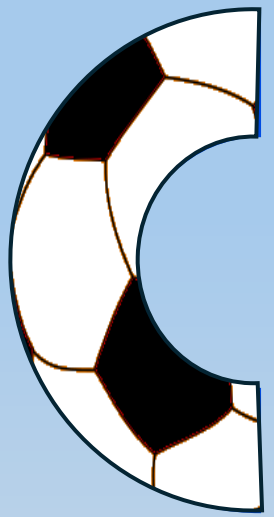
Progetto di Ingegneria del Software

Realizzato da:

- Biscaro Alessandro [1087892]
- Fabbris Thomas [1086063]
- Gambirasio Lorenzo Umberto [1087441]

Obiettivo del progetto

- SportMate ha l'obiettivo di rendere ancora più semplice e veloce la prenotazione di campi per organizzare partite di calcio a 5, calcio a 7 e basket 3v3, eliminando la necessità di lunghe telefonate e offrendo un'esperienza più interattiva e autonoma per gli utenti.
- Inoltre, offre ai gestori dei centri sportivi la possibilità di registrare il proprio polo, rendendolo disponibile agli utenti che utilizzano l’applicazione.



Difficoltà riscontrate

- Durante lo sviluppo del software, le maggiori difficoltà riscontrate hanno riguardato l'utilizzo di Jooq, in particolare durante la generazione automatica delle classi partendo dalle tabelle del database realizzato con SQLite.

- Un'altra difficoltà è stata riscontrata durante l'utilizzo del tool di modellazione Papyrus per la realizzazione dei diagrammi UML.

- Durante la stesura del codice, la maggiore difficoltà è stata far comunicare i tre livelli della nostra architettura.

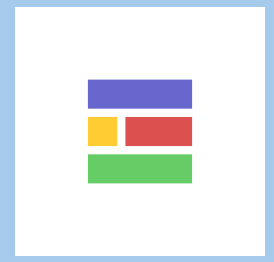
Paradigmi di programmazione e modellazione

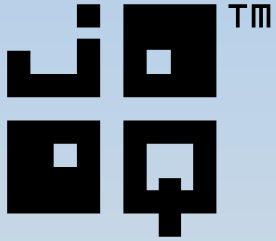
- Per lo sviluppo del progetto è stato adottato un paradigma orientato agli oggetti (OOP), utile per applicazioni con interfaccia utente, che si basa sull'utilizzo di oggetti, che rappresentano entità del mondo reale con attributi e metodi, per aumentare incapsulamento e polimorfismo.
- Il paradigma di modellazione utilizzato è UML, utile per descrivere e specificare in diagrammi i vari aspetti dell'applicazione, focalizzandosi su diversi punti di vista.
- I linguaggi di programmazione utilizzati sono stati: Java, per lo sviluppo del backend del progetto; HTML e CSS, per lo sviluppo dell'interfaccia web e dell'interfaccia utente (frontend del progetto); D, per i diagrammi UML.

UNIFIED MODELING LANGUAGE™

Tool utilizzati

- Per lo sviluppo dell'applicazione sono stati usati diversi tool:
  - Eclipse IDE: utile per lo sviluppo in Java
  - Maven: utile per gestire e automatizzare il progetto
  - Log4j: utile per la gestione dei log
  - Papyrus: utile per realizzare i diagrammi UML
  - SQLite: utile per creare il database per l'applicazione
  - Jooq: utile per rappresentare in classi Java le tabelle del database
  - HackMD: utile per la stesura dei documenti in file markdown
  - Stan4j: utile per analizzare la struttura del codice Java
  - JUnit: utile per realizzare test unitari o di integrazione
  - Vaadin: utile per lo sviluppo dell'interfaccia utente

