ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

# IDEAZIONE DI SISTEMI DISTRIBUITI COLLABORATIVI BASATI SU REALTÀ AUMENTATA MOBILE: UN CASO DI STUDIO, CON APPROFONDIMENTO RELATIVO AL LIVELLO DELLA COOPERAZIONE

*Relazione finale in*
PROGRAMMAZIONE DI SISTEMI EMBEDDED

*Relatore*
Prof. ALESSANDRO RICCI

*Presentata da*
FILIPPO BERLINI

*Co-relatori*
Ing. PIETRO BRUNETTI
Ing. ANGELO CROATTI

Prima Sessione di Laurea
Anno Accademico 2014 – 2015

# PAROLE CHIAVE

Augmented Reality

Location Awareness

CSCW

Cooperation

Distributed System

*"It's the real world, only better."*
*Jay Wright on Augmented Reality*

# Indice

# Introduzione

Dall'inizio del nuovo millennio lo sviluppo di tecnologie nel campo del mobile computing, della rete internet, lo sviluppo dell'Internet of things e pure il cloud computing hanno reso possibile l'innovazione dei metodi di lavoro e collaborazione.

L'evoluzione del mobile computing e della realtà aumentata che sta avvenendo in tempi più recenti apre potenzialmente nuovi orizzonti nello sviluppo di sistemi distribuiti collaborativi.

Esistono oggi diversi framework a supporto della realtà aumentata, Wikitude, Metaio, Layar, ma l'interesse primario di queste librerie è quello di fornire una serie di API fondamentali per il rendering di immagini 3D attraverso i dispositivi, per lo studio dello spazio in cui inserire queste immagini e per il riconoscimento di marker.

Questo tipo di funzionalità sono state un grande passo per quanto riguarda la Computer Graphics e la realtà aumentata chiaramente, però aprono la strada ad una Augmented Reality(AR) ancora più aumentata.

Questa tesi si propone proprio di presentare l'ideazione, l'analisi, la progettazione e la prototipazione di un sistema distribuito situato a supporto della collaborazione basato su realtà aumentata.

Lo studio di questa applicazione vuole mettere in luce molti aspetti innovativi e che ancora oggi non sono stati approfonditi né tanto meno sviluppati come API o forniti da librerie riguardo alla realtà aumentata e alle sue possibili applicazioni.

Ovvero con il nostro studio ci siamo proposti di creare un software che di base non avesse il concetto di una AR statica come per lo più è stata vista fino ad oggi, bensì creare un sistema dinamico, Object-Oriented, in cui gli oggetti aumentati abbiano uno stato, il quale può cambiare mediante le interazioni con la realtà. Il sistema che presentiamo si inserisce nel campo della realtà aumentata mobile, in quanto è sviluppato come un sistema distribuito dotato do un dominio condiviso, inserito in un'applicazione location aware. Infine è da sottolineare il fatto che noi non utilizzeremo né tecniche di riconoscimento di marker, né tecniche di rendering, né tantomeno algoritmi di localizzazione di immagini nello spazio, bensì il nostro sistema si basa su messaggi che vengono

visualizzati sulla User Interface, nel nostro caso gli smartglasses forniranno il supporto per l'interfaccia. Proprio per questo il nostro software è da considerarsi basato su realtà aumentata a causa delle ragioni suddette nonché delle motivazioni che ci hanno portato a questo studio.

L'applicazione progettata si basa su tecnologie hands-free, in particolare nel nostro caso si prevede l'utilizzo di smartglasses, questo per permettere all'utente di essere libero nei movimenti e non vincolato, potendo interagire senza usare le mani.

Altro aspetto fondamentale del software che andiamo a presentare è il dominio distribuito e condiviso. All'interno del sistema distribuito vi sono infatti anche gli oggetti aumentati, ovvero non più solo immagini renderizzate, ma veri è propri entità computazionali.

Ora vado ad'illustrare velocemente come è strutturata questa tesi e le parti in cui è divisa.

Prima di tutto verrà presentato lo stato dell'arte cercando di mostrare un po' quella che è la panoramica generale in cui si inserisce il nostro progetto. Inizialmente verrà fatto un piccolo excursus storico, seguito dall'esplorazione delle tecnologie e dei sistemi basati su realtà aumentata e location aware, accennando infine il concetto di Mirror World.

La seconda fase sarà di presentazione del caso di studio della tesi, ovvero un location based game basato su realtà aumentata con lo scopo di mettere in mostra scenari di cooperazione e di interazione fra utenti e oggetti aumentati.

Dopo questa fase verrà analizzato il caso di studio evincendone i casi d'uso e proseguendo con la modellazione del dominio. All'interno di questa parte verranno messi in mostra gli scenari fondamentali dell'applicazione.

Nella fase successiva si metterà in mostra la progettazione di questo sistema ed in particolare nella mia verrà mostrato in dettaglio lo strato di cooperazione del sistema che abbiamo ideato.

Alla fine di questa fase verrà mostrata l'implementazione del prototipo che abbiamo progettato e infine le valutazioni sul lavoro che abbiamo svolto e su quello che potrà migliorarlo in futuro.

# Capitolo 1

# Stato dell'Arte

## 1.1 Realtà Aumentata

### 1.1.1 Definizione e panoramica generale

La realtà aumentata è definita come la vista in real-time, diretta o indiretta, del mondo reale che viene però "aumentata" da informazioni virtuali computer-generated. Chi entra nel mondo aumentato percepisce un mondo fisico esteso dal mondo virtuale, ovvero un mondo che ha informazioni aggiuntive rispetto al mondo reale. Non dobbiamo considerare la realtà aumentata come il semplice rendering di immagini 2d o 3d attraverso un device, ma piuttosto una realtà che sopra lo "strato" fisico mette uno "strato" virtuale. La realtà aumentata (Augmented Reality o AR) non è nemmeno da intendersi solamente limitata al senso della vista. Potenzialmente la realtà aumentata può essere applicata a tutti i sensi, o addirittura sostituire i sensi mancanti [1].

L'Augmented Reality ha come scopo quello di semplificare la vita dell'utente in tutti momenti della sua vita portando la percezione del reale ad un livello rivoluzionario. La realtà non è più tutto ciò che l'uomo percepisce ma c'è di più. La realtà aumentata è la tecnologia che crea una "next generation, reality-based interface" [2] ,ovvero un'interfaccia di nuova generazione che sia insita e basata sulla realtà.

Le possibili applicazioni per questo tipo di tecnologia sono innumerevoli: condivisione di informazioni personali e assistenza personale, come per esempio il riconoscimento delle persone che incontri, turismo, navigazione gps, industria sia per quanto riguarda l'assemblaggio che le dimostrazioni ai clienti che la manutenzione, in campo medico, per esempio durante interventi, nella collaborazione, learning e educazione, gaming etc [3].

### 1.1.2 Cenni Storici

Il primo tentativo di realtà aumentata risale agli anni 50 quando Morton Heilig ipotizzò "The Cinema of the Future", ovvero lo spettatore doveva essere coinvolto con tutti i sensi in ciò che stava vedendo. Nel 1955 descrisse il suo modello che poi fu prototipato nel 1962 con il nome di "Sensorma". Il primo sistema di realtà aumentata nasce però nel 1966 quando Sutherland realizzò un schermo montato sulla testa dell'utilizzatore da cui si poteva vedere attraverso.

Figura 1.1: Sistema di realtà aumentata realizzato da Sutherland

L'interesse per questa nuova dimensione portò nel decennio tra il 1970 e il 1980 ad uno studio ed una continua ricerca in particolare da parte del Ames Research Center della NASA, del'università del North Carolina, del Massachussetts Institute of Technology [3]. Durante questo periodo, più precisamente nel 1975, Myron Krugger creò "Videoplace", ovvero una stanza in cui gli utenti potevano interagire con oggetti virtuali per la prima volta. Ma il concetto di Augmented Reality nasce più tardi, negli anni 90, durante un dialogo tra Tom Caudell e David Mizell. Nel 1992 L.B Rosenberg sviluppò uno dei primi esemplari funzionanti di sistemi per realtà aumentata, chiamato "Virtual Fixtures": Solo un anno più tardi Steven Feiner, Blair MacIntyre e Doree Seligmann svilupparono il prototipo KARMA (Knowledge-based Augmented Reality for Maintenance Assistance) ovvero un sistema per la manutenzione di una stampante laser di cui potevano essere dotati gli studenti, sempre basato su un supporto see-through head-mounted display. Questa invenzione fu importante in quanto fu la prima a presentare un paper sulla realtà aumentata di quello spessore [1]. Nel 1994 Paul Milgram e Fumio Kishino introdussero il concetto di Mixed Reality, ovvero di un continuo che va dall'ambiente reale, il

mondo fisico, a quello virtuale. Il mondo reale viene esteso ed aumentato dalla realtà aumentata, ovvero AR fornisce una virtualità locale, mentre la virtualità aumentata estende il mondo virtuale con quello fisico, ovvero il mondo reale, per mezzo per esempio di sensori, interagisce e modifica lo stato del mondo virtuale [3].

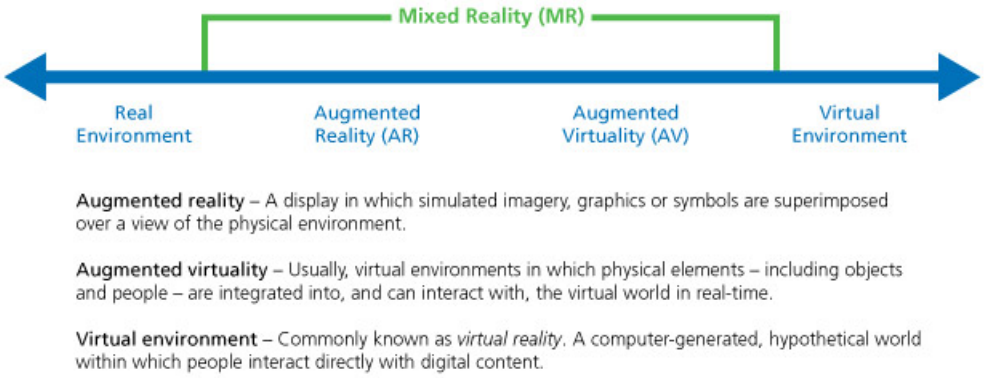


Figura 1.2: Diagramma esplicativo sulla Mixed Reality

Nel 1997 Ronald Azuma scrisse il primo saggio di Augmented Reality definendo AR come combinazione di reale e virtuale entrambi strutturati mediante le tecniche 3D, e capaci di interagire tra loro in tempo reale. La realtà aumentata diviene sempre di più fonte di interesse e di studio anche da parte degli amatori. Infatti è in questo contesto che intorno alla fine degli anni 90 vennero introdotti software toolkit a supporto dello sviluppo. Il più famoso di questi è ARToolKit che fu rilasciato nel 1999.

Gli inizi del nuovo millennio hanno dato vita al primo gioco outdoor che supporta realtà aumentata, ARQuake, creazione di Bruce Thomas, ovvero una versione "aumentata" del famoso gioco Quake, di cui fu fatta una dimostrazione durante l'International Symposium on Wearable Computers. The Horizon Report nel 2005 dichiarò che le tecnologie di realtà aumentata si sarebbero sviluppate molto più velocemente a partire da quel momento. Sempre in quell'anno furono sviluppati sistemi per fotocamere in grado di analizzare il mondo fisico in tempo reale e le posizioni degli oggetti e come sono in relazione tra loro. I sistemi basati su fotocamera sono diventati la base per le tecnologie a supporto della realtà aumentata [1].

Negli anni seguenti AR si diffuse anche nell'ambito mobile in particolare grazie alle librerie innovative Wikitude e Metaio in grado di fornire API agli sviluppatori che semplificano l'approcio alla realtà aumentata. In tempi recenti MIT "6th sense", le nuove tecnologie a supporto della realtà aumentata, come lo sviluppo nell'ambito degli smart glasses e di tutti i dispositivi wearable,

stanno portando ad uno sviluppo ed una ricerca costante, che molto probablimente porterà queste tecnologie ad un instaurarsi nella vita quotidiana di tutti.

## 1.2 Applicazioni Location Based

Le applicazioni location based sono applicazioni che basano le loro funzionalità sulla localizzazione e quindi la posizione dell'utente. Ad ogni user sono fornite delle funzionalità a partire dalla propria latitudine e longitudine.

La tecnologia che regge questo ambito di sviluppo, in particolare nella sua accezione outdoor, è il GPS, ovvero la localizzazione mediante satelliti. Questo tipo di tecnologia è ancora in sviluppo in quanto la precisione è ancora approssimativa, gli attuali sistemi hanno un errore di 3 - 5 m nei luoghi dove il segnale è elevato, ma sono stati fatti grandi passi rispetto alle prime tecnologie GPS [4]. DGPS (Diffierential Global Positioning System) utilizza postazione fisse per eliminare o diminuire il margine di errore attraverso la combinazione di più osservazioni. AGPS (Assisted Global Positioning System) è invece un sistema che permette di diminuire i tempi per la prima localizzazione appoggiandosi alle celle GSM (Global System for Mobile Communication).

