "Modifica dati" (mockup Pagina Dati palestra) 6. Inserisce il nuovo nome nella casella di testo "Nome" e preme il pulsante "Conferma" (mockup Pagina Modifica palestra) 7. Verrà visualizzata una pagina di conferma di avvenuta modifica (mockup Pagina Conferma modifica palestra)
Alternative course 1	1a. L'utente preme su "Recupera password", sarà inviata una mail con la password
Alternative course 6	6a. Se i dati inseriti non rispettano le convenzioni rispettivi ad ogni campo verrà creato un messaggio di errore e rimarrà sulla stessa pagina per rieseguire l'immissione dei dati

Figure 2.3: Modifica nome palestra Template

2.5 Mockups

Modifica dati

Federazione X

https://www.federazionex.org/login.html

Federazione X

Scegli un'operazione:

Ricerca eventi Ricerca palestre

Visualizza dati

Ricerca insegnanti Ricerca discipline

Figure 2.4: Pagina Login

Figure 2.5: Homepage

Federazione X

https://www.federazionex.org/visualizza-dati.html

Federazione X

Modifica dati personali:

Nome Cognome Data di nascita Codice fiscale Telefono Email Password

1A 12 10/10/2000 1234 00000 123@univ.it 1234

Conferma

Figure 2.6: Pagina Visualizza dati

Figure 2.7: Pagina Modifica dati



Federazione X

Modifica dati personali:

Nome Cognome Data di nascita Codice fiscale Telefono Email Password

✓ Dati modificati!

Figure 2.8: Pagina Conferma cambiamento dati

Iscrizione evento



Federazione X

Email: Password: Recupero password

Login

Figure 2.9: Pagina Login



Federazione X

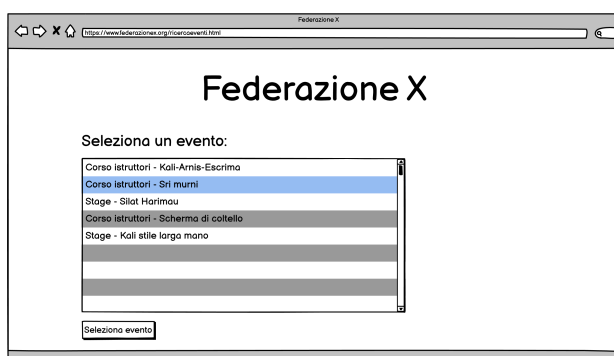
Scegli un'operazione:

Ricerca eventi Ricerca palestre

Visualizza dati

Ricerca insegnanti Ricerca discipline

Figure 2.10: Homepage



Federazione X

Seleziona un evento:

Corso istruttori - Kali-Arnis-Escrima

Corso istruttori - Sri mumi

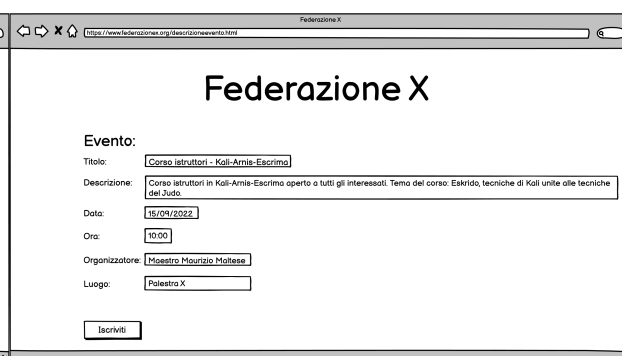
Stage - Silat Harimau

Corso istruttori - Scherma di coltello

Stage - Kali stile largo mano

Seleziona evento

Figure 2.11: Pagina Ricerca eventi



Federazione X

Evento:

Titolo: Corso istruttori - Kali-Arnis-Escrima

Descrizione: Corso istruttori in Kali-Arnis-Escrima aperto a tutti gli interessati. Tema del corso: Eskrido, tecniche di Kali unite alle tecniche del Judo.

