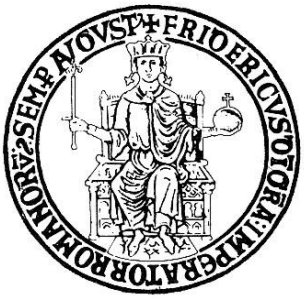
Elaborato in

Software Architecture Design



Università degli Studi di Napoli Federico II

Corso di Laurea Magistrale in Ingegneria Informatica

Prof.ssa:

Annarita Fasolino

Candidati:

Nunzio Tarallo

Riccardo Arnone

Roberto Maiello

Indice

Introduzione 5
Capitolo 1 5
Processo di sviluppo e tools adottati 5
1.1 Processo di sviluppo software 5
1.1.1 Up Agile 6
1.2 Organizzazione del lavoro 6
1.3 Tool utilizzati 9
1.4 Stima dei costi 10
1.5 Primo workshop dei requisiti e prima iterazione 13
1.6 Secondo workshop dei requisiti e seconda iterazione 13
1.7 Terzo workshop dei requisiti e terza iterazione 13
Capitolo 2 14
Fase di Avvio del progetto 14
2.1 Descrizione degli obiettivi 14
2.2 Analisi del testo 15
2.3 Tabella attori-obiettivi 16
2.4 Requisiti Funzionali in formato breve 17
2.5 Requisiti non Funzionali 19
2.6 Vincoli generali 20
2.7 Diagramma dei Casi d'Uso 21
Capitolo 3 22
Analisi e Specifica dei Requisiti 22
3.1 Identificazione degli attori 22
3.2 Descrizione dettagliata dei casi d'uso 22
3.3 Diagramma di Dominio 28
3.4 Diagrammi di Sequenza di Analisi 29
3.4.1 Sequence Diagram - Effettua Registrazione 29
3.4.2 Sequence Diagram – Effettua Login 30
3.4.3 Sequence Diagram – Visualizza Campi 31
3.4.4 Sequence Diagram – Visualizza Disponibilità 31
3.4.5 Sequence Diagram - Prenota Campo 32
3.4.5 Sequence Diagram - Gestisci Campo 33

3.5 Activity Diagram 34
3.5.1 Activity Diagram - Effettua Registrazione 34
3.5.2 Activity Diagram - Effettua Login 35
3.5.3 Activity Diagram - Visualizza Campi 36
3.5.4 Activity Diagram - Visualizza Disponibilità 37
3.5.5 Activity Diagram - Prenota Campi 38
3.5.6 Activity Diagram - Gestisci Campi 39
3.6 Diagramma delle Classi 40
3.7 Diagramma di Contesto 41
Capitolo 4 42
Architettura Software e scelte di progetto 42
4.1 Stile Architetturale 42
4.1.1 Diagramma dei Componenti 42
4.1.2 Diagramma dei Package 43
4.2 Pattern Architetturale 43
4.3 Client 44
4.3.1 Model-View-ViewModel (MVVM) 45
4.3.2 Android's Architecture Components 47
4.4 Server 49
4.5 Comunicazione Remota con Socket 50
4.6 Pattern Proxy-Skeleton e Thread 50
4.7 Diagrammi Architetturali 51
4.7.1 Vista Architetturale - Server 51
4.7.2 Vista Architetturale - Client 53
4.8 Diagramma di Sequenza Raffinati 54
4.8.1 Sequence Diagram Raffinato – Effettua Login 54
4.8.2 Sequence Diagram Raffinato - Visualizza Campi 55
4.8.3 Sequence Diagram Raffinato - Visualizza Disponibilità 56
4.8.4 Sequence Diagram Raffinato - Prenota Campo 57
4.9 Schema del Database 58
Capitolo 5 59
Implementazione del software 59
5.1 Documentazione dell'implementazione 59
5.1.1 Diagramma di deployment 59
5.2 Manuale di configurazione e avvio 60

5.2.1 Database 60
5.2.2 Server 61
5.2.3 Client 62
Capitolo 6 64
Testing dell'applicazione 64
6.1 Test Eseguiti 64

Introduzione

Il progetto è stato sviluppato nell'ambito del corso "Software Architecture Design" durante l'anno accademico 2021/2022.

Questo ha previsto la realizzazione un'applicazione Android per la gestione del complesso sportivo "Activity World". L'obiettivo è stato quello di migliorare i servizi offerti ai clienti del complesso, a partire dalla visualizzazione dei vari campi sportivi presenti nella struttura, fino alla procedura completa di prenotazione con il relativo pagamento.

L'applicazione richiede l'interazione con gli utenti e con l'amministratore del sistema, che può modificare le informazioni e i servizi offerti dal complesso.

Capitolo 1

Processo di sviluppo e tools adottati

1.1 Processo di sviluppo software

Per quanto riguarda la metodologia di lavoro scelta, si è deciso di adottare il processo di sviluppo software iterativo ed evolutivo UP (Unified Process) . Tale processo, ha permesso di ridurre il rischio di fallimento e di gestire opportunamente le modifiche dovute al cambiamento dei requisiti in seguito a rilasci incrementali del software, riducendo i costi e il lavoro necessari.

1.1.1 Up Agile

Lo sviluppo Agile ha permesso di focalizzare l'attenzione sul codice piuttosto che su design e documentazione, superando la "gravosità" dei precedenti approcci guidati dai piani e consentendo il rapido rilascio di software funzionante per ottenere feedback.
Quindi, in ogni iterazione, è stato selezionato un piccolo insieme di attività principali da elaborare ed è stata sviluppata solo la documentazione necessaria per la comprensione del task, invece i problemi semplici sono stati risolti direttamente in fase di programmazione.

Lo sviluppo UP si basa sul raffinamento e perfezionamento del sistema con il susseguirsi di molteplici iterazioni, con continui feedback e adattamenti ciclici. Il processo è costituito da quattro fasi: la prima è la fase di ideazione, nella quale si identificano i principali casi d'uso e le priorità relative e viene fatta un'analisi di fattibilità e una stima di costi e tempi; la seconda è la fase di elaborazione, nella quale viene fatta un'implementazione iterativa del nucleo dell'architettura; la terza è la fase di costruzione, ovvero dell'implementazione iterativa degli elementi rimanenti; l'ultima è la fase di transizione, ovvero quella di testing e successivo rilascio del software.

1.2 Organizzazione del lavoro

Lo sviluppo è stato eseguito da un team composto da tre studenti ed ha avuto una durata di 6 settimane, durante le quali sono state eseguite 3 iterazioni che hanno permesso una sua evoluzione attraverso vari raffinamenti successivi.

Il team si è riunito telematicamente 5 giorni a settimana per 6 ore al giorno circa, per un totale di 30 ore settimanali e 180 complessive per ogni componente del progetto. Di seguito si mostrano le fasi del processo UP con le relative durate ed elaborati prodotti:

Fase	Durata	Iterazioni	Attività	Milestones
Ideazione	1 settimana	1	Studio di fattibilità (stima costi e tempi), esplorazione dei requisiti	Specifica dei requisiti funzionali e non, Modello dei casi d'uso, Piano di iterazione.
Elaborazione	2 settimane	3	Analisi e specifica dei requisiti, implementazione iterativa e sviluppo del prototipo architetturale	Modello di dominio, Diagrammi di progettazione, Documento di architettura, Prototipo architetturale
Costruzione	2 settimane	3	Implementazione iterativa degli elementi rimanenti a minor rischio e raffinamento dei diagrammi	Modello di dominio, Diagrammi di progettazione, documento di architettura, Prototipo funzionante
Transizione	1 settimana	3	Completamento funzionale e test	Test e rilascio

Per tener traccia del lavoro svolto quotidianamente è stato utilizzato Smartsheet: una web Platform, utilizzata specialmente in ambito agile, che aiuta i team nella divisione e coordinazione del lavoro.

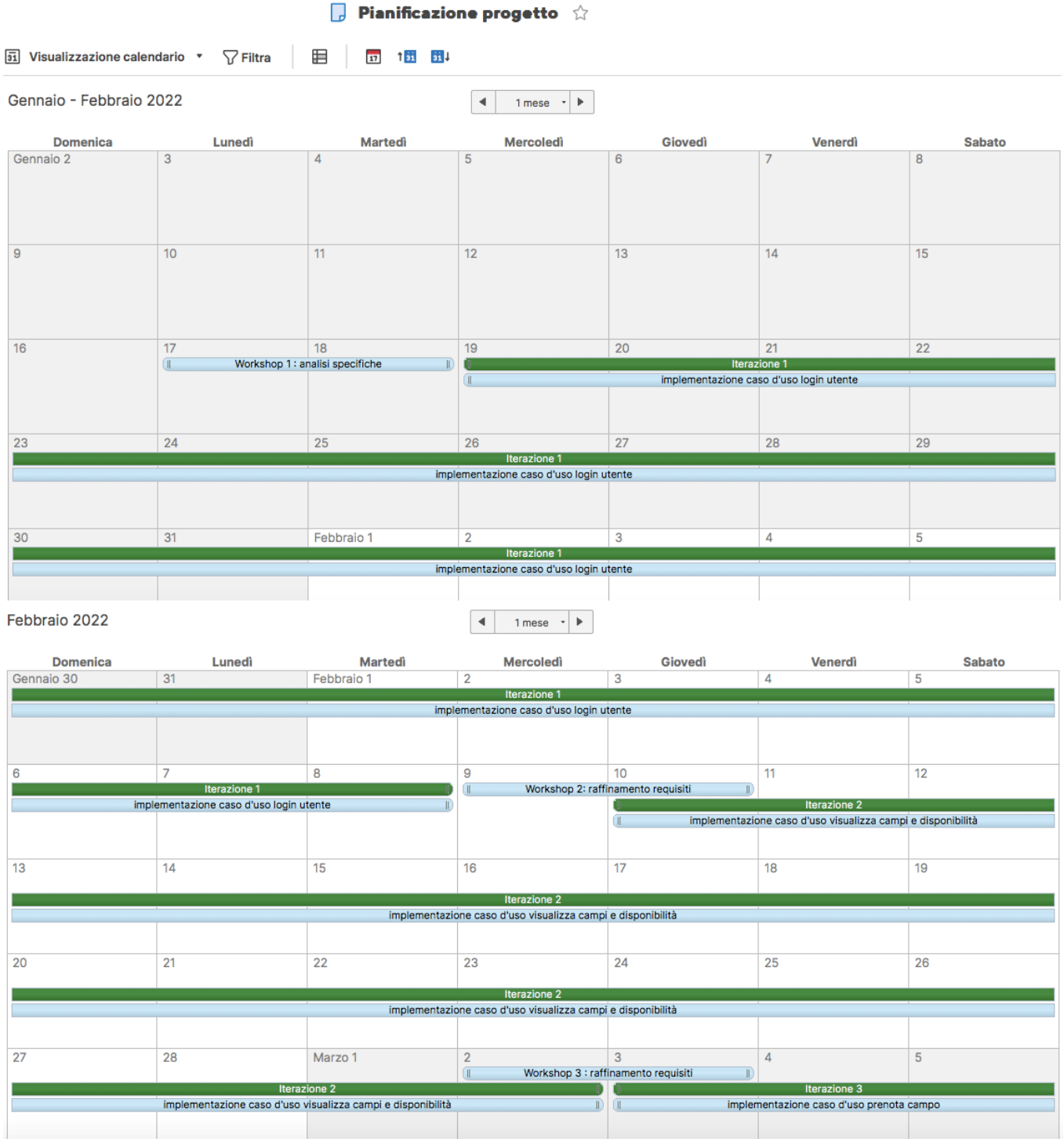


Figura 1 : Timeboxing 1

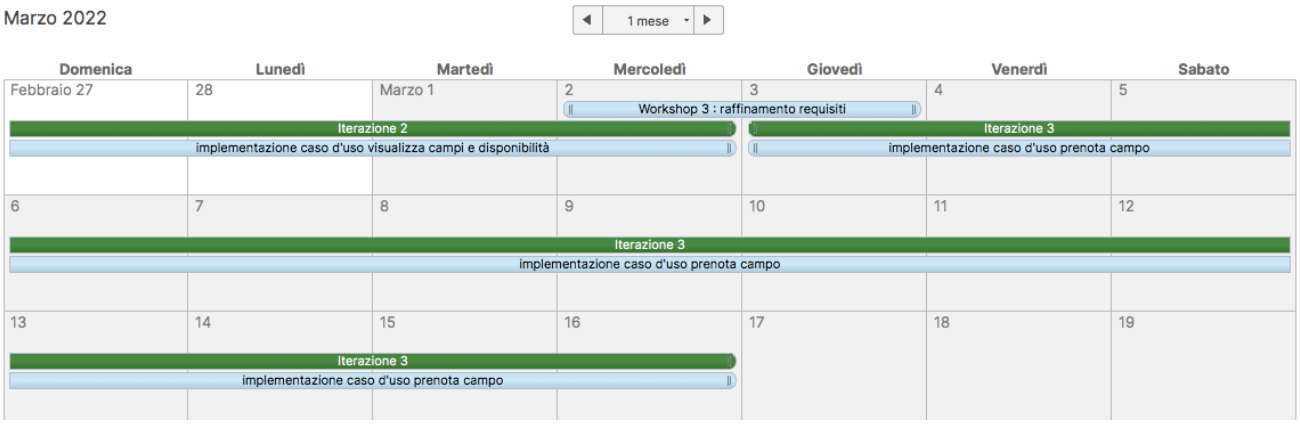


Figura 2: Timeboxing 2

1.3 Tool utilizzati

- **Microsoft Teams** per organizzare riunioni e discutere delle decisioni da prendere.
- **Visual Paradigm** per la modellazione in UML.
- **Google doc** per la documentazione.
- **XAMPP,** una piattaforma software costituita dal server Apache per la connessione al DB.
- **Database Mysql** per la memorizzazione dei dati.
- **Android Studio** per lo sviluppo del lato client dell'applicazione mobile realizzata.
- **Eclipse** per lo sviluppo del lato server.
- **GitHub** per il version control, la collaborazione degli sviluppatori, e per lo storage dell'applicazione realizzata
- **Smartsheet**, web application, utilizzata specialmente in campo agile, che aiuta gli sviluppatori nell'organizzazione del lavoro

1.4 Stima dei costi

Lo sviluppo ha previsto una fase iniziale in cui è stata discussa e poi definita l'idea da realizzare in gruppo. Inoltre in questa fase, si è valutata l'effettiva realizzabilità del progetto, principalmente in relazione al tempo, ai costi e al lavoro necessari. Per prevedere la dimensione del software e di conseguenza stimarne i tempi e i costi, indipendentemente dal team di sviluppo, è stato utilizzato il metodo degli Use Case Points.

Come primo parametro è stato calcolato l'UUCP (Unadjusted Use Case Points), dato dalla somma degli UUCW (Unadjusted Use Case Weight) e UAW (Unadjusted Actor Weight):

UUCP = UUCW + UAW = 150 + 11 = 161

Use Case Complexity	Weight	Number of Use Cases	Product
Simple	**5**	**2**	**10**
Average	**10**	**2**	**20**
Complex	**15**	**8**	**120**
Total (UUCW)			**150**

Actor Type	Weight	Number of Actor	Product
Simple	**1**	**2**	**2**
Average	**2**	**0**	**0**
Complex	**3**	**3**	**9**
Total (UAW)			**11**

A questo punto è stato determinato il fattore di complessità tecnica, meglio noto come TFactor, attribuendo ad ognuno dei tredici fattori un punteggio indicativo della rilevanza degli stessi: 0 irrilevante - 5 molto importante.

Fattore	Peso	Valutazione	Impatto
Sistema distribuito	**2**	**2**	**4**
Prestazioni	**2**	**3**	**6**
Efficienza end-user	**1**	**3**	**3**
Complessità di elaborazione	**1**	**2**	**2**
Riusabilità del codice	**1**	**2**	**2**
Facilità di installazione	**0.5**	**4**	**2**
Facilità di uso	**0.5**	**4**	**2**
Portabilità	**2**	**3**	**6**
Facilità di cambiamento	**1**	**3**	**3**
Uso concorrente	**1**	**4**	**4**
Sicurezza	**1**	**3**	**3**
Accesso per terze parti	**1**	**1**	**1**
Necessità di formazione	**1**	**0**	**0**
Totale (TFactor)			**38**

TFC = 0,6 + (0,01 * TFactor) = 0,6 + (0,01 * 38,0) = 0,98.

Successivamente, è stato ricavato L'EFactor, meglio noto come Environmental Factor:

Fattore	Peso	Valutazione	Impatto
Familiarità con processo di sviluppo	**1.5**	**3**	**4.5**
Esperienza applicativa	**0.5**	**1**	**0.5**
Esperienza orientata ad oggetti	**1**	**3**	**3**
Capacità di condurre analisi	**0.5**	**3**	**1.5**
Motivazione	**1**	**5**	**5**
Requisiti stabili	**2**	**3**	**6**
Staff part-time	**-1**	**0**	**0**
Difficoltà linguaggio di programmazione	**-1**	**3**	**-3**
Totale (EFactor)			**17.5**

EF = 1,4 + (-0,03 * EFactor) = 1,4 + (-0,03 * 17,5) = 1,4 - 0,525 = 0,875.

Infine, una volta ricavati il TFC e l'EF possiamo individuare gli Use Case Point, moltiplicando gli UUCP per i due fattori corretti:

UCP = UUCP * TFC * EF = 138

Supponendo un rapporto di 20 ore per UCP (Metodo Karner), otteniamo un quantitativo di ore pari a:

Ore totali = 20 * 138 = 2760h

1.5 Primo workshop dei requisiti e prima iterazione

Inizialmente c'è stata una riunione con lo stakeholder per definire i requisiti di progetto e stilare un primo documento di specifica dei requisiti.

Prima della prima iterazione, si è tenuto il primo workshop dei requisiti elencando i nomi dei casi d'uso e assegnando ad ognuno una priorità. Si è scelto come caso d'uso cardine dell'applicazione da realizzare quello relativo alla funzionalità di "Prenota Campo", che risulta significativo rispetto all'architettura e dunque ad elevato valore di business e di rischio.

Quindi nel primo workshop sono stati stabiliti gli obiettivi della prima iterazione e in particolare, si è posta l'attenzione sui requisiti che avrebbero potuto garantire un funzionamento corretto del sistema nel minor tempo possibile, separandoli da quelli che sarebbero potuti essere implementati in una fase avanzata del processo di sviluppo. Pertanto, nella prima iterazione si è scelto di concentrarsi sui casi d'uso relativi all'utente, e si è scelto di implementare il caso d'uso Effettua Login con i relativi test di controllo.

