UNIVERSITÀ DEGLI STUDI DI SALERNO

Ingegneria del Software

SYSTEM DESIGN DOCUMENT

ANNO ACCADEMICO 2017/2018

Versione1.4

Partecipanti:

NOME	MATRICOLA
Vecchione Angela	**0512102274**
Albano Eugenio	**0512102480**

Revision History:

DATA	VERSIONE	DESCRIZIONE	AUTORE
18/12/17	1.0	Stesura del System Design Document	Membri del team.
23/12/17	1.1	Stesura obiettivi di design e decomposizioni in sottosistemi.	Membri del team.
27/12/17	1.2	Stesura Mapping H/S e Gestione dati persistenti	Membri del team.
28/13/17	1.3	Stesura Boundary Conditions	Membri del team.
2/2/18	1.4	Completamento documento	Membri del team.

Indice

1. INTRODUZIONE

1.1 Scopo del sistema

Oggi giorno lo sport occupa un posto sempre più importante nella vita quotidiana. È molto comune trovare gruppi di persone alla ricerca di un campo sportivo libero per poter praticare il proprio sport preferito, da questa impellente necessità nasce la nostra web application.
Il nostro sito permetterà agli utenti di cercare i campi sportivi liberi della propria zona, facilitando così le operazioni di prenotazione e pagamento. Inoltre pagando online l'utente potrà usufruire di uno sconto.
Per i Partner sportivi è importante registrarsi al sito inserendo le proprie strutture sportive e le fasce orarie disponibili per le prenotazioni, in modo da pubblicizzare il suo campo ed avere maggiore affluenza.

1.2 Obiettivi di design

Di seguito sono elencati, in base ai criteri a cui si riferiscono, i diversi obiettivi di design del sistema Found It!.

1.2.1 Criteri di Performance

Tempo di risposta
Il Sistema deve assicurare una risposta rapida alle varie richieste degli utenti, con tempi di risposta non superiori a 2 secondi se si ha a disposizione una connessione

Internet stabile.

Throughput
Il Sistema sarà in grado di gestire più utenti in modo concorrente. Inoltre verrà progettato per poter resistere a richieste maggiori da parte degli utenti, utilizzando strutture scalabili per i server.

Memoria
Il sistema richiede lo spazio necessario per supportare il web server e lo spazio alla memorizzazione e all'archiviazione dei dati nell'unico database presente.
E' difficile prevedere una media dei campi sportivi inseriti da ogni Società Sportiva, ma possiamo prevedere che ogni Partner Sportivo inserisce in media 5 Campi Sportivi.

1.2.2 Criteri di Affidabilità

Robustezza
Found It! deve gestire eventuali input errati da parte dell'utente, attivando politiche di controllo lato Client (Javascript).
Siccome questi controlli possono essere evitati disabilitando alcune funzionalità, i controlli più importanti verranno effettuati lato Server (Servlet e JSP) garantendo così anche un secondo livello di robustezza. Nonostante questo comporti un overhead e appesantisca leggermente il carico del sistema, ciò permette all'utente di accorgersi subito di eventuali errori commessi.

Disponibilità
Il sistema deve essere disponibile ovunque, in qualsiasi momento e accessibile da qualsiasi dispositivo. Quindi, il sistema sarà disponibile 24 ore su 24 e 7 giorni su 7, a meno di interruzioni del servizio dovuti alla manutenzione ordinaria del sistema.

Sicurezza
Il sistema disporrà di una form di login che negherà l'accesso agli utenti non autorizzati, esclusi i visitatori che non intendono effettuare prenotazioni per cui si necessita un riconoscimento ed una registrazione.
La digitazione della password verrà nascosta attraverso caratteri speciali.
Il sistema si proteggerà da attacchi "brute force" vincolando l'utente a verificare un captcha ogni volta che tenterà di recuperare la password.
Il sistema garantirà la buona riuscita dei pagamenti poichè effettuati attraverso pagamento con carta prepagata.

1.2.3 Criteri di Manutenzione

Estendibilità
Found It! deve essere progettato per accogliere al meglio nuove funzionalità, integrando al meglio nuove funzionalità, integrandole al meglio con i moduli già presenti. Quindi è necessario che il codice sia ben strutturato in moduli, con il giusto trade-off tra coesione ed accoppiamento.

Modificabilità
Deve essere possibile modificare il codice in qualsiasi momento, per apportare modifiche o aggiustamenti.

1.2.4 Criteri per l'Utente Finale

Usabilità
Il Sistema deve avere un'interfaccia semplice ed intuitiva, garantendo una piacevole interazione tra l'utente e il sistema.

Il layout di Found It! si adatterà dinamicamente ai diversi dispositivi e alle diverse dimensioni dei display che si interfacciano con il sistema.

1.3 Definizioni, acronimi e abbreviazioni

- Found It!: Nome del sistema che verrà sviluppato.
- Utente Semplice: utente del sistema che effettua ricerche e prenotazioni.
- Partner Sportivo: utente del sistema che registra una Società Sportiva ed inserisce Campi Sportivi.
- Moderatore: utente del sistema che visualizza gli utenti iscritti e può bannarli.
- Utente: generalizzazione che racchiude Utenti Semplici, Partner Sportivi e Moderatori.
- Amministratore: gestisce il database, crea i moderatori ed effettua i backup.
- RAD: Requirement Analysis Document
- DBMS: Database Management System
- JSP: Java Server Page
- MVC: Model View Controller
- SDD: System Design Document
- DM: Data Model

1.4 Riferimenti

- Documento Requirement Analysis Document - Found It!
- B.Bruegge, A.H. Dutoit, Object Oriented So□ware Engineering - Using UML,Patterns and Java, Prentice Hall, 3rd edition, 2010
- Guida Html, Css, JavaScript, Xml: https://www.w3schools.com/

2. ARCHITETTURA SOFTWARE CORRENTE

Il sistema software ideato per Found It! è alla sua prima realizzazione e non sostituisce alcun sistema preesistente.

