UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

DOCUMENTAZIONE

Componenti del gruppo:

Carminati Matteo n°matricola: 1066354

Torri Lorenzo n°matricola: 1069047

CAPITOLO 3 – Software Life Cycle

La scelta del modello di vita del nostro software per quanto riguarda la gestione del lavoro del team e la suddivisione dei compiti, è di tipo agile, in particolare XP. Questa scelta è legata alla struttura del team, il quale è composto esclusivamente da due persone, e alla necessità di apportare al progetto rapidi e funzionali cambiamenti.

Per certi aspetti Extreme Programming è ottimale per il team in quanto la gestione del lavoro si basa interamente sul pair programming. I due componenti del team infatti lavorano per la maggior parte del tempo insieme riuniti in calls di Microsoft Teams. Entrambi conoscono perfettamente ogni aspetto del progetto sviluppato. Ogni componente del team è in grado di modificare il codice in quanto lo conosce e lo sa manipolare correttamente.

Per lo sviluppo del codice i due sviluppatori hanno deciso di dividersi il lavoro, tuttavia le modifiche saranno sempre sottoposte a verifica da parte dell'altro membro. Tali verifiche saranno effettuate durante brevi riunioni da remoto in modo che si possa avere un confronto diretto e immediato.

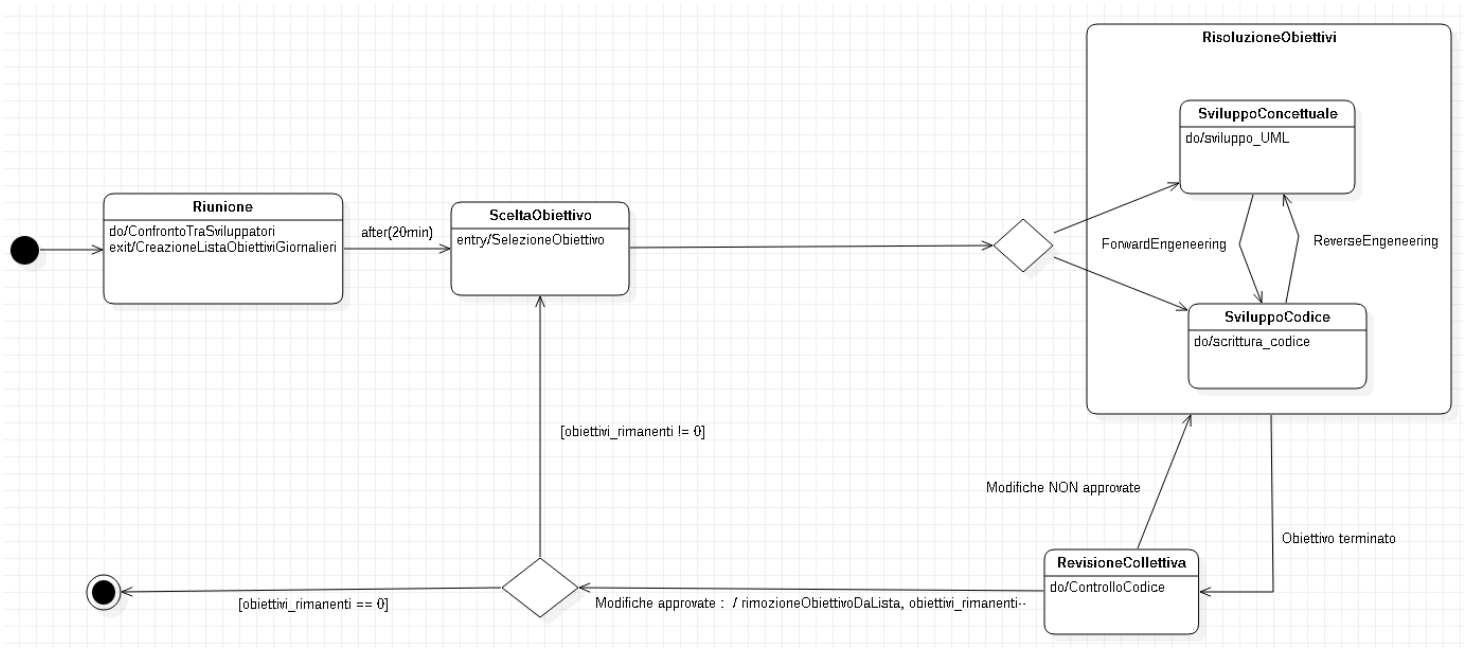
L'idea è quella di iniziare a lavorare su una prima versione basilare ma funzionante dell'applicazione, per poi rilasciare costantemente delle piccole versioni del sistema che vadano a migliorare la situazione di partenza e che permettano di verificare le funzionalità implementate.

Al termine dell'introduzione di una nuova funzionalità all'interno della applicazione bisognerà fare alcuni test. La fase di testing ha un ruolo chiave e verrà affrontata con più precisione in seguito (capitolo 13).

Il focus principale del team è la parte di progettazione e realizzazione del software, ciò nonostante verrà comunque implementata una parte grafica tramite Java Swing. Pertanto tutte le analisi che verranno condotte e quindi riportate non riguarderanno la parte grafica.

Come ultima precisazione il team predilige, come già specificato nel Project Plan al punto 8, un approccio di tipo Model driven architecture, dunque, quando possibile, i cambiamenti dell'applicativo vengono effettuati sui modelli UML implementati e non direttamente sul codice.

La software life cycle è stata parzialmente implementata da parte degli sviluppatori anche mediante formalizzazione eseguita con il diagramma statechart del tool UML.



CAPITOLO 5 – People Management and team Organization

L'organizzazione del team e la gestione delle persone che ne fanno parte è riconducibile perlopiù all'approccio agile, i due sviluppatori infatti collaborano in ogni aspetto del loro lavoro, applicando anche pair programming.

Per certi aspetti la gestione dei membri del team viene affrontata seguendo lo schema proposto da Mintzberg; in particolare vi sono alcune analogie con il meccanismo di coordinazione di adhocracy. L'organizzazione del lavoro avviene infatti prediligendo meeting e momenti di scambio di idee tra i due membri del team.

Altro aspetto tipico di adhocracy, e ripreso dal team di sviluppo, è la scelta di coordinare il lavoro attraverso mutual adjustment che i membri svolgono con molta regolarità.

Tali correzioni avvengono però tenendo sempre presente quelli che sono gli obiettivi prefissati, ponendo il focus sul prodotto finale considerato nella sua completezza e mai suddiviso in parti.

Per quanto riguarda la strutturazione del team di lavoro, si specifica che all'interno del team non esiste alcun tipo di gerarchia o di rigida suddivisione dei compiti.

Questo approccio è stato ritenuto quello più adatto poiché i due componenti del team hanno percorsi formativi e skills molto simili, pertanto svolgono compiti analoghi e godono entrambi di grande autonomia non dovendo rispondere alle direttive di un superiore.

Lo svolgimento del lavoro segue sempre gli standard legati ai linguaggi utilizzati.

Il team ha solo specifiche indicative riguardanti l'ammontare di ore di lavoro richieste per lo sviluppo della piattaforma.

Per poter organizzare meglio il workflow gli sviluppatori hanno deciso di utilizzare il servizio Kanban integrato a Github che permette, con le sue boards, di visualizzare e scegliere gli obiettivi di giornata, estratta dalla lista generale contenente tutti gli obiettivi del progetto.

CAPITOLO 6 - Software Quality

Il cliente non ha specificato dei requisiti di qualità relativi alla piattaforma richiesta; pertanto, questi sono stati determinati direttamente dal team di lavoro.

In particolare i due sviluppatori hanno deciso di seguire le direttive del modello McCall per avere dei criteri qualitativi da rispettare durante il lavoro svolto.

Alcuni parametri qualitativi saranno valutati in fase di sviluppo, altri necessariamente verranno presi in considerazione solo al termine dell'applicativo, o comunque durante le fasi finali di sviluppo.

Si riportano alcuni fattori di qualità perseguiti dal team e classificati secondo il modello McCall:

- Product operation
  - Correctness
    Il team vuole garantire un applicativo correttamente funzionante e che rispetti i requisiti riposti
  - Reliability
    Per la reliability si rimanda alle considerazioni fatte durante la fase di Software Testing del CAPITOLO 13
  - Efficiency
    La piattaforma non richiede particolari risorse, per cui ci si aspetta che possa performare in maniera uguale su ogni dispositivo, indipendentemente dall'hardware di quest'ultimo.
  - Integrity
    Non ci sono criticità particolari nell'ambito della sicurezza.
  - Usability
    La piattaforma, per poter essere eseguita, necessita di una JVM installata sul dispositivo.
- Product revision
  - Maintainability
    Per la mantainability si rimanda alle considerazioni fatte durante la fase di Software Maintainance del CAPITOLO 14
  - Testability
    Per la testability si rimanda alle considerazioni fatte durante la fase di Software Testing del CAPITOLO 13

- Product transition
    - Portability
      E' possibile utilizzare la piattaforma su qualsiasi dispositivo che abbia installata una JVM.