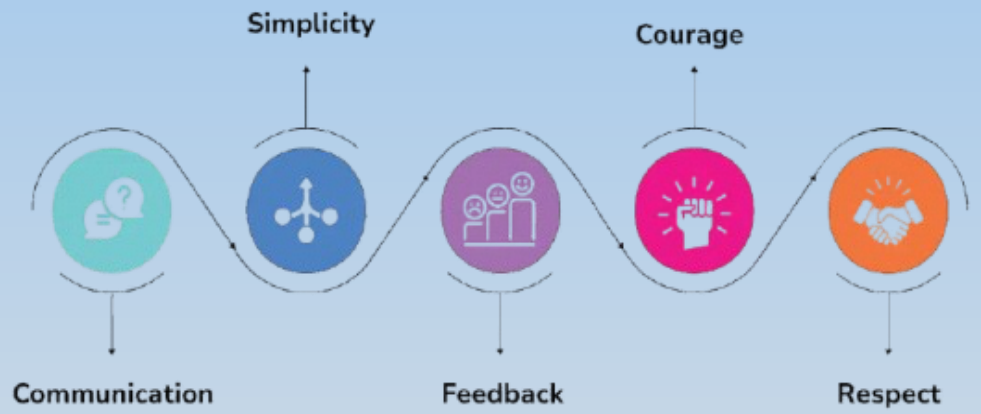




Software Configuration Management

- Durante la realizzazione dell'applicazione è stato utlizzato Git come sistema di gestione della configurazione, insieme a GitHub Desktop.
- Git ci ha permesso di tenere tracciati i progressi attraverso una Kanban Board, costantemente aggiornata.
- Sono stati utilizzati alcuni branch per sviluppare nuove funzionalità in modo isolato, evitando di compromettere la stabilità del branch principale.
- Sono stati usati diversi issue per monitorare e gestire eventuali errori, garantendo una tracciabilità chiara dei problemi e delle soluzioni

Software Life Cycle



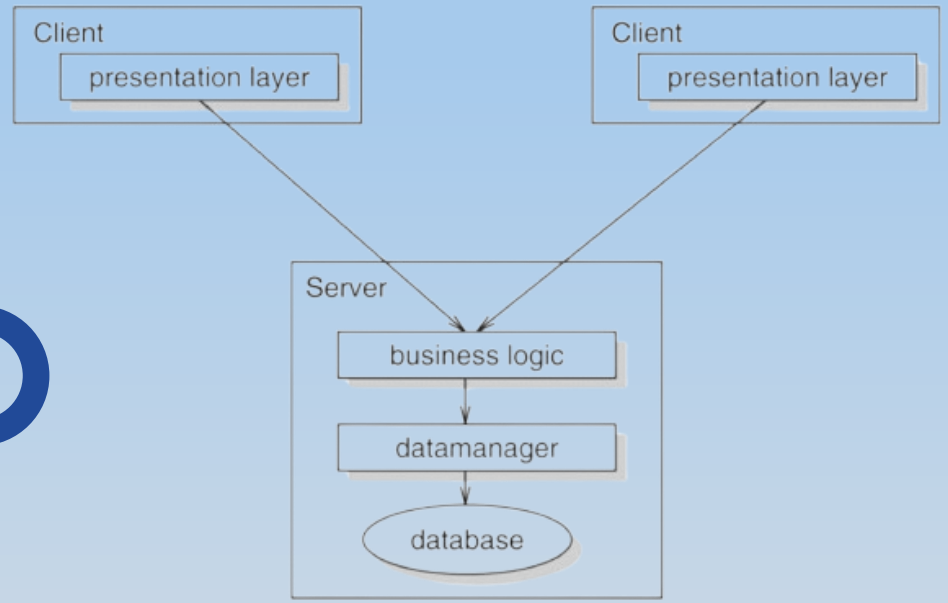
- Il progetto si è basato sul ciclo di vita del software *Extreme Programming* (XP), un metodo agile puro basato su una serie di *best practices* consolidate nel tempo, che mirano a ottenere un sistema sempre in esecuzione attraverso un approccio incrementale e un processo di integrazione continua del codice.
- Durante lo sviluppo del sistema software, sono state seguite tutte le linee guida di XP, in particolare la proprietà collettiva del codice, l'adozione di un ritmo sostenibile e la programmazione in coppia.