3. ARCHITETTURA SOFTWARE PROPOSTA

3.1 Panoramica

L'approccio utilizzato è di tipo client/server. Il server riceve le richieste da parte del client e risponde in tempo utile.
Ogni progetto software viene sviluppato seguendo una particolare architettura. Ciò favorisce lo sviluppo e le modifiche successive del sistema, inoltre facilita l'individuazione di problemi e la risoluzione degli stessi.
Nel nostro sistema verrà utilizzato l'architettura MVC, Model View Control, generalmente applicata ai sistemi web, che separa gli oggetti in tre livelli:

- Model, rappresenta il sistema di gestione dei dati. Si occupa della memorizzazione dei dati e delle interazioni con il database;
- View, rappresenta il sistema di interazione diretta con l'utente; rappresenta in tutto e per tutto l'interfacciamento che il sistema ha con tutti gli utenti che possono interagire con il sistema.
- Controller,è responsabile della sequenza di iterazioni con l'utente e notifica alle varie View i cambiamenti che avvengono nel modello.;

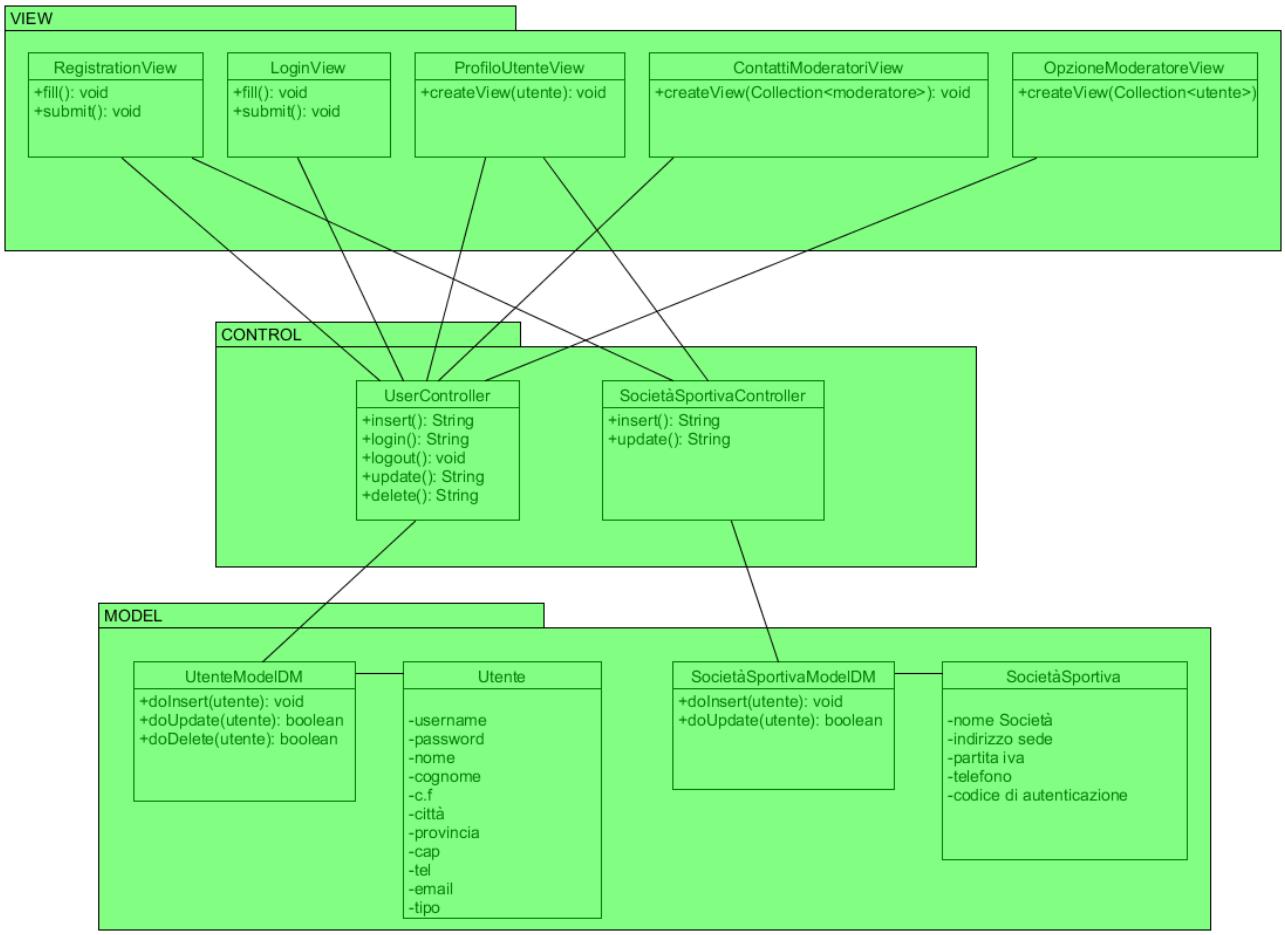
Questo tipo di architettura implica anche la tradizionale separazione fra logica applicativa , a carico del controller e del model, e l'interfaccia utente a carico del view.
I vantaggi che caratterizzano l'utilizzo di questo stile architetturale sono notevoli: eliminazione delle incongruenze nei dati, maggiore sicurezza delle informazioni

(soprattutto per quanto riguarda i dati personali degli utenti il cui trattamento dovrà essere conforme alle norme vigenti).
Una soluzione Client/Server offre anche un notevole risparmio economico. Inoltre, si ha una maggiore flessibilità delle esigenze di mercato, soprattutto per quanto riguarda i prodotti di supporto al software.