CAPITOLO 9 – Requirements Engineering

Il lavoro non è stato commissionato dal Presidente dell'associazione, ma è una proposta che i due sviluppatori hanno deciso di presentare. Pertanto non sono stati forniti dal Presidente requisiti particolari inerenti alla piattaforma stessa. Il team tuttavia, basandosi anche sull'esperienza dei singoli membri nell'ambito delle prenotazioni di strutture sportive, ha provveduto a stilare una serie di requisiti che vorrebbero vedere implementati in una piattaforma di questo genere.

I requisiti nella fase di elicitation sono stati organizzati in ordine gerarchico seguendo il modello Moscow:

- Must have
    - L'utente ha la possibilità di scegliere lo sport che vuole praticare.
    - L'utente ha la possibilità di scegliere l'orario di prenotazione.
    - L'utente ha la possibilità di scegliere la struttura che vuole prenotare.
    - La piattaforma cerca strutture con le indicazioni dell'utente.
    - L'utente conferma la prenotazione.
    - La piattaforma calcola il costo della prenotazione.

- Should have
    - L'utente ha la possibilità di scegliere se desidera avere lo spogliatoio o no.
    - Le strutture già registrate possono registrare nuovi campi.
    - Le strutture già registrate possono registrare nuovi spogliatoi.
    - La piattaforma distingue tra clienti standard e clienti convenzionati.
    - La piattaforma calcola il costo della prenotazione rispetto alla tipologia di cliente.

- Could have
    - La piattaforma, qualora il campo non sia disponibile, propone una serie di strutture disponibili e che rispettano i canoni indicati precedentemente dall'utente.
    - Le strutture possono bandire utenti sgraditi.

- Won't have
    - La piattaforma informa l'utente se la struttura fornisce al cliente materiale tecnico oppure no.
    - L'utente ha la possibilità di disdire prenotazioni già effettuate.

Il team ha deciso inoltre di riferirsi allo standard IEEE 830 per la specifica dei requisiti al fine di definirli in maniera più ordinata e precisa.

1. *Introduction*
    1.1 Purpose
    Questo documento riporta i requisiti definiti per la realizzazione di un prototipo di una piattaforma di prenotazioni online di attività sportive. Il documento costituisce una linea guida per il successivo sviluppo dell'applicativo stesso.
    1.2 Scope
    Questo prototipo vuole creare una prima simulazione semplificata di quella che potrebbe essere una piattaforma più completa ed elaborata, che possa permettere agli utenti frequentatori di centri sportivi di poter prenotare con facilità i campi adibiti a tale scopo. Ogni struttura possiede dei campi e degli spogliatoi, inoltre le strutture hanno la possibilità di aggiungere nella piattaforma eventuali nuovi campi e spogliatoi.
    Le strutture possono impedire che utenti indesiderati possano effettuare prenotazioni presso i loro campi. La piattaforma ha come obiettivo futuro quello di riunire il maggior numero possibile di strutture sportive della provincia di Bergamo, così da poter offrire un servizio migliore.
    1.3 Definition, acronyms and abbreviations

Utente standard: cliente frequentatore di centri sportivi che vuole ricercare un campo ed effettuare una prenotazione.

Utente convenzionato: cliente uguale a quello standard, ma dispone di uno sconto nel calcolo del costo totale della prenotazione dopo cinque prenotazioni effettuate presso la medesima struttura.

Campi: impianto di proprietà di una certa struttura e prenotabile dall'utente per praticare un determinato sport.

Spogliatoio: locale di proprietà di una certa struttura messa a disposizione ai clienti per cambiarsi.

Strutture: centro sportivo formato da campi e spogliatoi.

Ban: una struttura può decidere di impedire a determinati utenti indesiderati di effettuare prenotazioni a loro nome presso gli impianti gestititi dalla suddetta struttura.

1.4 Overview

La sezione 2 serve per dare una visione generale del funzionamento della piattaforma, mentre la sezione 3 da maggiori specifiche riguardo ai requisiti dell'applicativo. All'interno dello standard sono stati riportati solamente i requisiti identificati come must have e should have dal modello Moscow, gli altri requisiti di could e won't have verranno presi in considerazione solo qualora ci fosse tempo per implementarli.

2. Overall description

2.1 Product Perspective

Il prototipo verrà interamente sviluppato in linguaggio Java. Non avrà nessuna interazione con altri sistemi, dunque tutte le informazioni e i salvataggi legati all'applicativo saranno conservati mediante strutture proposte dal linguaggio Java senza l'ausilio di sistemi di database.

2.2 Product function

Il sistema mette a disposizione queste funzionalità

- L'utente può ricercare un campo in una determinata struttura, a un orario stabilito per poter praticare un certo sport
- L'utente può prenotare oltre a un campo anche uno spogliatoio
- La piattaforma calcola preventivamente il costo totale della prenotazione basandosi sulla tipologia di utente (convenzionato o non convenzionato)
- Le strutture possono registrare nuovi spogliatoi e campi
- Le strutture possono controllare le prenotazioni dei loro campi e spogliatoi

2.3 Constraints

Le strutture per poter entrare nella loro zona dedicata necessitano di un identificativo che viene fornito dall'applicativo nel momento in cui vengono inserite per la prima volta nel sistema.

Il cliente invece, per poter effettuare le prenotazioni ed essere identificato in maniera univoca, deve indicare il suo codice fiscale.

3. Specific requirements

3.1 Functional requirements

3.1.1 Selezione tipologia

Il sistema propone un menù iniziale da cui ci si identifica come cliente o come struttura. In base alla selezione fatta, il sistema entra nella sezione dedicata. Ad ogni momento è possibile tornare al menù iniziale.

3.1.2 Funzioni utente

3.1.2.1 Ricerca Campo

Questa funzionalità permette all'utente di ricercare un campo sportivo in un determinato giorno e orario presso una struttura e per praticare un determinato sport.

3.1.2.2 Prenotazione Campo e Spogliatoio

Si permette all'utente di prenotare il campo ottenuto dalla funzione ricerca campo utilizzando il proprio codice fiscale. Qualora l'utente fosse interessato anche a utilizzare uno spogliatoio, la piattaforma verifica se la struttura ha a disposizione uno stesso nell'orario specificato e in caso positivo lo prenota.

3.1.3 Funzioni struttura

3.1.3.1 Identificazione Struttura

Per entrare nella propria area dedicata, una struttura deve superare una fase di identificazione, dove è necessario inserire un identificativo che le è stato fornito dal sistema quando è stata inserita nello stesso.

3.1.3.2 Aggiunta Campi e/o Spogliatoi

Ogni struttura registrata nel sistema ha la possibilità di aggiungere nuovi campi e spogliatoi nel suo elenco dedicato.

3.1.3.3 Ban Utenti

La struttura ha la possibilità di vietare ulteriori prenotazioni da parte di un utente, identificato in maniera univoca dal suo codice fiscale.

3.1.3.4 Visualizzare Elenco Prenotazioni

Ogni struttura ha la possibilità di controllare tutte le prenotazioni effettuate presso i suoi campi e spogliatoi.

3.2 Performance requirements

Non esistono particolari limitazioni legate al sistema, se non quelli imposti dalle classi java che verranno impiegate per la realizzazione.

Non ci sono particolari vincoli temporali per quanto riguarda l'esecuzione, l'interesse è unicamente quello di assicurare la consistenza dei dati.

3.3 Software system attributes

3.3.1 Security

Dato che l'applicativo è un prototipo che non viene utilizzato contemporaneamente da più utenti, ma viene esclusivamente eseguito in locale e da un utente alla volta, non necessita di meccanismi di mutua esclusione per la gestione dell'accesso a risorse condivise, in quanto non esistono accessi concorrenti.

CAPITOLO 10 – Modelling

Per visualizzare i diagrammi UML si rimanda al file "Diagrammi UML.mdj".

USE CASE DIAGRAM

Il diagramma mostra gli attori che possono interagire con la Piattaforma e cosa possano fare nella stessa. In particolare l'attore Campo è astratto e generalizza i diversi campi adibiti ai diversi sport.

Il Cliente può effettuare una Prenotazione presso una Struttura.

La prenotazione effettua obbligatoriamente un controllo di disponibilità per il campo, per mostrare tale necessità è stata utilizzata un'associazione Include. La prenotazione può inoltre includere anche il controllo di disponibilità per uno spogliatoio qualora l'utente fosse interessato a prenotarlo. Per tale motivo si è scelta l'associazione Extend. Il cliente potrebbe avere diritto ad uno sconto nel momento in cui effettua la prenotazione, dunque è stata introdotta un'associazione di tipo extend tra Prenotazione e Sconto.