Le applicazioni location-based hanno grande sbocco soprattutto nel gaming. L'utilizzo della realtà aumentata e della localizzazione hanno dato vita ad un'esplosione dei location-based games. Ma prima di presentare questo tipo di applicazioni è secondo me utile soffermarsi sul concetto di collaborazione e sullo sviluppo di applicazioni basaste su Augmented Reality a supporto di essa.

## 1.3 Collaborazione

### 1.3.1 Panoramica Generale

La collaborazione, in quanto atteggiamento insito e indispensabile nella realtà umana, è stato fin dall'inizio degli studi sulla realtà aumentata un punto cruciale di ricerca. La necessità di riuscire a creare ambienti aumentati in cui più utilizzatori fossero in grado di interagire tra loro, collaborare e cooperare ha portato alla prototipazione di diverse applicazioni che in qualche modo hanno cercato di fornire in parte alcune di queste funzionalità. Ma prima ancora dobbiamo capire come può essere la collaborazione all'interno di un'applicazione di questo tipo. Distinguiamo tre stili fondamentali di interazione nel mondo della collaborazione: "attesa", "parzialmente inattesa" e "inattesa".

La collaborazione "attesa" ha interazioni che necessitano l'intervento degli utenti che stanno agendo all'interno del processo collaborativo. Di solito questo tipo di utenti rimane sostanzialmente stazionario durante l'interazione. Un esempio di collaborazione "attesa" può essere la chat in un ambiente collaborativo.

La collaborazione "parzialmente inattesa" invece non ha bisogno di utenti sempre presenti ma di utenti "nomadi" che hanno interazione con uno o più device, ma non direttamente con altri utenti. Questo tipo di collaborazione è utile quando un utente deve agire sulla sua applicazione per creare, modificare o distribuire file condivisi. Tutti gli utenti hanno sul loro device una replica di questi file e non sono consapevoli del processo di cambiamento che sta avvenendo nell'applicazione. In questo scenario solo l'utente che sta interagendo con l'applicazione è stazionario mentre gli altri possono essere mobili.

La collaborazione "inattesa" infine coinvolge utenti che sono in movimento mentre i loro device interagiscono tra loro per fornire qualche servizio cooperativo. Gli utenti sono tipicamente non consapevoli del processo che sta avvenendo all'interno dell'applicazione. Questo tipo di collaborazione è utilizzata per implementare meccanismi di condivisione che sono incorporati nell'applicazione collaborativa. Per esempio la localizzazione, la connessione, la disponibilità di file condivisi etc [5].

### 1.3.2 Augemented Reality in Computer Supported Collaborative Work

"How collaborative activities and their coordination can be supported by means of computer systems." [6]

"CSCW a generic term, which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques." [7]

Computer Supported Collaborative Work (CSCW) comprende tutto l'ambito di ricerca in cui si cerca di migliorare e rendere più efficiente il lavoro in team, e la cooperazione al suo interno attraverso le tecnologie informatiche in tutti i loro aspetti, dall'hardware al software, dal networking a tutti i servizi e le tecnologie che possono essere offerte.

La realtà aumentata offre uno scenario innovativo e intrigante per quanto riguarda il CSCW. Tecnologie hands-free e wearable che supportano realtà aumentata, l'interazione e la manipolazione di oggetti virtuali in team attraverso la collaborazione face-to-face, la collaborazione da remoto avendo un utente che funge da osservatore di quello che il suo compagno sta svolgendo, la collaborazione multiscala [8] sono tutti aspetti in cui AR può dare fortemente il suo contributo all'interno dei sistemi CSCW.

Schmalstieg [10] ha identificato cinque proprietà fondamentali nei sistemi collaborativi basati su realtà aumentata: virtuality, augmentation, multi-user support, independence, individuality [11].

Virtuality ovvero deve essere possibile l'esaminazione e la vista di oggetti non direttamente accessibili o non esistenti nel mondo reale.

L'augmentation è proprio il concetto di base della realtà aumentata ovvero l'estensione agli oggetti reali di proprietà virtuali, e o anche elementi virtuali, come descrizioni per esempio.

Con multi-user support invece si fa riferimento all'essenza dei CSCW ovvero la collaborazione. Perchè ci possa essere collaborazione ci devono essere più utenti attivi sullo stesso problema.

Independence d'altro canto è invece il fatto che ogni utente deve avere una sua indipendenza. Non bisogna fare l'errore di considerare un sistema collaborativo basato su realtà aumentata come un'utente attivo e n utenti che osservano ma bensì che ogni utente sia libero di muoversi all'interno del sistema. Deve poter decidere cosa vedere, con cosa interagire.

L'individuality tiene conto del fatto che l'applicazione può avere bisogno che l'utente veda i dati condivisi in modo diverso dagli altri [11].

In fine non bisogna dimenticare che questi sistemi necessitano dell'interazione con l'utente e ancora prima di essere interattivi [10].

Un esempio importante di interfaccia che permette collaborazione è Shared Space. Questo progetto unisce la realtà aumentata assieme ad un'interfaccia utente fisica e spatial 3D per fornire appunto un ambiente computazionale in grado di aiutare e supportare la cooperazione e la collaborazione. Shared Space permette all'utente di vedere ed interagire con immagini virtuali facendo riferimento però a oggetti reali, diagrammi. In più permette agli utenti di lavorare simultaneamente sul mondo reale e virtuale. Gli utenti per poter fare questo sono dotati di un database di immagini virtuali che vengono "renderizzate" sopra dei marker. Chiunque possieda questo database è in grado di vedere nello stesso momento l'immagine. In più l'utente è in grado di manipolare gli oggetti virtuali manipolando i marker fisici su cui sono visualizzati gli oggetti aumentati [9].

## 1.4 Location Based Games e Pervasive Games

### 1.4.1 Location Based Games

La diffusione di device mobili con capacità di computazione discrete a portato ad una proliferazione di giochi situati o basati sulla localizzazione. Questi giochi sono dotati di uno spazio fisico, ma allo stesso tempo di azioni ed eventi in uno spazio virtuale. Questi giochi, che vengono spesso anche definiti urban

games o street games, di base prevedono l'interazione sia tra mondo reale e mondo virtuale, sia tra i diversi utenti [12]. Molto spesso tra le specifiche dei location based games c'è la collaborazione tra gli user. Lo spazio in cui questi giochi si svolgono viene detto game space o narrative space se gli elementi narrativi prevalgono nel gioco [12].

### 1.4.2 Pervasive Games

Il concetto di pervasive game amplia ciò che si intende per location based game. Ovvero pervasive game estendono spazialmente e temporalmente i confini del gioco [12]. L'idea di base di questo tipo di giochi è che si possano giocare dovunque e in qualsiasi momento senza essere limitati nel tempo e nello spazio. Naturalmente i requisiti per un sistema di questo tipo sono tecnologie di localizzazione affidabili e comunicazione tra utenti [13].

### 1.4.3 Alcuni Esempi

I software sviluppati per il gaming basati su localizzazione spaziano dal gioco a puro fine ludico ai giochi per l'apprendimento o con fini pedagogici. Tra questi vi sono tutta una serie di giochi che si basano sulla collaborazione in team e sulla cooperazione all'interno di essi. Nell'ambito del learning vi sono presenti svariati esempi di location based game collaborativi. Per esempio Mystery at the Museum, ovvero un gioco creato per il Boston Museum of Science che richiede la collaborazione fra i giocatori per risolvere casi di furto all'interno del museo [12]. Un esempio di AR game collaborativo può invece essere Time Warp. Time Warp è un gioco all'aperto situato nel centro di Colonia. I giocatori sono equipaggiati di un sistema per realtà aumentata e un PDA (Personal Digital Assistent). L'ambiente reale in alcune zone è arricchito di personaggi virtuali e ricostruzioni di edifici storici virtuali. Il palmare ha la funzione di terminale delle informazioni, ovvero mostra la propria posizione corrente e la posizione dei portali temporali. In più una pagina web offre agli osservatori di vedere il gioco in streaming. Tutto questo basato sul MORGAN AR/VR framework, che utilizza CORBA e design patterns come il publish-subsrcibe. CORBA (Common Object Request Broker Architecture) è uno standard che facilita la comunicazione tra sistemi che sono sviluppati su piattaforme diverse. Il publish-subscribe pattern permette a processi interessati a dati provenienti da particolari componenti di "abbonarsi" (subscribe) al componente per poi ricevere i suddetti dati da lui "pubblicati" (publish). La localizzazione dei giocatori è basata su sistemi GPS.

Le problematiche maggiori di questi due esempi sono la difficile adattabilità ad altri luoghi in primis, e in più la necessità di un'infrastruttura di rete forte e resistente alle cadute di linea e alle basse latenze.

## 1.5 Tecnologie a supporto della realtà aumentata

In questa sezione si descrivono i principali tools a supporto della realtà aumentata, ovvero Wikitude, Metaio e Layar.

Wikitude è un tool che fornisce tecnologie utili alla realtà aumentata nato in Austria nel 2008. Inizialmente nato come supporto ad applicazioni AR location-based, dal 2012 è stato indirizzato anche verso il riconoscimento di immagini e markers, con il lancio del Wikitude SDK, che sfrutta anche tecnologie di geolocalizzazione.

Per quanto riguarda le applicazioni location-based, Wikitude fornisce esperienze di realtà aumentata utilizzando la localizzazione GPS e sensori quali giroscopio e accelerometro per permettere il calcolo della posizione degli oggetti da visualizzare sullo schermo di smartphone o smart-glasses.

Metaio, compagnia fondata nel 2003 a Monaco di Baviera, si occupa dello sviluppo di software che supportino la realtà aumentata. Ha prodotto il Metaio SDK, che permette di sviluppare applicazioni AR in svariati ambienti come, ad esempio, iOS, Android e Windows. Nel 2009 rilascia Junaio, un browser studiato per la realtà aumentata, che ancora oggi è ampiamente diffuso.

Sempre nell'ottica dei browser si inserisce Layar, compagnia olandese fondata nel 2009 ad Amsterdam. Layar si è imposto fin da subito come uno dei browser per realtà aumentata dominanti nel mercato, offrendo esperienze di augmented reality accessibili a un numero molto elevato di utenti.

Tutti questi strumenti offrono esperienze significative nell'ambito dell'AR, ma sono limitati dalla necessità di utilizzare markers o tecniche avanzate di riconoscimento delle immagini per risultare efficaci.

## 1.6 Mirror Worlds

Un altro concetto fondamentale che si può trarre dallo studio della realtà aumentata è quello di Mirror World. Il software aggiunge al mondo reale informazioni che lo ampliano e si crea in questo modo un continuo che va dal reale al virtuale che aumenta anche ciò che l'utente percepisce. Ma non è solo l'utente a percepire il mondo bensì è anche il mondo che percepise il reale, e quindi l'utente. In questo contesto il mondo è sì esteso dal virtuale, ma allo

stesso tempo il reale diviene estensione del virtuale. Ovvero il mondo virtuale coglie i cambiamenti della realtà che lo circonda e in base questo varia il suo stato.

In questo ambito la realtà aumentata è la tecnologia che rende possibile sviluppare un mondo di questo tipo. L'estensione del reale è proprio il concetto di informazione aumentata che poi verrà visualizzata attraverso il dispositivo.

Un esempio di Mirror World è Ghost Game in a City. All'interno di questo gioco vi sono due team, dotati di smart glasses a supporto della realtà aumentata, i quali sono alla ricerca di tesori. Ma in questo ambiente vi sono dei nemici ovvero i fantasmi. Il fantasma, a differenza dei personaggi virtuali all'interno di giochi di Augmented Reality, è in grado di percepire il reale e reagire di conseguenza. Un esempio molto chiaro su questo fatto è che il fantasma coglie lo stato di luminosità del mondo reale e nel momento in cui è sopra ad una certa soglia diviene vulnerabile. O ancora percependo come è fatto il mondo reale il fantasma può valutare diverse strategie per catturare i giocatori.

Il modo in cui oggi vediamo la realtà può essere molto rivoluzionato dall'introduzione di questi aspetti e dallo sviluppo di tecnologie per realtà aumentata [14].

# Capitolo 2

# Caso di Studio

## 2.1 Caso di Studio

L'idea di base è quella di realizzare un sistema che permetta ad un team formato da più elementi di cooperare al fine del raggiungimento e dell'analisi di determinati punti d'interesse, sfruttando la realtà aumentata.

Un punto può essere visto come uno "scrigno" che contiene un determinato oggetto (può anche essere vuoto). L'obiettivo del team è quello di raggiungere i punti d'interesse per trovare tutte le monete che compongono il "tesoro" finale.

Uno scrigno può contenere delle monete oppure una chiave per un altro scrigno, per cui solo il giocatore che ha raggiunto il punto in cui vi è una chiave può ottenere l'oggetto dello scrigno aperto da tale chiave.

Una volta aperti tutti gli scrigni il gioco è concluso. Tutte le informazioni relative ai punti già raggiunti e allo stato di ogni punto dovranno essere condivise tra i membri del team.