3.2 Decomposizione in sottosistemi

I tre livelli, Model View Controller, devono essere rispettati nello sviluppo del sistema considerando l'indipendenza esterna ed un forte accoppiamento interno, in modo da permettere una facile manutenibilità ed un'estendibilità semplice e flessibile.

3.2.1 Gestione Utente



Di seguito è riportata una breve descrizione delle view del sottosistema:

- **RegistrationView** permette all'Utente di inserire i dati e sottometterli al sistema.
- **LoginView** permette all'utente di inserire i propri dati di accesso e di sottometterli per la richiesta di autenticazione al sistema.
- **ProfiloUtenteView** permette all'utente di interfacciarsi con il sistema.
- **ContattiModeratoriView** permette all'utente di visualizzare i contatti dei moderatori.
- **OpzioneModeratoreView** permette al moderatore di visualizzare la lista dei PartnerSportivi, degli Utenti Semplici e di bannarli, sbannarli o cancellarli.

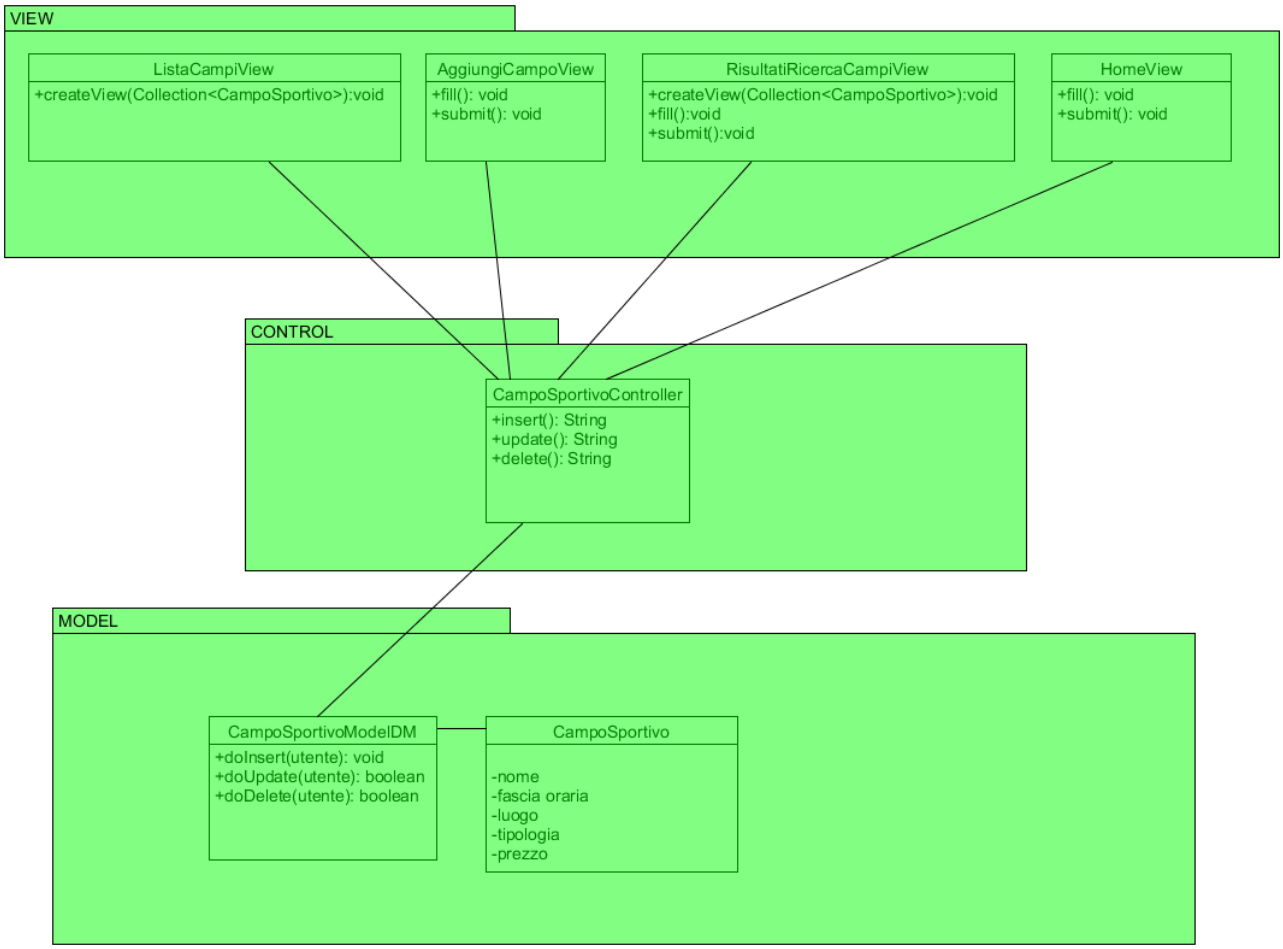
I Controller sono: **UserController** e **SocietàSportivaController.**
Le principali funzioni del UserController sono:

- **Login** consente all'Utente di autenticarsi al sistema.
- **Logout** consente all'Utente di terminare la sessione di lavoro e disconnettersi dal sistema.
- **Insert** consente di inserire un nuovo Utente presente all'interno del sistema.
- **Delete** consente al Moderatore di cancellare un Utente presente nel sistema.
- **Update** consente di aggiornare le informazioni relative ad un Utente presente nel sistema.

Le principali funzioni della SocietàSportivaController sono:

- **Insert** consente al PartnerSportivo di inserire una nuova Società Sportiva nel sistema in fase di registrazione.
- **Update** consente al Partner Sportivo di aggiornare le informazioni relative alla Società Sportiva pesente nel sistema.