1.6 Secondo workshop dei requisiti e seconda iterazione

Verso la fine della prima iterazione si è svolta una riunione in cui il team ha verificato se gli obiettivi della prima iterazione fossero stati raggiunti e se fosse necessario apportare dei miglioramenti.

Durante il secondo workshop dei requisiti tutto il lavoro del primo workshop è stato rivisto e raffinato, sono stati scelti altri requisiti da approfondire e sono stati implementati i casi d'uso "Visualizza Campi" e "Visualizza Disponibilità".

1.7 Terzo workshop dei requisiti e terza iterazione

Nel terzo workshop, dopo un ulteriore raffinamento, è stato scelto di descrivere il caso d'uso Prenota Campo, e dopo una fase di analisi e progettazione, si è passati alla programmazione e al test.

Capitolo 2

Fase di Avvio del progetto

2.1 Descrizione degli obiettivi

Nella fase iniziale dell'avvio del progetto sono stati stabiliti gli obiettivi principali del sistema "Centro Sportivo ActivityWorld".

Si vuole realizzare un'applicazione mobile per la gestione automatizzata dei servizi offerti dal centro sportivo, di cui solo un utente registrato può beneficiare. Il complesso dispone di 20 campi sportivi di varie tipologie: tennis, paddle, calcio, basket e pallavolo, ognuno caratterizzato da un proprio identificativo numerico e uno status che può essere disponibile o prenotato. Inoltre, ad ogni campo è assegnato un numero massimo di partecipanti, dei quali occorre memorizzare nome e cognome (causa covid).

In particolare il sistema deve permettere agli utenti di visualizzare le tipologie dei campi e, dopo aver eseguito la procedura di autenticazione, di verificarne la disponibilità, consultare gli orari accessibili e le tariffe per ogni tipologia di campo. Inoltre, per ognuna di queste tipologie, è associata l'attrezzatura da poter noleggiare nel complesso, ma previa richiesta. L'utente registrato, selezionata la tipologia e l'orario, può prenotare il campo e l'attrezzatura ,eventualmente necessaria, e fornire la lista completa dei partecipanti con nome e cognome. Il sistema aggiorna in maniera automatica la disponibilità dei campi e dell'attrezzatura relativa; in base alla tipologia del campo l'utente vedrà una vista specifica dell'attrezzatura.

Il sistema deve interagire con un sistema esterno per gestire il pagamento, che l'utente può effettuare tramite carta di credito o servizi bancari come paypal. Se il pagamento va a buon fine, allora il sistema invia all'utente registrato una mail contenente i dettagli della prenotazione tramite un sistema email esterno. Eventualmente l'utente registrato deve poter modificare o disdire la prenotazione entro e non oltre 3 giorni prima della data di prenotazione e visualizzare le prenotazioni effettuate in un apposito storico delle prenotazioni.

Il gestore del complesso deve poter visualizzare e modificare la lista dei campi, aggiungendo o rimuovendo nuovi campi disponibili, e aggiornare l'attrezzatura da noleggiare. Infine, il sistema deve mostrare un report delle prenotazioni, in modo tale da consentire al gestore del campo di vedere quali campi con la rendita maggiore e gli utenti più "attivi".

2.2 Analisi del testo

Ai fini di una precisa comprensione del dominio del sistema da sviluppare, è stata effettuata l'analisi testuale dei requisiti informali iniziali, attraverso il tool offerto da Visual Paradigm. Questa ha permesso di evidenziare gli attori, i casi d'uso e le classi coinvolte nel sistema in esame e di determinarne il numero di occorrenze nel testo.

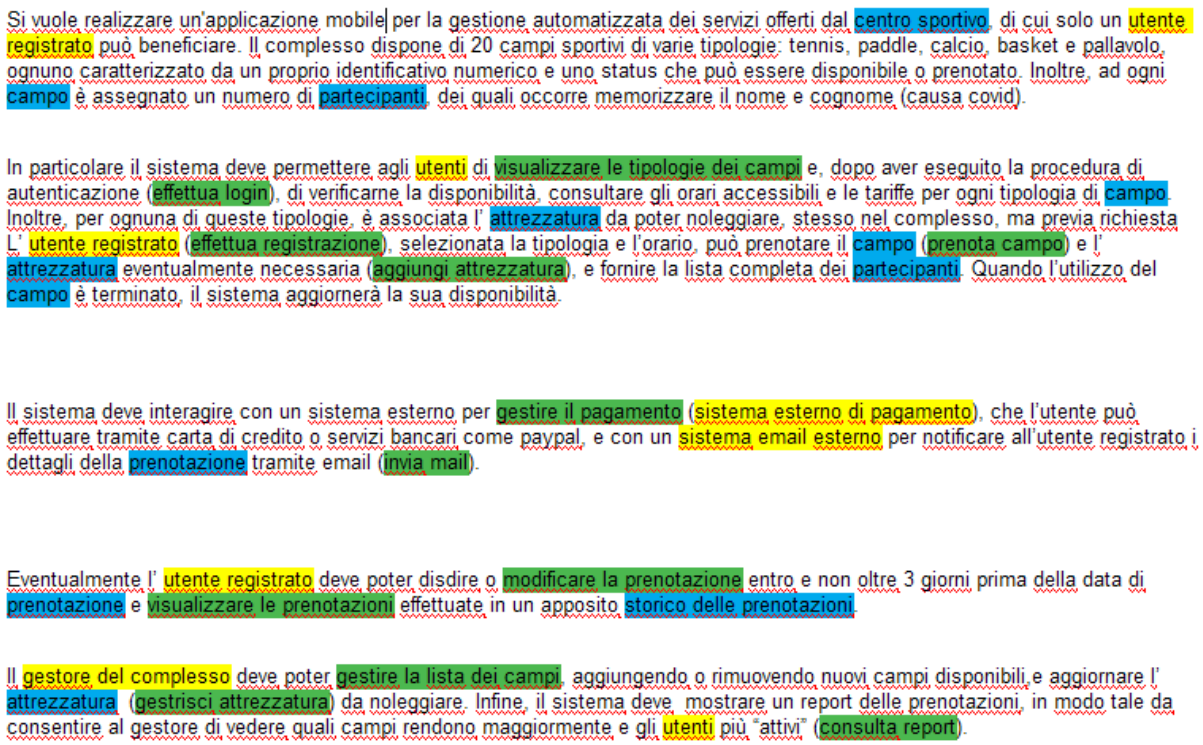


Figura 3: Analisi testuale 1

No.	Candidate Class	Extracted Text	Type	Occurrence	Highlight
1	utente registrato	utente registrato	Actor	3	
2	gestore del complesso	gestore del complesso	Actor	1	
3	sistema esterno di pagamento	sistema esterno di pagamento	Actor	1	
4	sistema email esterno	sistema email esterno	Actor	1	
5	utenti	utenti	Actor	2	
6	aggiungi attrezzatura	aggiungi attrezzatura	Use Case	1	
7	effettua registrazione	effettua registrazione	Use Case	1	
8	prenota campo	prenota campo	Use Case	1	
9	modificare la prenotazione	modificare la prenotazione	Use Case	1	
10	gestire il pagamento	gestire il pagamento	Use Case	1	
11	invia mail	invia mail	Use Case	1	
12	visualizzare le prenotazioni	visualizzare le prenotazioni	Use Case	1	
13	gestrisci attrezzatura	gestrisci attrezzatura	Use Case	1	
14	gestire la lista dei campi	gestire la lista dei campi	Use Case	1	
15	consulta report	consulta report	Use Case	1	
16	effettua login	effettua login	Use Case	1	
17	visualizzare le tipologie dei campi	visualizzare le tipologie dei campi	Use Case	1	
18	centro sportivo	centro sportivo	Class	1	
19	campo	campo	Class	4	
20	partecipanti	partecipanti	Class	2	
21	attrezzatura	attrezzatura	Class	3	
22	prenotazione	prenotazione	Class	2	
23	storico delle prenotazioni	storico delle prenotazioni	Class	1	

Figura 4: Analisi testuale 2

2.3 Tabella attori-obiettivi

Attore	Obiettivi
Utente	· Effettuare la registrazione · Visualizzare i campi
Utente Registrato	· Visualizzare i campi · Prenotare un campo · Aggiungere l'attrezzatura · Visualizzare le prenotazioni · Annullare la prenotazione · Effettuare il login
Amministratore	· Gestire i campi · Gestire l'attrezzatura · Consultare un report
Sistema esterno di pagamento	· Gestire il pagamento
Sistema email esterno	· Inviare la mail

2.4 Requisiti Funzionali in formato breve

Dai risultati dell'analisi testuale, sono stati individuati i seguenti casi d'uso:

- **Effettua registrazione**

Gli **utenti** possono ***registrarsi*** nel sistema fornendo le credenziali: nome, cognome, cf, email, password e numero di telefono.

- **Effettua login**

Gli **utenti registrati** possono ***loggarsi*** nel sistema fornendo le proprie credenziali di email e password.

- **Visualizza campi**

Gli **utenti, registrati e non,** possono ***visualizzare i campi*** presenti in struttura, con relativi orari e tariffe.

- **Prenota campo**

Gli **utenti registrati**, dopo aver visualizzato le disponibilità, possono ***prenotare*** il campo richiesto. Tale funzionalità include l'utilizzo dei casi d'uso "Gestisci pagamento" e "Invia email".

- **Aggiungi attrezzatura**

Gli **utenti registrati**, in fase di prenotazione, possono ***aggiungere l'attrezzatura*** per il campo scelto, qualora sia necessaria.

- **Visualizza prenotazioni**

Gli **utenti registrati** possono ***visualizzare lo storico*** delle proprie prenotazioni.

- **Annulla prenotazione**

Gli **utenti registrati** possono ***disdire la prenotazione***.

- **Gestisci campi**

L'**amministratore** può ***gestire la lista dei campi*** e in particolare può aggiungere, rimuovere, modificare e leggere i campi presenti nella lista.

- **Gestisci attrezzatura**

L'**amministratore** può ***gestire la lista dell'attrezzatura*** e in particolare può aggiungere, rimuovere, modificare e leggere l'attrezzatura presente nella lista.

- **Visualizza report**

L'**amministratore** può ***monitorare*** le richieste degli utenti e l'andamento generale del complesso sportivo.

- **Gestisci pagamento**

Il **sistema esterno di pagamento** può ***verificare*** il buon esito della transazione.

- **Invia mail**

Il **sistema mail esterno** può i***nviare una mail*** di conferma dell'avvenuta prenotazione.

2.5 Requisiti non Funzionali

- **Sicurezza**

Ogni attore del sistema, escluso l'utente che non esegue operazioni critiche, deve autenticarsi per poter accedere alle funzionalità offerte dall'applicazione. In questo modo è possibile garantire la confidenzialità e l'integrità dei dati.

- **Prestazioni**

Un aspetto importante delle prestazioni è il tempo di risposta, ovvero la quantità totale di tempo necessaria per rispondere a una richiesta di servizio. Nel nostro caso per far sì che la nostra applicazione abbia tempi di risposta brevi, abbiamo utilizzato il pattern Proxy-Skeleton in modo tale da alleggerire il Server dalle troppe richieste provenienti da più Client e limitare i malfunzionamenti.

Anche dal lato Client le richieste vengono eseguite su un thread separato, in modo da garantire aumentando il throughput complessivo e la responsiveness, evitando il freeze dell'interfaccia ad ogni richiesta.

- **Usabilità**

L'interfaccia dell'applicazione per i clienti dovrà essere user friendly, semplificando e rendendo intuitive le schermate e la grafica ad esse associate.

- **Evolvibilità**

Per ogni funzionalità offerta dal sistema, si garantisce il disaccoppiamento tra applicazione e servizio. Ciò permette una facile modifica dell'applicazione qualora fosse necessario cambiare i fornitori dei servizi.

- **Persistenza**

I dati relativi alle diverse entità, sono memorizzate in modo persistente su database, in modo da evitare la perdita degli stessi.

In fase di prenotazione, dopo che l'utente ha selezionato il campo che desidera prenotare, il sistema deve bloccarlo per alcuni minuti. Se in questo intervallo di tempo l'utente non procede con il pagamento, il sistema deve annullare la prenotazione e rendere nuovamente disponibile il campo. Ovviamente, il sistema rifiuterà il pagamento se sono passati i minuti e l'utente dovrà rifare la procedura di acquisto.

Il sistema mostra tutti i campi con una data che parte da quella corrente fino ad un massimo di un mese, mantenendo la persistenza per tutti i campi.

- **Robustezza**

Durante la fase di test del sistema sono state gestite diverse eccezioni, per cui il sistema si presenta robusto nel caso in cui gli stessi dovessero presentarsi.

2.6 Vincoli generali

- I dispositivi mobili devono possedere il sistema operativo Android a partire dalla versione 5.0 in poi.
- Tutti gli attori devono necessariamente autenticarsi per usufruire delle funzionalità del sistema, ad eccezione dell'attore Utente il quale può registrarsi e visualizzare la lista delle disponibilità dei campi senza autenticarsi.

2.7 Diagramma dei Casi d’Uso

Tramite il Diagramma dei casi d’uso sono stati definiti i principali servizi del sistema, come si mostra di seguito.

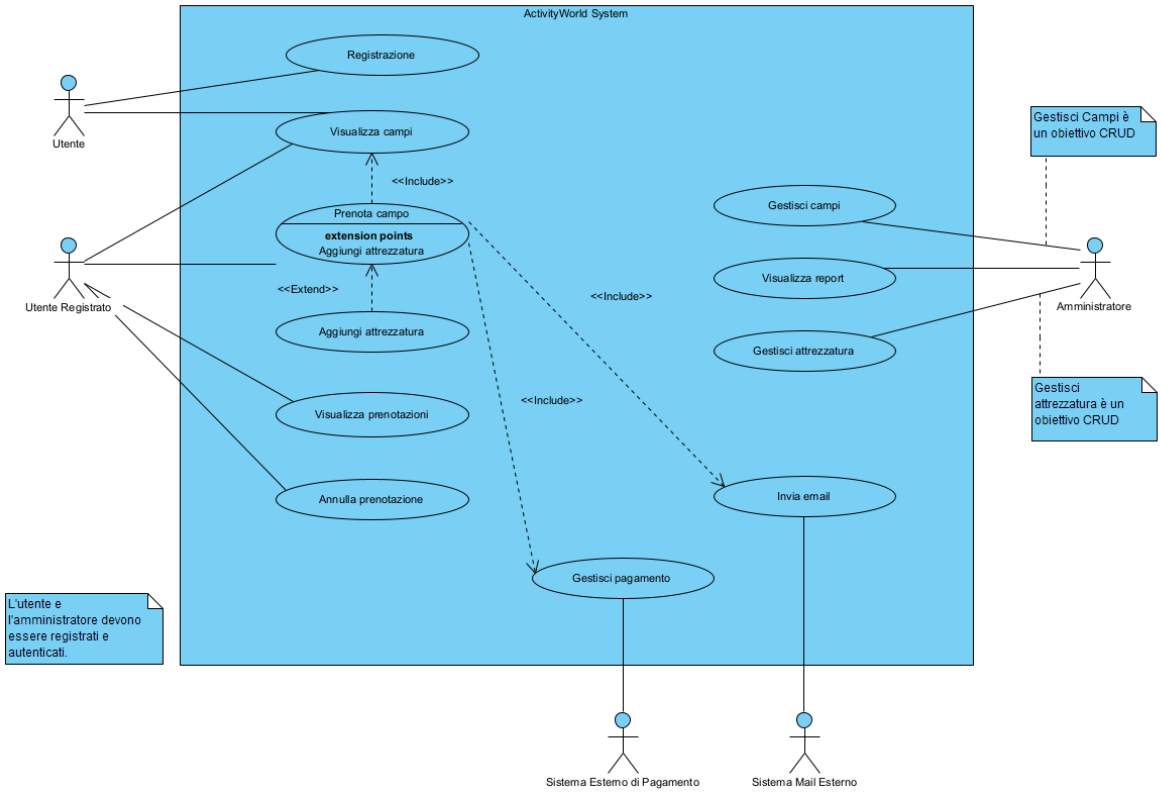


Figura 5: Use Case Diagram

Il caso d’uso “Visualizza campi” è incluso in “Prenota campo”, quindi questo va a precisare che prima di prenotare un campo, il cliente ha la necessità di visualizzare e dunque scegliere una delle tipologie ad esso associate.

Allo stesso modo, i casi d’uso “Gestisci pagamento” e “Invia email ” sono entrambi inclusi in “Prenota campo”. Con questa inclusione, si vuole intendere che, quando il cliente completa la prenotazione di un campo sportivo, dovrà effettuare il pagamento e ricevere la mail relativa alla prenotazione.

Il caso d’uso “Aggiungi attrezzatura” è un'estensione del caso d’uso “Prenota campo”, che denota la possibilità di aggiungere, durante la prenotazione del campo, l’attrezzatura eventualmente necessaria.

Capitolo 3

Analisi e Specifica dei Requisiti

3.1 Identificazione degli attori

Il sistema prevede i seguenti **attori primari**:

- Utente
- Utente registrato
- Amministratore del centro sportivo

Il sistema prevede i seguenti **attori di supporto**:

- Sistema esterno di pagamento
- Sistema responsabile dell'invio delle mail

3.2 Descrizione dettagliata dei casi d'uso

Sono stati forniti gli scenari dei casi d'uso descritti precedentemente, evidenziando sia i flussi di successo che i possibili scenari alternativi.

3.2.1 Caso d'uso Effettua Registrazione

Portata: Applicazione mobile
Livello: Obiettivo Utente
Attore Primario: Utente
Parti interessate e interessi: Utente
Pre-condizioni: nessuna
Post-condizioni: il sistema completa la registrazione con successo

Scenario principale di successo:

1. L’utente richiede la registrazione.
2. Il sistema avvia la registrazione.
3. L’utente inserisce i dati anagrafici.
4. L’utente inserisce un recapito telefonico.
5. L’utente inserisce la sua email.
6. L’utente inserisce la password.
7. L’utente conferma i dati.
8. Il sistema verifica i dati inseriti.
9. Il sistema registra l’utente.

Scenari alternativi:

8.a Se i dati anagrafici non rispettano i vincoli imposti:

1. Il sistema mostra all’utente un messaggio di errore, si ritorna al passo 3.
2. Il sistema richiede all’utente di fornire nuovamente i dati anagrafici non corretti.

8.b Se il recapito telefonico non rispetta i vincoli imposti:

1. Il sistema mostra all’utente un messaggio di errore, si ritorna al passo 4.
2. Il sistema richiede all’utente di fornire nuovamente il recapito telefonico.

8.c Se l’email non rispetta i vincoli imposti:

1. Il sistema mostra all’utente un messaggio di errore, si ritorna al passo 5.
2. Il sistema richiede all’utente di fornire nuovamente l’email.