Le prenotazioni sono associate agli spogliatoi e ai campi. Tuttavia, dato che una struttura possiede i singoli campi e spogliatoi, nello use case diagram si associa la Prenotazione direttamente alla Struttura corrispondente.

Una Struttura ha la possibilità di aggiungere nuovi campi e nuovi spogliatoi, con la condizione che questo avvenga per una unità per volta.

Per ultimo una Struttura ha la possibilità di effettuare un Ban nei confronti di uno o più clienti.

CLASS DIAGRAM

Le classi che verranno presentate di seguito presentano attributi e metodi che verranno spiegati più in dettaglio nella sezione dedicata allo standard IEEE 1016 del CAPITOLO 12.

La classe struttura con i suoi attributi e metodi descrive le caratteristiche delle singole strutture sportive registrate nel sistema, salvate all'interno della classe Registro Strutture attraverso una lista.

La struttura ha inoltre una serie di Campi e Spogliatoi prenotabili dai Clienti. Non possono esistere campi o spogliatoi non associati a una struttura, di conseguenza usiamo una composition. Inoltre una prenotazione non può esistere senza un campo, quindi avrà anche in questo caso una composition, mentre nel caso degli

spogliatoi, possono esistere prenotazioni senza spogliatoio, di conseguenza si utilizza una semplice aggregazione condivisa.
La classe Campo è astratta poiché generalizza altre classi più specifiche che rappresentano le tipologie di campi sportivi che possono esserci in una struttura.
La classe Cliente rappresenta sempre un nuovo utente che utilizza la piattaforma e che vuole effettuare una nuova prenotazione, di conseguenza i metodi di ricerca per una prenotazione saranno implementati da questa classe.
La classe Prenotazione possiede un metodo calcolcaPrezzo(), il quale basandosi sull'attributo Prezzo delle istanze delle classi Campo e Spogliatoio associate alla prenotazione e sul possibile sconto applicabile al cliente, determina il costo totale da sostenere.

STATE MACHINE
Il sistema passa subito dall'initial state allo stato rinominato "Menu Iniziale", il quale rappresenta la situazione iniziale in cui non è ancora avvenuto l'accesso alla piattaforma come utente utilizzatore o come struttura registrata. Da "Menu iniziale" quindi, a seconda di chi sta utilizzando la piattaforma (un utente oppure un proprietario di una struttura), è possibile passare a diversi stati.
Il sistema può tornare in ogni momento allo stato "Menu Iniziale" nel caso in cui si effettui il logout.

Attività Struttura
In particolare, in caso di accesso da parte di una struttura, il sistema entra in uno stato di "Check ID" nel quale si verifica che la Password inserita coincida con uno degli ID delle strutture salvate nel sistema. Solamente nel caso in cui l'autenticazione vada a buon fine il sistema può passare nello stato "Attività_Struttura".
All'interno dello stato "Attività Struttura" sono stati definiti una serie di altri stati; inizialmente il sistema passa allo stato "Scelte_iniziali_struttura" nel quale, mediante un'entry activity, viene richiesto al proprietario della struttura di scegliere l'operazione da effettuare. A seconda di tale scelta il sistema può entrare in quattro possibili stati: "Ban_utente", "Aggiunta_campi", "Aggiunta_spogliatoi" e "Consulta_prenotazioni". In particolare nel caso dei primi tre state, il sistema torna allo stato "Scelte_iniziali_struttura" subito dopo aver terminato l'attività di entry; nel caso invece di "Consulta_prenotazioni" il sistema torna allo stato iniziale solo quando specificato dal proprietario della struttura.

Attività Utente
Nel caso in cui un utente abbia fatto accesso alla piattaforma, il sistema passa all'interno dello stato "Attività_Utente" nel quale si trovano diversi stati che rappresentano i passaggi richiesti per poter effettuare una prenotazione.
Per prima cosa quindi il sistema entra nello stato "Scelte_iniziali", dove l'utente setta i parametri della propria ricerca. Nel caso in cui l'utente abbia scelto una struttura da cui è stato bannato il sistema non potrà uscire da questo stato e l'utente sarà costretto a ridefinire i suoi parametri di prenotazione.
Si esce dallo stato "Scelte_iniziali" solo se l'utente ha indicato una struttura congrua (non è bannato) e si passa al fork. Si separa infatti la ricerca del campo da quella dello spogliatoio, consentendo però uno sviluppo parallelo delle due all'interno delle regioni dell'orthogonal state. La scelta è legata al fatto che la ricerca di un campo è obbligatoria, mentre quella dello spogliatoio è facoltativa e dovrà essere fatta in modo parallelo a quella del campo solo se specificatamente richiesto dall'utente.
Qualora il campo non dovesse essere libero, il sistema ritorna nello stato "Scelte_iniziali".
In caso contrario invece il sistema deve tenere conto di queste diverse situazioni:

- il cliente non richiedeva lo spogliatoio, allora il sistema procede al passo successivo
- il cliente aveva richiesto lo spogliatoio ed è libero, quindi la ricerca si riunisce con quella del campo con un join e il sistema procede al passo successivo
- il cliente aveva richiesto uno spogliatoio il quale però non è libero nell'orario specificato, di conseguenza si entra nello stato "Decisioni_spogliatoio", dove il cliente decide se prenotare comunque senza lo spogliatoio e quindi portare il sistema al passo successivo, oppure ritornare allo stato "Scelte_iniziali".

Al termine il sistema entra nello stato "Conferma_finale", dove calcola il costo della prenotazione.

SEQUENCE DIAGRAM

Il sequence diagram è stato suddiviso in tre diagrammi diversi rinominati: Login Struttura, Attività Struttura e Attività Utente.

Login Struttura

Questo diagramma rappresenta le sequenze di messaggi scambiati tra le tre lifeline necessarie per poter effettuare un login per una struttura (Struttura, Interfaccia Struttura e Registro Strutture).
Nel caso in cui l'ID inserito non sia presente nel sistema si rimane all'interno del loop e viene notificato un messaggio di errore.

Attività Struttura

Per poter svolgere le attività riservate alle strutture, è necessario prima superare l'identificazione, per questo è stato introdotto il blocco ref che richiama il Login Struttura.
Successivamente si entra nel loop che termina solo quando il proprietario della struttura effettua il logout dalla pagina dedicata. Al suo interno c'è un blocco alt che propone una scelta tra le varie attività che la lifeline indicata con Struttura può svolgere andando ad interagire con la sua interfaccia dedicata, la quale a sua volta interagisce con la corrispettiva classe Struttura.

Attività Utente

L'attività utente coincide con la fase di ricerca e successiva conferma della prenotazione. In particolare anche in questo caso si entra in un blocco loop che termina solo se viene confermata la prenotazione o se il cliente esce dalla sua zona dedicata. L'utente dovrà quindi settare i parametri di prenotazione desiderati, il sistema potrà effettuare la ricerca solo a settaggio terminato, in caso contrario infatti verrà catturata l'eccezione e restituito un messaggio di errore.
Al termine del controllo della disponibilità si mostreranno al cliente i risultati di tale ricerca; l'utente quindi potrà scegliere se confermare la prenotazione o annullarla.

ACTIVITY DIAGRAM

Il diagramma è suddiviso in due diverse partizioni che rappresentano le Attività dell'utente e le Attività della struttura. Si entra in una o nell'altra partizione a seconda della modalità di utilizzo che viene settata all'inizio (tramite set mode).
In particolare è possibile passare da una partizione ad un'altra attraverso degli accept event action che permettono di captare gli eventi di set mode struttura o utente, interrompendo tutte le attività di quella partizione e passando a quelle dell'altra.

Attività Utente

Si ripartisce l'attività dell'utente nelle varie parti del settaggio concorrenti tra loro dei dati e si passa alla fase di ricerca solo quando sono state settate tutte correttamente. Qualora anche solo un campo dovesse risultare non settato, si gestisce l'eccezione interrompendo le attività di settaggio e tornando alla fase precedente di ripartizione delle attività di settaggio.
Si procede in maniera analoga anche qualora la ricerca della prenotazione desiderata abbia esito negativo.
Al contrario in caso di esito positivo della fase di ricerca, si conferma la prenotazione e si terminano le attività della partizione "Attività Utente".