Data: 15/04/2022

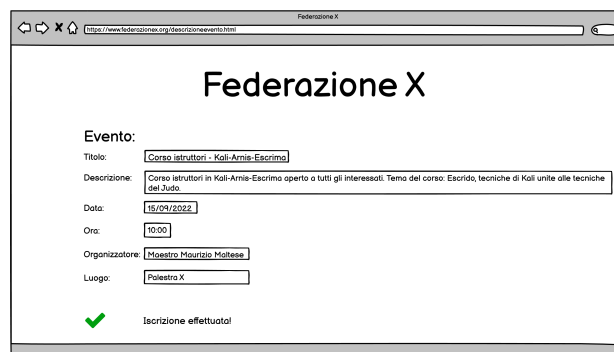
Ora: 10.00

Organizzatore: Maestro Maurizio Maltese

Luogo: Palestra X

Iscriviti

Figure 2.12: Pagina Descrizione evento



Federazione X

Evento:

Titolo:

Descrizione:

Data:

Ora:

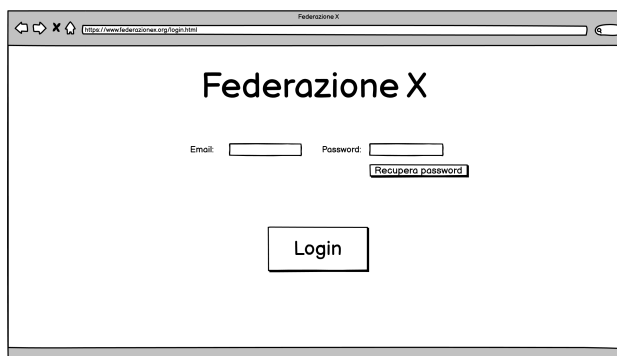
Organizzatore:

Luoogo:

✓ Iscrizione effettuata!

Figure 2.13: Pagina Iscrizione confermata

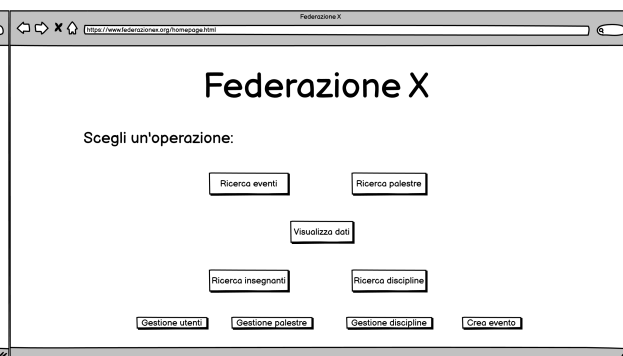
Modifica nome palestra



Federazione X

Email: Password:

Figure 2.14: Pagina Login



Federazione X

Scegli un'operazione:

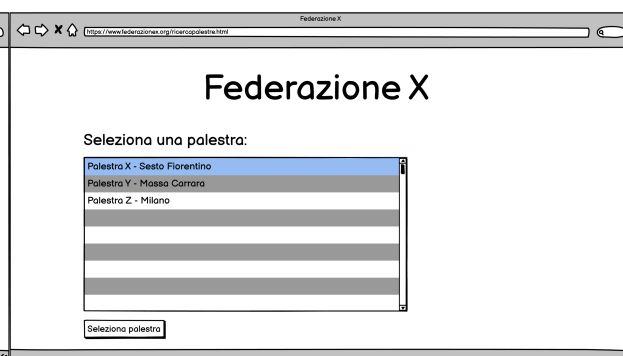
Figure 2.15: Homepage Maestro



Federazione X

Gestione palestre:

Figure 2.16: Pagina Gestione palestre



Federazione X

Seleziona una palestra:

Figure 2.17: Pagina Ricerca palestra

Figure 2.18: Pagina Dati palestra

Figure 2.19: Pagina Modifica palestra

Figure 2.20: Pagina Conferma modifica palestra

2.6 Descrizione classi

2.6.1 Package Federazione

In questo package sono presenti tutte le entità presenti in una federazione e che dovranno essere quindi salvate all'interno di un database, come tali possiedono tutte un id intero se non diversamente specificato.