Requisiti e tecniche di elicitazione

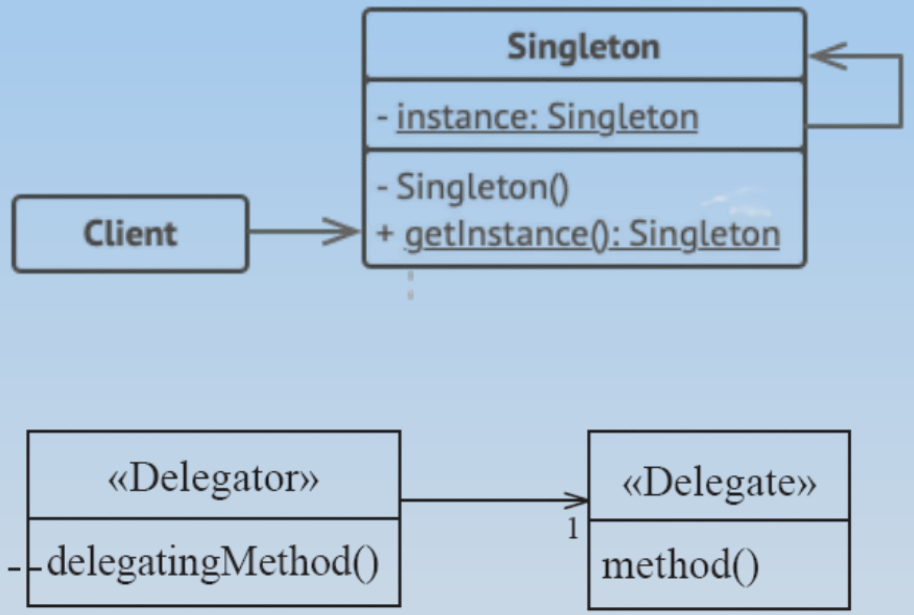
- La principale fonte di informazione per il processo di elicitazione dei requisiti sono gli utenti dell'applicazione e i gestori dei centri sportivi.
- La principale tecnica di elicitazione dei requisiti utilizzata con gli utenti è stata l'*intervista aperta*; al contrario, abbiamo preferito un'analisi basata su scenari creati artificialmente e discussi con i gestori.
- I requisiti sono stati documentati all'interno della Specifica dei Requisiti e classificati in base alla loro priorità utilizzando la tecnica di prioritizzazione MoSCoW.

Stile architetturale

- Lo stile architettonico utilizzato durante l'implementazione del progetto software è stato uno stile architettonico layered (a strati), articolato in 3 livelli:
  - SportMateDBLayer: corrisponde al livello dati in cui vengono archiviate e gestite su un database le informazioni elaborate dall'applicazione
  - SportMateBusinessLayer: corrispondente al *layer* di business e persistenza, si occupa dell'elaborazione delle informazioni raccolte nel livello di presentazione attraverso la logica di business; inoltre, può anche manipolare i dati presenti nel *layer* dati.
  - SportMatePresentationLayer: corrispondente al layer di presentazione, rappresenta l'interfaccia utente ed il livello di comunicazione dell'applicazione con cui interagisce direttamente l'utente finale.