Attività Struttura

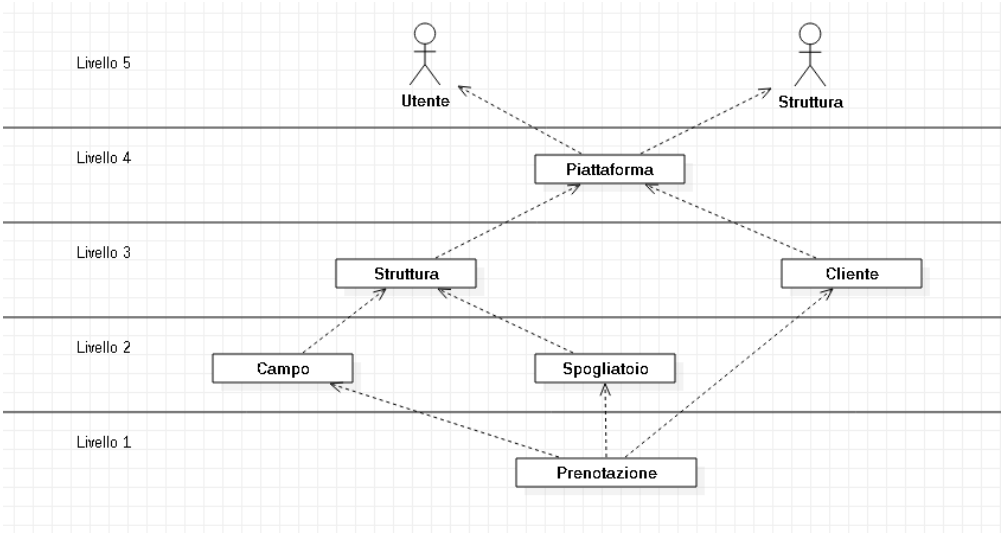
Nella partizione "Attività Struttura" si procede per prima cosa all'inserimento della Password; qualora quest'ultima non fosse corretta si solleva un'eccezione per tornare alla fase di inserimento della Password.
Qualora l'autentificazione avvenga con successo, si procede scegliendo la prossima attività da svolgere. Ciascuna di queste attività può essere terminata in qualsiasi momento, a tale scopo infatti è stato introdotto un accept event action rinominato "Torna al Menu Struttura" che permette di poter tornare alla scelta dell'azione da effettuare.

Nel caso in cui la scelta ricada sull'attività di Ban, qualora il codice fiscale indicato non dovesse esistere, allora si solleva un'eccezione che permette di interrompere l'attività e tornare alla fase di scelta.
Le altre attività invece non presentano criticità, pertanto non sono richiesti degli exception handler e una volta terminate possono concludersi anche tutte le attività della partizione "Attività Struttura".

CAPITOLO 11 – Software Architecture

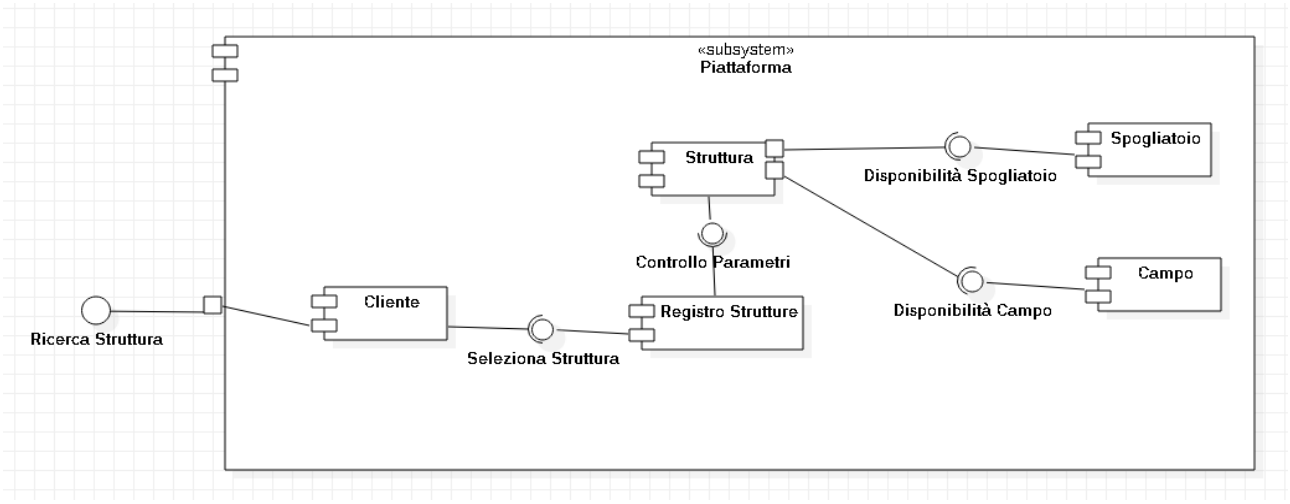
Il team ha deciso di sviluppare un'architettura del software seguendo lo standard IEEE 1471, per poter suddividere l'applicativo in componenti per poterne semplificare il funzionamento e come modello guida per la realizzazione della piattaforma.
Per l'analisi delle viewpoint si è seguito il modello di Bass, nel quale si suddividono le viewpoint in tre classi: Module Viewpoint, Component and Connector Viewpoint e Allocation Viewpoint.
In particolare il team si è soffermato sulle prime due tipologie.

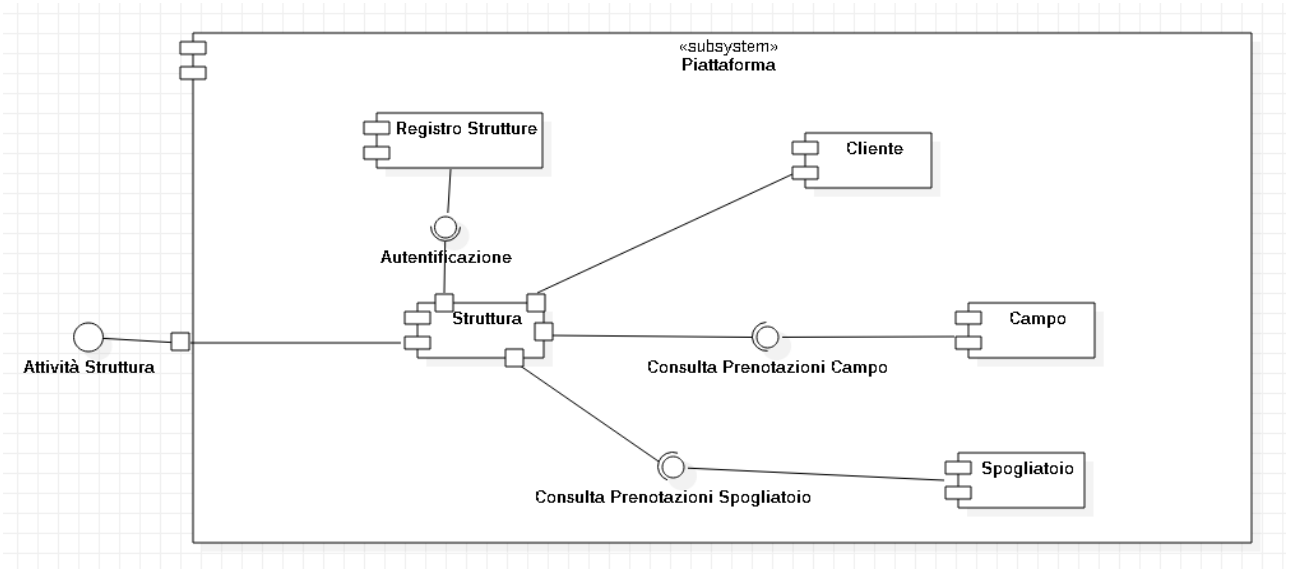
Il Module Viewpoint è stato utilizzato per rappresentare una vista statica dell'applicativo, il quale è stato diviso in livelli (Layered). In particolare sono stati individuati cinque diversi livelli, dove ogni livello può sfruttare i servizi offerti da quelli sottostanti. La piattaforma è pensata per poter essere utilizzata sia dagli utenti che vogliono effettuare le prenotazioni, sia per le strutture che vogliono invece accedere alla loro area riservata, di conseguenza i due attori faranno un utilizzo differente dell'applicativo, nonostante questo sia il medesimo per entrambi. Al quinto livello sono quindi presenti le due diverse tipologie di utilizzatori della piattaforma. A quello sottostante è presente la Piattaforma che rappresenta un'astrazione del nostro sistema.
Dal terzo livello in poi sono state specificate le principali classi che compongono il sistema, ponendo l'attenzione sulle loro interazioni.
Per quanto riguarda la classe struttura, questa sfrutta i servizi delle classi Campo e Spogliatoio, le quali a loro volta si fondano su quelli proposti dalla classe Prenotazione. La classe Cliente, invece, salta il secondo livello e sfrutta direttamente i servizi proposti dalla classe Prenotazione.
Per approfondire al meglio i servizi offerti dalle singole classi ci si rifà al Class diagram.