3.2.2 Gestione Campo Sportivo



Di seguito è riportata una breve descrizione delle view del sottosistema:

- **ListaCampiView** è la view che contiene la lista dei Campi Sportivi presenti nel sistema relativa ad un Partner Sportivo.
- **AggiungiCampoView** consente al Partner Sportivo di aggiungere un nuovo Campo Sportivo nel sistema.
- **RisultatiRicercaCampiView** è la view che contiene i risultati di una ricerca effettuata dall'Utente.
- **HomeView** contiene la form per effettuare ricerche di Campi Sportivi presenti

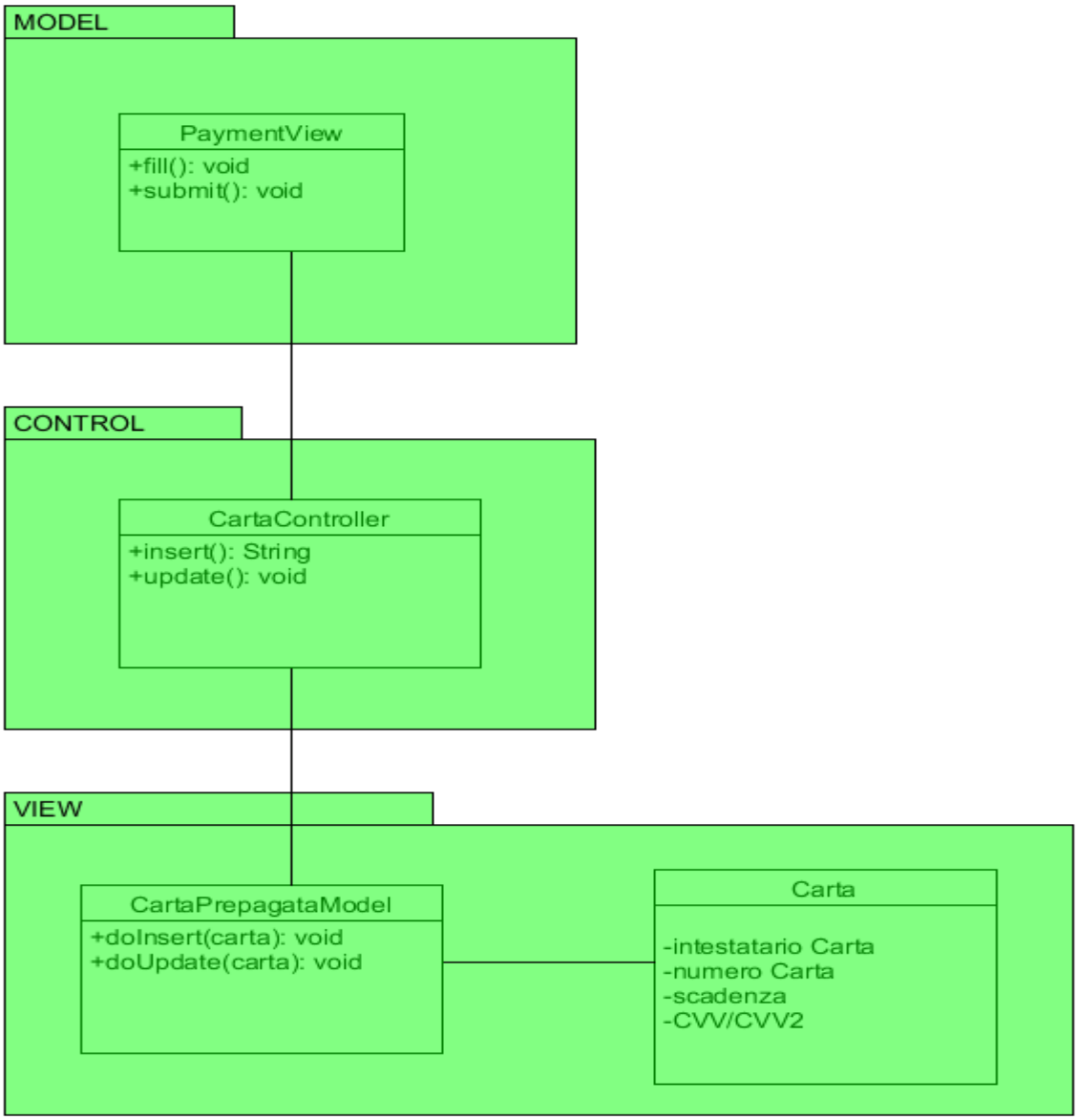
nell’area selezionata.

Il Controller che gestisce i Campi Sportivi è: **CampoSportivoController.**

Le principali funzioni del CampoSportivoController sono:

- **Insert** consente di inserire un nuovo Campo Sportivo all’interno del sistema.
- **Delete** consente al Partner Sportivo di cancellare un Campo Sportivo presente nel sistema.
- **Update** consente di aggiornare le informazioni relative ad un Campo Sportivo presente nel sistema.