8.d Se la password non rispetta i vincoli imposti:

1. Il sistema mostra all’utente un messaggio di errore, si ritorna al passo 6.
2. Il sistema richiede all’utente di fornire nuovamente la password.

3.2.1 Caso d'uso Effettua Login

Portata: Applicazione mobile

Livello: Obiettivo Utente

Attore Primario: Utente Registrato

Parti interessate e interessi: Utente Registrato

Pre-condizioni: l'utente deve essere registrato

Post-condizioni: l'utente è autenticato e accede al sistema

Scenario principale di successo:

1. L'utente richiede di effettuare il login.
2. Il sistema avvia l'autenticazione.
3. L'utente inserisce email.
4. L'utente inserisce password.
5. Il sistema verifica i dati inseriti.
6. Il sistema logga l'utente.

Scenari alternativi:

5.d Se l'email e/o la password non rispettano i vincoli imposti:

1. Il sistema mostra all'utente un messaggio di errore, si ritorna al passo 3.

3.2.2 Caso d'uso Visualizza Campi

Portata: Applicazione mobile

Livello: Obiettivo Utente

Attore primario: Utente e Utente Registrato

Parti interessate e interessi: Utente Registrato

Pre-condizioni: nessuna.

Post-condizioni: a seguito di una richiesta da parte dell'utente, il sistema mostrerà i campi disponibili.

Scenario principale:

1. L’utente richiede di visualizzare i campi.
2. Il sistema avvia la ricerca dei campi.
3. L’utente inserisce la tipologia del campo.
4. L’utente inserisce la data.
5. L’utente inserisce l’orario.
6. Il sistema mostra i campi disponibili.

Scenari alternativi:

6.a Non sono presenti campi disponibili all'utente che ha effettuato la richiesta di visualizzazione.

1. Il sistema mostra all’utente un messaggio di errore.
2. Il sistema rimanda l’utente al punto 3.

3.2.3 Caso d’uso Prenotazione Campi

Portata: Applicazione Mobile
Livello: Obiettivo Utente
Attore Primario: Utente Registrato
Parti interessate e interessi: Utente Registrato - Sistema esterno di pagamento - Sistema Mail esterno
Pre-condizioni: L’utente è registrato e autenticato
Post-condizioni: A seguito di una richiesta di prenotazione di uno o più campi dell’utente, il sistema si preoccuperà di effettuare la prenotazione

Scenario principale di successo:

1. **INCLUDE (**visualizza campi)
2. L’utente sceglie il campo desiderato.
3. Il sistema avvia la procedura di prenotazione.
4. L’utente fornisce il nome e cognome dei partecipanti.
5. **INCLUDE** (effettua pagamento).

6. Il sistema registra la prenotazione.
7. **INCLUDE** (invio mail).
8. Il sistema notifica l'avvenuta prenotazione.

Estensioni (o Flussi alternativi):

4.a Se nome e cognome dei partecipanti sono errati:

1. Il sistema mostra all'utente un messaggio di errore, si ritorna al passo 6.
2. Il sistema rimanda l'utente al punto 3.

5.a. Se il pagamento non è andato a buon fine:

1. Il sistema mostra all'utente un messaggio di errore, si ritorna al passo 6.
2. Il sistema rimanda l'utente al punto 1.

3.2.3 Caso d'uso Gestisci Campi

Portata: Applicazione Mobile

Livello: Obiettivo Amministratore

Attore Primario: Amministratore

Parti interessate e interessi: Amministratore

Pre-condizioni: L'amministratore ha effettuato l'autenticazione

Post-condizioni: le operazioni CRUD rispettano i vincoli

Scenario principale di successo:

IF CREATE (*L'amministratore richiede di aggiungere un campo)*

1. L'amministratore inserisce la tipologia di campo.
2. L'amministratore inserisce il prezzo.
3. L'amministratore inserisce la data.
4. L'amministratore inserisce l'orario.
5. Il sistema esegue l'operazione e fornisce un resoconto dell'esito.

IF READ (*L'amministratore richiede di vedere i campi).*

1. L'amministratore inserisce la tipologia di campo
2. il sistema restituisce i campi.

IF UPDATE (*L'amministratore richiede di modificare un campo*)

1. L'amministratore inserisce il campo che desidera modificare.
2. Il sistema restituisce le informazioni associate a tale campo.
3. L'amministratore effettua le modifiche di suo interesse.
4. Il sistema aggiorna il campo.

IF DELETE (*L'amministratore richiede di eliminare un campo*)

1. L'amministratore seleziona il campo che intende cancellare.
2. Il sistema cancella il campo.

Estensioni (o Flussi alternativi):

Gestione degli eventuali errori per le varie operazioni CRUD

3.3 Diagramma di Dominio

In questo diagramma sono mostrate le entità in gioco del sistema software iniziale e le associazioni tra di esse. In questo modo vengono evidenziati gli aspetti essenziali del dominio di interesse, tralasciando momentaneamente le funzionalità da implementare in un secondo momento.

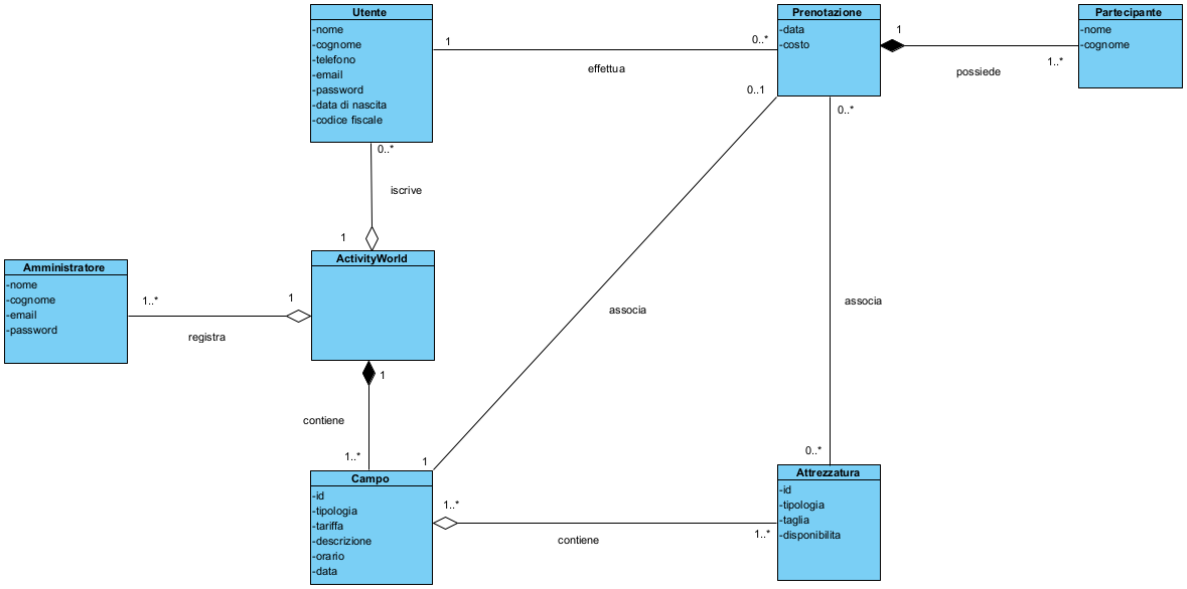


Figura 6: System Domain Model 1

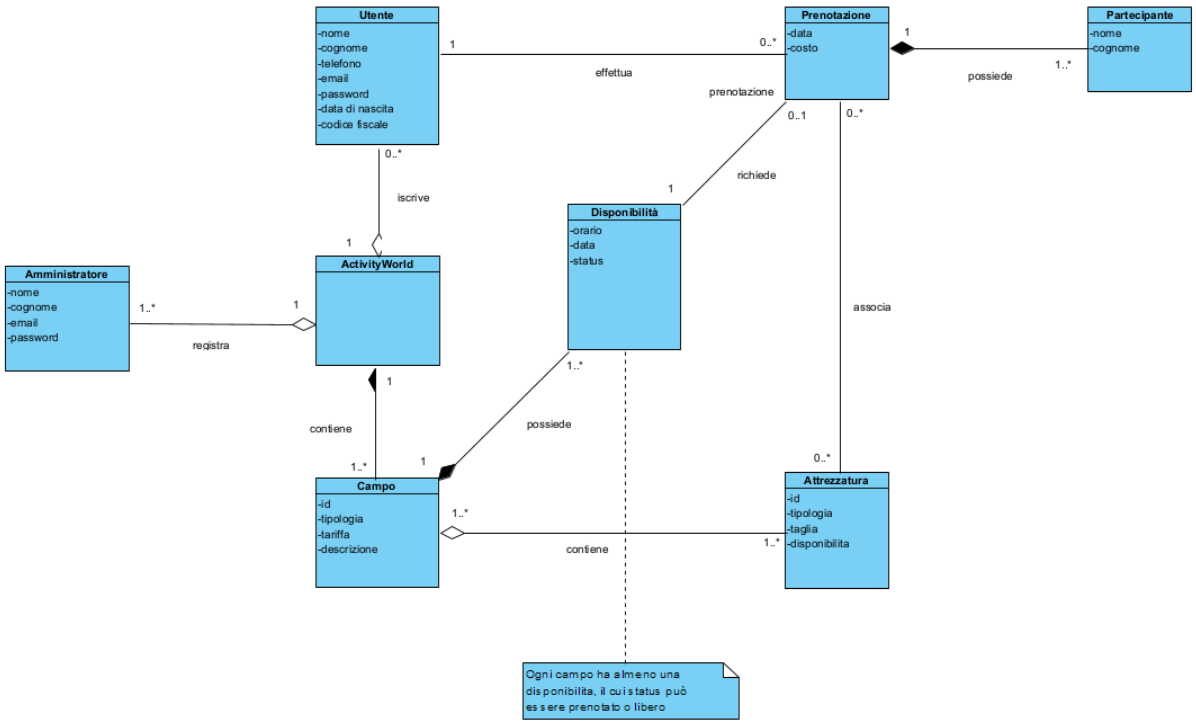


Figura 7: System Domain Model 2

Durante la seconda iterazione, che si focalizzata sul caso d'uso "Visualizza Campi", il team si è reso conto fosse più corretto distinguere il concetto di "campo" da quello di "disponibilità" ad esso associata. Ogni campo sarà caratterizzato dagli attributi di: tipologia, tariffa e descrizione e gli sarà associata una o più disponibilità, contente: orario, data e status. Quindi, come risultato, il caso d'uso in esame, è stato suddiviso in due casi d'uso distinti "Visualizza Campi" e "Visualizza Disponibilità".

3.4 Diagrammi di Sequenza di Analisi

I diagrammi di sequenza di alto livello mostrano la sequenza degli eventi generati dall'interazione dell'utente con il sistema.