Gli sviluppatori hanno realizzato anche un Component and Connector Viewpoint, per poter avere una vista dinamica del sistema, individuando i componenti principali dell'applicativo e mostrandone le interazioni.
Si propongono due punti di vista differenti: quello dell'utente e quello della struttura iscritta alla piattaforma.
Per quanto riguarda la vista dell'utente, c'è una funzionalità di Ricerca Struttura che viene erogata dalla Piattaforma. Questa permette di individuare all'interno della piattaforma una struttura, presente tra le strutture del registro, che rispetti i parametri di prenotazione indicati dall'utente.
La vista della struttura, invece, permette di svolgere alcune attività, indicate con Attività Struttura, sempre erogate dalla Piattaforma. In particolare è necessario che prima si effettui un'autentificazione, per poter poi interagire in completa libertà con gli altri componenti del sistema stesso.

Si specifica che il connector che lega Struttura con Cliente vuole indicare la possibilità di una struttura di eseguire un ban nei confronti di un oggetto della classe Cliente.





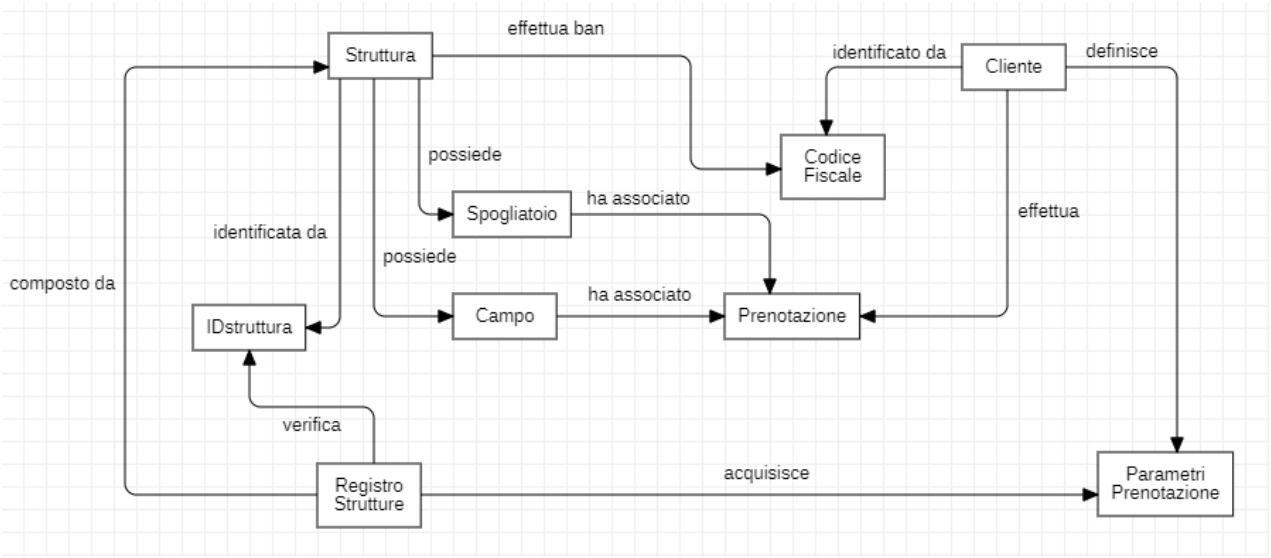
Il team ha inoltre seguito come pattern architetturale quello del Model View Controller, infatti ha suddiviso l'applicativo in due parti:

- View, costituita solamente dalle classi che implementano l'interfaccia grafica tramite Java Swing. Tale porzione non viene esaminata all'interno della documentazione, in quanto il team non le ha dedicato un progettazione strutturata come invece ha fatto per la parte di controller.
- Controller, porzione dell'applicativo funzionale che è stata progettata con attenzione e quindi anche riportata nella documentazione.

CAPITOLO 12 – Software Design

Per la realizzazione della piattaforma si utilizza il linguaggio di programmazione Java, essendo questo un linguaggio object-oriented, la squadra di lavoro ha deciso di definire la parte di design dell'applicativo attraverso l'object-oriented analysis and design methods (OOAD).

Precisando che non tutti gli oggetti qui riportati faranno parte delle classi effettive che compongono la piattaforma realizzata (ad esempio non esisterà una classe codice fiscale, questo sarà invece un attributo di un'altra classe), di seguito si riporta il modello UML definito individuando gli oggetti del sistema e le loro interazioni.



Questo diagramma non permette di definire in maniera chiara la dinamicità del sistema. Ad esempio, il diagramma non permette di specificare il procedimento temporale con cui avviene una prenotazione (settaggio parametri, seguito da controllo disponibilità ed infine conferma o respinta della prenotazione).

Per questo motivo gli sviluppatori rimandano al sequence diagram qualora si volesse avere una migliore visione del funzionamento del sistema.

Le entità identificate nella prima parte della fase di design vengono esplicitate seguendo lo standard del modello IEEE 1016 il quale suddivide ogni entità in una serie di attributi.

Identification	Struttura
Type	Classe
Purpose & Function	Permette di identificare univocamente le strutture sportive del sistema. Ha associato una lista di campi e una lista di spogliatoi, entrambe modificabili. Può specificare utenti indesiderati (ban). Può consultare le prenotazioni associate ai suoi campi e spogliatoi. Tiene conto del numero di prenotazioni fatte da ciascun cliente presso i suoi campi.
Dependencies	Usa le classi Cliente, Spogliatoio e Campo. È usata dalla classe Registro Strutture.
Interfaces	Utilizza i costruttori di Campo, Spogliatoio e Cliente qualora si aggiungano nuovi oggetti nelle liste di Struttura indicate in precedenza. Sfrutta i metodi di Spogliatoio e di Campo per poter stampare le prenotazioni.
Data	Gli attributi identificatovi della Struttura sono: Nome, Indirizzo, Città. Ha un attributo IDstruttura generato in automatico dal sistema e che funziona da password. Possiede le liste: ListaCampi, ListaSpogliatoi, ListaBan. Possiede una hashmap ConteggioPrenotazioni che associa ad ogni cliente il numero di prenotazioni fatte presso la struttura.

Identification	Campo
Type	Classe astratta
Purpose & Function	È una classe astratta perché generalizza le classi specifiche Campo Calcetto, Campo Basket e Campo Tennis. Riporta solo attributi e metodi comuni a tutte le tipologie di campi. Permette di aggiungere prenotazioni modificando la lista ListaPrenotazioneCampi associata. Permette di stampare le prenotazioni associate.
Dependencies	Usa la classe Prenotazione. È usata dalla classe Struttura.
Interfaces	Sfrutta il costruttore di prenotazione per poter aggiungere prenotazioni alla lista associata.
Data	Ha un attributo IDcampo univoco rispetto alla struttura di appartenenza. Ha un attributo Sport che specifica lo sport praticabile. Ha un attributo Prezzo che specifica il costo orario. Ha un attributo nroMaxPerosne che specifica il numero massimo di persone che possono giocare contemporaneamente. Possiede la lista ListaPrenotazioneCampi.

Identification	Spogliatoio
Type	Classe
Purpose & Function	Rappresenta gli spogliatoi di una struttura e prenotabili dagli utenti. Permette di aggiungere prenotazioni modificando la lista ListaPrenotazioneSpogliatoi associata. Permette di stampare le prenotazioni associate.
Dependencies	Usa la classe Prenotazione. È usata dalla classe Struttura.
Interfaces	Sfrutta il costruttore di prenotazione per poter aggiungere prenotazioni alla lista associata.
Data	Ha un attributo Numero univoco rispetto alla struttura di appartenenza. Ha un attributo Prezzo che specifica il costo orario. Possiede una lista ListaPrenotazioneSpogliatoi.

Identification	Registro Strutture
Type	Classe
Purpose & Function	Raccoglie tutte le strutture della piattaforma. Permette di effettuare l'autentificazione di una struttura.
Dependencies	Usa la classe Struttura. È usata dalla classe Cliente quando si effettuano prenotazioni.
Interfaces	Non essendoci la possibilità di aggiungere nuove strutture alla piattaforma non utilizza il costruttore della classe Struttura.
Data	Ha una lista di strutture chiamata ListaStrutture.

Identification	Cliente
Type	Classe
Purpose & Function	Identifica un utente della piattaforma. Permette di settare i parametri di prenotazione e quindi di effettuare nuove prenotazioni presso le strutture della piattaforma. Controlla se è stato bannato da una determinata struttura.
Dependencies	Usa la classe Prenotazione. È usata dalla classe Struttura.
Interfaces	Sfrutta il costruttore di prenotazione per poter creare nuove prenotazioni.

Data	Ha un attributo Nome e uno Cognome che identificano l'utente. Ha un attributo CodiceFiscale univoco.

Identification	Parametri Prenotazione
Type	Parametro
Purpose & Function	Permette al cliente di specificare le preferenze di prenotazione. Permette al Registro Strutture di controllare la disponibilità rispettando le scelte del cliente.
Dependencies	È un parametro di una funzione della classe Cliente (controlloDisponibilità). Usato dalla classe Cliente.