A seconda delle informazioni in esso contenute, lo scrigno può essere aperto secondo diverse modalità:

- può essere aperto direttamente giungendo al punto di interesse
- può essere aperto solo se si è in possesso della chiave dello scrigno
- può essere aperto solo se almeno due membri del team sono presenti nel punto d'interesse
- può essere aperto solo se tutti gli altri scrigni sono stati aperti e almeno due membri del team sono presenti nel punto d'interesse (in questo caso si tratta dello scrigno finale)

Inizialmente non si conoscono le informazioni contenute all'interno dei punti d'interesse e il loro stato è "non visitato". Quando avviene l'arrivo di un

giocatore in un punto, lo stato può cambiare in "scrigno aperto", specificando anche il tipo di informazione ivi ottenuta (monete, chiave, nessuna informazione), o "scrigno non apribile", specificando la causa, ovvero mancanza di chiave o bisogno della presenza di un altro giocatore.

Ogni giocatore visualizza sullo schermo i dieci punti di interesse che rappresentano gli "scrigni" da aprire (attraverso una mappa, un radar o un elenco) e la sua posizione.

Viene anche visualizzata la quantità totale di "monete" raggiungibili, data dalla somma dei contenuti di ogni scrigno (la quantità di ogni scrigno diminuisce col passare del tempo o con l'azione di una "centrale di controllo").

Nel momento in cui un giocatore raggiunge uno dei punti, se il giocatore è autorizzato ad aprire lo scrigno (possiede la eventuale chiave o è in compagnia dell'altro giocatore se è necessaria la cooperazione) può visualizzare il contenuto dello scrigno il quale viene aggiunto al totale delle monete e anche scalato dal totale disponibile.

Se lo scrigno contiene anche una chiave, questa viene visualizzata e aggiunta alle chiavi già possedute dal giocatore.

Se lo scrigno non è apribile, l'eventuale esigenza di cooperare o di possedere una determinata chiave viene condivisa tra i due giocatori, ovvero entrambi possono conoscere lo stato di ogni punto in ogni momento tramite un'apposita schermata e vengono notificati di ogni loro cambiamento di stato.

Sullo scenario di gioco agiscono anche dei "ladri fantasma", posizionati in delle zone predefinite. All'arrivo di un giocatore in quelle zone, il ladro assale il giocatore. Il ladro ruba parte del suo denaro (se ne possiede) che viene scalato dal totale. Al termine dell'aggressione, il giocatore può lasciare un segnale di pericolo sul campo, in modo tale che quando il compagno arriverà in tale punto sarà conscio del pericolo.

Il gioco termina quando tutti gli scrigni sono stati aperti. L'idea è quella di inserire dieci scrigni nello scenario (dieci punti d'interesse). Quattro di questi sono apribili solo possedendo la chiave (per cui ci saranno quattro scrigni che possiedono una chiave al loro interno). Quattro sono apribili semplicemente raggiungendo il punto. Uno è apribile solo con la presenza simultanea di entrambi i giocatori. Uno (il punto finale) è apribile solo con la presenza di entrambi i giocatori e solo dopo aver aperto tutti gli altri.

## 2.2 Considerazioni

Tale caso di studio offre spunti interessanti riguardo i principali aspetti della realtà aumentata.

Innanzitutto, il sistema da realizzare può essere definito come un gioco location-based. Difatti la localizzazione è cruciale per il sistema. La posizione dei giocatori deve essere nota in ogni momento e deve essere confrontata in continuazione con la posizione degli scrigni. Inoltre, anche i ladri sono dotati di una propria posizione. Sarà quindi necessario individuare tecniche di localizzazione adeguate e studiare un sistema di condivisione delle posizioni.

Anche la comunicazione giocherà un ruolo fondamentale. Infatti il sistema da sviluppare sarà a tutti gli effetti distribuito, in quanto informazioni sugli oggetti (come ad esempio il loro stato) dovranno essere necessariamente condivise tra tutti gli utenti. Per cui sarà necessaria un'infrastruttura di rete robusta e in grado di reagire a eventuali malfunzionamenti.

Un altro aspetto interessante e cruciale riguarda la cooperazione. Molti degli scenari di gioco prevedono l'interazione, sotto forma di collaborazione, di diversi utenti. Infatti il fatto che alcuni scrigni possano essere aperti solamente con la presenza di due giocatori, o anche che un giocatore possa lasciare una notifica di pericolo per il compagno, implica che vengano sviluppati meccanismi più o meno espliciti di cooperazione. Più in generale, è evidente che il sistema dovrà soddisfare tutti i cinque punti ipotizzati nel progetto Studierstube sulle applicazioni AR collaborative.

Dovrà far leva sulla virtuality, in quanto i giocatori dovranno poter interagire con gli scrigni e i ladri, non presenti "fisicamente" sul campo.

Dovrà far uso dell'augmentation, per aggiungere informazioni al reale (in questo caso il reale può essere visto come una posizione ben precisa nel mondo fisico data da latitudine e longitudine).

Dovrà, come detto precedentemente, basarsi sulla cooperation.

Dovrà rispettare l'indipendence, poichè i giocatori vedranno il mondo aumentato da diverse prospettive.

Infine, dovrà integrare aspetti dell'individuality, in quanto i giocatori potranno vedere gli stessi oggetti in maniera differente (ad esempio, il giocatore in possesso della chiave per aprire un determinato scrigno lo vedrà come "apribile", un altro giocatore non in possesso della chiave no).

Tutte queste caratteristiche rendono il gioco che verrà sviluppato un'applicazione AR collaborativa a tutti gli effetti.

# Capitolo 3

# Analisi e Modellazione

## 3.1 Casi d'Uso

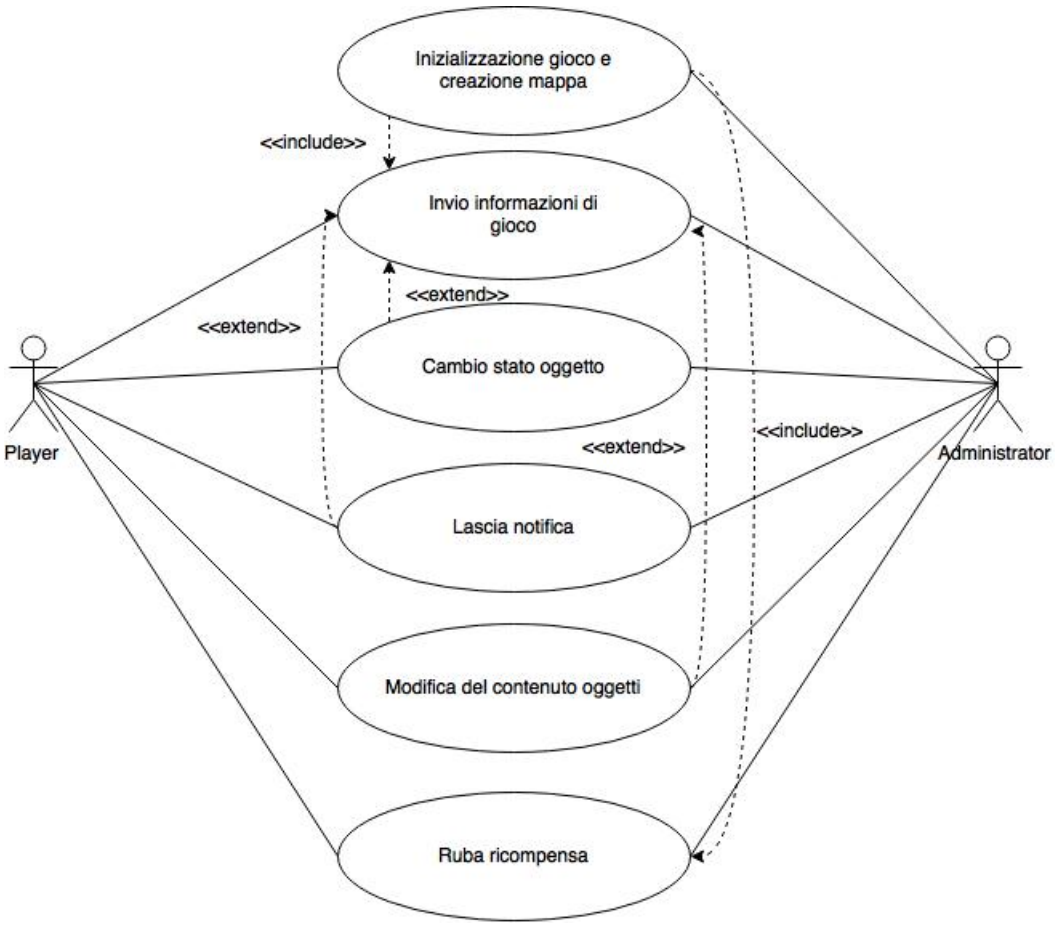


Figura 3.1: Casi d'uso del nostro sistema

### 3.1.1 Inizializzazione del gioco e creazione della mappa

Il server, una volta avviato, crea gli oggetti e fornisce loro delle coordinate, andando a definire così la mappa di gioco. Le informazioni contenute negli oggetti saranno il fulcro del sistema e verranno modificate e condivise con gli utenti.

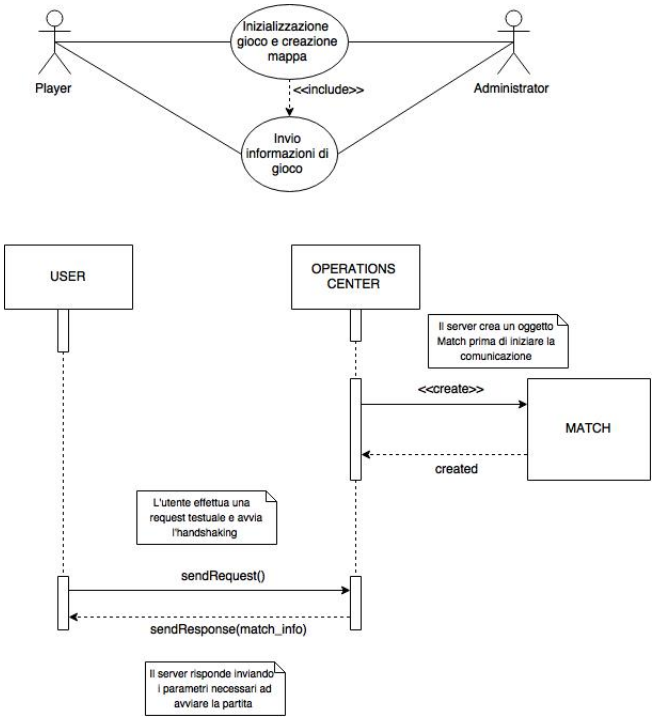


Figura 3.2: Diagramma di sequenza relativo all'inizializzazione del gioco

### 3.1.2 Invio delle informazioni di gioco

Il server viene contattato da un utente e risponde inviando le informazioni relative a tutti i punti di interesse precedentemente inizializzati. Una volta completato il download delle informazioni da entrambi i giocatori il gioco ha inizio. Si noti che inizialmente le informazioni inviate agli utenti danno indicazioni solo sulla posizione dei punti ma non sul loro contenuto.

### 3.1.3 Cambio stato oggetto

L'oggetto è localizzato attraverso due zone circolari, il cui centro è dato dalle coordinate dell'oggetto. La zona più ampia indica la vicinanza all'oggetto, per cui quando l'utente vi entra è avvisato di essere in prossimità dell'oggetto. La zona più piccola indica l'oggetto stesso, per cui quando l'utente entrerà in

tale zona andrà ad agire direttamente sull'oggetto, causandone eventualmente il cambio di stato.

Gli stati che caratterizzano un oggetto sono i seguenti:

- UNVISITED: oggetto che si trova in un punto non ancora visitato
- OPEN: oggetto aperto (il suo contenuto è stato ottenuto)
- LOCKEDKEY: oggetto visitato ma chiuso e da aprire con la chiave specificata
- LOCKEDCOOPERATION: oggetto visitato ma chiuso e da aprire con cooperazione attraverso gli utenti
- FINAL: oggetto finale, visitato ma da aprire solamente con la cooperazione di entrambi gli utenti e dopo aver aperto tutti gli altri oggetti

L'arrivo di un utente in prossimità di un oggetto ne determina l'eventuale cambio di stato.