3.4.1 Sequence Diagram - Effettua Registrazione

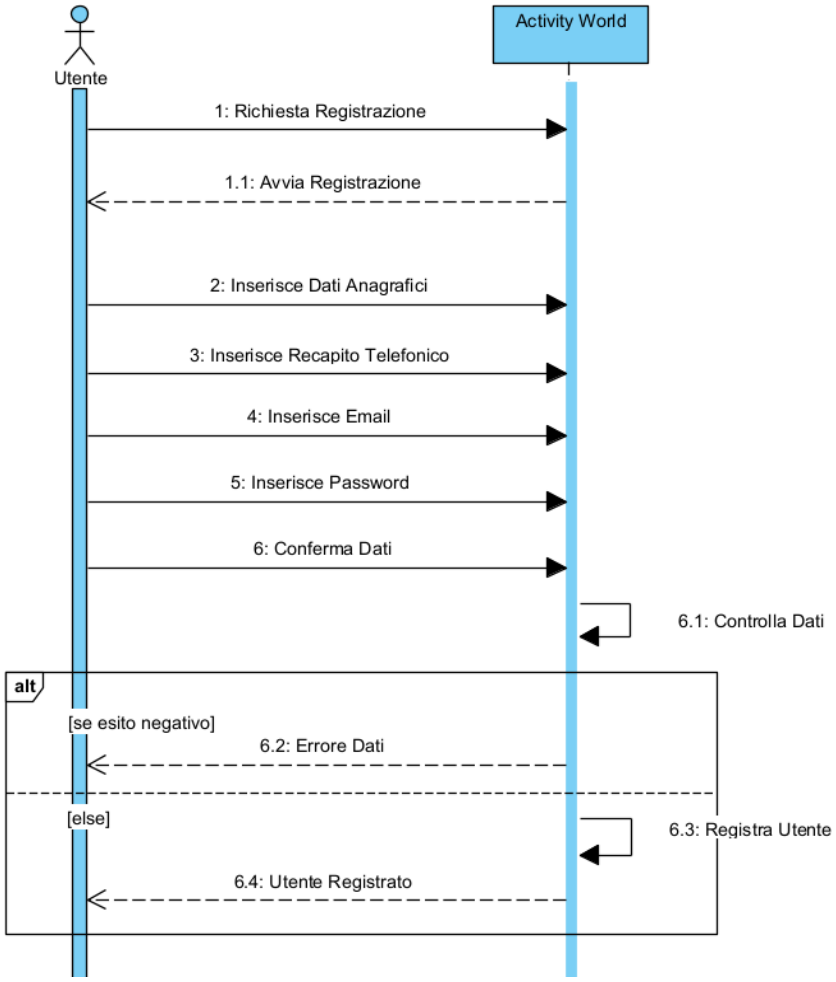


Figura 8: Sequence Diagram - Effettua Registrazione

3.4.2 Sequence Diagram – Effettua Login

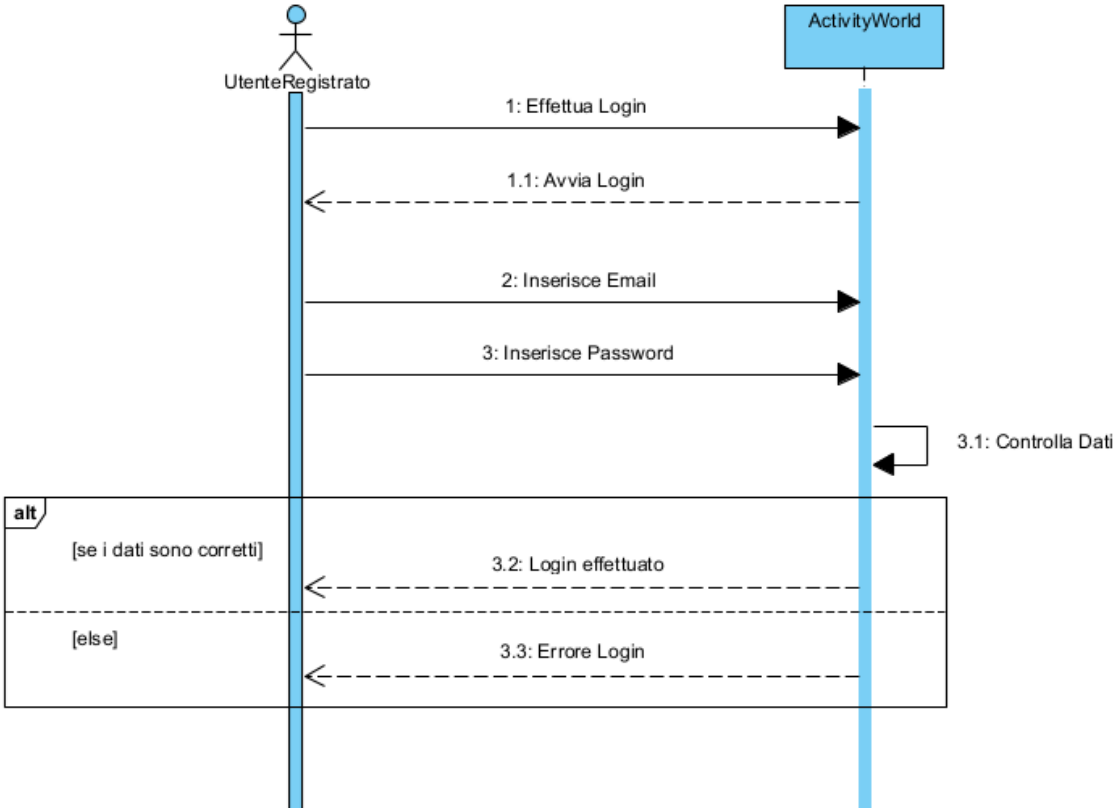


Figura 9: Sequence Diagram - Effettua Login

3.4.3 Sequence Diagram – Visualizza Campi

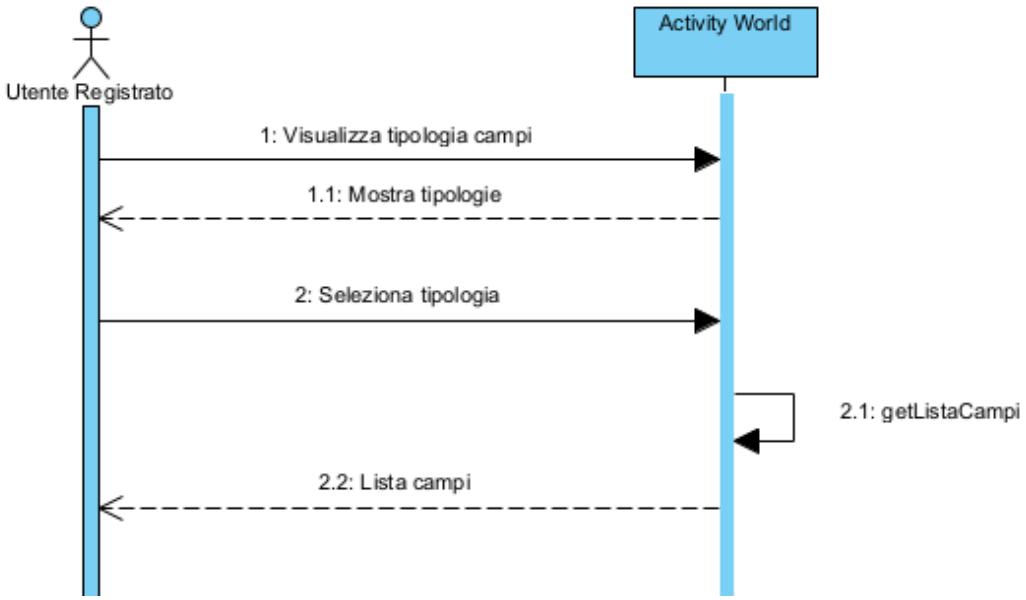


Figura 10: Sequence Diagram - Visualizza Campi

3.4.4 Sequence Diagram – Visualizza Disponibilità

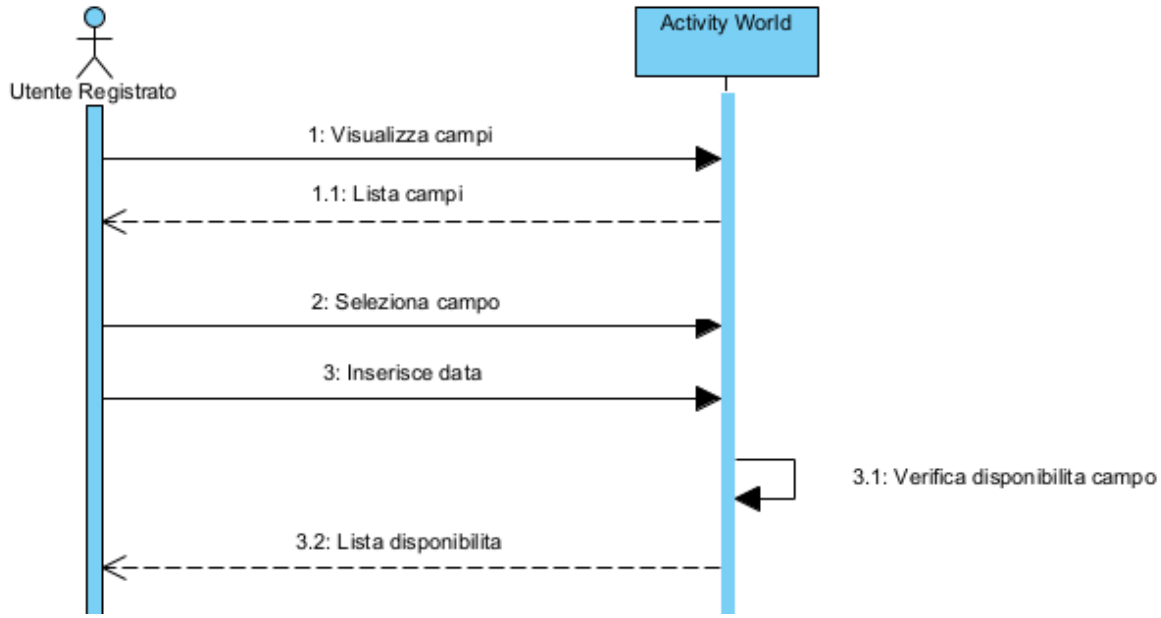


Figura 11: Sequence Diagram - Visualizza Disponibilità

3.4.5 Sequence Diagram - Prenota Campo

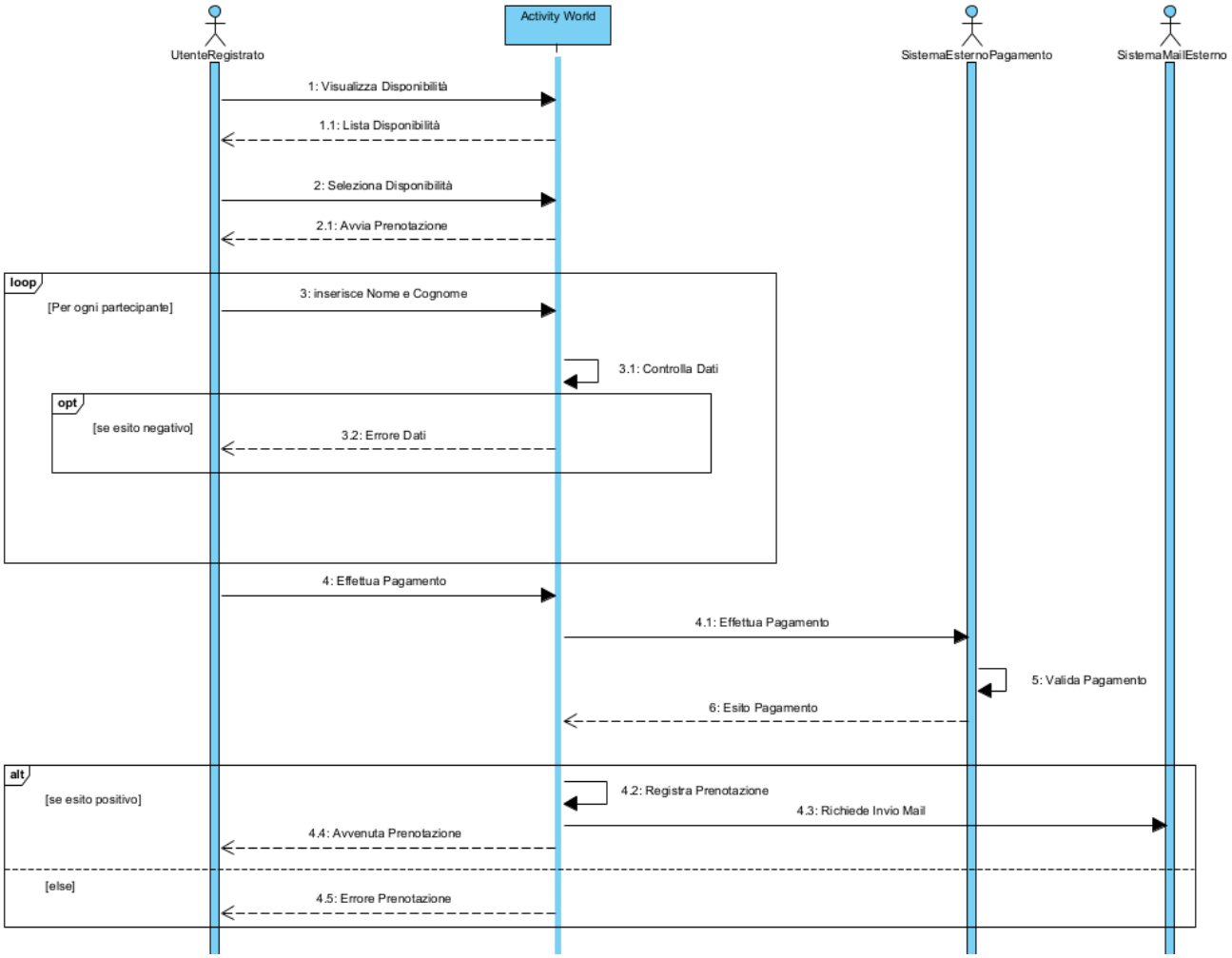


Figura 12: Sequence Diagram - Prenota Campo

3.4.5 Sequence Diagram - Gestisci Campo

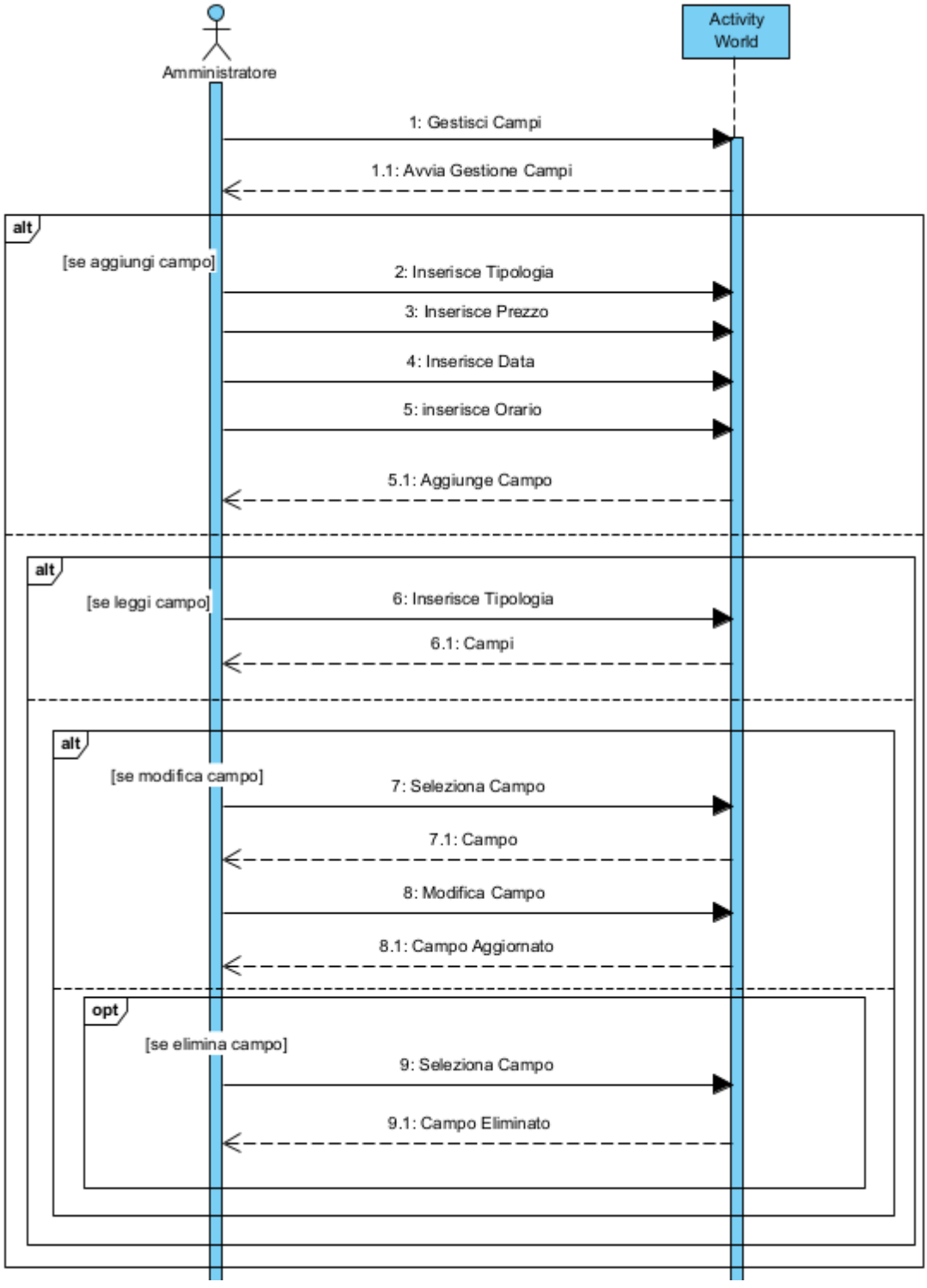


Figura 13: Sequence Diagram - Gestisci Campo

3.5 Activity Diagram

I diagrammi di attività mostrano la rappresentazione grafica degli scenari principali dei casi d'uso descritti in precedenza.