Allievo

La classe Allievo è una Entità e rappresenta la classe base da cui deriveranno tutti gli altri utenti, ovvero Istruttore e Maestro, in quanto anche questi ultimi sono degli allievi a loro volta che possono iscriversi in delle palestre per imparare nuove discipline. Come attributi conterrà i dati anagrafici e di contatto, oltre a riferimenti ad eventi, discipline e palestra a cui è iscritto, in particolare conterrà anche l'attributo "iscrizionePalestra" che serve ad indicare se è già iscritto in una palestra, in caso lo fosse non potrà iscriversi ad una nuova palestra. I metodi che possiede sono solo getter e setter in

quanto svolge il ruolo di entità le cui funzioni sono esposte dal relativo controller (ControllerAllievo). Questi metodi in particolare saranno utilizzati dal rispettivo Data Access Object (DAOUtente) per poter inizializzare l'entità quando messa in vita a runtime.

Istruttore

Deriva dalla classe Allievo. In aggiunta possiede riferimenti alle palestre in cui insegna e agli eventi creati, oltre che ai metodi getter e setter di quest'ultimi.

Maestro

Deriva da Istruttore. Non possiede riferimenti in più rispetto all'istruttore, è però necessaria in quanto il fatto di essere un maestro predispone ad avere delle funzionalità in più esposte nel rispettivo controller (ControllerMaestro).

Evento

La classe Evento è una Entità ed è astratta, rappresenta tutti i dati necessari ad un evento come il titolo, descrizione, orario, data, luogo e organizzatore. Possiede gli unici metodi getter e setter usati per mettere in vita una istanza di quest'ultimo da parte del rispettivo Data Access Object (DAOEvento).

Stage

Deriva dalla classe Evento, non aggiunge funzionalità, è un tipo di evento creabile sia da istruttori che maestri.

CorsoIstruttori

Deriva dalla classe Evento, non aggiunge funzionalità, è un evento creabile solo da maestri.

EventoFactory

Interfaccia che espone due metodi per poter creare una istanza di un Evento attraverso il passaggio di tutti gli attributi che questo deve possedere oppure come copia di un altro evento. Non possiede default implementations.

StageFactory

Classe che implementa l'interfaccia EventoFactory. Crea una istanza di un evento Stage.

CorsoIstruttoriFactory

Classe che implementa l'interfaccia EventoFactory. Crea una istanza di un evento CorsoIstruttori.

Palestra

La classe Palestra è una Entità che possiede gli attributi come il nome, indirizzo e contatti, possiede anche riferimenti alle discipline che possiede, gli insegnanti, gli iscritti e gli eventi che ospita.

Disciplina

La classe Disciplina è una Entità atta a descrivere una disciplina, possiede un nome (utilizzato anche come chiave primaria non essendoci discipline diverse con lo stesso nome al posto di un id intero) e una descrizione, possiede inoltre riferimenti agli iscritti a quella disciplina, agli insegnanti e alle palestre in cui è insegnata.

2.6.2 Package DAOs

Tutte le classi all'interno di questo package sono necessarie per svolgere tutte le operazioni CRUD tipiche di un database oltre che ad alcune funzionalità aggiuntive per ogni DAO. Tutte quante invieranno i propri comandi ad un EntityManager che si interfacerà direttamente con un DBMS.

DAOUtente

DAOUtente è una classe che gestisce le operazioni CRUD per gli utenti, come gli allievi, istruttori e maestri, è infatti in grado di reperire i dati per creare istanze di queste tipologie oltre che aggiungerne di nuovi, modificarli o eliminarli. Una funzionalità aggiuntiva è quella di reperire tutti gli insegnanti presenti nella federazione e tutti gli utenti in una sola volta.

DAOPalestra

DAOPalestra è una classe che gestisce le operazioni CRUD per le palestre. Come funzionalità aggiuntiva alle CRUD può iscrivere un utente ad una palestra.

DAOEvento

DAOEvento è una classe che gestisce le operazioni CRUD per gli eventi. Come funzionalità aggiuntiva alle CRUD può iscrivere un utente ad un evento.

DAODisciplina

DAODisciplina è una classe che gestisce le operazioni CRUD per le discipline. Come funzionalità aggiuntiva alle CRUD può iscrivere un utente ad una disciplina.

2.6.3 Package Controllers

In questo package sono presenti le classi controller che espongono le funzionalità all'esterno dell'applicazione, rappresentano quindi la business logic e la linea di separazione tra backend e frontend.