- boolean key_needed
- boolean cooperation_needed
- boolean final
- boolean visited
- boolean have_key
- int number_of_user
- boolean all_open

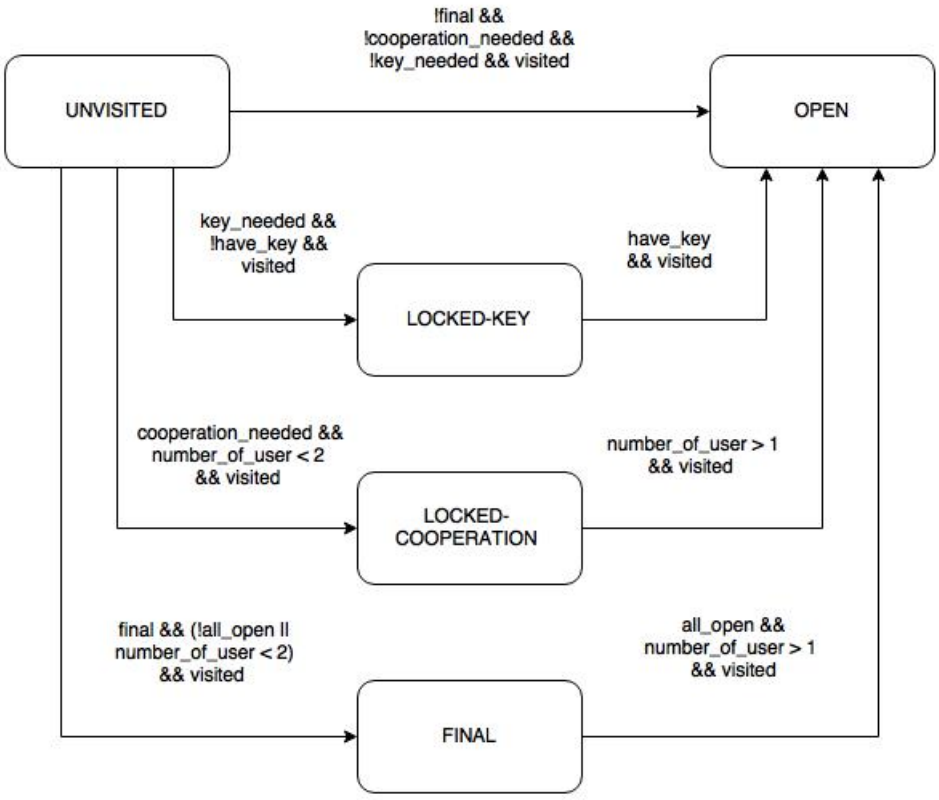


Figura 3.3: Diagramma a stati del Treasure Chest

Ecco il diagramma di sequenza che mostra le interazioni che avvengono al momento del cambio di stato di un oggetto:

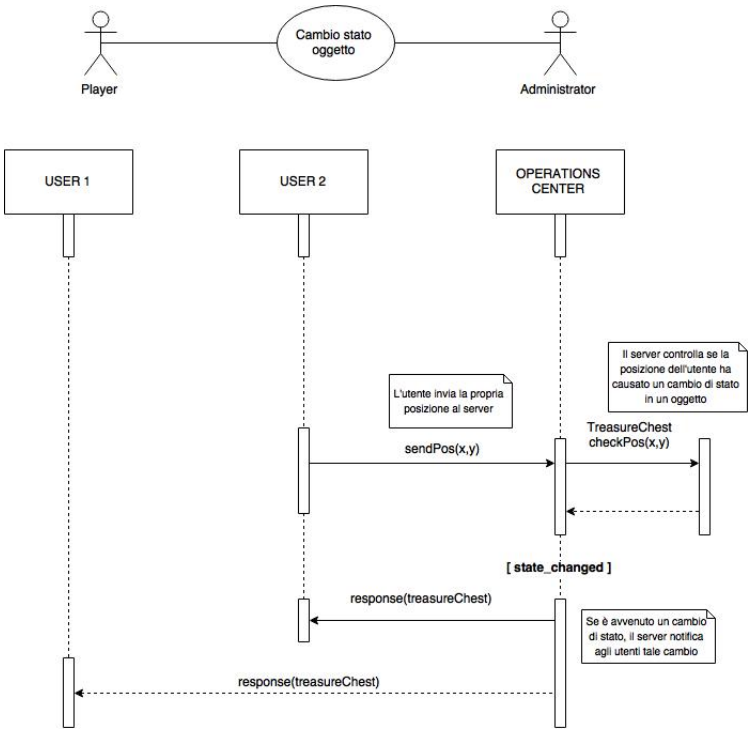


Figura 3.4: Diagramma di sequenza relativo al cambio di stato di un oggetto

Ecco i vari scenari:

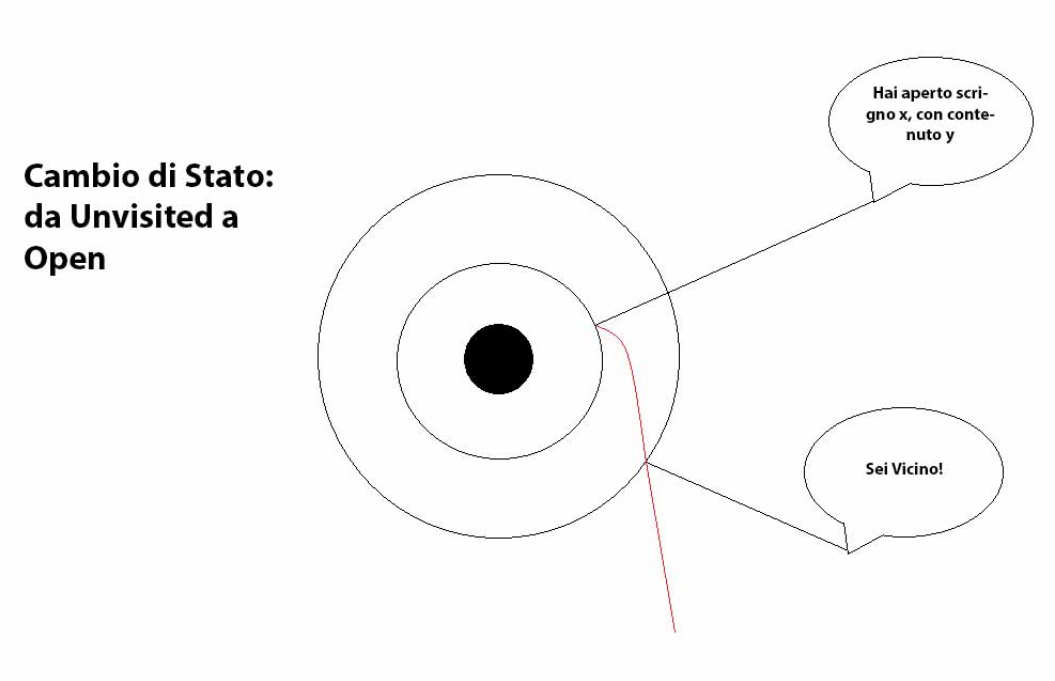


Figura 3.5: Caso in cui lo scrigno può essere aperto

**Caso in cui lo scrigno può essere aperto** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto. Tale notifica non è generata dal server ma direttamente lato utente, siccome la posizione dell'oggetto è nota a priori grazie all'handshaking iniziale.

Una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server (che conosce la posizione dell'utente in ogni istante) che ha aperto con successo l'oggetto e ne ha ottenuto il contenuto.

Il cambio di stato dell'oggetto da UNVISITED a OPEN viene notificato ad entrambi gli utenti insieme all'informazione relativa al suo contenuto (monete o chiavi).

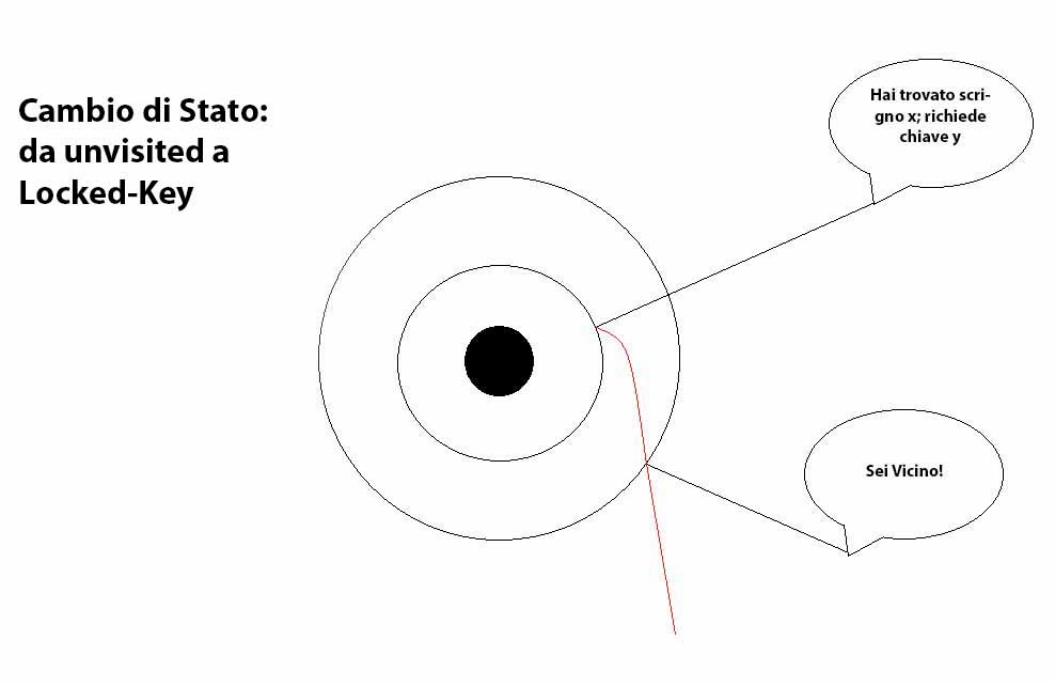


Figura 3.6: Caso in cui l'apertura dell'oggetto richiede una chiave Y, NON POSSEDUTA dall'utente

**Caso in cui l'apertura dell'oggetto richiede una chiave Y, NON POSSEDUTA dall'utente** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto. Tale notifica non è generata dal server ma direttamente lato utente.

Una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server che è richiesta la chiave Y per poter aprire l'oggetto.

Il cambio di stato dell'oggetto da UNVISITED a LOCKEDKEY viene notificato ad entrambi gli utenti insieme all'informazione relativa alla chiave necessaria per aprirlo.

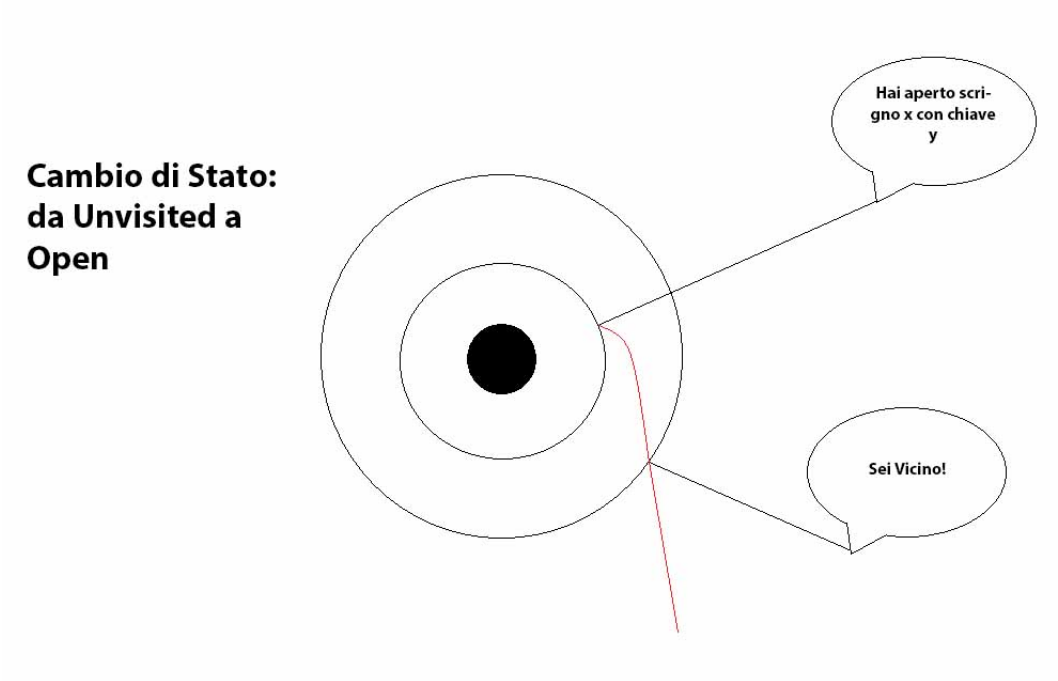


Figura 3.7: Caso in cui l'apertura dell'oggetto richiede una chiave Y, POSSEDUTA dall'utente

**Caso in cui l'apertura dell'oggetto richiede una chiave Y, POSSEDUTA dall'utente** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto. Tale notifica non è generata dal server ma direttamente lato utente.

Una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server che ha aperto con successo l'oggetto grazie all'ausilio della chiave Y.

Il cambio di stato dell'oggetto da UNVISITED a OPEN viene notificato ad entrambi gli utenti insieme all'informazione relativa relativa al suo contenuto (monete o chiavi).