3.5.1 Activity Diagram - Effettua Registrazione

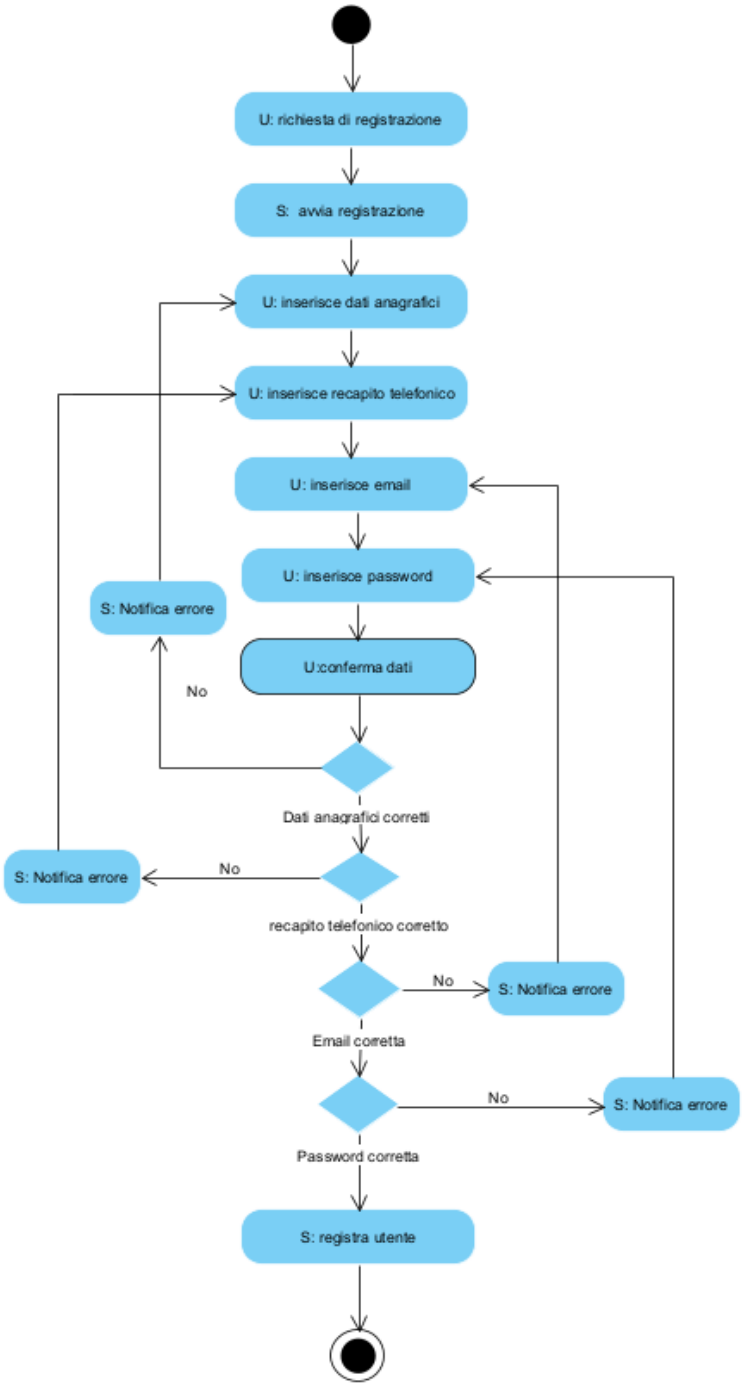


Figura 14: Activity Diagram - Effettua Registrazione

3.5.2 Activity Diagram - Effettua Login

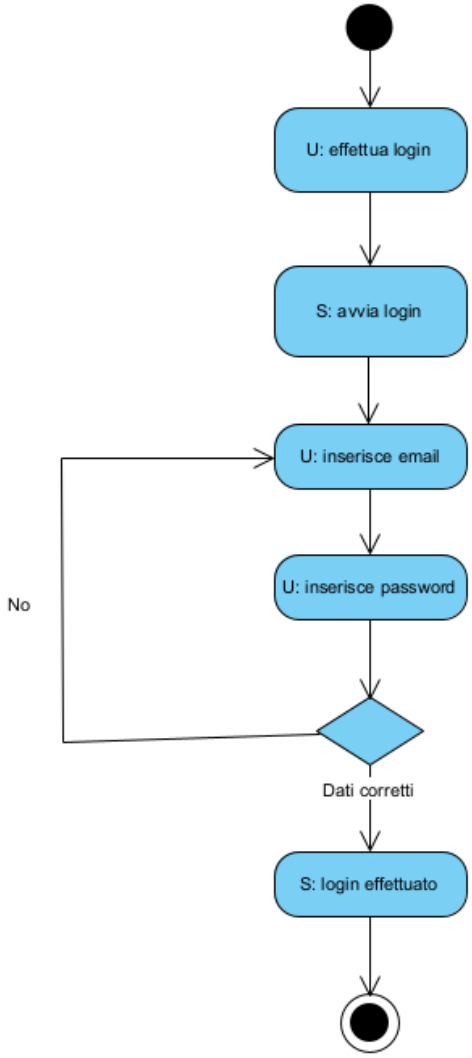


Figura 15: Activity Diagram - Effettua Login

3.5.3 Activity Diagram - Visualizza Campi

Figura 16: Activity Diagram - Visualizza Campi

3.5.4 Activity Diagram - Visualizza Disponibilità

Figura 17: Activity Diagram - Visualizza Disponibilità

3.5.5 Activity Diagram - Prenota Campi

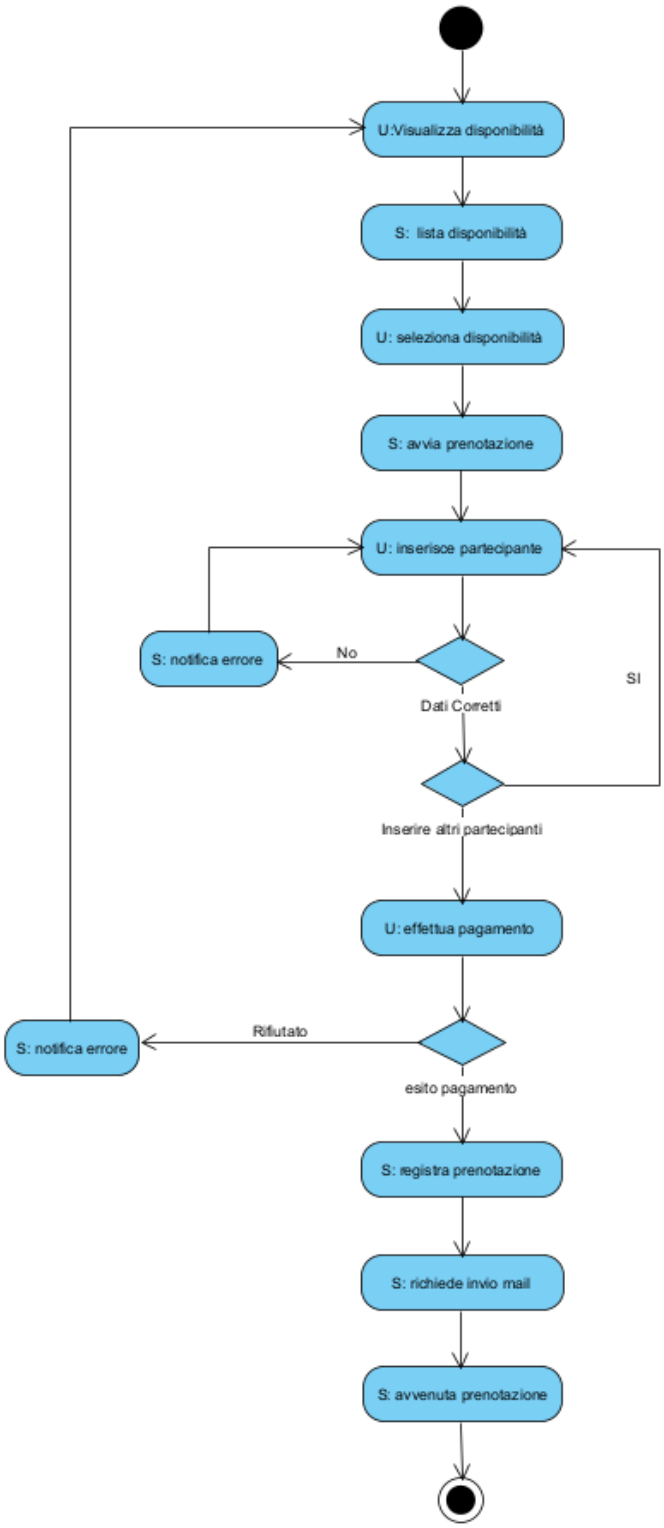


Figura 18: Activity Diagram - Prenota Campi

3.5.6 Activity Diagram - Gestisci Campi

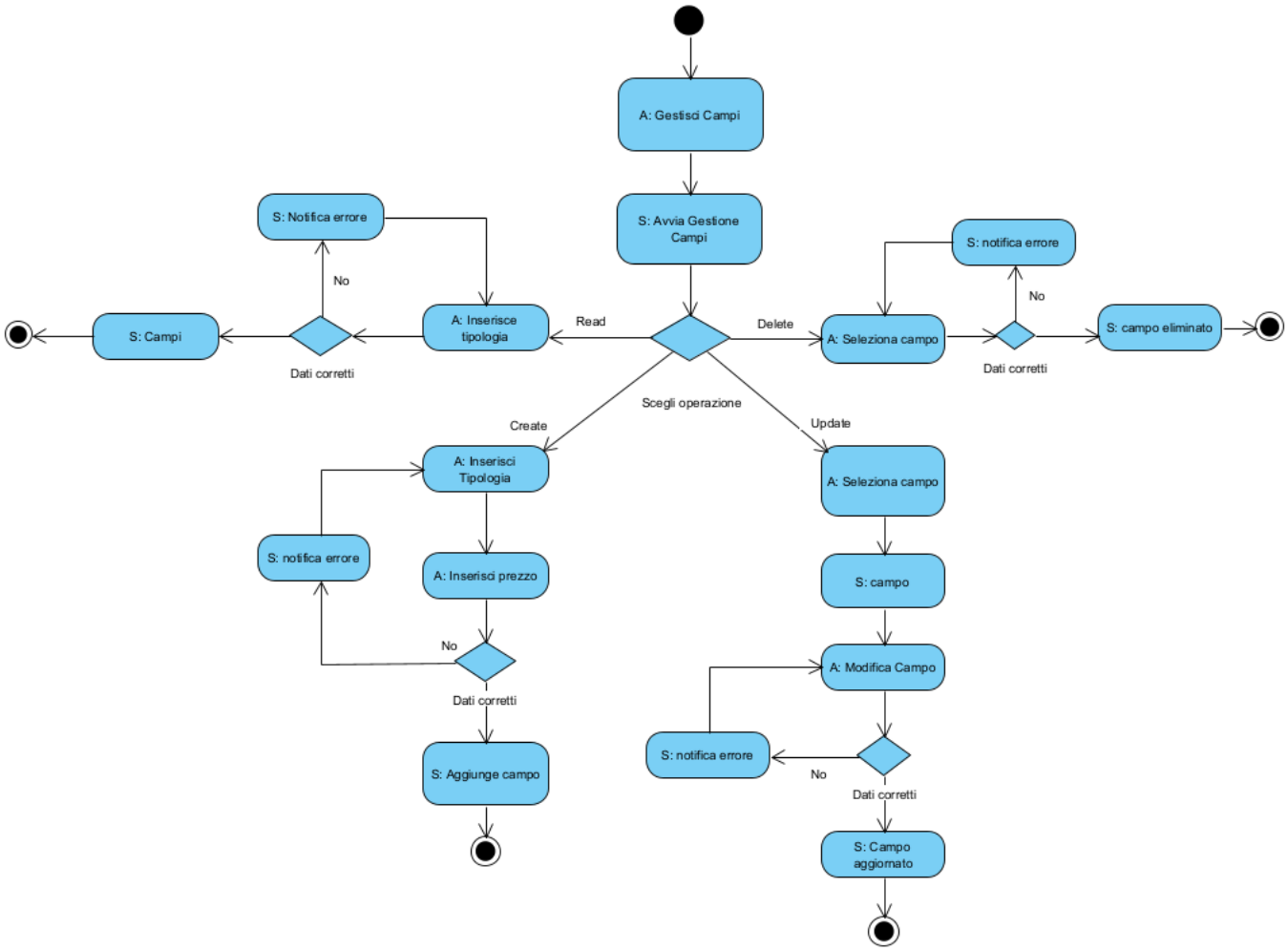


Figura 19: Activity Diagram - Gestisci Campi

3.6 Diagramma delle Classi

In seguito allo sviluppo dei sequence di analisi sono state evidenziate le operazioni da assegnare ad ogni classe del sistema.

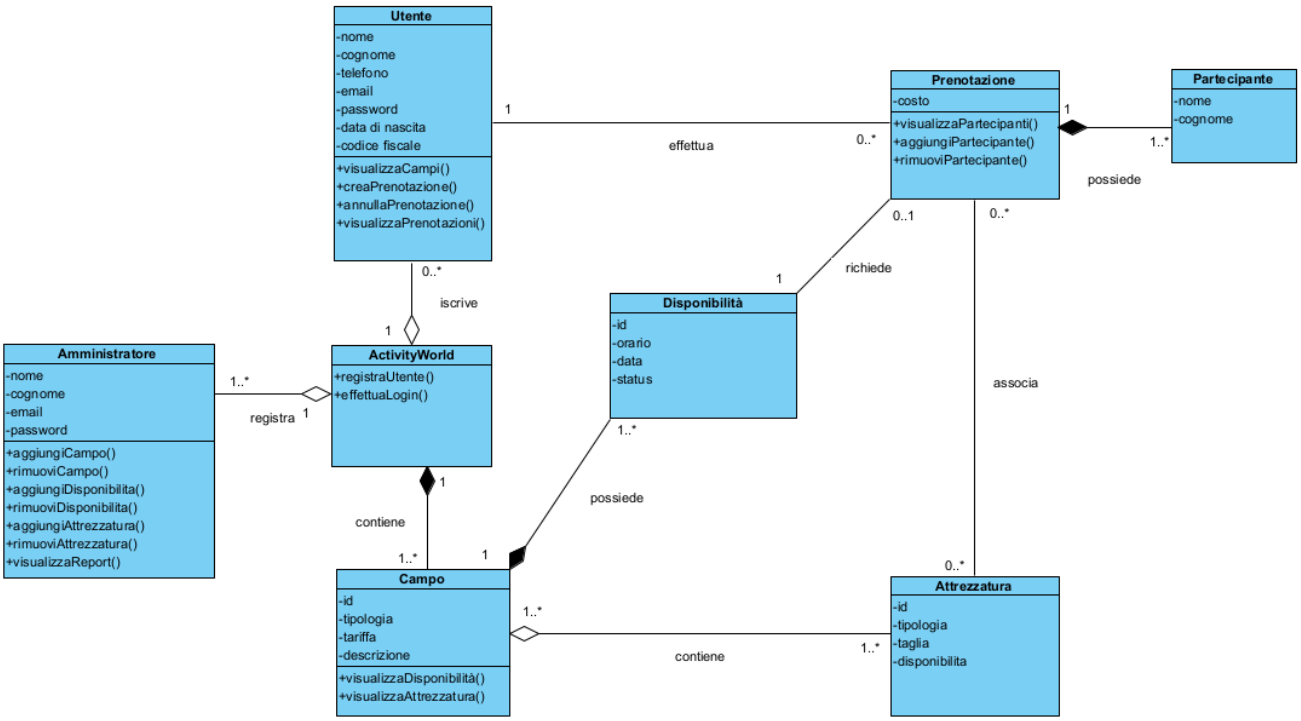


Figura 20: Class Diagram

Le scelte effettuate sono state dettate dalle seguenti considerazioni:

1. Assegnare le operazioni alla classe che è in grado di "soddisfare" la responsabilità

2. Assegnare le operazioni alla classe che invoca il metodo.

In ogni caso, si è evitato di assegnare i metodi a classi molto “distanti”, che quindi necessitano di più passaggi per recuperare i dati utili all'esecuzione dell'operazione, favorendo il "Low Coupling".

3.7 Diagramma di Contesto

Il seguente diagramma mostra il contesto del sistema, ovvero le interazioni tra il sistema di ActivityWorld e gli attori esterni.

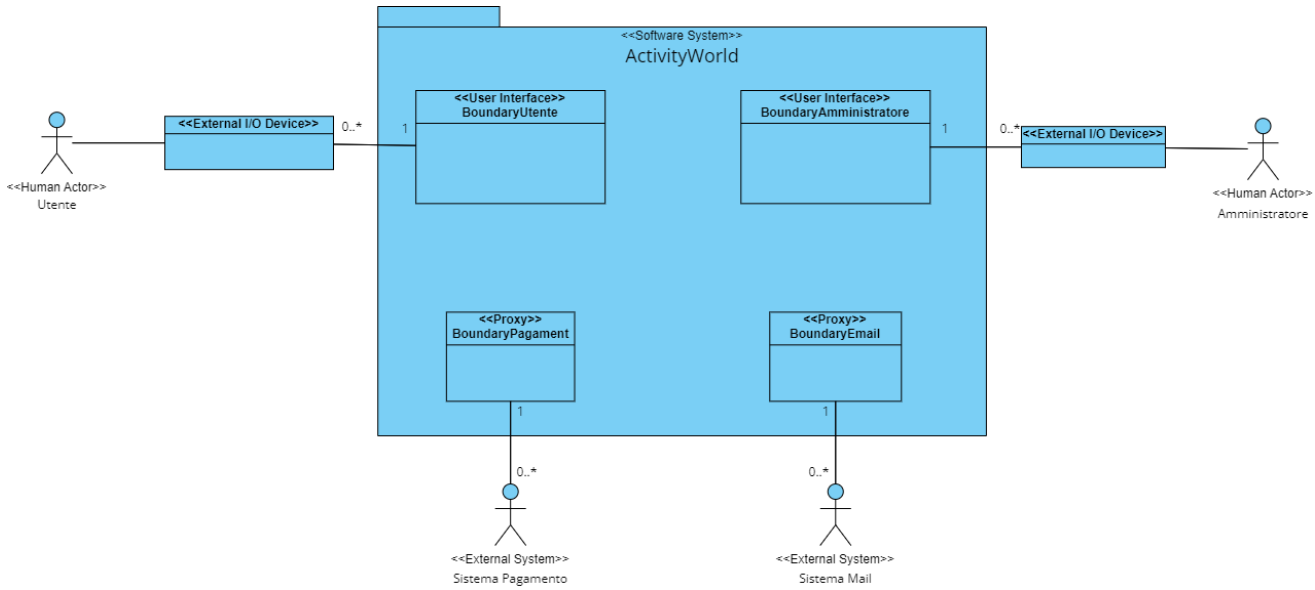


Figura 21: Context Diagram

Capitolo 4

Architettura Software e scelte di progetto

4.1 Stile Architetturale

Quello proposto è un sistema distribuito basato su un'Architettura Client-Server. Questa ha permesso di mantenere i ruoli separati tramite diverse interfacce e client indipendenti, eseguite su dispositivi diversi, che inviano richieste al server tramite un connettore basato su TCP/IP. In questo modo è stato possibile ottenere una forte scalabilità, necessaria anche nell'ottica futura di aggiungere altre tipologie di utenti al sistema, e l'indipendenza delle parti.

4.1.1 Diagramma dei Componenti

Il component diagram ha come scopo quello di rappresentare la struttura interna del sistema software modellato in termini dei suoi componenti principali e delle relazioni fra di essi.

Diagrammi dei componenti che definiscono l'architettura generale del sistema software da sviluppare

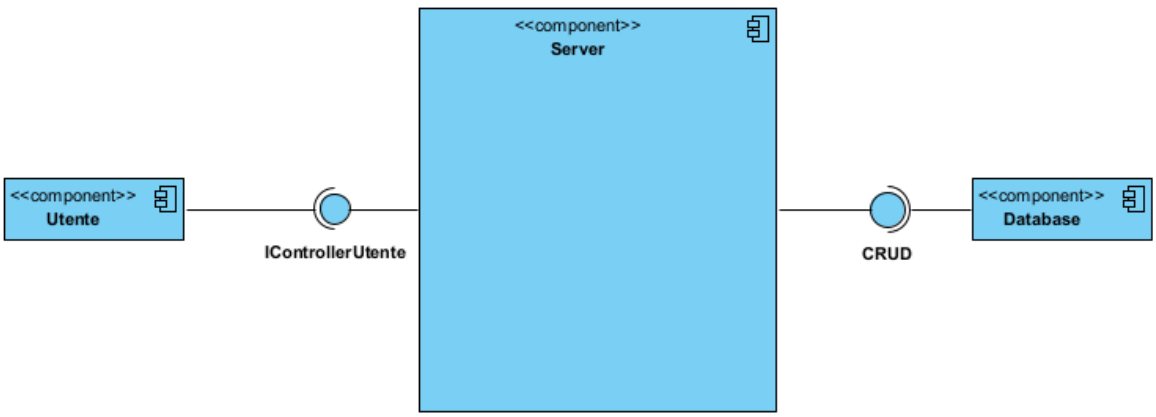


Figura 22: Component Diagram

4.1.2 Diagramma dei Package

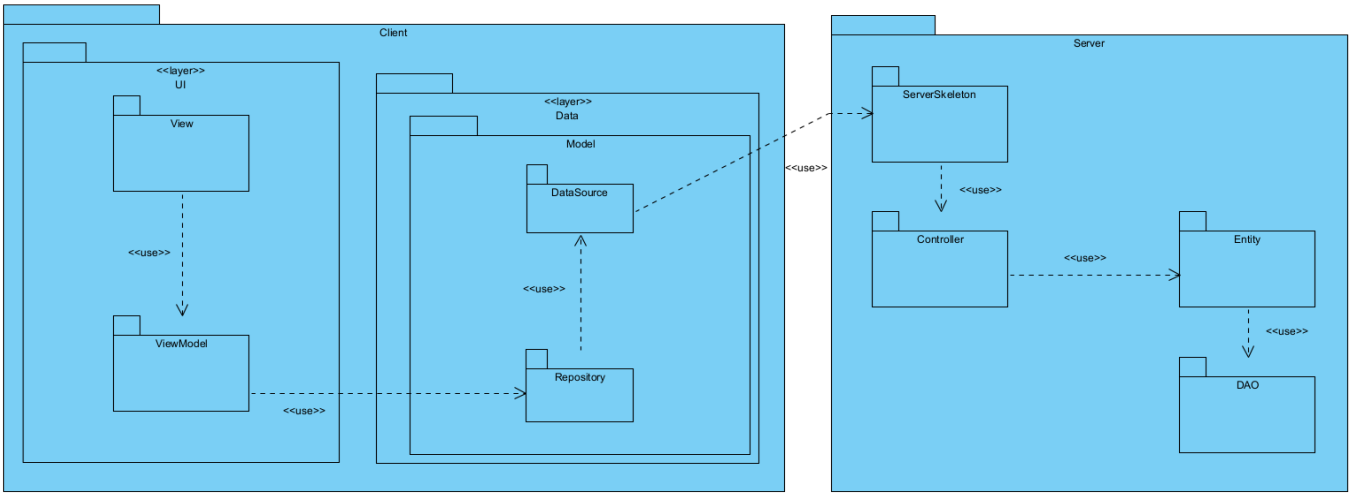


Figura 23: Package Diagram

4.2 Pattern Architetturale

Three-tiered pattern

Avendo un'architettura client server, ci si è trovati dinanzi all'esigenza di decidere come distribuire le funzionalità applicative tra client e server.

Il three-tier pattern si presta bene a risolvere le nostre esigenze andando a dividere l'architettura in three-tier:

- **Front tier**: l'interfaccia utente, tramite la quale gli utenti accedono ai servizi del sistema.
- **Middle tier**: dove risiede tutta la logica di business per realizzare le funzionalità del sistema.
- **Back tier**: corrisponde ad un RDBMS server per memorizzare le informazioni utili alla realizzazione dei servizi.

4.3 Client

Ogni client effettua delle operazioni di richiesta verso il server, richiede quindi, un terminale con capacità elaborative (uno smartphone). Il client gestisce la porzione di interfaccia utente dell'applicazione, verificando i dati inseriti e richiedendo al server ciò di cui ha bisogno; è dunque la parte dell'applicazione che l'utente vede e con la quale interagisce.

Lato client, si è optato per l'ambiente di sviluppo Android rispetto a quello iOS per due ragioni principali:

- **Target:** una buona fetta di devices mobili gira su un sistema operativo Android permettendoci di raggiungere quindi un numero maggiore di utenti.
- **Strumenti a disposizione:** sviluppare applicazione iOS richiede infatti dispositivi Apple di cui non tutto il team era provvisto.

La scelta del pattern architetturale da utilizzare per l'applicazione mobile è stata poi frutto di una lunga consultazione della documentazione ufficiale di Google e vari articoli scritti da sviluppatori di rilievo. È stato infatti cruciale comprendere come il sistema operativo gestisca le risorse del device e le sue applicazioni.

Per poterne ottimizzare tale utilizzo, favorire la scalabilità, la robustezza e il testing, l'architettura deve rispettare i seguenti principi cardine.

- **Separation of concerns**

Ogni sezione della nostra applicazione dovrebbe occuparsi di un problema specifico e distinto dagli altri: deve esistere una separazione a livello software in funzione delle operazioni eseguite. Infatti ad esempio, le classi relative all'interfaccia (UI) dovrebbero contenere unicamente la logica che gestisce le interazioni con la stessa e il sistema operativo.

Dal punto di vista dell'architettura, le applicazioni possono essere create logicamente per seguire questo principio attraverso una struttura a livelli.

- **Drive UI from data models**

L'interfaccia dovrebbe essere 'guidata' a partire da modelli di dati persistenti. Tali modelli rappresentano i dati dell'applicazione e sono indipendenti dagli elementi dell'interfaccia e dal ciclo di vita dei vari componenti dell'applicazione.

4.3.1 Model-View-ViewModel (MVVM)

Tenendo conto allora dei principi sopra descritti e delle linee guida fornite da Google si è scelti di utilizzare il pattern architetturale **Model-View-ViewModel (MVVM)** combinato agli Android's Architecture Components.

A differenza del MVC o MVP, l'MVVP disaccoppia ulteriormente la View dal ViewModel attraverso un componente chiamato Binder gestito da una libreria nativa di Android chiamata DataBinding.

Il binder lega i componenti dell'interfaccia ai data sources. Questo implica che le modifiche ai dati apportate dall'utente attraverso la view verranno automaticamente riportate nel view model.
Allo stesso modo, eventuali modifiche apportate ai dati contenuti nel view model verranno automaticamente rappresentate nella view.