3.2.3 Gestione Pagamento



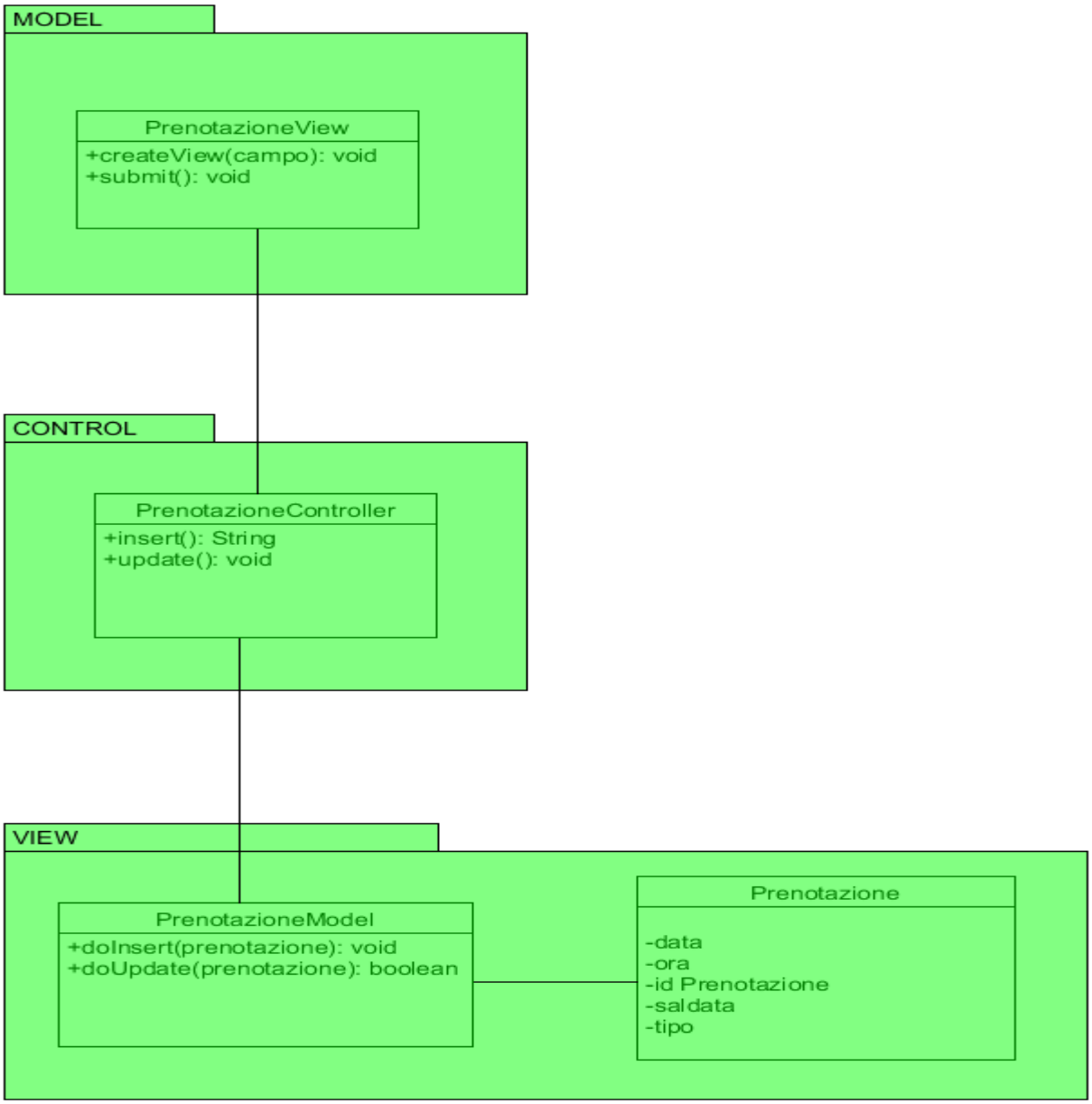
Di seguito è riportata una breve descrizione delle view del sottosistema:

- **PaymentView** permette all'Utente di inserire i dati della Carta e sottometterli al sistema.

Il Controller che gestisce le Carte Prepagate è: **CartaController.**
Le principali funzioni di CartaController sono:

- **Insert** consente di inserire una nuova Carta Prepagata all'interno del sistema.
- **Update** consente di aggiornare le informazioni relative ad una Carta Prepagata presente nel sistema.

3.2.4 Gestione Prenotazione



Di seguito è riportata una breve descrizione delle view del sottosistema:

- **PrenotazioneView** permette all'Utente di visualizzare un riepilogo della Prenotazione scelta e sottometterla al sistema.

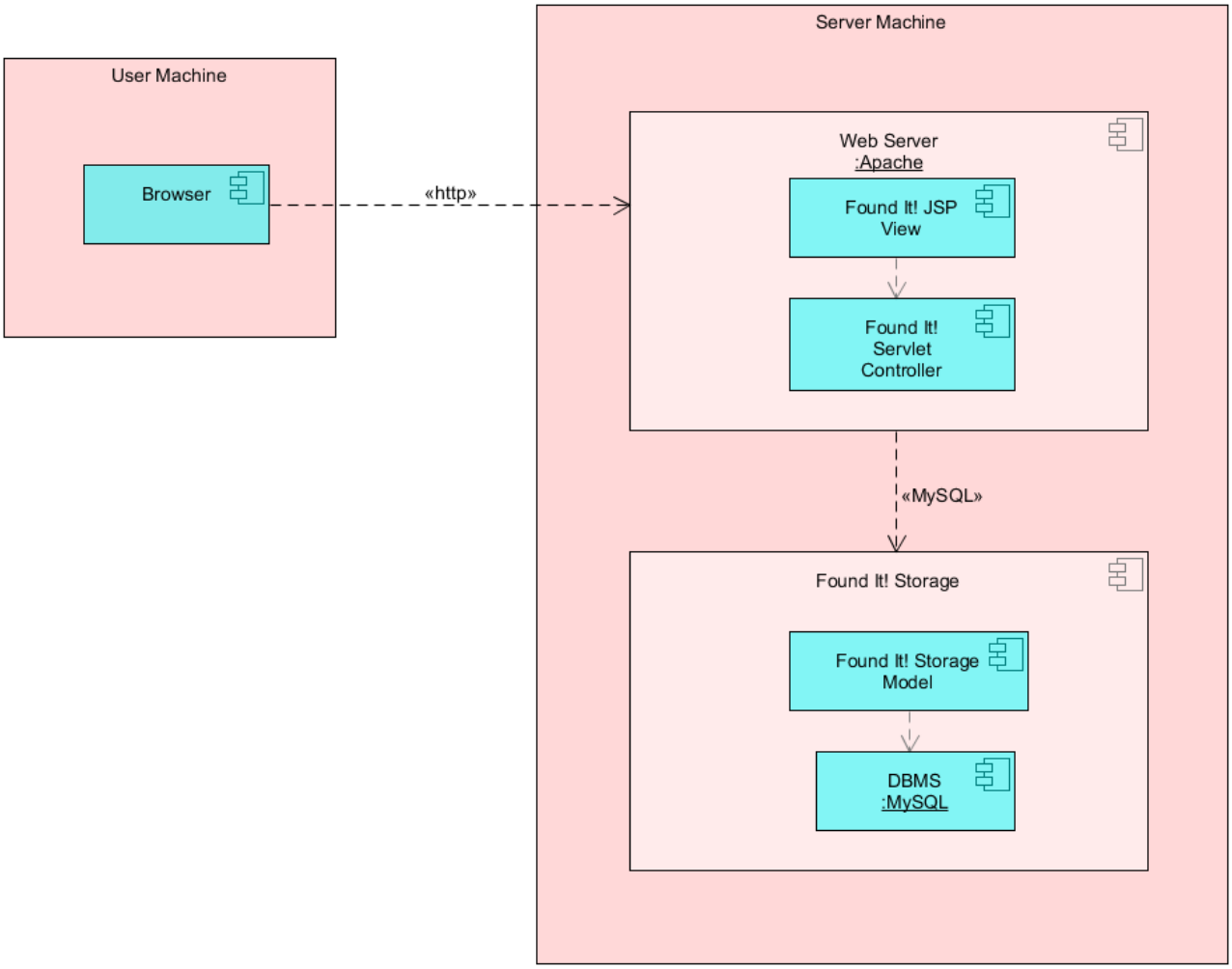
Il Controller che gestisce le Prenotazioni è: **PrenotazioneController.**
Le principali funzioni di PrenotazioneController sono:

- **Insert** consente di inserire una nuova Prenotazione all'interno del sistema.
- **Update** consente al Partner Sportivo di aggiornare le informazioni relative ad una Prenotazione presente nel sistema.

3.3 Mapping Hardware/Software

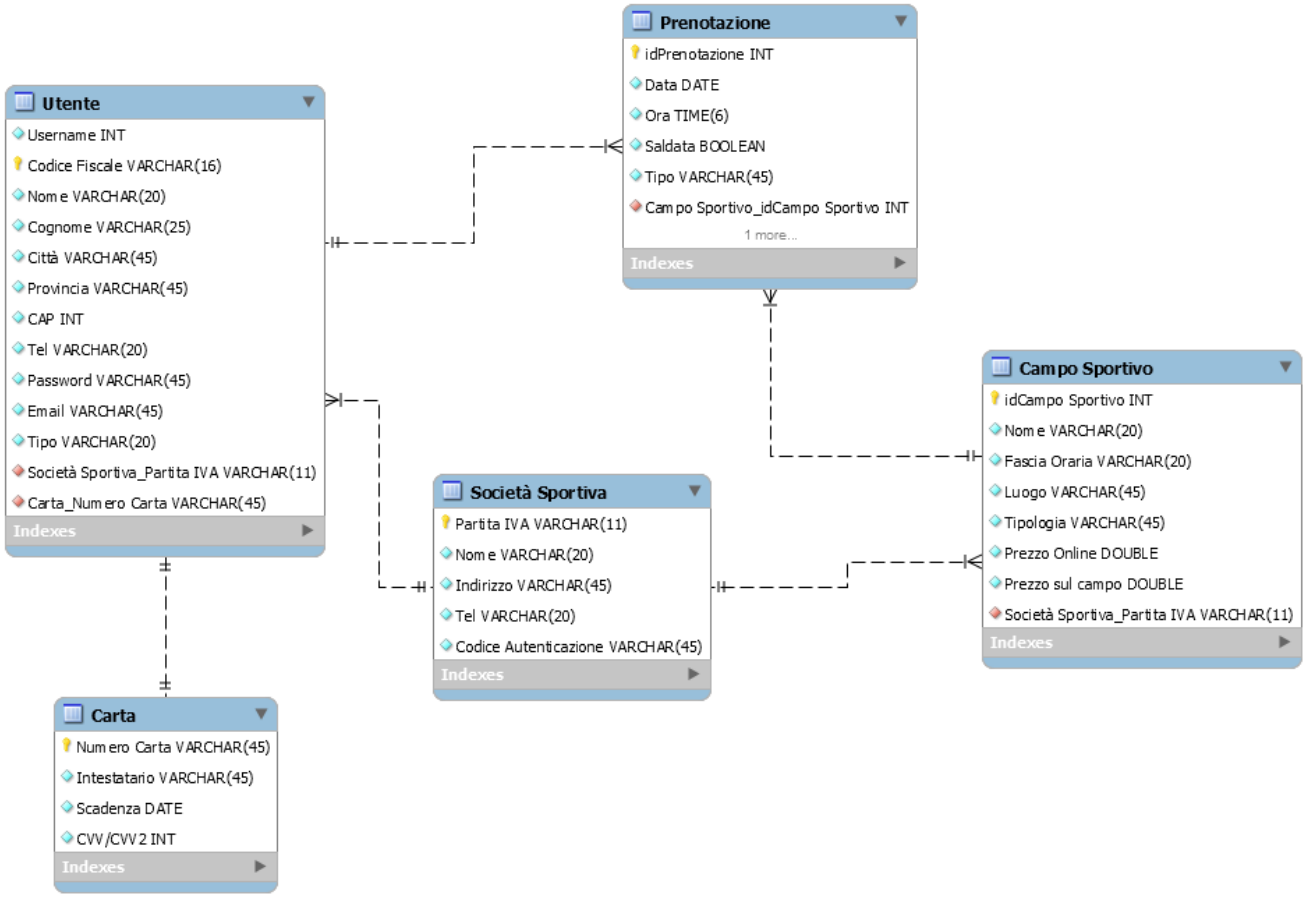
Il Sistema deve essere scomposto in due nodi fondamentali, seguendo l'architettura Client/Server:

- Nodo Client: L'Utente accede da remoto al Sistema attraverso il browser.
- Nodo Server: Contiene il web server su cui viene installato il Sistema e dal quale vengono erogati i servizi.