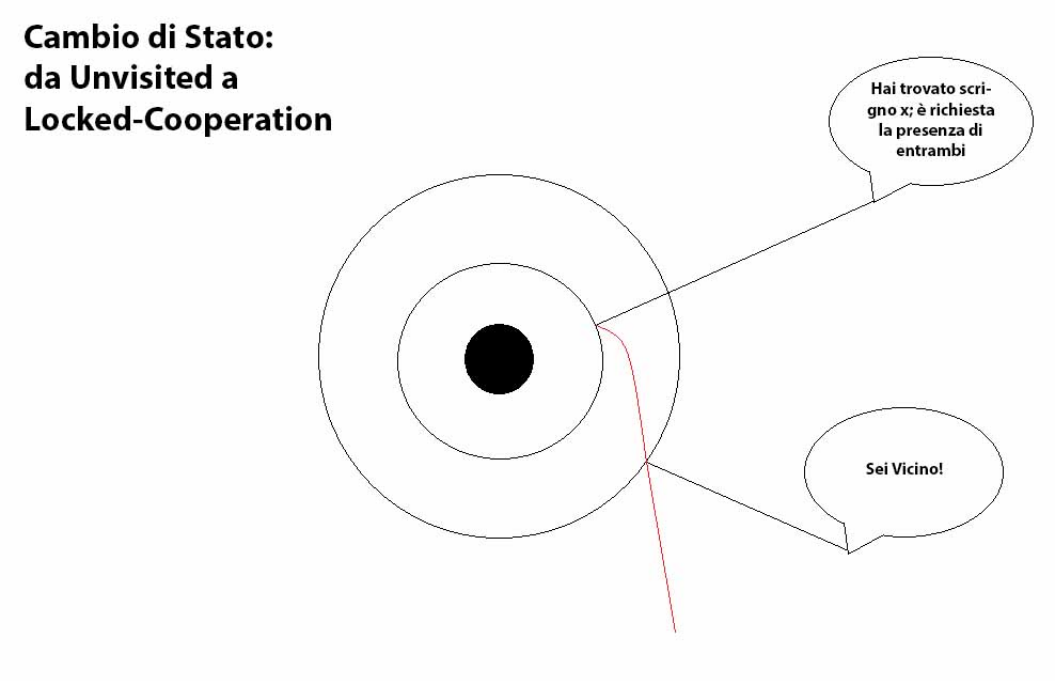


Figura 3.8: Caso in cui l'apertura dell'oggetto richiede cooperazione

**Caso in cui l'apertura dell'oggetto richiede cooperazione** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto. Tale notifica non è generata dal server ma direttamente lato utente.

Una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server che per aprire l'oggetto è necessaria la cooperazione tra i due utenti.

Il cambio di stato dell'oggetto da UNVISITED a LOCKEDCOOPERATION viene notificato ad entrambi gli utenti. All'utente che non si trova nel punto viene richiesto di esplicitare se è intenzionato a recarsi nel punto o meno. In base alla risposta l'utente che si trova nel punto saprà se attendere o meno.

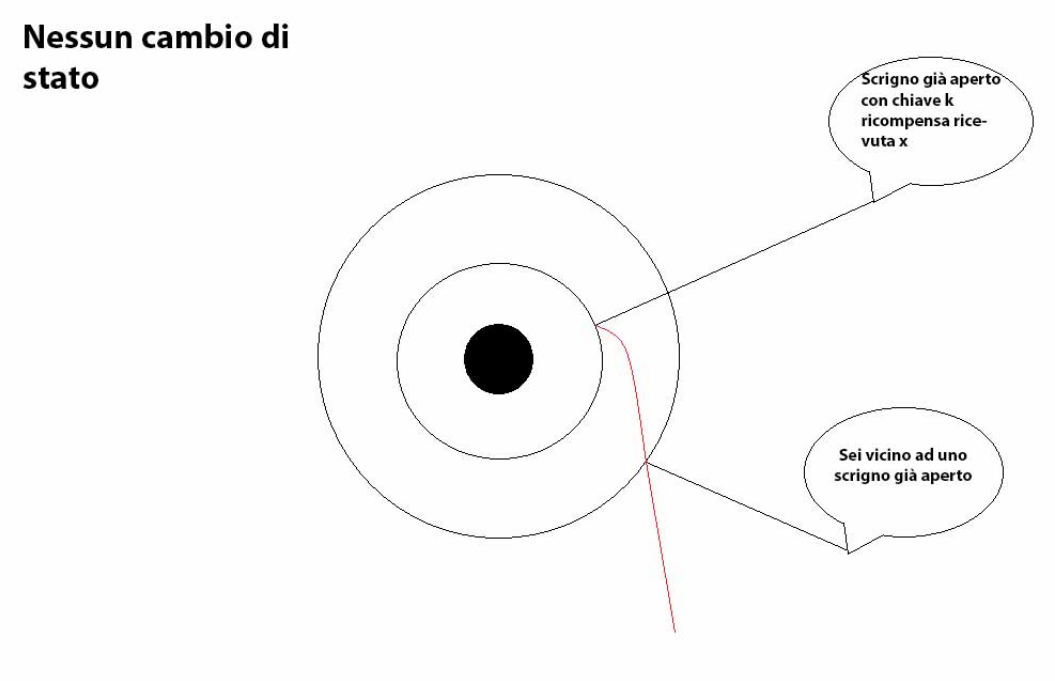


Figura 3.9: Caso in cui l'oggetto è stato già aperto

**Caso in cui l'oggetto è stato già aperto** Interazioni: in tale caso non avviene alcuna interazione tra utente e server: l'utente era già stato notificato dell'apertura dell'oggetto per cui possiede già tale informazione, quindi una volta giunto entro il raggio più ampio viene notificato che si trova in prossimità di un oggetto che è già stato aperto.

Una volta arrivato entro il raggio più ristretto viene notificato che l'oggetto a cui si trova di fronte è già stato aperto precedentemente con conseguente guadagno di X monete.

Non avviene alcun cambio di stato dell'oggetto.

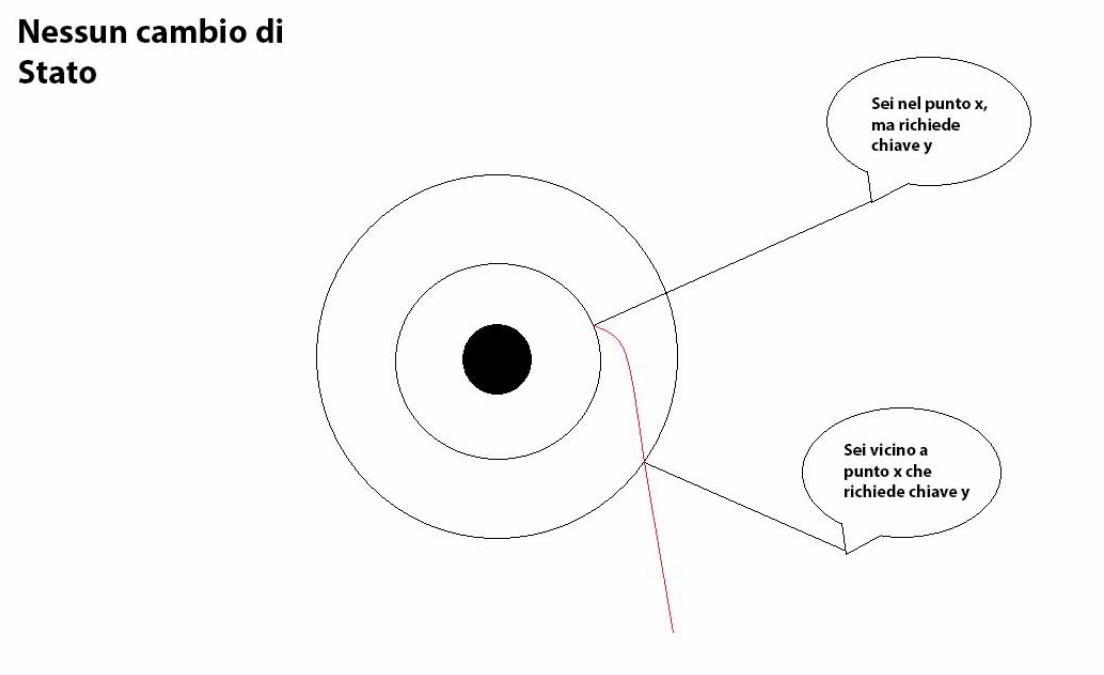


Figura 3.10: Caso in cui l'oggetto richiede una chiave Y, NON POSSEDUTA dall'utente, ed è già stato precedentemente visitato

**Caso in cui l'oggetto richiede una chiave Y, NON POSSEDUTA dall'utente, ed è già stato precedentemente visitato** Interazioni: anche in tale caso non avviene alcuna interazione tra utente e server. Infatti l'utente era già stato precedentemente notificato dell'avvenuta visita all'oggetto (da parte sua o del compagno) e dell'impossibilità nell'aprirlo se non attraverso la chiave Y.

Per cui in locale l'utente verrà prima notificato di essere in prossimità di un punto che richiede la chiave Y, successivamente verrà notificato (sempre in locale) di trovarsi di fronte ad un oggetto che attualmente non può aprire poichè non possiede la chiave Y.

Non avviene alcun cambio dui stato dell'oggetto.

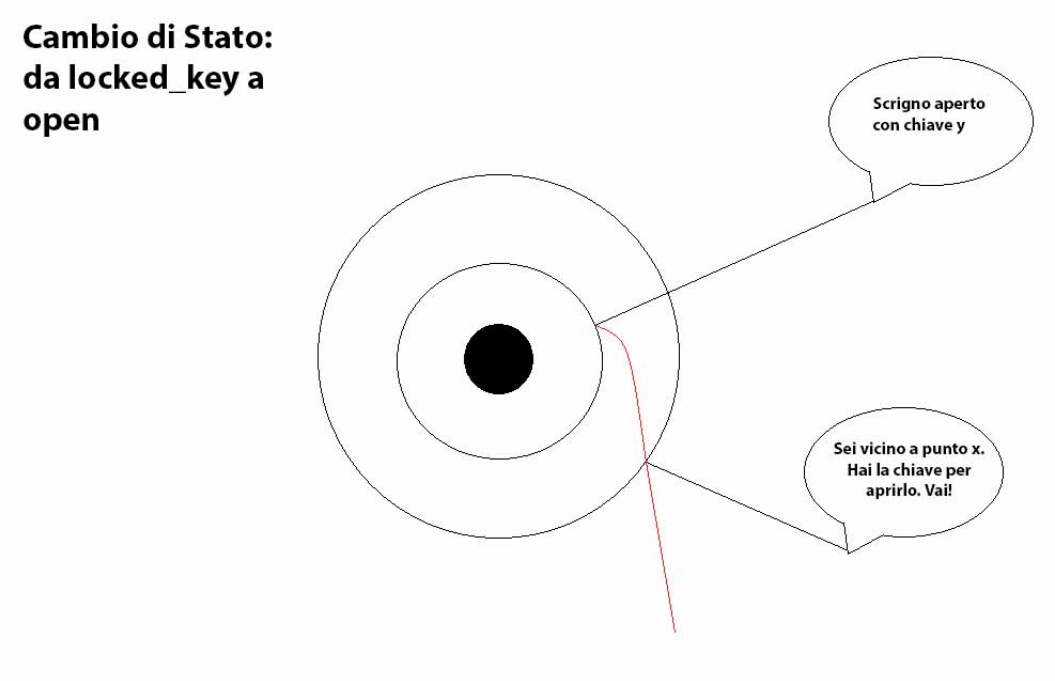


Figura 3.11: Caso in cui l'oggetto richiede una chiave Y, POSSEDUTA dall'utente, ed è già stato precedentemente visitato

**Caso in cui l'oggetto richiede una chiave Y, POSSEDUTA dall'utente, ed è già stato precedentemente visitato** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto e di possedere la chiave Y necessaria ad aprirlo.

Tale notifica non è generata dal server ma direttamente lato utente, poichè l'utente aveva già ricevuto la notifica della visita a tale oggetto (da parte sua o del compagno).

Una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server che ha aperto l'oggetto grazie all'ausilio della chiave Y, ottenendo il suo contenuto (monete o chiave). Il cambio di stato dell'oggetto da LOCKED-KEY a OPEN viene notificato ad entrambi gli utenti insieme all'informazione relativa relativa al suo contenuto (monete o chiavi).

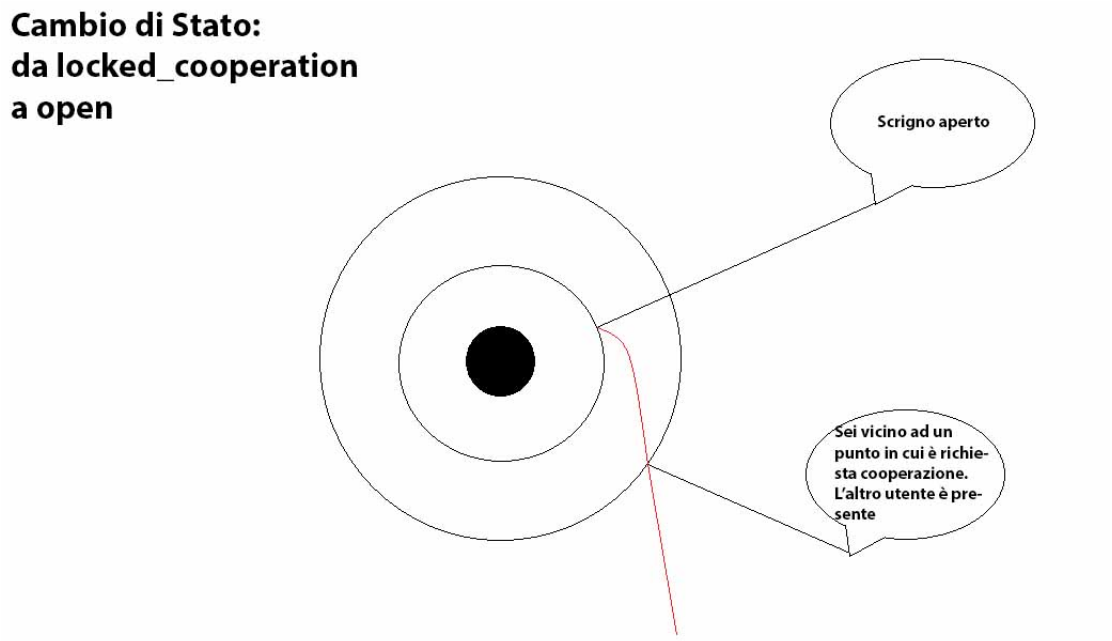


Figura 3.12: Caso in cui l'oggetto richiede cooperazione ed è già stato precedentemente visitato (l'altro utente è PRESENTE entro il raggio più ristretto dell'oggetto)

**Caso in cui l'oggetto richiede cooperazione ed è già stato precedentemente visitato (l'altro utente è PRESENTE entro il raggio più ristretto dell'oggetto)** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità dell'oggetto e che il proprio compagno è già in corrispondenza di tale oggetto.