- Data Binding (dal codice alle views)
- View binding (dalle view al codice)

Pertanto, il ViewModel non sarà a conoscenza delle Views che ne osservano i cambiamenti.

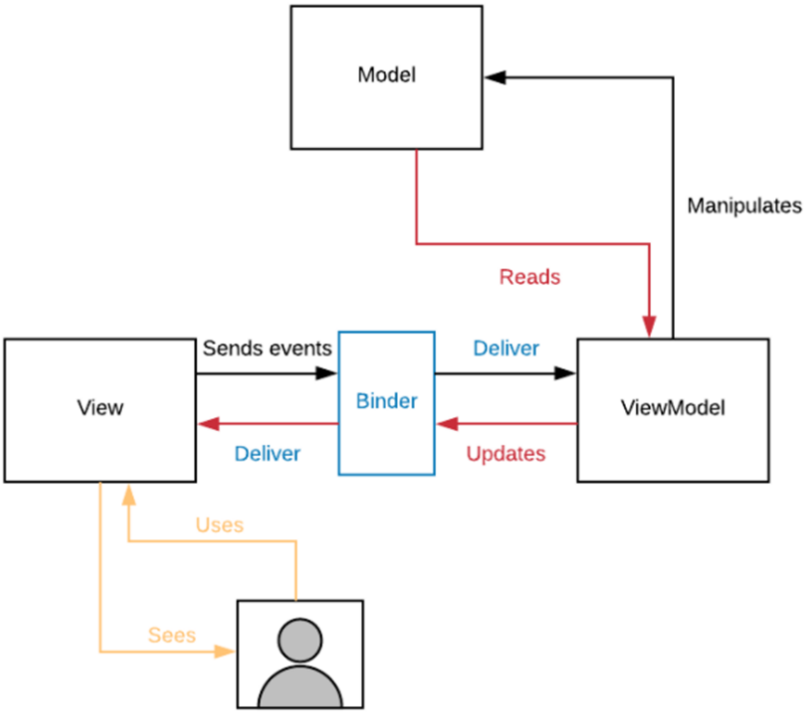


Figura 24: MVVM

- **View**

Nell'MVVM si cerca di mantenere la View il più leggera possibile non dovendo contenere quasi alcuna logica ma semplicemente la gestione dell'UI.

In Android, solitamente la View corrisponde ad un ***Activity*** o un ***Fragment*** il cui ruolo è quello di mostrare ciò che viene ricevuto dal ViewModel o viceversa inviargli degli input.

- **Model**

Il modello è il componente più leggero in assoluto in quanto non è a conoscenza di nessun altro componente dell'applicazione. Il suo ruolo è quello di salvare/memorizzare lo stato del sistema e lasciare che il ViewModel lo legga. Il modello viene spesso creato utilizzando il **Repository** pattern.

- **ViewModel**

Infine, il ViewModel è il cuore del pattern MVVM. Include tutta la business logic. Si occupa di adattare le informazioni prelevate dal Model affinché la View possa mostrarle correttamente.

Viceversa, si occupa anche di trasformare gli input events ricevuti dalla View in dati che verranno salvati dal Model.

Per lo sviluppo Android, il pattern MVVM viene utilizzato in un'architettura avente *almeno due strati*:

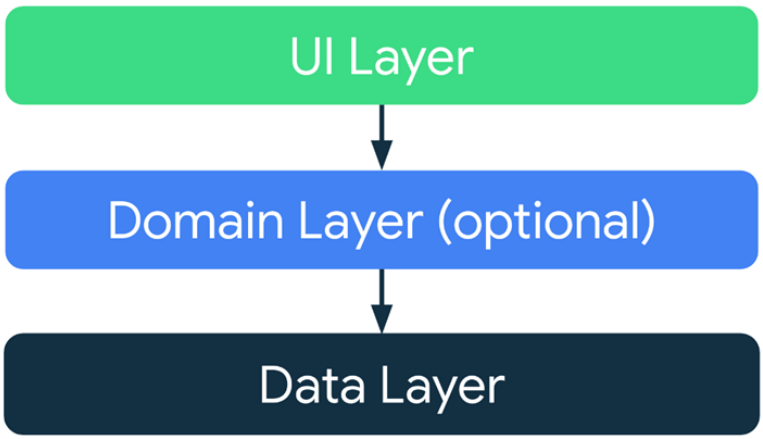


Figura 25: Layer Architettura

4.3.2 Android's Architecture Components

- **UI Layer**

Il ruolo del livello UI (o *presentazione*) è quello di mostrare i dati dell'applicazione sullo schermo. Ogni volta che i dati mutano/cambiano, che sia a causa dell'interazione dell'utente (ad esempio attraverso la pressione di un pulsante) o input esterni (come la risposta ad una chiamata sulla rete), l'interfaccia deve aggiornarsi per poter riflettere tali cambiamenti.

Il layer UI è composto da due cose:

· Gli elementi dell'interfaccia che presentano i dati sullo schermo (***Views***)

· State Holders (come il ***ViewModel***) che conservano i dati, li espongono all'interfaccia e ne gestiscono la logica.

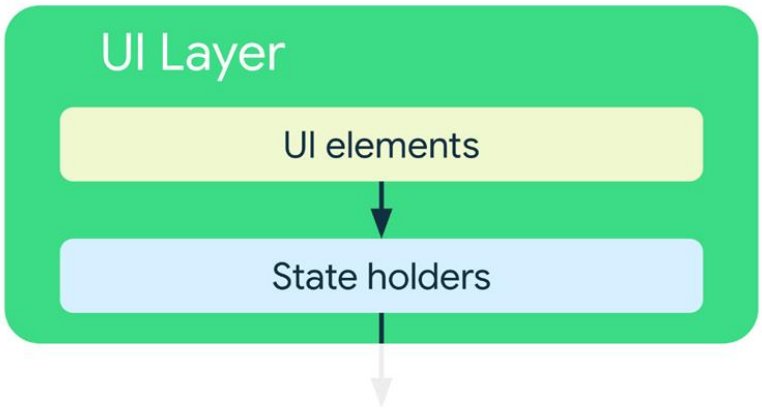


Figura 26: UI Layer

- **Data Layer**

Il data layer contiene la cosiddetta ***business logic***, cioè l'insieme di regole che determinano come l'applicazione debba creare, memorizzare o modificare i dati.

Come anticipato, spesso si utilizza, per questo layer il pattern **Repository**. Ogni repository può contenere 0 o più ***Data sources***. Il pattern prevede la creazione di un repository per ogni tipo di dato differente che si gestisce nell'applicazione.

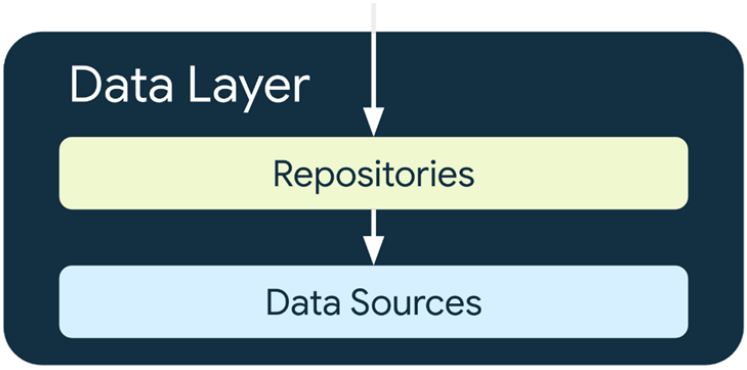


Figura 27: Data Layer

Le classi Repository sono responsabili dei seguenti tasks:

· Esporre i dati al resto dell'applicazione

· Centralizzare i cambiamenti dei dati

· Astrarre le sorgenti dei dati dal resto dell'applicazione

4.4 Server

Per migliorare ulteriormente il sistema, anche per il lato server è stato deciso di adottare una struttura a livelli. In questo modo, ogni livello è client del livello sottostante e ciò consente di avere una maggiore flessibilità, interconnettività e modificabilità tra i vari livelli e di rispettare il principio della separation, trattato in precedenza.

- ### Layer Controller

La comunicazione avviene con il livello più alto del server, ovvero con il layer control, che si occupa di gestire la logica di business del sistema andando a smistare le apposite richieste client.

Abbiamo utilizzato il pattern strutturale **Facade**, attraverso la classe Archivio si fornisce un'interfaccia più semplice all'esterno, unificando le interfacce.Tale pattern ci permette di nascondere la complessità del sistema, riducendo la comunicazione e dipendenza tra classi.

- ### Layer Entity

Tutta la logica implementativa del sistema è stata delegata al layer immediatamente sottostante al controller. Questo sarà il cuore del nostro server dove verranno realizzate tutte le procedure per poter offrire i servizi ai client.

- ### Layer DAO

Per gestire la logica di accesso al database è stato usato il pattern Data Access Object (DAO). Questo ha permesso nascondere agli strati superiori la complessità delle operazioni CRUD verso il database, di evitare l'interazione tra le componenti della logica di business con il database, aumentando la manutenibilità, ed evitare di stravolgere il codice dell'applicazione in caso di modifiche al Db.

In questo layer vengono realizzate tutte le query utili allo scambio di dati persistenti da conservare per i servizi offerti.

4.5 Comunicazione Remota con Socket

Gestiamo la comunicazione tra i processi client e server, con apposite Socket con protocollo di rete tcp. Il client implementa l'interfaccia del controller del server per accedere ai servizi del sistema, quindi attraverso appositi flussi o stream, che client e server apriranno, verranno trasmesse tutte le informazioni relative al tipo di servizio a cui si desidera accedere e tutti i parametri essenziali che il client deve fornire per accedere a questi servizi e infine tutti i parametri di ritorno che il server fornirà dopo la elaborazione di queste informazioni.

4.6 Pattern Proxy-Skeleton e Thread

Per separare ulteriormente i client dal server e aumentare il parallelismo, abbiamo adoperato un pattern proxy lato client, ovvero ogni client utilizzerà per inviare le richieste al server un apposito proxy, il quale implementa l'interfaccia corrispondente, in questo modo abbiamo maggior separazione tra i due blocchi. Sul server, visto che il numero di client che effettueranno richieste di servizi saranno numerosi, per gestire questa situazione abbiamo sviluppato uno skeleton che creerà un thread per ogni socketUtente che richiederà servizi.

NB. Per una migliore visione dei successivi diagrammi architetturali e dei sequence di dettaglio, si consiglia di prendere visione dal file .vpp in allegato.