Identification	Codice Fiscale
Type	Attributo
Purpose & Function	Identifica univocamente il cliente. Serve alla classe Struttura per fare il conteggio delle prenotazioni ed eventualmente per effettuare il ban.
Dependencies	È usato dalla classe Struttura.

Identification	IDstruttura
Type	Attributo
Purpose & Function	Identifica univocamente una struttura. Serve alla classe Registro Strutture per fare l'autentificazione della struttura.
Dependencies	È usato dalla classe Registro Strutture.

Identification	Prenotazione
Type	Classe
Purpose & Function	Serve per poter creare una nuova prenotazione relativa ad un campo ed eventualmente anche ad uno spogliatoio. Permette di calcolare il costo della prenotazione basandosi sul prezzo orario del campo (e dello spogliatoio) e tenendo presente un eventuale sconto del cliente.
Dependencies	È usata dalle classi Cliente, Campo, Spogliatoio.
Data	Ha un attributo univoco CodicePrenotazione. Ha gli attributi NomeStruttura, Data, Orario, Sport, Spogliatoio che la caratterizzano e rappresentano le preferenze del cliente. Ha gli attributi NomeCampo, NomeSpogliatoio che vengono assegnati in base alla disponibilità dalla struttura specificata durante la fase di ricerca. Ha un attributo prezzo da calcolare con il metodo calcolaPrezzo. Ha un attributo CodiceFiscale dell'utente che ha effettuato la prenotazione.

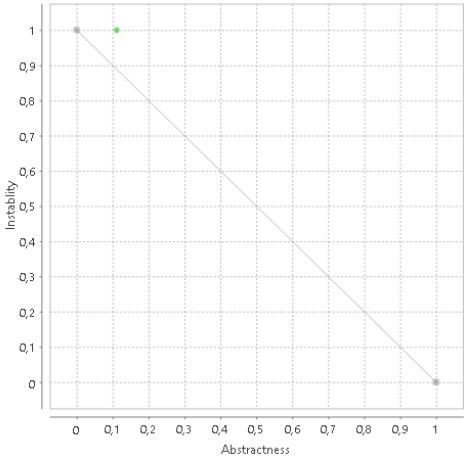
Design Pattern:

Si riporta di seguito una breve descrizione dei due design pattern utilizzati nello sviluppo dell'applicativo:

- Singleton: questo pattern è stato utilizzato per poter garantire che la classe RegistroStrutture potesse essere istanziata una sola volta, affinché l'applicativo risulti corretto infatti è fondamentale che il registro contenente le strutture iscritte alla piattaforma sia unico.
  A tale scopo, per poter creare un'istanza di questa classe, non è possibile richiamare il suo costruttore ma è presente il campo statico "registro" che rappresenta l'unica istanza possibile della classe RegistroStrutture.
  È consentito solamente richiamare in maniera statica il metodo getIstance() che, qualora "registro" sia null richiama il costruttore privato e ritorna la variabile "registro" istanziata, altrimenti ritorna il "registro" già istanziato in precedenza.

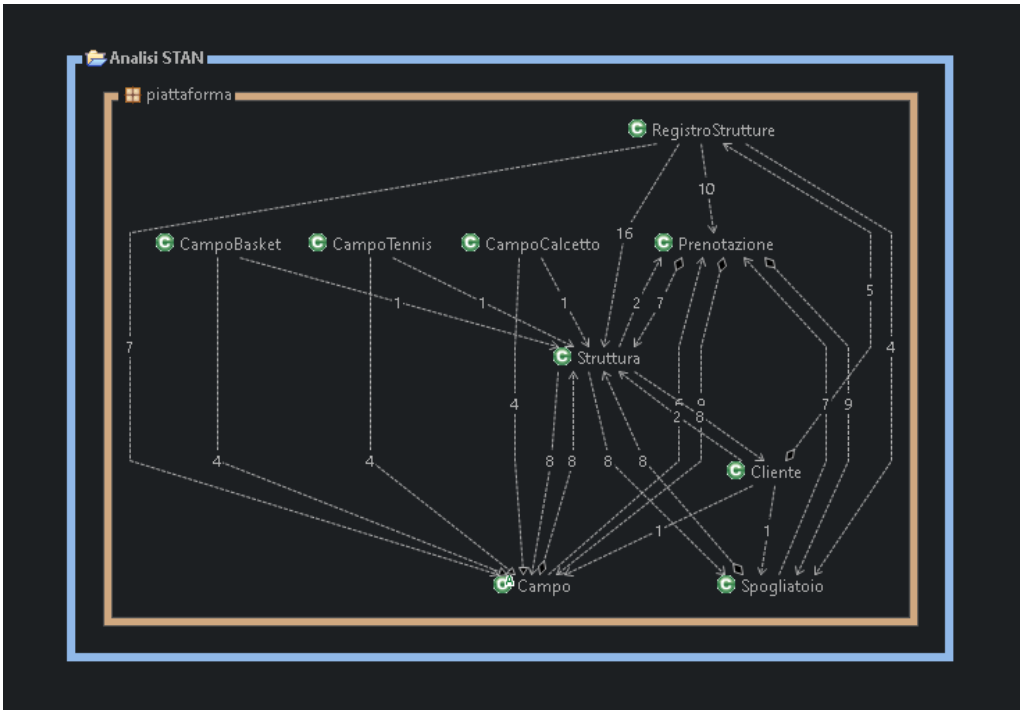
- Delegation: è stato utilizzato questo pattern per poter delegare delle attività richieste dalla classe Cliente (controlloDisponibilitàCampo() e controlloDisponibilitaSpogliatoio()) alla classe RegistroStrutture. Tale soluzione è stata impiegata per evitare che la classe Cliente, che rappresenta gli utenti dell'applicativo, potesse interagire direttamente con aspetti critici della piattaforma di prenotazioni. Attraverso la delegation è stato possibile interporre una classe gestita dal sistema che potesse mediare alle richieste dell'utilizzatore.
- Abstraction-Occurrence: per gestire la possibilità di una struttura di avere una lista di campi e una di spogliatoi a lei associati, il team ha deciso di seguire il pattern di abstraction-occurrence. Infatti campi e spogliatoi devono avere le stesse caratteristiche della struttura alla quale appartengono, ma devono anche essere identificabili univocamente. Per questo Campo e Spogliatoio sono rappresentati da due diverse classi che si legano a quella di Struttura con una relazione uno molti. Ogni spogliatoio e campo avrà quindi un riferimento alla struttura di appartenenza, e ogni struttura avrà invece una lista di campi e spogliatoi. Nel momento in cui un campo o uno spogliatoio viene creato, viene anche aggiunto alla lista corrispondente della struttura a cui appartiene.

Il team ha deciso inoltre di utilizzare il tool STAN4j per poter avere una misurazione della complessità del progetto.



Lo studio è stato effettuato solo per il package piattaforma contenuto nel folder "src", infatti non sono stati prese in considerazione le classi appartenenti ai folder "grafica" e "test". Il team infatti ha voluto rimanere conforme alla linea decisionale tenuta durante tutto lo sviluppo del progetto, ovvero quella di focalizzare l'analisi esclusivamente sulla parte dell'applicativo e non sulla parte grafica o dei test effettuati.

Dal grafico a sinistra si può notare che il package piattaforma ha un livello di astrazione pari a 0.1, mentre un livello di instabilità pari a 1. Dato che la linea ottimale per un progetto sarebbe quella di ottenere una somma delle due variabili uguale a 1, il team avrebbe potuto non utilizzare l'unica classe astratta, oppure aumentare il numero totale di classi presenti nel package stesso, per poter avvicinare ulteriormente la misurazione con il pallino in verde alla linea in diagonale (situazione ottimale).



Inoltre il team ha riportato una paronamica delle dipendenze delle varie classi presenti nel package generata dal tool. In particolare, andando a visualizzare le freccie tratteggiate con la finestra "Dependencies", il team ha potuto osservare più facilmente le interazioni tra le varie classi del package.

CAPITOLO 13 – Software Testing

Per poter testare l'applicativo manualmente, è possibile utilizzare l'interfaccia grafica sia come utente, sia come struttura iscritta, verificando così tutte le funzionalità implementate nella piattaforma.

Per poter avviare l'applicativo è necessario aprire con il tool Eclipse il file "piattaformaJava\grafica\piattaforma.Main.java" e selezionare la voce "Run As Java Application".

Al momento nell'interfaccia grafica, all'interno della classe "MainFrame", sono stati istanziati pochi oggetti, l'obiettivo infatti era quello di poter creare una simulazione del funzionamento del sistema.

Inoltre, non avendo un database di supporto, le modifiche fatte tramite l'interfaccia grafica non potranno essere rese permanenti, pertanto saranno perse ad ogni nuova esecuzione.