Tale notifica proviene dal server, che possiede le informazioni circa la posizione dei due utenti ad ogni loro spostamento. Una volta giunto entro il raggio più ristretto, l'utente viene notificato dell'avvenuta apertura dell'oggetto insieme al proprio compagno.

Se è presente una chiave, verrà assegnata ad uno solo dei due utenti, mentre all'altro verrà notificata la sua scoperta. Il cambio di stato dell'oggetto da LOCKEDCOOPERATION a OPEN viene notificato ad entrambi gli utenti insieme all'informazione relativa relativa al suo contenuto monetario.

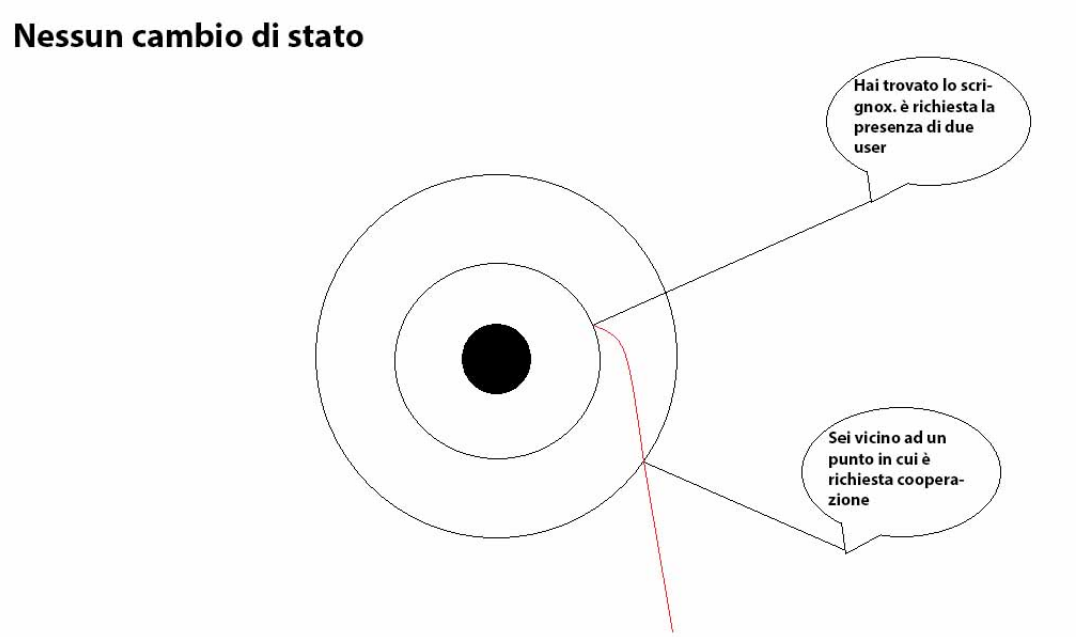


Figura 3.13: Caso in cui l'oggetto richiede cooperazione ed è già stato precedentemente visitato (l'altro utente NON è PRESENTE entro il raggio più ristretto dell'oggetto)

**Caso in cui l'oggetto richiede cooperazione ed è già stato precedentemente visitato (l'altro utente NON è PRESENTE entro il raggio più ristretto dell'oggetto)** Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato di essere in prossimità di un oggetto che richiede cooperazione. Tale notifica non è generata dal server ma direttamente lato utente.

Si ripresenta lo scenario della scoperta di un oggetto che richiede cooperazione, per cui una volta arrivato entro il raggio più ristretto, l'utente viene notificato dal server che per aprire l'oggetto è necessaria la cooperazione tra i due utenti.

All'utente che non si trova nel punto viene richiesto di esplicitare se è intenzionato a recarsi nel punto o meno. In base alla risposta l'utente che si trova nel punto saprà se attendere o meno. Non avviene alcun cambio dui stato dell'oggetto.

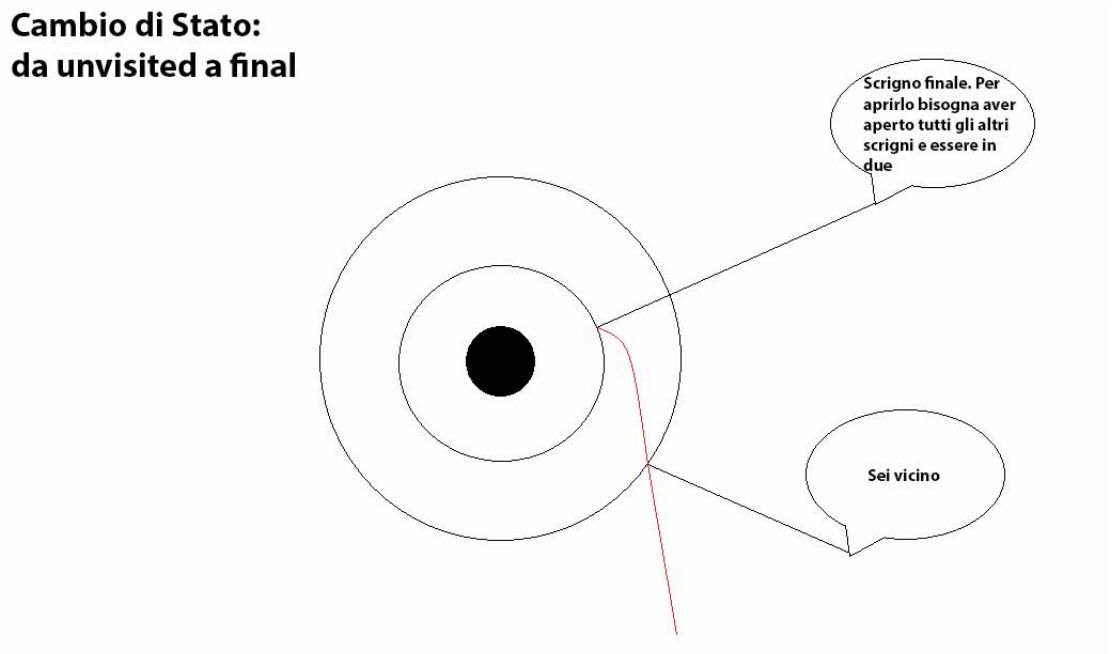


Figura 3.14: Caso in cui l'oggetto finale viene scoperto ma non può essere aperto

**Caso in cui l'oggetto finale viene scoperto ma non può essere aperto**
Interazioni: una volta arrivato entro il raggio più ampio, l'utente viene notificato in locale di essere in prossimità di un punto. Una volta arrivato entro il raggio più ristretto, avviene il cambio di stato da UNVISITED a FINAL che viene notificato agli utenti.

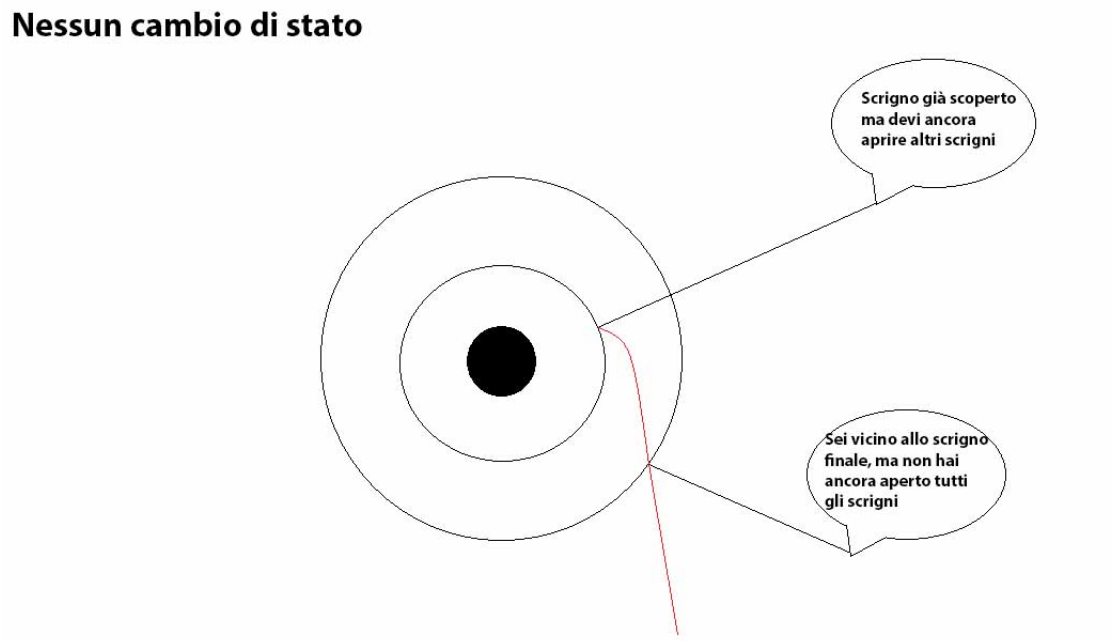


Figura 3.15: Caso in cui l'oggetto finale è già stato visitato ma non può essere aperto perchè non tutti gli altri oggetti sono già stati aperti

**Caso in cui l'oggetto finale è già stato visitato ma non può essere aperto perchè non tutti gli altri oggetti sono già stati aperti** Interazioni: non avvengono interazioni, siccome gli utenti possiedono già l'informazione relativa al numero di scrigni già aperti, per cui le notifiche riguardo la prossimità all'oggetto finale e il bisogno di aprire altri oggetti vengono generate in locale. Non avviene alcun cambio di stato.

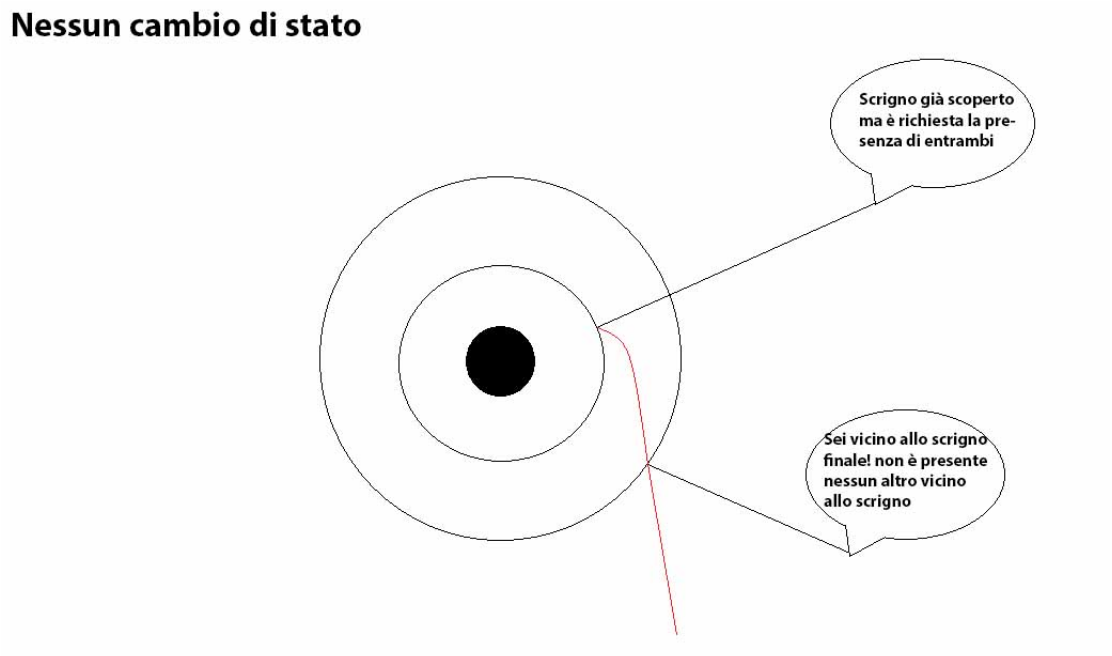


Figura 3.16: Caso in cui l'oggetto finale è già stato visitato ma non può essere aperto perchè il compagno non è presente (tutti gli altri oggetti sono già stati aperti)

**Caso in cui l'oggetto finale è già stato visitato ma non può essere aperto perchè il compagno non è presente (tutti gli altri oggetti sono già stati aperti)** Interazioni: in questo caso, siccome tutti gli altri oggetti sono già stati aperti, la situazione è analoga al caso in cui sia richiesta cooperazone per l'apertura dello scrigno (per cui avviene la richiesta di disponibilità all'altro utente).