4.7 Diagrammi Architetturali

4.7.1 Vista Architetturale - Server

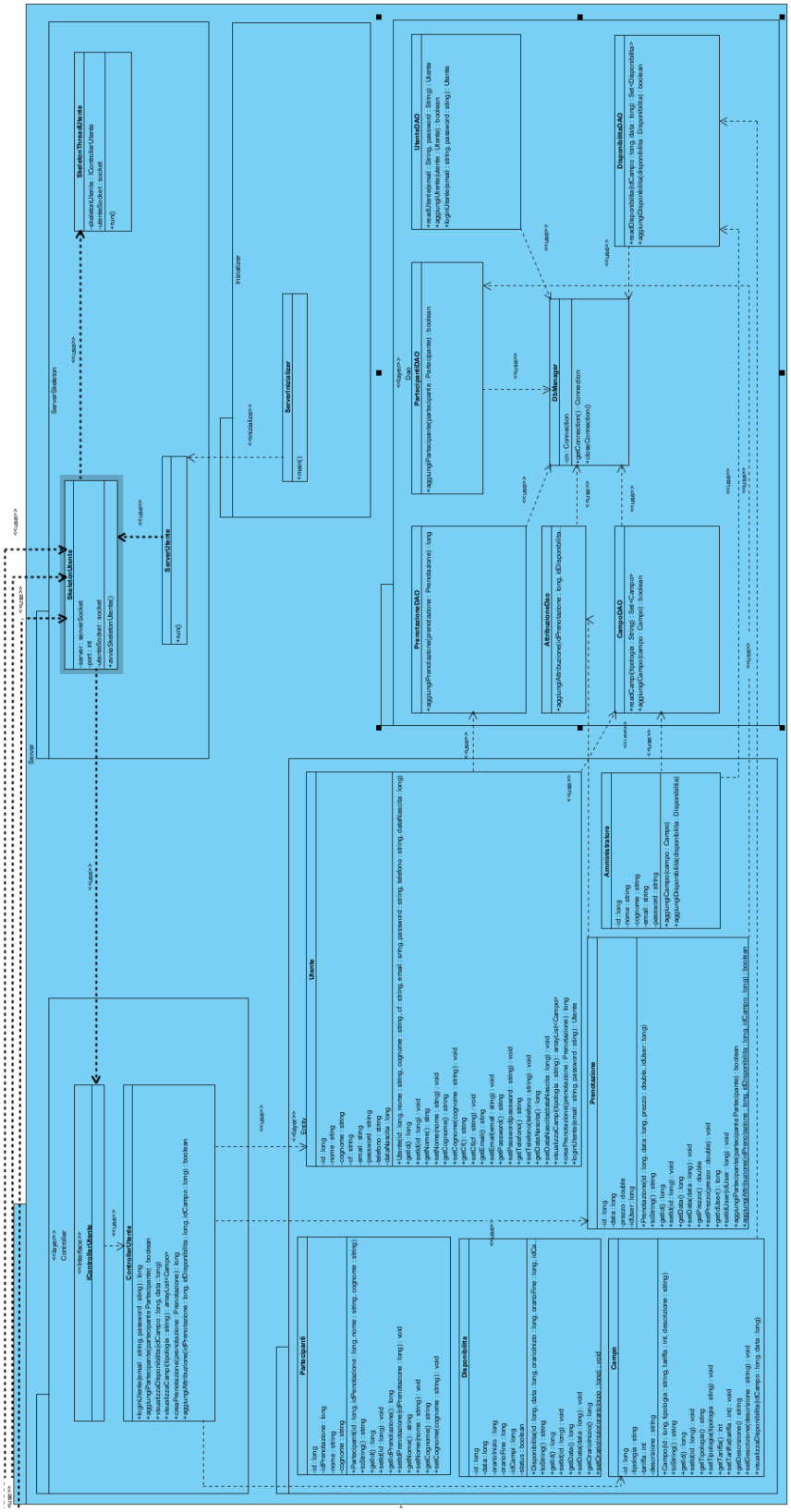


Figura 28: Architettura Client-Server 1

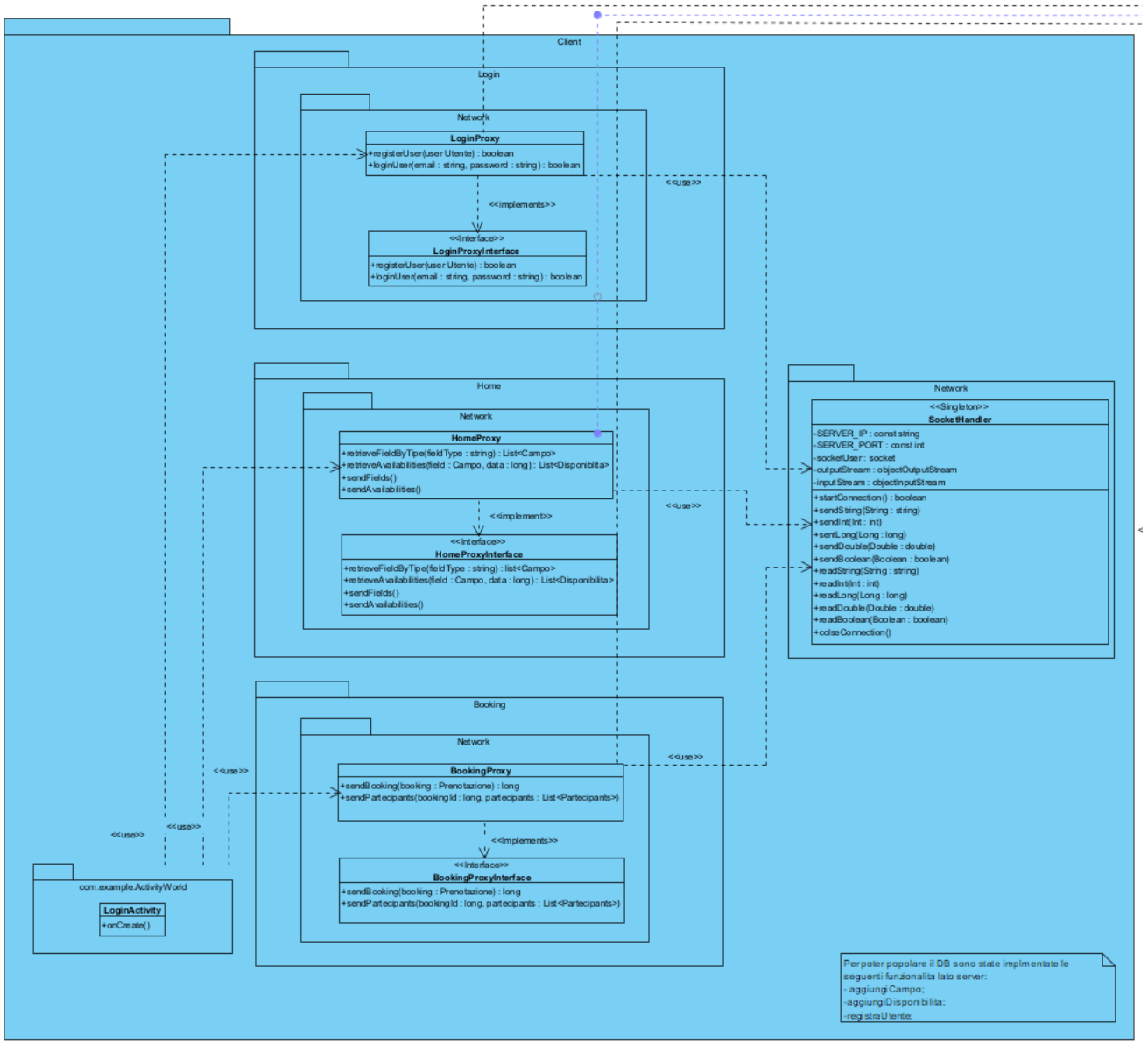


Figura 29: Architettura Client-Server 2

4.7.2 Vista Architetturale - Client

Figura 30: Architettura Client

4.8 Diagramma di Sequenza Raffinati

4.8.1 Sequence Diagram Raffinato – Effettua Login

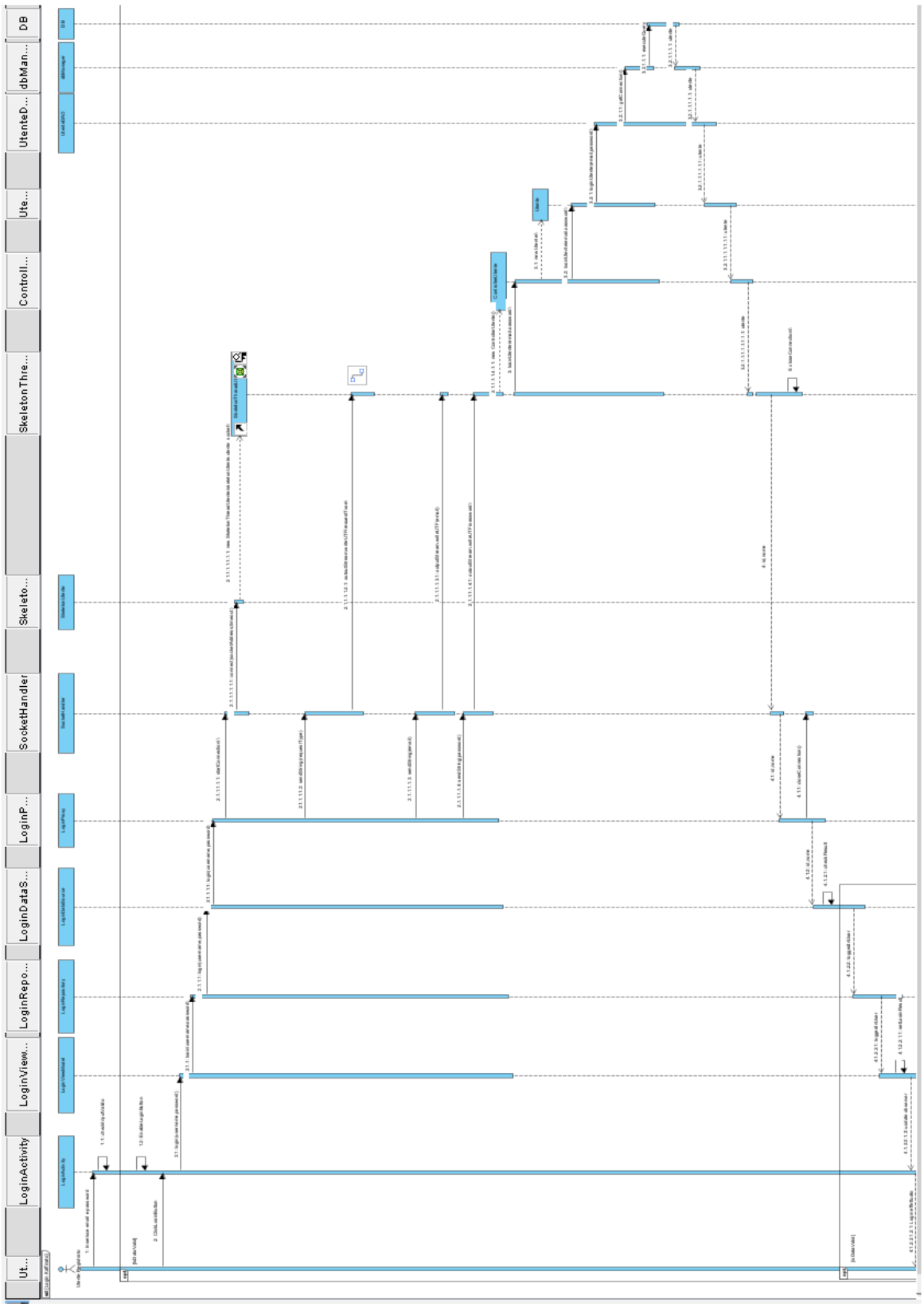


Figura 31: Sequence Diagram Raffinato - Effettua Login

4.8.2 Sequence Diagram Raffinato - Visualizza Campi

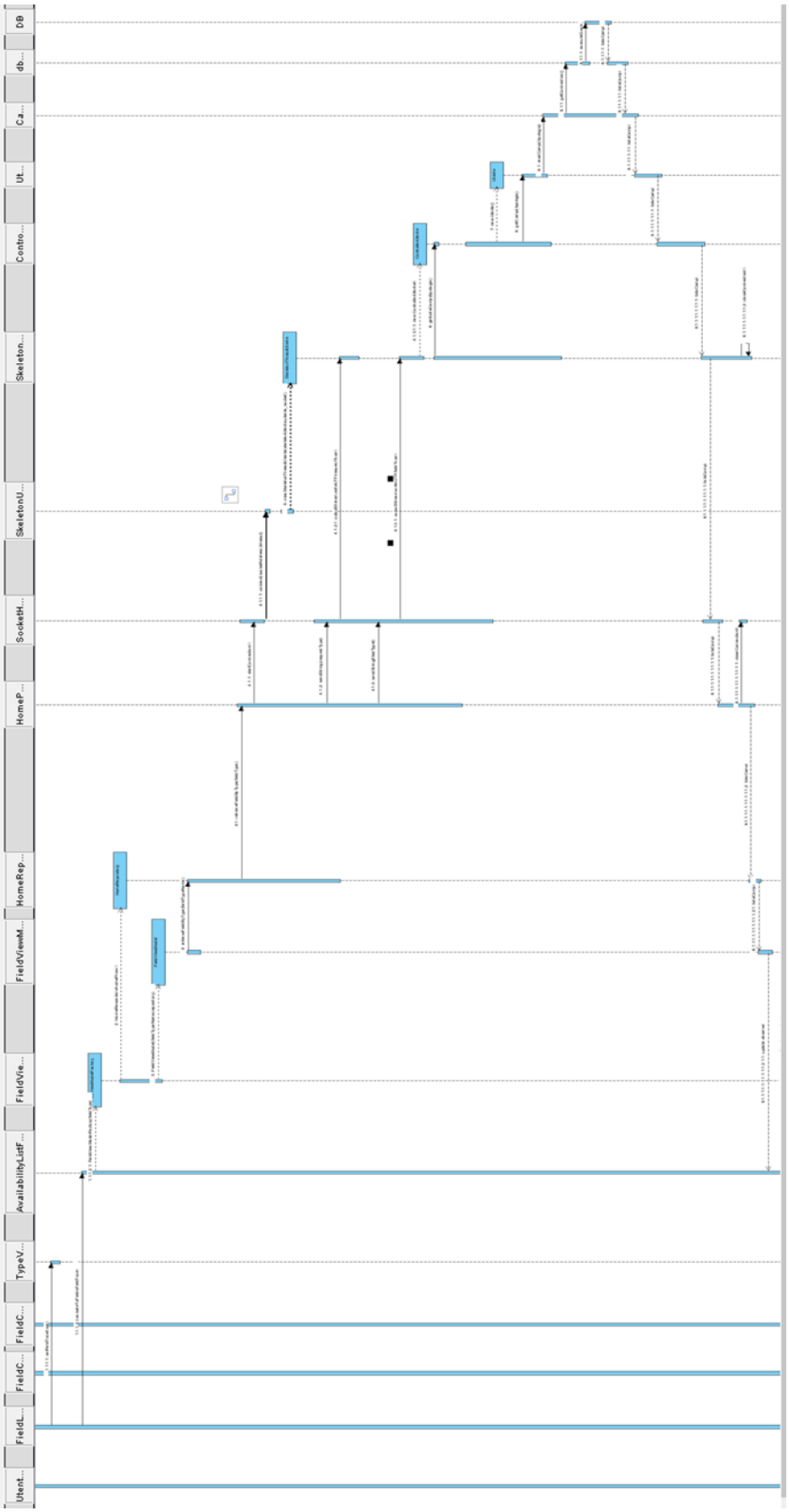


Figura 32: Sequence Diagram Raffinato - Visualizza Campi

4.8.3 Sequence Diagram Raffinato - Visualizza Disponibilità

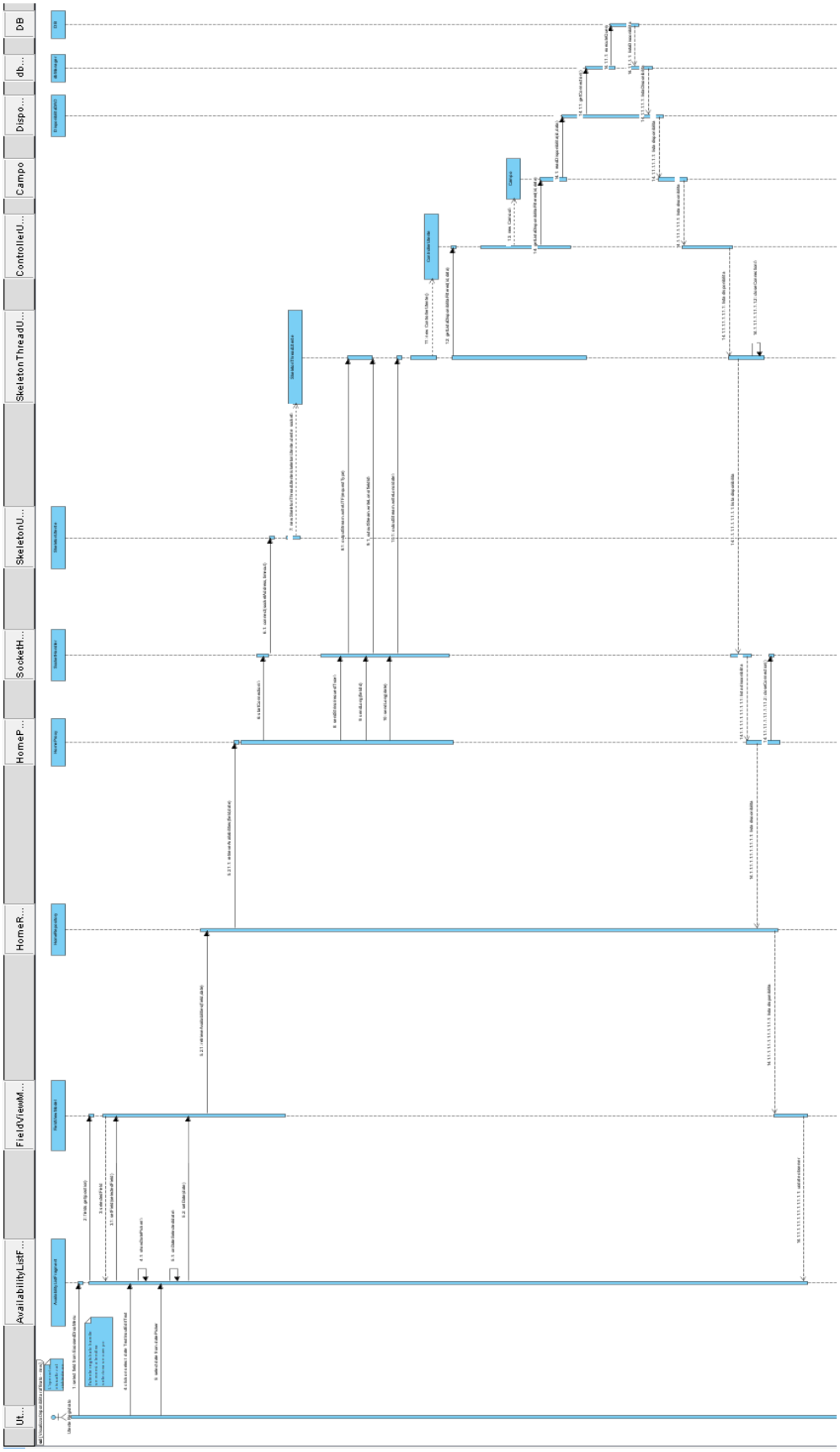


Figura 33: Sequence Diagram Raffinato - Visualizza Disponibilità

4.8.4 Sequence Diagram Raffinato - Prenota Campo

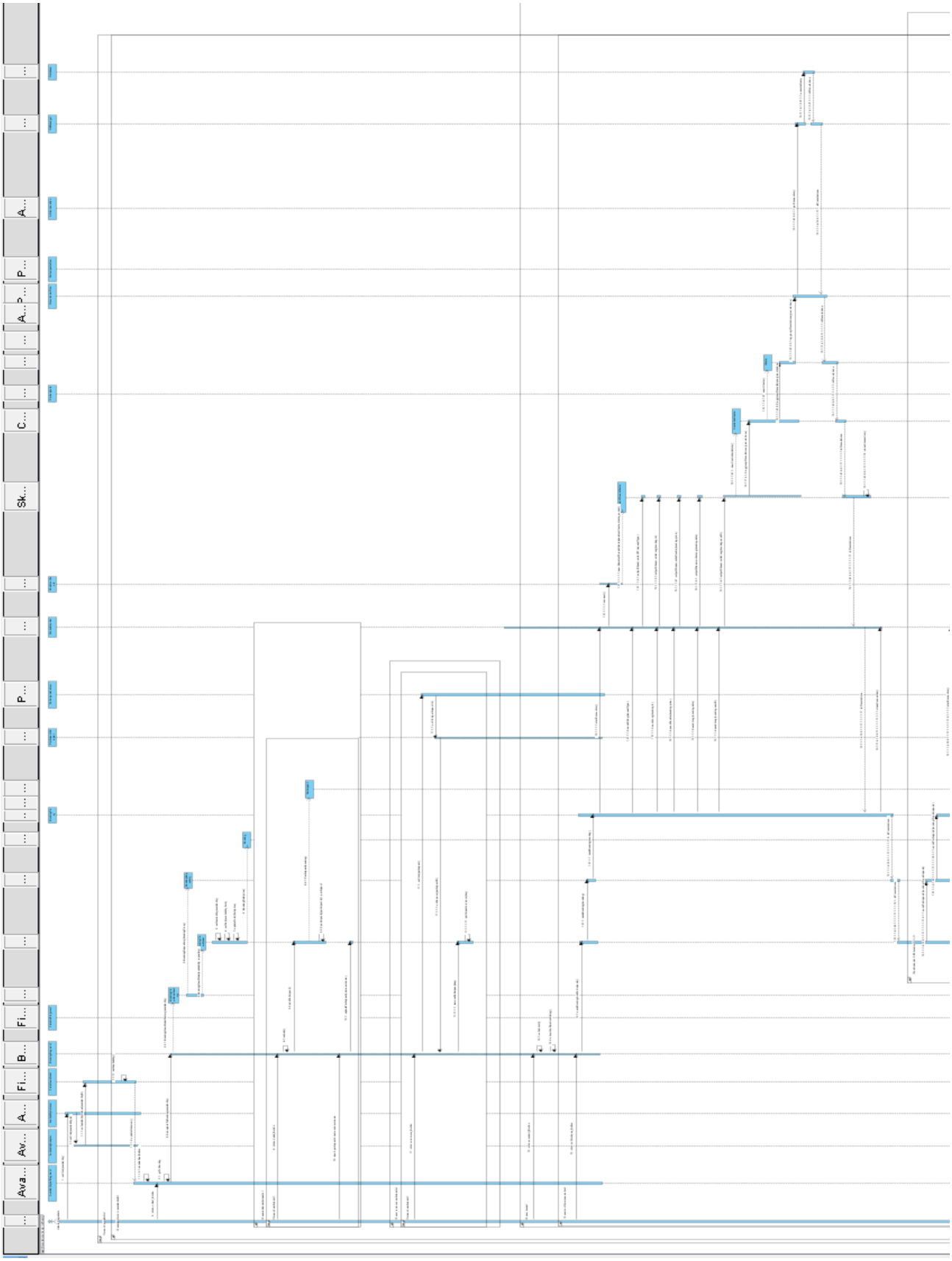


Figura 34: Sequence Diagram Raffinato - Prenota Campo

4.9 Schema del Database

Il Database utilizzato per la persistenza dei dati segue il seguente schema ER:

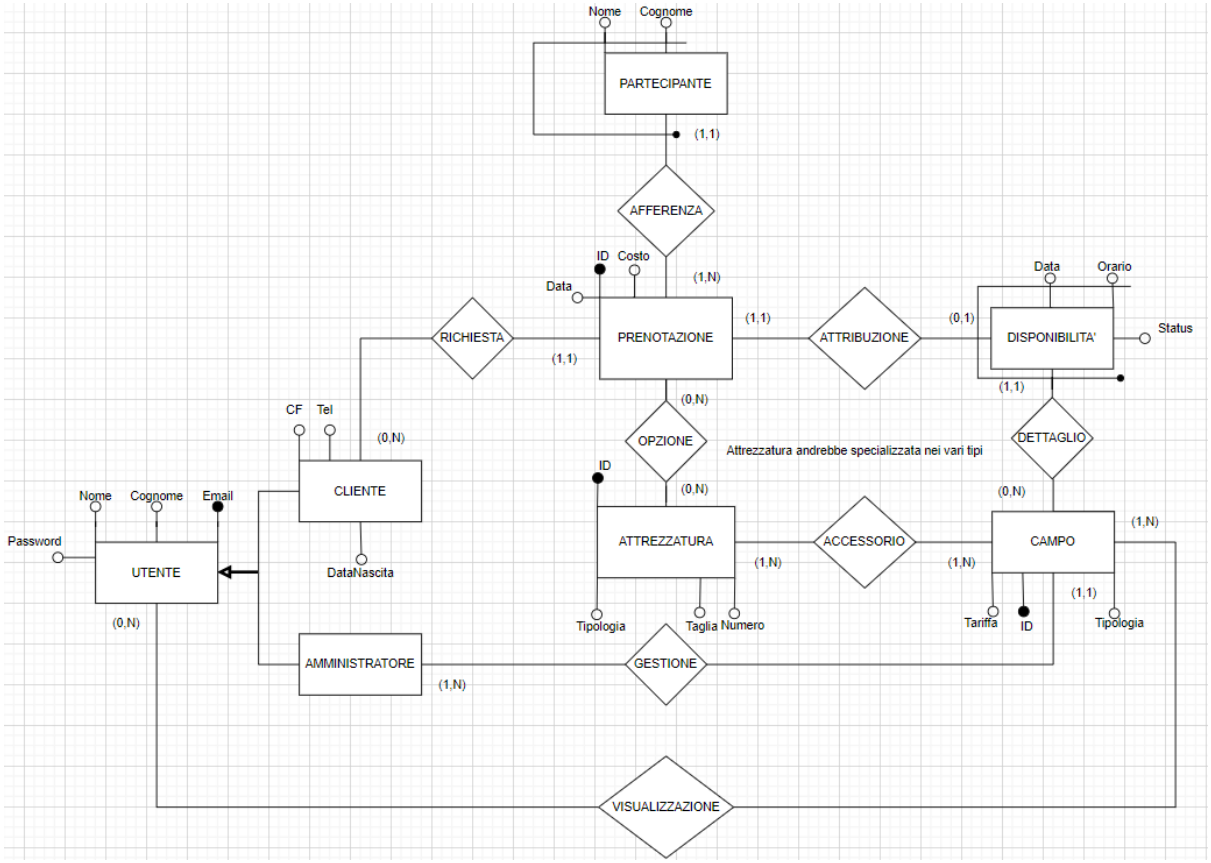


Figura 35: Diagramma ER

Progettazione Logica

Partecipanti (Id,*Prenotazione*:PRENOTAZIONE,*Nome*,*Cognome*,)
Prenotazioni (*Id*, Costo, Data, *Cliente*:CLIENTI)
Disponibilità (Id, *Data*, *Orario*, *Campo*:CAMPO, Status)
Campi (*Id*,Tariffa, Tipologia, Descrizione)
Attrezzature (*Id*, Tipologia, Taglia, Numero, Status)
Clienti (*Email*, Nome, Cognome, Password, CF, Telefono, DataNascita)
Amministratori (*Email*, Nome, Cognome, Password)

Gestioni (*Amministratore*:AMMINISTRATORI, *Campo*:CAMPI)
VisualizzazzioniA (*Amministratore*:AMMINISTRATORI, *Campo*:CAMPI)
VisualizzazzioniC (*Cliente*:CLIENTE, *Campo*:CAMPI)
Opzioni (*Prenotazione*:PRENOTAZIONI, *Attrezzatura*:ATTREZZATURE, Quantità)
Accessori (*Attrezzatura*:ATTREZZATURE, *Campo*:CAMPI)
Attribuzioni (*Prenotazione*:PRENOTAZIONI, ***IdDisponibilità***:DISPONIBILITA,**IdCampo**:CAMPI)

Capitolo 5

Implementazione del software

5.1 Documentazione dell'implementazione

Essendo il sistema Object-Oriented, il linguaggio di programmazione scelto per l'implementazione è Kotlin lato client e Java lato server. L'applicazione client è stata sviluppata utilizzando l'IDE Android Studio, mentre per il server è stato utilizzato l'IDE Eclipse.