Di seguito viene fornito un elenco, suddiviso nelle due diverse modalità di utilizzo della piattaforma, contenente i passi definiti appositamente dagli sviluppatori al fine di condurre un test manuale completo. In particolare, a seguito di una breve spiegazione, vengono riportate in blu le specifiche istruzioni da svolgere.

Testing lato struttura

Per poter accedere all'area privata di una struttura è necessario cliccare il bottone denominato "STRUTTURA" che si trova in alto nella schermata dell'interfaccia grafica.

Successivamente è necessario inserire una delle 5 possibili password delle 5 diverse strutture iscritte temporaneamente al sistema:

- Password = "Struttura_1", accesso all'area privata della struttura "Campi Sportivi Stezzano"
- Password = "Struttura_2", accesso all'area privata della struttura "Centro Sportivo Levate"
- Password = "Struttura_3", accesso all'area privata della struttura "Campo CONI"
- Password = "Struttura_4", accesso all'area privata della struttura "Palestra Comunale Grassobbio"
- Password = "Struttura_5", accesso all'area privata della struttura "Oratorio Seriate"

Il sistema, in caso di password errata, mostrerà un pop up di errore.

Inserire una password errata. Successivamente inserirne una corretta a scelta.

1. Se inserita una password corretta, il sistema entra nell'area dedicata alla struttura da cui potrà scegliere diverse attività da svolgere quali:
    a) "Aggiungi", per l'aggiunta di un campo o di uno spogliatoio (punto 3).
    b) "Ban Utente", per inserire un ban ad un utente identificato dal suo codice fiscale (punto 4).
    c) "Controlla Prenotazioni", per consultare le prenotazioni relative ai suoi campi e spogliatoi (punto 5).
    d) "Menu Struttura", per tornare alla schermata iniziale dell'area privata della struttura (punto 6).

    In ogni momento è possibile tornare al menu principale, a quello dedicato dell'utente o alla pagina di accesso del punto 1, utilizzando il menu presente in alto. In tal caso il sistema mostra un pop up che chiede se si è intenzionati ad abbandonare la propria area privata e a confermare la propria scelta. Qualora si prema "Annulla" o il tasto di chiusura del pop up il sistema rimane all'interno dell'area privata, mentre se si dovesse premere "OK", il sistema esce dall'area privata e apre la nuova schermata a seconda del pulsante che si è premuto. Ovviamente qualora si decidesse di abbandonare la propria area privata, e si eseguisse nuovamente il punto 1, il sistema chiederà nuovamente la Password.

    Selezionare il pulsante "Aggiungi".
2. Questa area permette di poter aggiungere dei nuovi campi o spogliatoi alla struttura.
    In particolare il sistema propone una scelta esclusiva tra una delle tipologie di campi supportate dalla piattaforma (tennis, calcetto, basket) e lo spogliatoio. Per il campo da tennis o da calcetto inoltre sono previsti dei parametri aggiuntivi, mostrati nella stessa linea orizzontale, che devono essere necessariamente specificati in questa fase.
    Inoltre, indipendentemente dalla scelta, è necessario inserire un prezzo orario.
    Una volta terminato il settaggio è possibile premere il pulsante "Submit" per poter creare una nuova istanza. Qualora però il settaggio non sia corretto, il sistema mostra un pop di errore e l'aggiunta viene annullata.
    Provare a inserire un nuovo campo omettendo alcuni parametri.

Provare ad inserire un campo con tutte le specifiche richieste.
Ripetere le stesse operazioni anche per l'inserimento di un nuovo spogliatoio.
Verificare il risultato delle proprie operazioni nel recap riportato in Menu Struttura.
Selezionare il pulsante "Ban Utente".

3. Questa attività richiede di specificare il nome, cognome e codice fiscale dell'utente che si vorrebbe bannare. In particolare si ricorda che il ban viene fatto sul codice fiscale. Qualora i campi non vengano tutti settati il sistema mostra un pop up di errore. Il sistema non effettua nessun controllo sulla correttezza del codice fiscale rispetto al nome e cognome inseriti.
   Nel caso in cui si dovesse ripetere lo stesso codice fiscale di un utente già bannato, il sistema mostra un pop up di errore.
   Aggiungere un nuovo ban.
   Indicare nuovamente lo stesso codice fiscale e osservare la comparsa del pop up di errore.
   Selezionare il pulsante "Controlla prenotazioni".
4. La pagina permette di poter stampare le prenotazioni presenti per i campi e spogliatoi della struttura. In particolare prima di poter premere il pulsante "Submit" è necessario scegliere l'id del campo o spogliatoio di cui stampare le prenotazioni. Se non viene fatta tale scelta e si preme comunque il pulsante di "Submit" il sistema mostra un pop up di errore. Qualora invece la scelta sia corretta, il sistema stampa le prenotazioni associate. Inoltre si mostra come la lista di campi e spogliatoi associata alla struttura sia aggiornata rispetto alla modifica effettuata al punto 3.
   Provare a premere submit senza specificare il campo/spogliatoio.
   Scegliere un campo o uno spogliatoio tra quelli elencati.
5. *Premere in qualiasi momento il pulsante "Menu Struttura" per tornare alla schermata iniziale dell'area privata della struttura*.

Selezionare uno dei pulsanti in alto per poter testare la corretta uscita dall'area della struttura.

<u>Testing lato utente</u>
Per poter accedere all'area dedicata all'utente è necessario cliccare il bottone denominato "UTENTE" che si trova in alto nella schermata dell'interfaccia grafica.

1. In questa prima pagina è necessario settare tutti i parametri per poter effettuare una ricerca per una prenotazione presso una delle 5 strutture salvate nel sistema. In particolare qualora anche solo uno dei campi non venga settato correttamente, il sistema mostra un pop up di errore. Inoltre c'è un controllo più specifico legato alla data per verificare che venga scritta correttamente e che, accoppiata all'orario specificato, sia una data futura rispetto al momento in cui si sta effettuando la prenotazione. Gli orari sono fissi e non variano a seconda della struttura. Il settaggio dello spogliatoio è invece facoltativo.
   Se il campo richiesto è già occupato nell'orario mostrato, il sistema mostra un messaggio di errore.
   Premere "Submit" e osservare l'errore generato: "Campi anagrafici non settati correttamente".
   Settare solo i campi anagrafici avendo cura che il codice fiscale non ricada nel ban inserito al punto 3 del test della struttura. Premere quindi "Submit" per osservare l'errore: "Campi non settati correttamente".
   Con i campi anagrafici di prima inserire "Calcetto" in "Campi Sportivi Stezzano", scegliere una combinazione data+orario che sia già passata e osservare l'errore.
   Inserire "Calcetto" in "Palestra Comunale Grassobbio", scegliendo come data "12-05-2022" (o qualsiasi altra data futura rispetto al momento di utilizzo) e orario qualsiasi. Osservare l'errore: "Non esiste questa tipologia di campo nella struttura".
   Inserire dei valori dei campi anagrafici uguali al punto 3 del test della struttura, specificando gli altri campi a proprio piacimento, tranne per la struttura che deve essere uguale a quella con cui si è deciso di lavorare nel test Struttura. Osservare l'errore dovuto all'impossibilità di prenotare in tale struttura per la presenza di un ban.
   Inserire "Calcetto" in "Campi Sportivi Stezzano", scegliendo come data "10-12-2022" e orario"11:00". Osservare l'errore in quanto si sovrappone a un'altra prenotazione salvata dagli sviluppatori.

Inserire "Tennis" in "Palestra Comunale Grassobbio", scegliendo come data "12-05-2022", orario "20:00", con Spogliatoio = NO. (non ci saranno errori).

2. Questa pagina viene mostrata solo se la ricerca del campo effettuata al punto 1 ha esito positivo. Ci sono tre possibili esiti a seconda delle preferenze indicate dall'utente:
    - Se non ha specificato uno spogliatoio, allora il sistema mostra che la ricerca ha avuto esito positivo e permette di osservare un recap della prenotazione stessa.
    - Se ha specificato uno spogliatoio ed è disponibile nell'orario richiesto, allora anche la ricerca ha avuto risultato positivo e si mostra anche in questo caso un recap.
    - Se ha specificato uno spogliatoio, ma questo non è disponibile all'orario del campo, il sistema propone all'utente di prenotare comunque il campo presso la struttura ma senza lo spogliatoio

    Qualora l'utente non sia soddisfatto può annullare la propria prenotazione cliccando uno dei bottoni in alto del menu. Se invece è intenzionato a confermare la prenotazione dovrà premere "Conferma". Una volta confermata la prenotazione viene mostrato un pop up con il costo totale della prenotazione.
    Confermare la prenotazione che è passata senza errori. Si torna in automatico alla condizione mostrata al punto 1.
    Inserire "Tennis" in "Palestra Comunale Grassobbio", scegliendo come data "12-05-2022", orario "09:00", con Spogliatoio = SI. Arrivati alla schermata 2 premere Conferma.
    Inserire "Basket" in "Palestra Comunale Grassobbio", scegliendo come data "12-05-2022", orario "09:00", con Spogliatoio = SI. Non ci saranno errori ma lo spogliatoio non sarà disponibile e si passerà alla schermata del punto 2. Confermare quindi la prenotazione anche senza spogliatoio.