ControllerLogin

ControllerLogin è la prima classe di cui si va a creare una istanza al momento dell'avvio dell'applicazione, necessaria per poter far autenticare un utente e poter creare il controller corretto in base a quale utente ha fatto l'accesso. Possiede un riferimento al DAOUtente per poter recuperare l'utente e svolgere le operazioni di autenticazione. In base a cosa gli è stato restituito creerà una derivata di ControllerUtente che funge da classe base per tutti gli altri controller degli utenti.

ControllerUtente

ControllerUtente è una classe astratta che possiede come unico attributo un riferimento alla classe Allievo con visibilità protected, in quanto ogni sua sottoclasse si occuperà di gestire le operazioni sull'utente che si è autenticato.

ControllerAllievo

ControllerAllievo è una classe derivata da ControllerUtente e possiede dei riferimenti al DAOUtente che richiamerà ogni volta che dovrà svolgere operazioni di modifica, come la modifica dei dati, e di recupero della lista degli insegnanti della federazione. Possiede un riferimento al ControllerPalestra tramite il quale può richiedere la lista delle palestre della federazione e visualizzare i dati di ogni palestra selezionata ed eventualmente iscriversi.

Possiede un riferimento a `ControllerEvento` tramite il quale può visualizzare gli eventi presenti, visualizzarne i dati e iscriversi a questi. Possiede inoltre un `ControllerDisciplina` tramite il quale può visualizzare le discipline disponibili nella federazione e iscriversi a quest'ultime in caso però siano iscritti ad una palestra e presenti anche in quest'ultima.

ControllerIstruttore

`ControllerIstruttore` è una classe derivata da `ControllerAllievo` e possiede metodi per poter creare `Stage` oltre che all'eliminazione e modifica di un evento creato. Inoltre possiede la capacità di poter aggiungere un nuovo allievo al database.

ControllerMaestro

`ControllerMaestro` è una classe che deriva da `ControllerAllievo` e permette ogni tipo di operazione sul database in quanto un maestro possiede i massimi privilegi di amministrazione della federazione.

2.7 Design patterns

2.7.1 Factory method Pattern

Dovendo creare due tipologie di eventi e non volendo esplicitare quale dei due andare a creare ho deciso di creare una interfaccia che esponesse i metodi necessari alla creazione degli eventi, in particolare un metodo serve per creare un evento da zero, l'altro serve per poter creare un evento come copia di un altro, in particolare viene utilizzato nella logica di modifica dei dati di un evento. Le classi che implementano i metodi che creano l'evento nello specifico sono `CorsoIstruttoriFactory` e `StageFactory`, le istanze di queste due classi le ho pensate come passate dalla pagina web, la quale in base alla pagina in cui si trova, ovvero di creazione evento stage o creazione evento corso istruttori, passerà al controller eventi la factory corretta insieme ai dati dell'evento e creerà quindi l'evento corretto che salverà poi nel database.

2.7.2 Front Controller Pattern

Non volendo esporre in modo diretto all'utente le entità disponibili nel domain model e volendo far gestire le richieste di quest'ultimi ad un handler invece che direttamente dalle istanze delle entità ho deciso di usare questo pattern tramite il quale saranno gestite le richieste degli utenti e in base a ciò

che viene richiesto eseguono un'operazione od un'altra. In base alla pagina in cui ci troviamo, saranno presenti o meno determinati controller che andranno a gestire le varie entità. Non è stato utilizzato il pattern Facade in quanto il controller non solo delega determinate operazioni agli interessati ma può eseguire anche ulteriori operazioni. Nel nostro caso i controller sono molto simili a delle facade se non per pochissimi metodi come la modifica dei dati in cui c'è della logica in più di controllo dei dati in ingresso prima di decidere se richiamare il metodo che effettivamente eseguirà la modifica dei dati.

2.7.3 Data Access Object Pattern

Essendoci la necessità di comunicare con un database ho deciso di spostare tutta la logica in delle classi apposite che avessero il compito di gestirne l'accesso ed eseguire le operazioni tipiche CRUD più delle operazioni specifiche per ogni Data Access Object. Ne sono presenti uno per ogni entità del domain model.