3.4 Gestione Dati Persistenti

Per la memorizzazione dei dati si è scelto di utilizzare un Database relazionale che tramite il concetto di tabelle e relazioni permette di gestire in modo ottimale l'archiviazione delle informazioni di varie entità del sistema e soprattutto permette una facile interrogazione per poter ottenere le informazioni.
Si è scelto di usare il DBMS open source MySQL dato che permette la creazione di "database relazionali" ossia consente la conservazione dei dati in tabelle separate anziché in un'unica grande entità. Grazie a questa tecnica si ha una buona flessibilità e velocità di accesso ai dati ed una maggiore modellazione delle basi di dati. MySQL presenta una struttura multi-thread comprendente un thread fisso che controlla la connessione in ingresso e un thread attivo per ogni connessione che fa in modo da impedire che due thread scrivano sulla stessa tabella nello stesso momento evitando così perdita di informazioni.
Si è scelto di utilizzare un DBMS per rendere la base di dati gestibile anche da un amministratore non avvezzo ai sistemi informatici complessi grazie ad un'interfaccia utente molto semplice da utilizzare, rispetto alla visione del codice sorgente di ogni tabella.



Legenda

	Chiave Primaria
	Relazione 1 a 1
	Relazione 1 a N

3.5 Controllo degli accessi e sicurezza

Found It! è un sistema multiutente, quindi attori differenti hanno il permesso di eseguire diverse operazioni su vari insiemi di oggetti.
Il sistema software è protetto da accessi grazie ad un meccanismo di autenticazione che, tramite una combinazione di login e password, regola gli accessi al sistema. Il sistema analizzerà i campi login e password e verificherà che nel database sia presente la tupla contenente login e password specificati.
Nel caso in cui l'accesso venga negato verrà presentato all'utente un messaggio di errore e riproposto l'inserimento dei dati necessari per l'accesso. Il sistema inoltre riconosce automaticamente quali sono i servizi destinati e permessi all'utente che si è autenticato nel sistema. Di seguito riportiamo la tabella degli accessi:

Tipologia Utente / Funzionalità	Utente non Registrato	Utente Semplice	Partner Sportivo	Moderatore
Registrazione Utente	V	X	X	X
Modifica Utente	X	V	V	V
Cancella Utente	X	X	X	V
Aggiungi CampoSpo	X	X	V	X
Modifica CampoSpo	X	X	V	X
Elimina CampoSpo	X	X	V	X
Ban Utente	X	X	X	V

Sban Utente	X	X	X	V
Ricerca Campo	V	V	V	V
Prenota Campo	X	V	V	V
Occupa Campo	X	X	V	X
Cancella Prenotazio	X	V	V	V

3.6 Controllo globale del software

Il flusso di controllo del sistema deve essere gestito all'interno del livello Controller:ciascuna servlet deve verificare, all'interno della sessione, la presenza delle informazioni necessarie affinché al passo attuale sia garantito uno stato consistente dell'operazione in corso. Al verificarsi di un avvenimento (il più delle volte generato da un utente tramite l'interfaccia grafica) il controllo viene trasferito al relativo oggetto che lo gestisce. Il suddetto oggetto, ricevuto il controllo, manda in esecuzione il relativo sottosistema. Nella maggior parte dei casi, ogni operazione termina con un messaggio di notifica all'utente utilizzatore del Client.

3.7 Boundary conditions

Per ogni oggetto persistente, si esamina in quale use case è creato o distrutto. Per ogni oggetto non creato o non distrutto in uno degli use case, si aggiunge uno use case invocato dall'amministratore di sistema.

- Utente: gli oggetti Utente di tipo **Utente Semplice** vengono creati nel caso d'uso UC_0.1 Registrazione Utente Semplice, archiviati all'uscita e cancellati dal Moderatore nel caso d'uso UC_2.16 Cancellazione Utente. Gli oggetti Utente di tipo **Partner Sportivo** vengono creati nel caso d'uso UC_0.2 Registrazione Partner Sportivo con Società, archiviati all'uscita e cancellati dal Moderatore nel caso d'uso UC_2.16 Cancellazione Utente. Gli oggetti Utente di tipo **Moderatore** vengono creati dall'Amministratore del sistema, archiviati all'uscita e cancellati dallo stesso.

- Campo Sportivo: gli oggetti Campo Sportivo vengono creati dal Partner Sportivo nel caso d'uso UC_2.7 Aggiunta campi sportivi e cancellati dallo stesso nel caso d'uso UC_2.10 Rimozione campi sportivi.
- Società Sportiva: gli oggetti Società Sportiva vengono creati dal Partner Sportivo nel caso d'uso UC_0.2 Registrazione Partner Sportivo con Società.
- Prenotazione: gli oggetti Prenotazione vengono creati nei casi d'uso UC_3.2 Prenotazione Online di un Campo Sportivo e UC_3.4 Occupazione fascia oraria.
- Carta: gli oggetti Carta vengono creati nel caso d'uso UC_3.2 Prenotazione Online di un Campo Sportivo.