La comunicazione tra i vari Client e il Server avviene attraverso l'uso di Socket TCP, invece per quanto riguarda la connessione tra il Server e il database MySQL, questa avviene attraverso il connettore MySQL JConnector.

5.1.1 Diagramma di deployment

Il diagramma di Deployment descrive il sistema in termini di risorse hardware e di relazioni tra di esse.

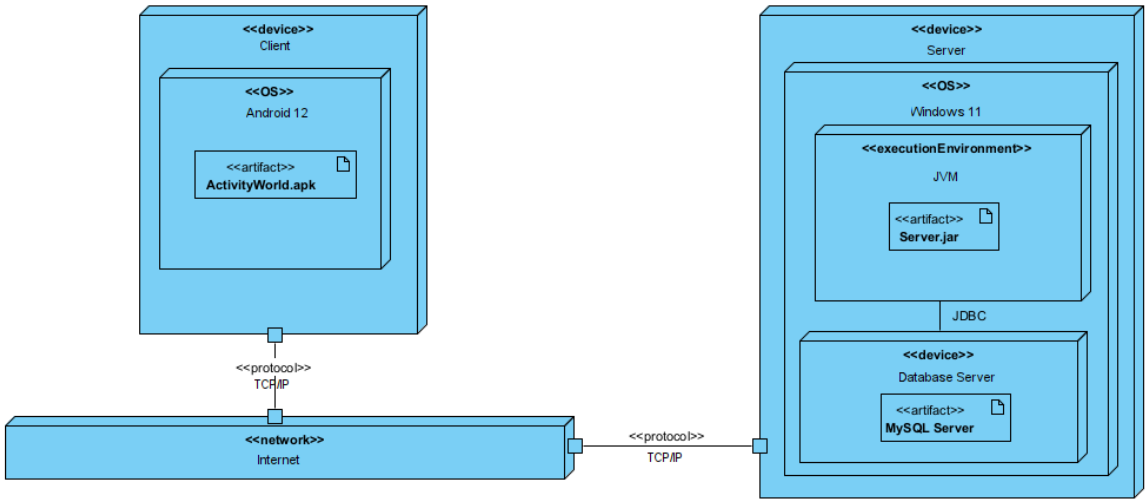


Figura 36: Deployment Diagram

5.2 Manuale di configurazione e avvio

5.2.1 Database

Prima di avviare qualsiasi altro componente della nostra applicazione, bisogna ricordarsi di mandare in esecuzione il database, popolato in precedenza. Per fare ciò apriamo XAMPP e una volta avviato il web server Apache, procediamo con l'avvio del database utilizzato, ovvero MySQL.

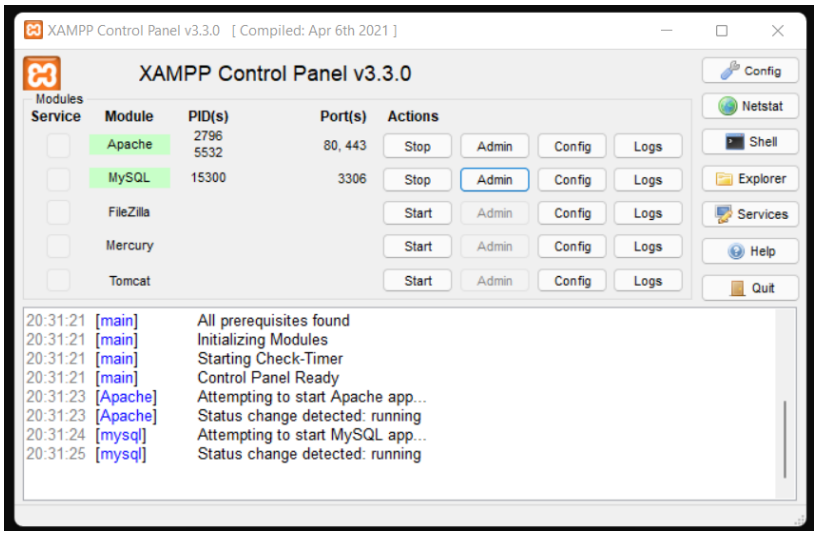


Figura 37: Schermata XAMPP

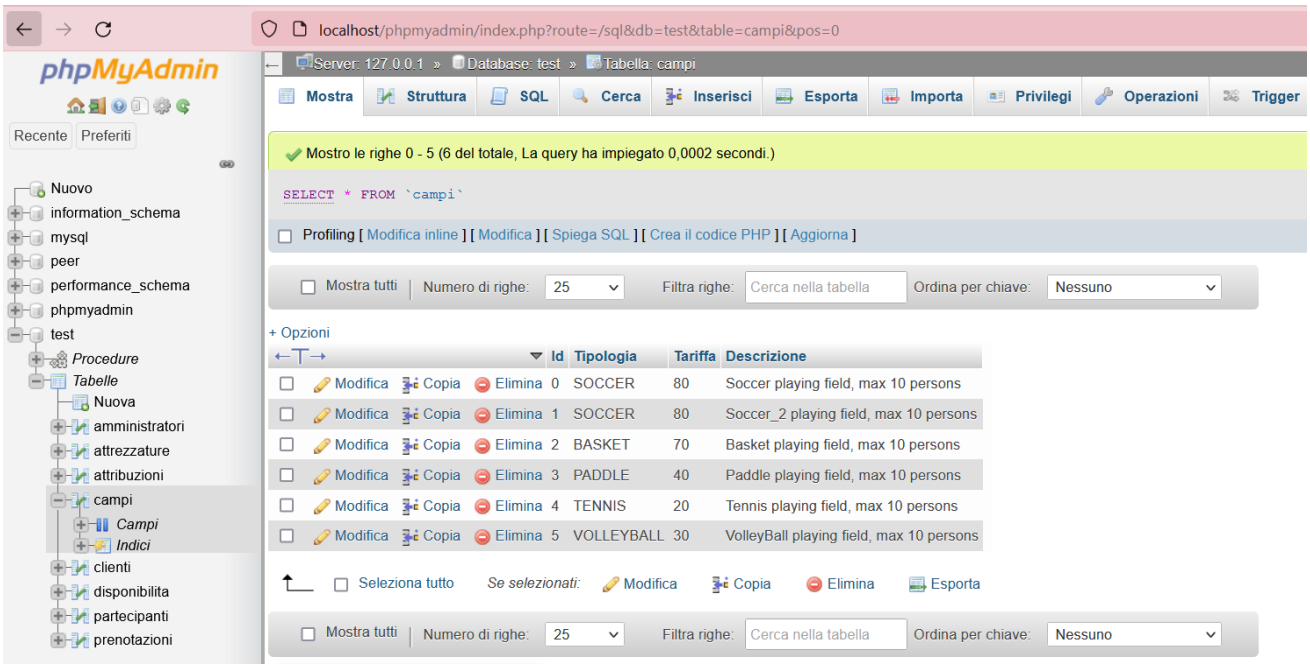


Figura 38: Database

Fatto ciò mandiamo in esecuzione il Server.

5.2.2 Server

Prima di descrivere come bisogna avviare il server, è opportuno definire i prerequisiti che devono essere necessariamente soddisfatti. Inoltre, è necessario ricordare che i comandi utilizzati sono relativi al sistema operativo Microsoft Windows, quindi potrebbero non funzionare in ambienti differenti.

Prerequisiti

- XAMPP (versione consigliata 7+)
- Java (versione consigliata 14)
- Java Development Kit (versione consigliata 13+)

Avvio

Per mandare in esecuzione il server, bisogna aprire il prompt dei comandi e spostarci nella cartella dove è presente l'eseguibile Server.jar. Successivamente avviamo l'eseguibile, grazie al comando:

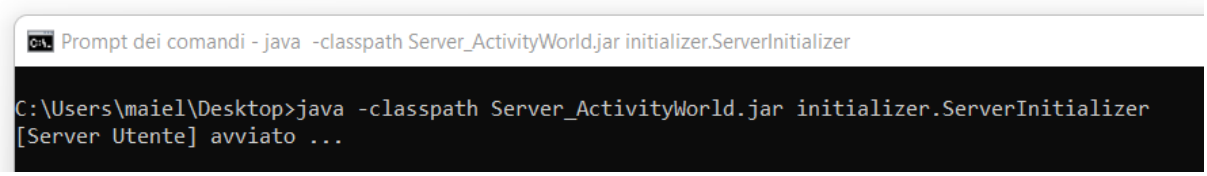


Figura 39: Lancio file jar

Al fine di rendere accessibile i servizi offerti dal server anche all'esterno della rete locale, e quindi per effettuare test remoti su vari dispositivi, si è configurato il gateway di rete in modo da effettuare un port-forwarding associando una porta esterna di comunicazione ad una interna della macchina server. Nel nostro caso abbiamo utilizzato la porta 7777. Con queste configurazioni di rete è stato possibile effettuare sia test nel quale tutti i client e il server, erano presenti nella rete locale, e sia in remoto nel quale tutti i dispositivi erano connessi a reti diverse tra loro.

5.2.3 Client

L'applicazione client è stata sviluppata e testata su dispositivi Android dotati di livello API compatibili a partire dalla versione 21, quindi Android 5.0. Per l'installazione dell'applicazione è necessario trasferire il pacchetto di installazione, file con estensione ".apk", sul dispositivo, disabilitare la protezione di "origini sconosciute" per i pacchetti di installazione e lanciare l'eseguibile.

Esecuzione del client

Terminata l'installazione, l'icona sulla "Home" di sistema consente di eseguire l'applicazione. Si fa presente che al termine dello sviluppo della suddetta versione Client, è stata prodotta una registrazione dimostrando quanto prodotto.

Figura 40: Schermate prenotazione 1

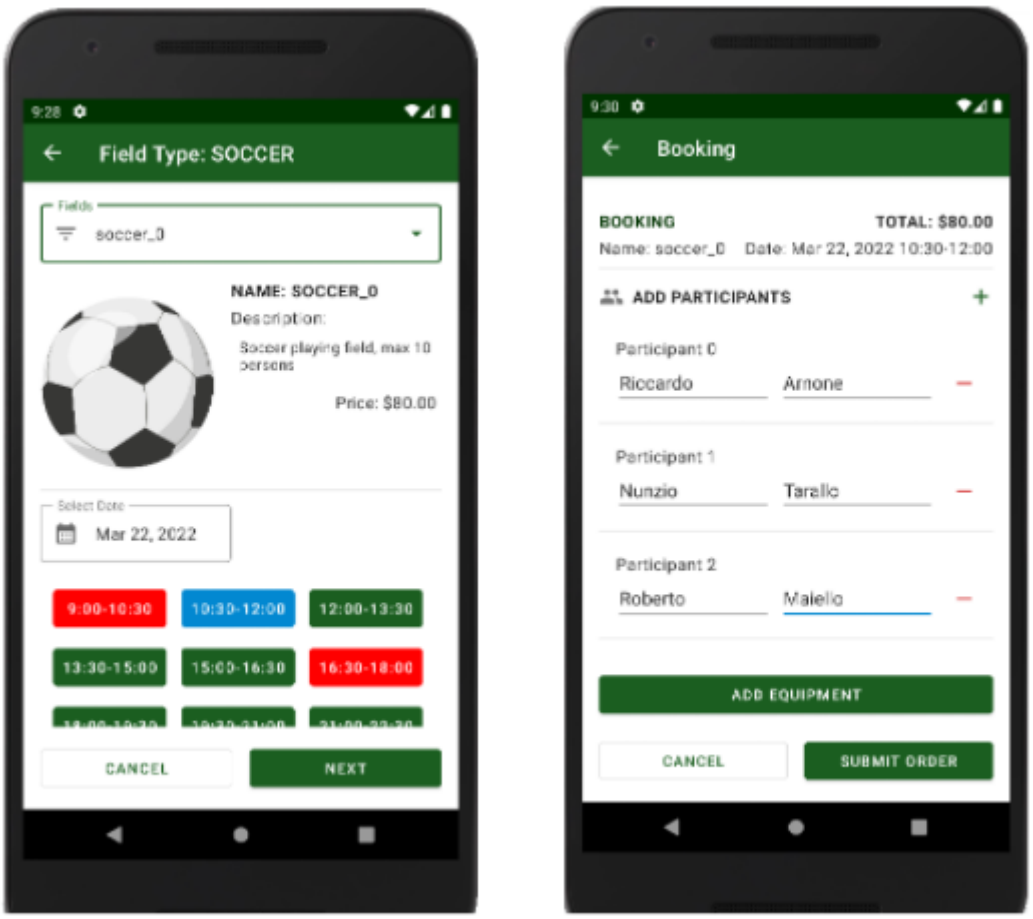


Figura 41: Schermate prenotazione 2

Capitolo 6

Testing dell’applicazione

6.1 Test Eseguiti

Al termine di ciascun iterazione, in parallelo con lo sviluppo del codice, è stata eseguita l’attività di testing; questa ha permesso di verificare la correttezza, completezza e affidabilità dell’applicazione realizzata.

Durante lo sviluppo dell’applicazione sono stati effettuati diversi test, tutti eseguiti manualmente. Sono stati effettuati dei test di accettazione per verificare il funzionamento complessivo del sistema, di tipo black box e dal punto di vista utente, ossia con riferimento a scenari dei casi d’uso.

test accettazione smartsheet

	Test n°	It.	Azione	Input	Output atteso	Output effettivo	Superato?
1	1	1	login utente	email non valida	Error: not a valid username	Error: not a valid username	☑
2	2	1	login utente	campo email vuoto	sign in button disabled	sign in button disabled	☑
3	3	1	login utente	campo password vuoto	sign in button disabled	sign in button disabled	☑
4	4	1	login utente	email e/o password non corrispondente a un utente registrato	Error: login failed!	Error: login failed!	☑
5	5	1	login utente	password < 5 caratteri	Error: password must be >5 characters	Error: password must be >5 characters	☑
6	6	1	login utente	email e password valide e corrispondenti a un utente registrato	indirizzamento a schermata visualizza tipologia	indirizzamento a schermata visualizza tipologia	☑
7	7	2	visualizza campi	click button tipologia soccer	indirizzamento a schermata campi soccer	indirizzamento a schermata campi soccer	☑
8	8	2	visualizza campi	click button tipologia tennis	indirizzamento a schermata campi tennis	indirizzamento a schermata campi tennis	☑
9	9	2	visualizza campi	click button tipologia paddle	indirizzamento a schermata campi paddle	indirizzamento a schermata campi paddle	☑
10	10	2	visualizza campi	click button tipologia basket	indirizzamento a schermata campi basket	indirizzamento a schermata campi basket	☑
11	11	2	visualizza campi	click button tipologia pallavolo	indirizzamento a schermata campi pallavolo	indirizzamento a schermata campi pallavolo	☑
12	12	2	visualizza disponibilità	campo	caricamento info e disponibilità per quel campo	caricamento info e disponibilità per quel campo	☑
13	13	2	visualizza disponibilità	data_correntei<= data <=data_corrente+30	caricamento disponibilità campo	caricamento disponibilità campo	☑
14	14	2	visualizza disponibilità	ora di inizio disponibilità antecedente all'orario attuale	disponibilità button disabled, grey	disponibilità button disabled, grey	☑
15	15	2	visualizza disponibilità	ora di inizio disponibilità uguale all'orario attuale	disponibilità button disabled, grey	disponibilità button disabled, grey	☑
16	16	2	visualizza disponibilità	ora di inizio disponibilità successivo all'orario attuale	disponibilità button enabled	disponibilità button enabled	☑
17	17	2	visualizza disponibilità	status disponibilità: available	disponibilità button enabled, green	disponibilità button enabled, green	☑
18	18	2	visualizza disponibilità	status disponibilità: reserved	disponibilità button disabled, red	disponibilità button disabled, red	☑
19	19	3	prenota campo	disponibilità selezionata	booking button enabled	booking button enabled	☑
20	20	3	prenota campo	disponibilità non selezionata	booking button disabled	booking button disabled	☑
21	21	3	prenota campo	click sul booking button	indirizzamento alla schermata prenotazione	indirizzamento alla schermata prenotazione	☑
22	22	3	prenota campo	nessun partecipante	Error: inserisci almeno un partecipante	Submit order button enabled	☐
23	23	3	prenota campo	nome e cognome del partecipante	Submit order button enabled	Submit order button enabled	☑
24	24	3	prenota campo	nome partecipante non valido	Error: invalid name!	Error: invalid name!	☑
25	25	3	prenota campo	cognome partecipante non valido	Error: invalid surname!	Error: invalid surname!	☑
26	26	3	prenota campo	click submit order button enabled	prenotazione effettuata	prenotazione effettuata	☑

Figura 42: Test Accettazione

È bene precisare che tutti i vari bug riscontrati durante i test, sono stati opportunamente eliminati.

Si è passata infine a una fase di “Refactoring” per migliorare il codice prodotto migliorandone l’espressività e la leggibilità.

I test sono stati rieseguiti e passati con successo anche dopo questa fase.