Design Pattern



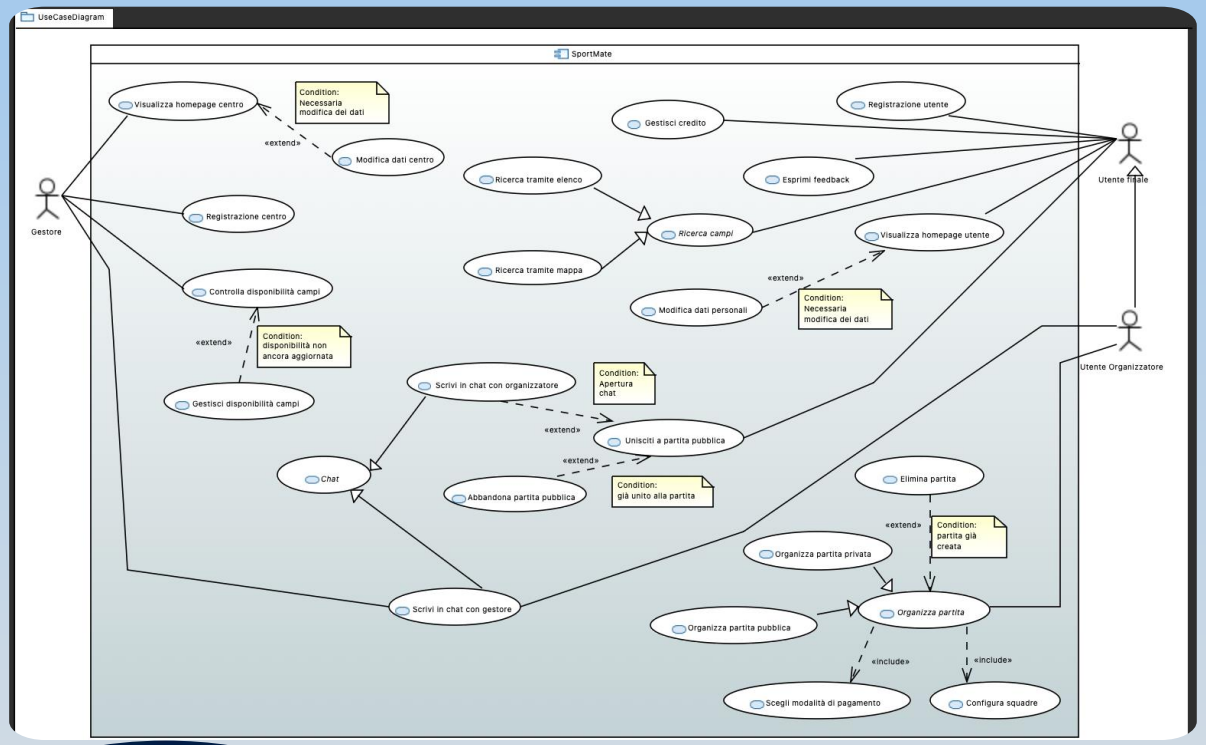
- I design pattern utilizzati durante lo sviluppo del codice del progetto software sono i seguenti:
  - **Singleton Pattern**: impiegato per fornire un'utile astrazione per gestire la connessione al database ***embedded*** usato in SportMate, denominato ***SportMateDB***.
  - **Delegation Pattern**: quando un utente interagisce con un bottone, l'oggetto ***Button*** di Vaadin delega la gestione dell'evento associato ad un ***listener***, implementando in questo modo il ***Delegation Pattern.*** Tale pattern è stato anche applicato nelle classi **HomePageView** e **FeedbackView** in **SportMatePresentationLayer**, per delegare la visualizzazione della lista dei feedback alla classe **MassageListDelegator**.

Metriche di complessità

- Per valutare la qualità del sistema software, sono state calcolate le seguenti metriche di complessità con l'ausilio del tool **Stan4j**, suddivise per pacchetto. Per semplicità di lettura, il suffisso *sportmateinc*, comune a tutti i pacchetti del progetto non è stato riportato.

Pacchetto	WMC	RFC	ACC	Coupling afferente	Coupling efferente	Astrattezza	Instabilità
sportmatedblayer	19	12	1.58	13	0	0	0
sportmatebusinesslayer.entities	10.07	9.64	1.06	25	0	0.07	0
sportmatebusinesslayer.services	6	15	1.33	11	26	0	0.70
sportmatebusinesslayergenerated	3	3	1	15	15	0	0.50
sportmatebusinesslayergenerated.tables	19.07	18.07	1.06	28	16	0	0.36
sportmatebusinesslayergenerated.records	36.60	36.60	1	15	15	0	0.5
sportmatepresentationlayer.application	2	2	1	0	0	0	1
sportmatepresentationlayer.application.security	3.25	3.75	1.3	1	2	0	0.67
sportmatepresentationlayer.application.services	3	5	1.2	7	4	0	0.36
sportmatepresentationlayer.application.views	8	9	1.14	0	2	0	1
sportmatepresentationlayer.application.views.comuni	5	5.8	1	0	1	0	1
sportmatepresentationlayer.application.views.gestore	20.33	28	1.33	0	12	0	1
sportmatepresentationlayer.application.views.utente	13.12	15.62	1.33	0	16	0	1

Modellazione



- Utilizzando il tool Papyrus, abbiamo realizzato i seguenti diagrammi UML:
  - Diagramma dei casi d'uso
  - Diagramma delle classi
  - Diagramma della macchina a stati
  - Diagramma di sequenza
  - Diagramma di comunicazione
  - Diagramma delle attività
  - Diagramma delle componenti
  - Diagramma dei pacchetti

Implementazione

- I requisiti non implementati durante la stesura del codice sono stati:
  - Possibilità di unirsi ad una partita pubblica organizzata da altri utenti
  - Possibilità di utenti finali e gestori di comunicare attraverso una chat
  - Possibilità di generare le squadre in maniera casuale, indicando il colore delle divise
  - Possibilità di abbandonare la partita pubblica prenotata, entro 6 ore dall'inizio della partita
  - Possibilità di cancellare la prenotazione del campo

Testing

- Attraverso diversi test in JUnit abbiamo potuto individuare alcune criticità nel nostro codice e di conseguenza migliorarlo, in modo da renderlo più sicuro.