Non avviene alcun cambio di stato.

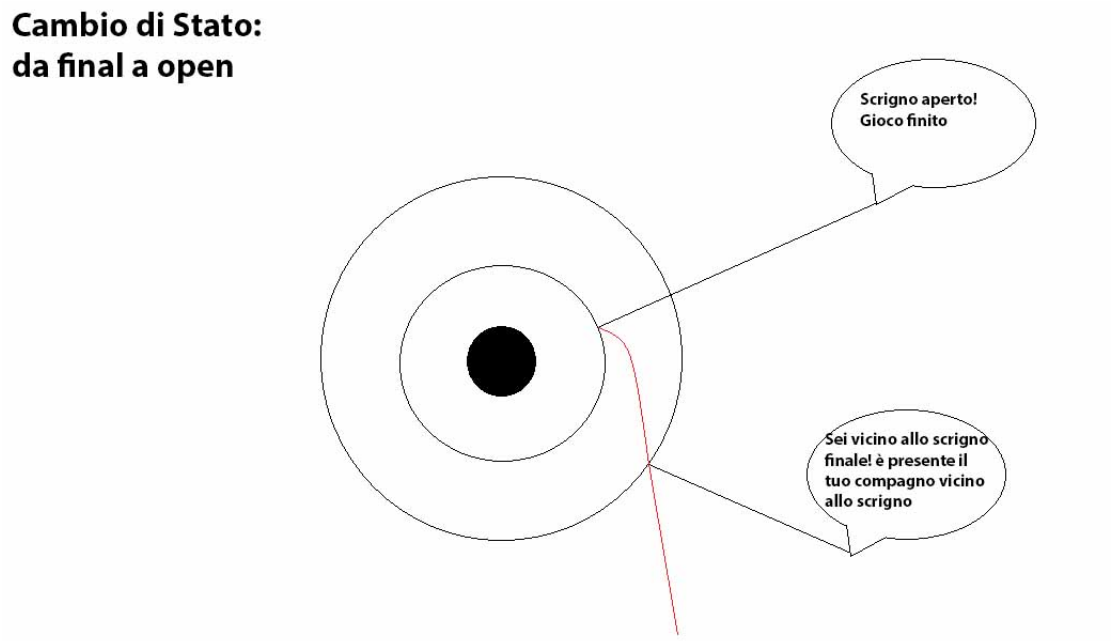


Figura 3.17: Caso in cui l'oggetto finale può essere aperto (è presente il compagno e tutti gli oggetti sono stati aperti)

**Caso in cui l'oggetto finale può essere aperto (è presente il compagno e tutti gli oggetti sono stati aperti)** Interazioni: in questo caso, siccome tutti gli altri oggetti sono già stati aperti e il compagno è presente nel punto dell'oggetto, la situazione è analoga a quella della cooperazione richiesta con presenza del compagno in loco.

Una volta raggiunto l'oggetto il gioco è concluso.

### 3.1.4 Lascia Notifica

L'utente ha la possibilità, nell'eventualità che ne abbia bisogno, di lasciare una notifica, di cui tiene traccia il server, in una zona dove è svolto il gioco. Il caso che verrà studiato è la presenza di un ladro virtuale in una certa zona della mappa.

Nel caso in cui uno degli utenti incappi nel ladro può lasciare un segnale in quella zona in modo tale che se l'altro utente si avvicini a quella zona il server gli notifichi il pericolo.

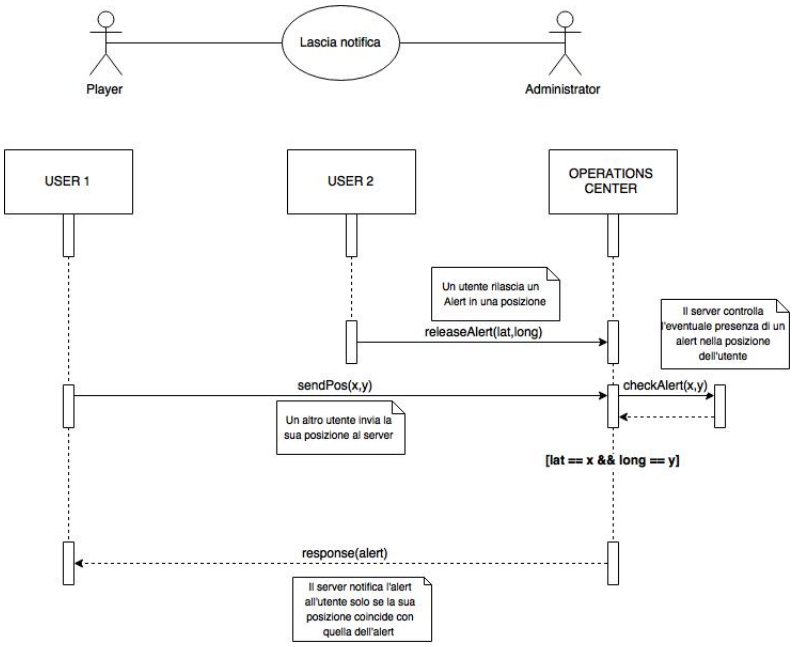


Figura 3.18: Diagramma di sequenza relativo al rilascio di una notifica

### 3.1.5 Modifica Contenuto

La centrale operativa può modificare ed in particolare diminuire il valore della ricompensa contenuta all'interno degli oggetti.

Nel nostro caso di studio verrà gestita la diminuzione delle ricompense con un certo timer, ovvero dopo intervalli di tempo prefissati verrà diminuito il valore del contenuto degli oggetti.

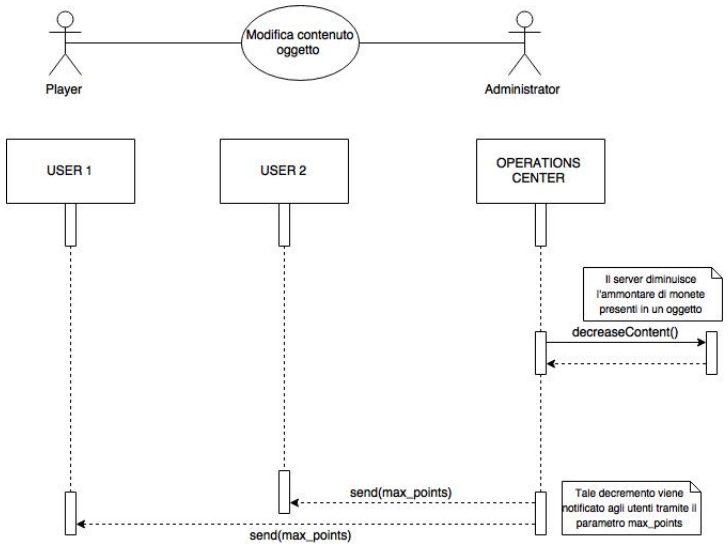


Figura 3.19: Diagramma di sequenza relativo alla modifica del contenuto degli oggetti

### 3.1.6 Ruba Ricompensa

L'attore ladro nel momento in cui l'utente entra nel suo raggio d'azione ha la possibilità di rubare una quantità prestabilita di ricompensa, e questo chiaramente solo se gli utenti hanno già aperto qualche scrigno .

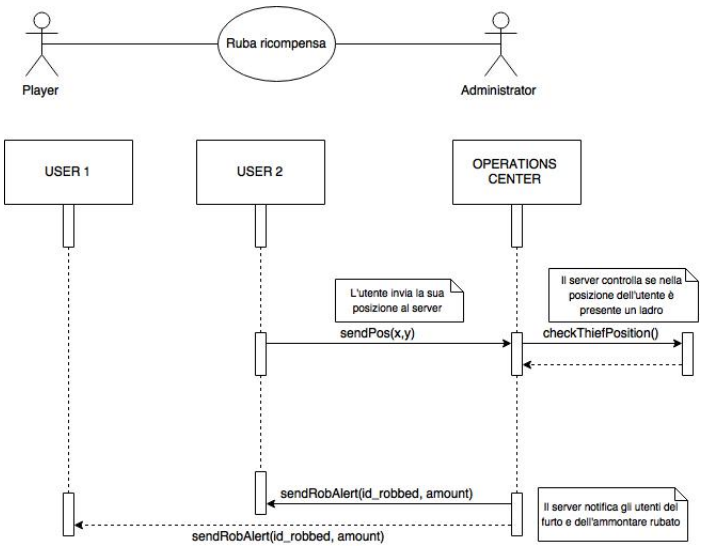


Figura 3.20: Diagramma di sequenza relativo all'azione del ladro

## 3.2 Dominio Applicativo

### 3.2.1 Lato Server

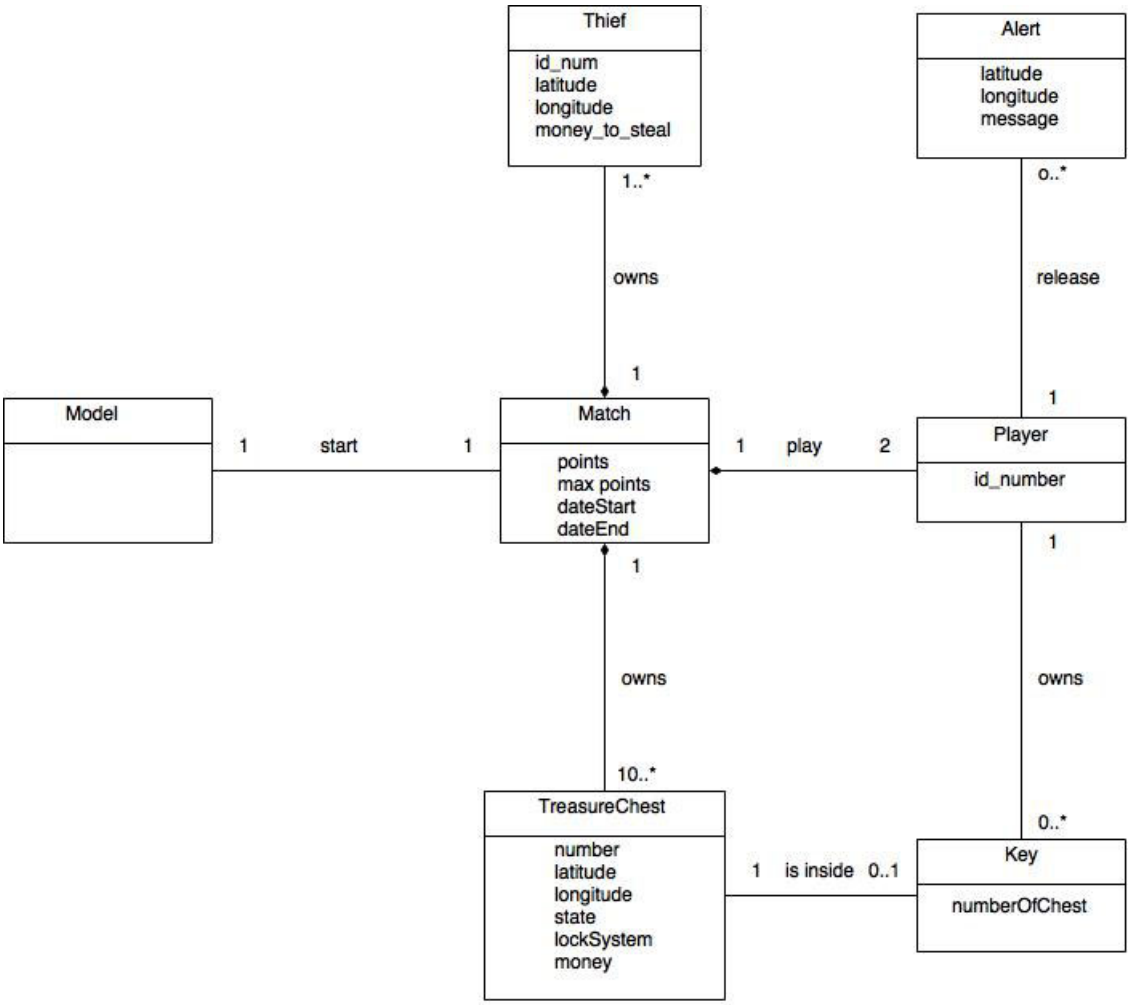


Figura 3.21: Diagramma delle classi del dominio applicativo lato server

**Model** E' il container del dominio applicativo, ovvero è la classe da cui verrà inizializzata la partita e gestito tutto il dominio.

**Match** E' la modellazione della partita. Al suo interno si tiene traccia del punteggio attuale, che all'inizio è zero, e del massimo punteggio raggiungibile che viene aggiornato dopo i furti e dopo le diminuzioni da parte della centrale operativa. In più tiene traccia del giorno di inizio e di fine.