Chapter 3

Testing

Per l'esecuzione dei test è stato utilizzato JUnit4 offerto direttamente dall'IDE NetBeans. Sono stati testati solo i codici riguardanti gli use case esposti anche come template nella sezione 2.4, in particolare vengono quindi testati i controller `ControllerAllievo` e `ControllerMaestro` al cui interno sono comunque utilizzati i DAO ed altri controller, ho quindi supposto che questi funzionassero. Nel test di modifica dei dati dovendo confrontare che due oggetti, quello modificato attuale e quello che mi aspetto di avere, siano uguali ho dovuto eseguire l'override del metodo `equals` della classe `Object`, in quanto il metodo `assertEquals` utilizza proprio questo metodo che nella sua versione di default, escluso che per le stringhe, va a confrontare solo i riferimenti dei due oggetti passati, invece nel mio caso era necessario che fossero controllati gli attributi e che questi corrispondessero tra loro.

3.1 Codici test

3.1.1 Test ControllerAllievo

```
public class ControllerAllievoTest {

    private static ControllerAllievo ca;
    private static ControllerEvento ce;

    public ControllerAllievoTest() {
    }

    @BeforeClass
    public static void setUpClass() {
        String nome = "Msttia";
        String cognome = "Pardeo";
        String dataNascita = "17-04-1996";
        String codFisc = "PRDMTT96D17D575E";
        String telefono = "3xxxxxxxxx";
        String email = "mattia.pardeo@stud.unifi.it";
        String password = "MattiaP.";
        ca = new ControllerAllievo(new Allievo(nome, cognome, dataNascita,
            codFisc, telefono, email, password), new DAOUtente());
        //Controller creato dalla pagina in realtà e poi settato
        ce = new ControllerEvento(null);
        ca.setControllerEvento(ce);
    }

    @Test
    public void testModificaDati() {
        //Setup
        String nome = "Mattia";
        String cognome = "";
        String dataNascita = "";
        String codFisc = "";
        String telefono = "";
        String email = "";
        String password = "";
        Allievo expected = new Allievo(ca.utente);
        expected.setNome(nome);

        //Test
        ca.modificaDati(nome, cognome, dataNascita, codFisc, telefono,
            email, password);

        Allievo actual = ca.utente;

        assertEquals(expected, actual);
    }

    @Test
    public void testIscriviAEvento(){

        //Evento in realtà inviato dalla pagina web al controller eventi
        Evento testStage = new Stage("12-12-2022", "10.30", "Stage",
            "Stage di Karambit", new Istruttore(), new Palestra());

        ca.iscriviAEvento(testStage);
        assertTrue(testStage.getIscritti().contains(ca.utente));
    }
}
```

3.1.2 Test ControllerMaestro

```

public class ControllerMaestroTest {

    public static ControllerMaestro cm;

    public ControllerMaestroTest() {
    }

    @BeforeClass
    public static void setUpClass() {
        String nome = "";
        String cognome = "";
        String dataNascita = "";
        String codFisc = "";
        String telefono = "0";
        String email = "";
        String password = "";
        cm = new ControllerMaestro(new Maestro(nome, cognome, dataNascita,
            codFisc, telefono, email, password), new DAOUtente());
    }

    @Test
    public void testModificaNomePalestra() {

        //Set up di palestra e recupero, in realtà saranno riprese da una
        // lista dalla pagina web
        String nome = "Renbukan";
        String indirizzo = "";
        int telefono = 0;
        String email = "";
        //Supponendo di aver già testato ControllerPalestra
        ControllerPalestra cp = new ControllerPalestra(new DAOPalestra());
        cp.aggiungiPalestra(nome, indirizzo, telefono, email);
        Palestra actual = cp.recuperaPalestreFederazione().get(0);

        nome = "Renbukan";
        Palestra expected = new Palestra(actual);
        expected.setNome(nome);

        //Test
        cm.setControllerPalestra(cp);
        cm.modificaPalestra(actual, nome, indirizzo, telefono, email);

        actual = cp.recuperaPalestreFederazione().get(0);

        assertEquals(expected, actual);
    }
}

```