JUnit 4

Copertura per DBLayer

SportMateDBLayer (30 gen 2025 14:44:32)

Element	Coverage	ed Instructions	ed Instructions	tal Instructions
SportMateDBLayer	81,5 %	238	54	292
src/main/java	70,7 %	111	46	157
sportmateinc.sportmatedblayer	70,7 %	111	46	157
SportMateDB.java	70,7 %	111	46	157
src/test/java	94,1 %	127	8	135
sportmateinc.sportmatedblayer	94,1 %	127	8	135
SportMateDBTest.java	94,1 %	127	8	135

Copertura per BusinessLayer

Problems Javadoc Declaration Console SonarQube On-The-Fly Coverage

SportMateBusinessLayer (30 gen 2025 14:40:18)

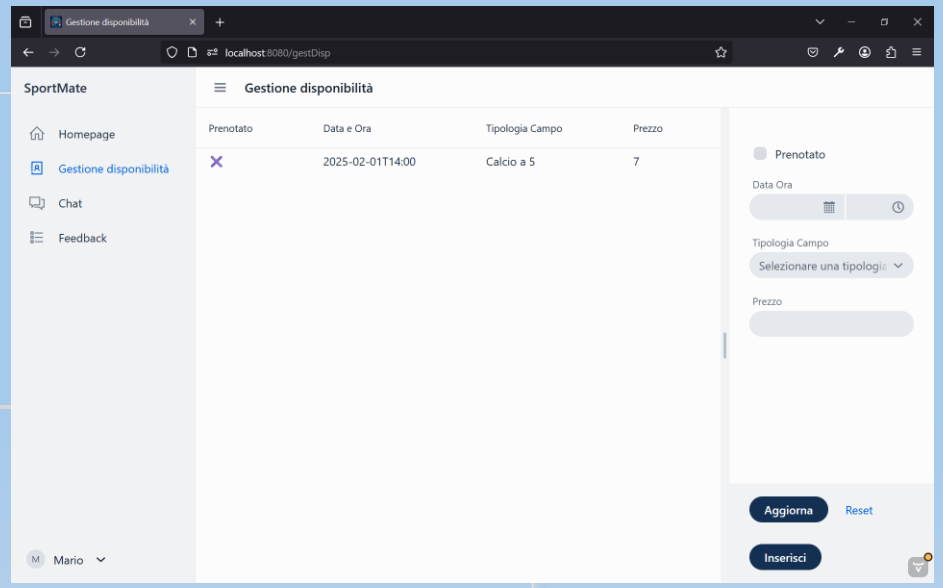
Element	Coverage	ed Instructions	ed Instructions	tal Instructions
src-generated	23,2 %	1.411	4.664	6.075
sportmateinc.sportmatebusinesslayergenerated.tables.records	1,6 %	44	2.705	2.749
sportmateinc.sportmatebusinesslayergenerated.tables	41,2 %	1.312	1.873	3.185
sportmateinc.sportmatebusinesslayergenerated	39,0 %	55	86	141
src/main/java	97,3 %	3.151	89	3.240
sportmateinc.sportmatebusinesslayer	0,0 %	0	49	49
sportmateinc.sportmatebusinesslayer.entities	95,3 %	704	35	739
AuthenticatedProfile.java	64,3 %	27	15	42
Feedback.java	78,6 %	55	15	70
Disponibilita.java	94,0 %	79	5	84
CentroSportivo.java	100,0 %	93	0	93
Gestore.java	100,0 %	10	0	10
InfoDisponibilita.java	100,0 %	53	0	53
InfoPartita.java	100,0 %	63	0	63
Livello.java	100,0 %	15	0	15
Location.java	100,0 %	53	0	53
Partita.java	100,0 %	103	0	103
Persona.java	100,0 %	73	0	73
ServiziAggiuntivi.java	100,0 %	21	0	21
TipoCampo.java	100,0 %	15	0	15
Utente.java	100,0 %	44	0	44
sportmateinc.sportmatebusinesslayer.services	99,8 %	2.447	5	2.452
LocationService.java	93,8 %	75	5	80
AuthenticatedProfileService.java	100,0 %	159	0	159
CentriSportiviService.java	100,0 %	275	0	275
DisponibilitaService.java	100,0 %	453	0	453
FeedbackService.java	100,0 %	71	0	71
GestoriService.java	100,0 %	220	0	220
LivelliService.java	100,0 %	90	0	90