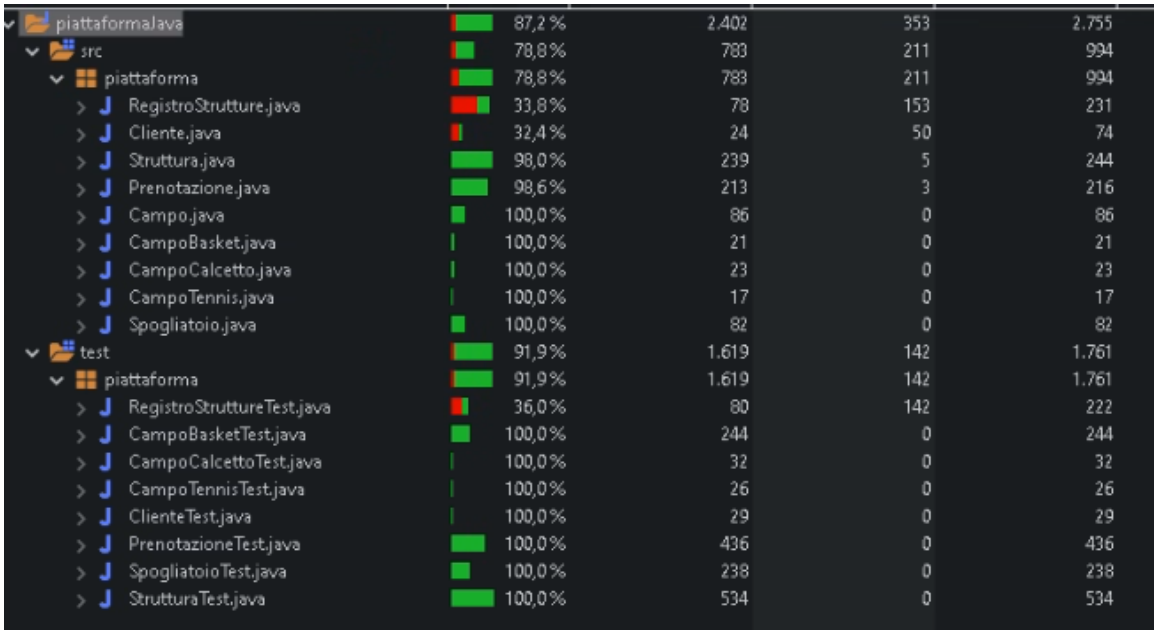
Provare a inserire cinque prenotazioni diverse presso la stessa struttura e con lo stesso utente. Notare che a partire dalla sesta prenotazione verrà applicato in automatico uno sconto al costo totale.

A questo punto
Per poter verificare il corretto funzionamento della gestione delle prenotazioni è consigliato infine di accedere all'area privata della struttura "Palestra Comunale Grassobbio" (con password: "Struttura_4") per verificare se è presente, seguendo il punto 4, le prenotazione appena creata nel punto 1 e 2 del test Utente.

Oltre al testing manuale, spiegato precedentemente, il team ha implementato una serie di test automatici in Eclipse sfruttando il tool Junit. In particolare si riportano le seguenti considerazioni:

- La classe Campo è astratta, dunque i suoi metodi sono stati testati nelle sue sottoclassi (in particolare il test è stato fatto nella classe CampoBasket).
- La classe Cliente nei metodi controlloDisponibilitàCampo() e controlloDisponibilitaSpogliatoio() si limita a richiamare i metodi omonimi della classe RegistroStrutture, pertanto non sono stati implementati i test per questi metodi nella classe Cliente, ma è stato fatto nella classe RegistroStrutture.
- Per poter verificare che i metodi catturassero correttamente le eccezioni (qualora ci siano) è stata utilizzata l'annotazione @Test (expected=Throwable.class).
- Essendo la classe RegistroStrutture realizzata attraverso il design pattern Singleton, il testing è stato fatto in un unico metodo.

Successivamente è stata analizzata la copertura del codice tramite il tool Eclemma; in particolare si nota che il progetto Java "piattaformaJava" ha una copertura pari all'87,2%, valore ritenuto più che soddisfacente.
Si specifica che una volta inserita la componente di interfaccia grafica tale percentuale diminuirà poiché i metodi dell'interfaccia non sono stati testati in quanto non ritenuto necessario da parte degli sviluppatori.
Infine si vuole precisare che la copertura misurata per la classe RegistroStruttura è piuttosto bassa, tuttavia il team si aspettava un risultato del genere poiché tale classe utilizza metodi testati in altre classi (ma non in RegistroStrutture per evitare un'inutile duplicazione di codice).
Considerazioni analoghe sono valide anche per la copertura della classe Cliente.



piattaformaJava	87,2 %	2.402	353	2.755
src	78,8 %	783	211	994
piattaforma	78,8 %	783	211	994
RegistroStrutture.java	33,8 %	78	153	231
Cliente.java	32,4 %	24	50	74
Struttura.java	98,0 %	239	5	244
Prenotazione.java	98,6 %	213	3	216
Campo.java	100,0 %	86	0	86
CampoBasket.java	100,0 %	21	0	21
CampoCalcetto.java	100,0 %	23	0	23
CampoTennis.java	100,0 %	17	0	17
Spogliatoio.java	100,0 %	82	0	82
test	91,9 %	1.619	142	1.761
piattaforma	91,9 %	1.619	142	1.761
RegistroStruttureTest.java	36,0 %	80	142	222
CampoBasketTest.java	100,0 %	244	0	244
CampoCalcettoTest.java	100,0 %	32	0	32
CampoTennisTest.java	100,0 %	26	0	26
ClienteTest.java	100,0 %	29	0	29
PrenotazioneTest.java	100,0 %	436	0	436
SpogliatoioTest.java	100,0 %	238	0	238
StrutturaTest.java	100,0 %	534	0	534

CAPITOLO 14 - Software Maintenance

Il codice iniziale dell'applicativo è stato generato in automatico a partire dal Class Diagram tramite il plug in Rebel.

Nel corso della fase di programmazione gli sviluppatori hanno avuto la necessità di fare alcune attività di reverse path engineering; infatti alcune porzioni del codice generato in automatico sono state modificate per renderle il più possibili conformi alle caratteristiche della piattaforma pensata.

Durante lo sviluppo inoltre sono state ritenute necessarie anche alcune modifiche ai diagrammi UML precedentemente definiti giungendo a soluzioni lievemente differenti rispetto a quelle pensate inizialmente, ma ritenute più adatte.

In particolare il Class Diagram è stato oggetto della maggior parte delle revisioni introdotte. Ad alcune classi infatti sono stati aggiunti dei metodi e degli attributi, mentre ad altre sono stati tolti. Ad esempio per poter integrare nel progetto i vari design pattern, i due sviluppatori hanno riconsiderato l'organizzazione dell'interazione tra le varie classi andando a spostare tutti i metodi della classe Cliente in quella di RegistroStrutture, per poter sviluppare il design pattern di Delegation. Questa scelta è giustificata dalla volontà di impedire al cliente di modificare direttamente dati sensibili riguardanti le strutture della piattaforma, si è quindi preferito utilizzare la classe Registro Strutture come "mediatore" per tale scopo.

In ogni caso, in seguito alle modifiche, non è stato generato nessun nuovo codice automatico con Rebel, il team ha infatti preferito mantenere quello originale e modificarlo di conseguenza in modo da evitare di perdere i progressi raggiunti fino a quel momento.

Anche l'Activity Diagram è stato rivisto in alcune sue parti, in particolare sono stati aggiunti alcuni exception handler che potessero gestire casi di errore che inizialmente non erano stati individuati dagli sviluppatori.

Nella fase finale che consiste nello sviluppo del codice dell'interfaccia grafica, è stato necessario perseguire delle attività di refactoring su alcuni metodi del codice applicativo con lo scopo di renderli più facilmente integrabili con l'interfaccia grafica realizzata.

Ad esempio i metodi ConfermaPrenotazioneNOSpogliatoio() e ConfermaPrenotazioneCONSpogliatoio() erano inizialmente pensati come di tipo void; successivamente il team ha ritenuto necessario che ritornassero il prezzo calcolato dall'oggetto Prenotazione che istanziavano nel loro corpo. In questo modo si riesce a mostrare all'utente, una volta confermata la sua prenotazione, anche il costo finale di quest'ultima.