3.7.1 Casi d'uso per Startup

Caso d'uso StartWebServer

Nome Use Case	*StartWebServer*
Partecipanti	*Amministratore del sistema*
Flusso degli eventi 1. *L'Amministratore del sistema accede alla macchina che contiene il Web Server.* 2. *L'Amministratore del sistema manda in esecuzione il Web Server.* 3. *Il Sistema avvia il Web Server.*	
Condizione d'ingresso	*L'Amministratore ha accesso al Sistema*
Condizione di uscita	*Il Web Server è stato avviato*

Caso d'uso StartMySql

Nome Use Case	*StartMySql*
Partecipanti	*Amministratore del sistema*
Flusso degli eventi 1. *L'Amministratore del sistema accede alla macchina che contiene il Server MySql.* 2. *L'Amministratore del sistema manda in esecuzione il Server MySql.* 3. *Il Sistema avvia il Server MySql.*	
Condizione d'ingresso	*L'Amministratore ha accesso al Sistema*
Condizione di uscita	*Il Server MySql è stato avviato*

3.7.2 Casi d'uso per lo Shutdown

Caso d'uso ShutdownWebServer

Nome Use Case	*ShutdownWebServer*
Partecipanti	*Amministratore del sistema*
Flusso degli eventi 1. *L'Amministratore del sistema accede alla macchina che contiene il Web Server.* 2. *L'Amministratore del sistema preme il pulsante "Spegni".* 3. *Il Sistema effettua una scansione per verificare se ci sono eventuali richieste in sospeso, porta a termine le richieste e avvia le procedure di arresto.* 4. *Il Sistema notifica il successo dell'operazione.*	
Condizione d'ingresso	*Il Web Server è in esecuzione*

Condizione di uscita	*Il Web Server è stato spento correttamente*

Caso d'uso ShutdownMySql

Nome Use Case	*ShutdownMySql*
Partecipanti	*Amministratore del sistema*
Flusso degli eventi 1. *L'Amministratore del sistema accede alla macchina che contiene il Server MySql.* 2. *L'Amministratore termina l'applicazione del Server MySql.* 3. *Il Sistema notifica il successo dell'operazione.*	
Condizione d'ingresso	*Il Server MySql è in esecuzione*
Condizione di uscita	*Il Server MySql è stato spento*

3.7.3 Comportamento del Sistema in situazioni eccezionali

Nel caso in cui si verifichi un crash del sistema (dovuto ad un errore hardware o software), si tenta un ripristino della configurazione precedente allo stato d'errore e un riavvio del sistema. Dato che i dati vengono gestiti dal DB non c'è rischio di perdita dei dati. Tuttavia in caso di guasti al supporto di memorizzazione dei dati nel database server, inevitabilmente i dati verranno persi.

Per rendere minimo tale rischio, verranno eseguiti periodicamente dei backup del database del sistema. Per evitare problemi è comunque consigliabile effettuare dei

controlli periodici sull'hardware. In caso di crash dovuto ad un bug nel codice del sistema, si distinguono tre casi:

- Il crash si è verificato nel client. Il sistema potrà, quindi, continuare a funzionare regolarmente.
- Il crash si è verificato nel server. Il sistema risulta quindi inutilizzabile, in quanto non è possibile comunicare con il database.
- Il crash si è verificato nel database server. Il sistema risulta inutilizzabile, in quanto non è possibile accedere ai dati.

In tutti e tre i casi, Found It! provvederà a segnalare nella maniera opportuna il tipo di problema riscontrato.

4. SERVIZI DEI SOTTOSISTEMI

4.1 Gestione Utente

Sottosistema	Descrizione
Gestione Utente	Sottosistema che gestisce la registrazione di un utente, la sua autenticazione e le operazioni necessarie alla sua gestione.
Servizio	**Descrizione**
creaUtente()	Permette di inserire un nuovo Utente

	all'interno del database
inoltraLogin()	Permette ad un utente di poter effettuare l'accesso al sistema.
inoltraLogout()	Permette ad un utente di uscire dal sistema.
cancellaDatiUtente()	Permette al Moderatore di cancellare un Utente presente nel sistema.
bannaUtente()	Permette al Moderatore di bannare un utente.
getUtentiBannati()	Permette di visualizzare la lista di tutti gli utenti bannati.
sbannaUtente()	Permette al Moderatore di sbannare un utente.
cambiaPassword()	Permette ad un Utente di cambiare la propria password.
getUtenti()	Permette al Moderatore di visualizzare una lista di utenti.
modificaDati()	Permette ad un Utente di modificare i propri dati.

4.2 Gestione Campo Sportivo

Sottosistema	Descrizione
Gestione Campo Sportivo	Sottosistema che gestisce la ricerca, l'inserimento, la rimozione e la modifica dei dati di un Campo Sportivo.
Servizio	**Descrizione**
creaCampoSportivo()	Permette di inserire un nuovo Campo Sportivo all'interno del database
rimuoviCampo()	Permette di cancellare un Campo Sportivo dal sistema.
aggiornaCampo()	Permette di modificare i dati di un Campo Sportivo presente sul database.
doRetrieveCampiSportivi()	Permette di cercare determinati Campi Sportivi presenti sul database.

4.3 Gestione Pagamento

Sottosistema	Descrizione
Gestione Pagamento	Sottosistema che gestisce l'inserimento di un metodo di Pagamento.
Servizio	**Descrizione**
inserisciPagamento()	Permette di inserire una nuova Carta Prepagata all'interno del sistema.
updatePagamento()	Permette di aggiornare le informazioni relative ad una Carta Prepagata presente nel sistema.

4.4 Gestione Prenotazione

Sottosistema	Descrizione
Gestione Prenotazione	Sottosistema che gestisce l'inserimento e

	la modifica dei dati di una Prenotazione.
Servizio	**Descrizione**
confermaPrenotazione()	Permette di inserire una nuova Prenotazione all'interno del database
prenota()	Permette al Partner Sportivo di occupare una fascia oraria.