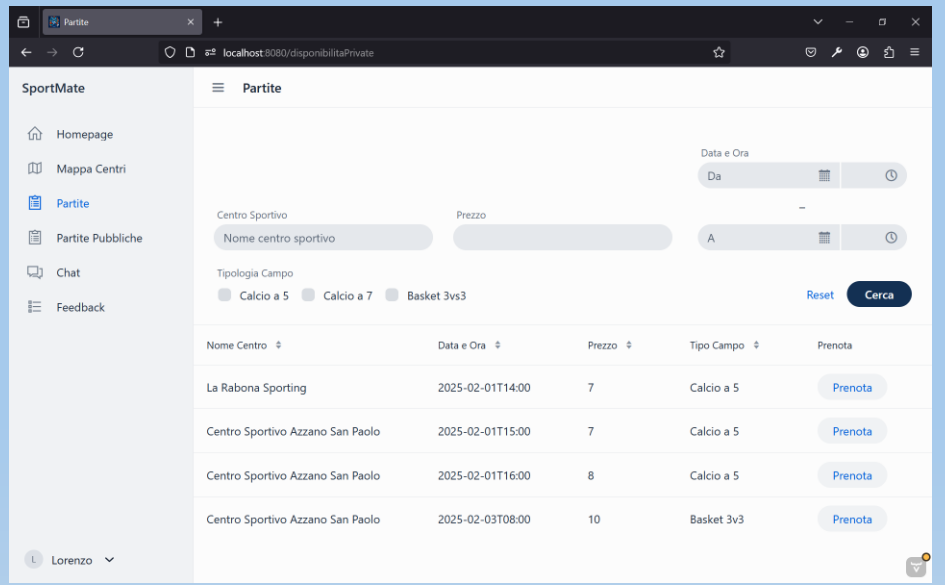
Problems Javadoc Declaration Console SonarQube On-The-Fly Coverage

SportMateBusinessLayer (30 gen 2025 14:40:18)

Element	Coverage	ed Instructions	ed Instructions	tal Instructions
UtentiService.java	100,0 %	349	0	349
src/test/java	98,6 %	4.313	61	4.374
SportMateInc.SportMateBusinessLayer.services	97,9 %	2.815	61	2.876
ServiziAggiuntiviServiceTest.java	93,1 %	242	18	260
AuthenticatedProfileServiceTest.java	93,7 %	192	13	205
TipoCampoServiceTest.java	95,1 %	174	9	183
LivelliServiceTest.java	92,0 %	81	7	88
GestoriServiceTest.java	97,6 %	247	6	253
UtentiServiceTest.java	97,4 %	227	6	233
DisponibilitaServiceTest.java	99,4 %	318	2	320
CentriSportiviServiceTest.java	100,0 %	268	0	268
FeedbackServiceTest.java	100,0 %	223	0	223
LocationServiceTest.java	100,0 %	171	0	171
PartitaServiceTest.java	100,0 %	427	0	427
PrenotazioneServiceTest.java	100,0 %	245	0	245
SportMateInc.SportMateBusinessLayer	100,0 %	137	0	137
GenerateJooqTest.java	100,0 %	137	0	137
SportMateInc.SportMateBusinessLayer.entities	100,0 %	1.361	0	1.361
AuthenticatedProfileTest.java	100,0 %	30	0	30
CentroSportivoTest.java	100,0 %	191	0	191
DisponibilitaTest.java	100,0 %	236	0	236
FeedbackTest.java	100,0 %	88	0	88
InfoDisponibilitaTest.java	100,0 %	126	0	126
InfoPartitaTest.java	100,0 %	145	0	145
LivelloTest.java	100,0 %	26	0	26
LocationTest.java	100,0 %	62	0	62
PartitaTest.java	100,0 %	172	0	172
PersonaTest.java	100,0 %	153	0	153
ServiziAggiuntiviTest.java	100,0 %	31	0	31

DEMO





- Di seguito sono riportate alcune schermate del nostro sito, con le funzionalità di prenotazione delle partite (utente) e gestione delle disponibilità (gestore).