**TreasureChest** E' la modellazione dello scrigno al cui interno si tiene traccia del numero dello scrigno, della posizione, del suo stato, del sistema di blocco che influenza poi il cambio di stato e l'ammontare di monete al suo interno. In più può contenere una chiave di cui tiene una referenza.

**Key** E' la modellazione della chiave e ha come unico campo il numero dello scrigno che può aprire.

**Player** E' la modellazione dei giocatori e tiene traccia dell'id, che lo identifica in tutti i suoi messaggi mandati, degli alert che rilascia ,delle chiavi che possiede e della sua posizione.

**Alert** E' la modellazione della notifica che un giocatore può lasciare in un certo punto della mappa. E' caratterizzato dalla posizione e dal messaggio lasciato dall'utente.

**Thief** E' la modellazione dell'entità nemica. Anche questa viene situata in un punto sulla mappa, per cui si tiene traccia della sua posizione, è identificata dal suo id e ha una quantità che può rubare quando "assale" un giocatore.

## 3.3 Lato Client

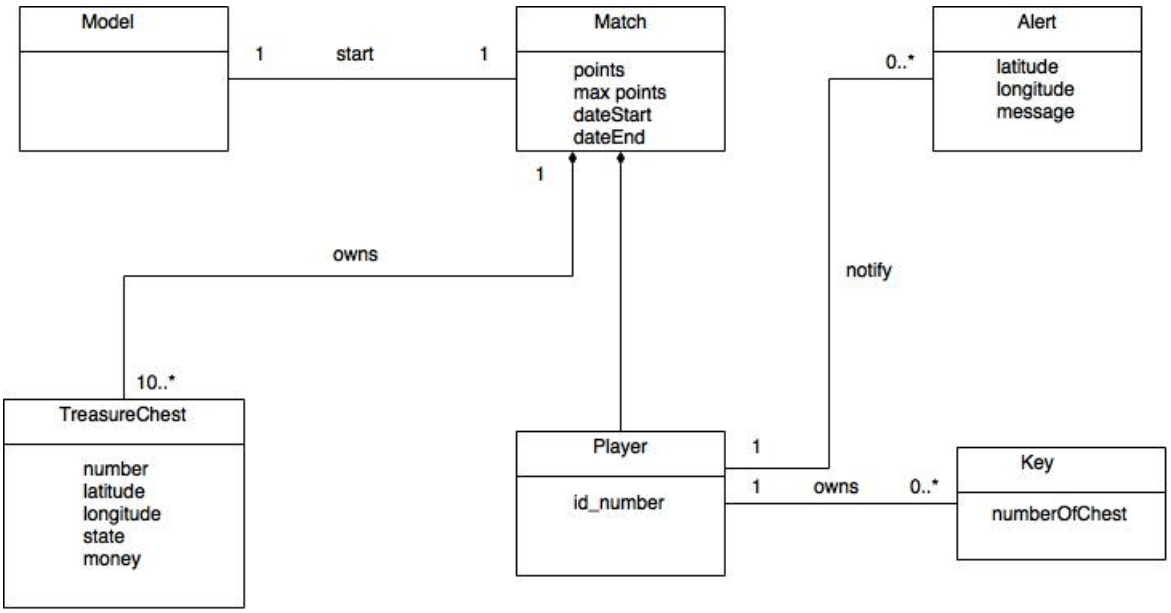


Figura 3.22: Diagramma delle classi del dominio applicativo lato client

**TreasureChest** E' l'unica modellazione che ha delle differenze dal lato server in quanto non tiene traccia del sistema di blocco. In più quando il gioco viene creato sono inizializzate solamente la posizione e lo stato, mentre le altre variabili vengono inizializzate solo nelle successive modifiche.

## 3.4 Dominio applicativo dei messaggi scambiati e tecniche di comunicazione e interazione

Di seguito vengono elencati tutti i tipi di messaggi scambiati tra server e utente, specificando il tipo di informazione inviata e la tecnica di invio per ognuno.

### 3.4.1 Inizializzazione

La comunicazione ha inizio tramite un "handshaking" che da il via all'invio dei dati relativi al mondo condiviso da server a user. L'utente effettua una **request**, composta da un semplice messaggio testuale destinato al server. Il server, dopo aver assegnato un ID all'utente, reagisce con una **response** contenente i seguenti dati:

- Tipo messaggio
- ID Utente
- Monete totali disponibili
- Lista contenente i TreasureChest in questa forma:

**TreasureChest**
longitude
latitude

Tali oggetti verranno poi mappati in oggetti di questo tipo, lasciando inizialmente vuoti i campi number e money.

**TreasureChest**
longitude
latitude
state
number

money

In questo modo l'utente ha ricevuto le informazioni necessarie ad iniziare il gioco, ovvero la posizione GPS degli oggetti (senza ancora conoscerne il contenuto), il totale di monete che può ottenere insieme al compagno e il proprio ID.

### 3.4.2 Invio posizione (da Client a Server)

L'utente comunica la propria posizione GPS ad ogni suo spostamento, inviando i seguenti dati:

- Tipo messaggio
- ID Utente
- Latitudine
- Longitudine

### 3.4.3 Cambio di stato dell'oggetto (da S a C)

Quando l'utente arriva in un punto d'interesse, il server invia informazioni riguardanti il cambiamento di stato del relativo oggetto (se è avvenuto un cambio di stato). Si distinguono due casi:

**L'oggetto viene aperto** In questo caso vengono inviati i seguenti dati:

- Tipo messaggio (object open)
- ID Utente
- Chiave contenuta (campo a zero se non contiene chiavi)
- TreasureChest in questa forma:

**TreasureChest**
longitude
latitude
state
number
money

In questo modo l'utente riceve tutte le informazioni contenute nell'oggetto.

**L'oggetto non viene aperto** In questo caso vengono inviati i seguenti dati:

- Tipo messaggio (object not open)
- ID Utente
- Chiave contenuta (campo a zero se non contiene chiavi)
- TreasureChest in questa forma:

**TreasureChest**
longitude
latitude
state
number

In questo modo l'utente non riceve le informazioni relative al contenuto dell'oggetto, ma solo sul suo stato e sul numero dell'oggetto.

### 3.4.4 Invio di conferma/rifiuto cooperazione (da C a S):

Messaggio generato dal cliente che riceve la richiesta di cooperazione. E' un semplice messaggio testuale con cui l'utente può accettare/rifiutare la richiesta di cooperazione.

**Cooperazione accettata**

- Tipo messaggio
- ID Utente
- Messaggio accetto

**Cooperazione accettata**

- Tipo messaggio
- ID Utente
- Messaggio rifiuto

### 3.4.5 Notifica di conferma/rifiuto cooperazione (da C a S):

Il server, una volta ricevuta la conferma/rifiuto di cooperazione da parte di U2, deve notificarla a U1. Viene "inoltrato" lo stesso messaggio precedentemente ricevuto.

**Cooperazione accettata**

- Tipo messaggio
- ID Utente
- Messaggio accetta cooperazione

U1 resterà nel punto in attesa del compagno.

**Cooperazione non accettata**

- Tipo messaggio
- ID Utente
- Messaggio rifiuta cooperazione

U1 sarà libero di proseguire, poiché il compagno non è intenzionato a recarsi lì.

### 3.4.6 Notifica di allerta (da C a S):

L'utente può segnalare degli Alert messages con cui aiutare il compagno a non incappare in zone pericolose. Può lasciare un messaggio che verrà visualizzato dall'altro utente quando entrerà nella zona in cui il messaggio è stato rilasciato.

Il messaggio ha la seguente struttura:

- Tipo messaggio
- ID Utente
- Alert in questa forma:

**Alert**
longitude
latitude
message

### 3.4.7 Notifica di allerta (da S a C):

Quando un utente raggiunge una zona in cui il compagno (o anche se stesso) ha lasciato un Alert message, il server gli notifica il messaggio in questo modo:

- Tipo messaggio
- ID Utente
- Alert in questa forma:

**Alert**
longitude
latitude
message

Si noti che in base all'ID ricevuto l'utente saprà se l'Alert message era stato rilasciato da lui stesso o dal compagno.

### 3.4.8 Messaggio Ladro (da S a C):

Quando un utente giunge in un'area in cui è presente un ladro, il server notifica l'avvenuta rapina, tramite un messaggio di questo tipo:

- Tipo messaggio
- ID Utente
- Quantità derubata

Si noti che in base all'ID ricevuto l'utente saprà se la vittima è stata lui stesso o il compagno. Entrambi gli utenti sapranno quale importo è stato derubato per cui l'importo verrà scalato dal totale delle monete fino a quel momento ottenute e dal totale di monete ottenibili.

### 3.4.9 Messaggio diminuzione ricompensa (da S a C):

Quando il server diminuisce la quantità di monete presenti negli oggetti notifica l'avvenimento inviando il nuovo importo totale disponibile:

- Tipo messaggio
- Nuovo importo massimo

# Capitolo 4

# Progettazione

## 4.1 Compito svolto all'interno del progetto

Nella progettazione di questo sistema io mi sono occupato in particolare dello strato che si occupa di fornire al nostro programma funzionalità che permettano la cooperazione e la collaborazione. In primis mettere a disposizione API al livello applicativo a supporto della cooperazione. In più creare uno strato nella sua interezza che sia in grado di gestire il dominio condiviso tra i vari client e il server e allo stesso tempo capace di supportare le interazioni tra utente e oggetto e tra più utenti. Infine, in quanto la nostra applicazione rientra tra i software location aware, parte del mio lavoro sarà anche dedita a sviluppare un sistema di localizzazione che permetta di gestire il nostro sistema come l'abbiamo ideato. Lo scopo del mio lavoro è quello quindi di creare una serie di funzionalità, in grado di fornire cooperatività, resistenti alle problematiche sopra accennate.

## 4.2 Problematiche principali

La prima fase della mia progettazione sarà mirata a identificare le problematiche principali verso le quali lo strato di cui mi occupo può incorrere.

Il problema principale della nostra applicazione, in quanto sistema distribuito, è certamente la coerenza dei dati. Nel nostro software Server e Client avranno un dominio condiviso o ancora meglio replicato. Infatti le due parti del nostro sistema necessitano di avere simultaneamente lo stesso dominio, con qualche differenza dal punto di vista dei dettagli sulle varie parti. Ad ogni modifica del sistema distribuito, tutti coloro che condividono quel dominio devono essere aggiornati.

L'esempio più facile da comprendere è il cambio di stato di un oggetto. Su questo verte tutto il sistema. Nel momento in cui un oggetto viene modificato, per una qualsiasi ragione, ogni altro utente connesso al sistema deve essere informato di questo cambiamento. Se non viene informato di questo può accadere che in prossimità dell'oggetto abbia informazioni diverse rispetto a quele ipoteticamente condivise da tutti e quindi che il sistema vada contro alla coerenza che è la condizione necessaria per la nostra applicazione.

Un'altra problematica fondamentale per quanto riguarda lo strato di cui mi occupo è la localizzazione dell'utente. Per rendere possibile tutte le interazioni tra utente e oggetto, ma anche tra utente e utente, è necessario che il server sia a conoscenza, con frequenze abbastanza elevate, della posizione dello user. Dando per assodato che stiamo discutendo di un'applicazione location aware è necessario che il server sia consapevole della posizione dell'utente.

Riprendiamo come esempio quello di un utente che si appropinqua ad un oggetto. Se il sistema non è a conoscenza con frequenze elevate della sua posizione può accadere che l'utente passi troppo velocemente in prossimità dell'oggetto e che il sistema non se ne renda conto. O ancora se il posizionamento non è preciso può capitare che un utente vicino ad un oggetto non sia però visto nella sua reale posizione e quindi non avvengano poi interazioni.

L'ultima problematica fondamentale per il nostro sistema è la gestione delle disconnessioni. Essendo un sistema distribuito, in ogni momento il server, ma anche i client, devono essere consapevoli della loro connessione. In particolare per quanto riguarda il livello di cooperazione il server per la distribuzione delle informazioni deve essere cosciente dello stato dei client, per poi decidere come operare nel momento della riconnessione.

Un esempio che mette bene in luce questa problematica è se un utente trova un oggetto che può essere aperto dalla chiave posseduta da un utente che è offline. Il server deve essere sempre e comunque in grado di rendersi conto della disconnessione dell'utente, per poi in seguito riaggiornare l'utente in base ai cambiamenti degli stati e rimandare le notifiche ritenute importanti.

## 4.3 Architettura logica generale del sistema

Il sistema sarà basato su un'architettura Client-Server, in cui il Server sarà una piattaforma che fungerà da gestore generale delle informazioni di gioco e delle interazioni, mentre il client sarà costituito dal sottosistema composto da uno smartphone e da un paio di smart-glasses per ogni utente.

Lo smartphone fungerà da localizzatore GPS e si occuperà della comunicazione col server.

Gli smart-glasses fungeranno da semplice interfaccia per l'utente su cui saranno visualizzate tutte le informazioni necessarie allo svolgimento del gioco.

L'architettura generale del sistema è perciò la seguente:

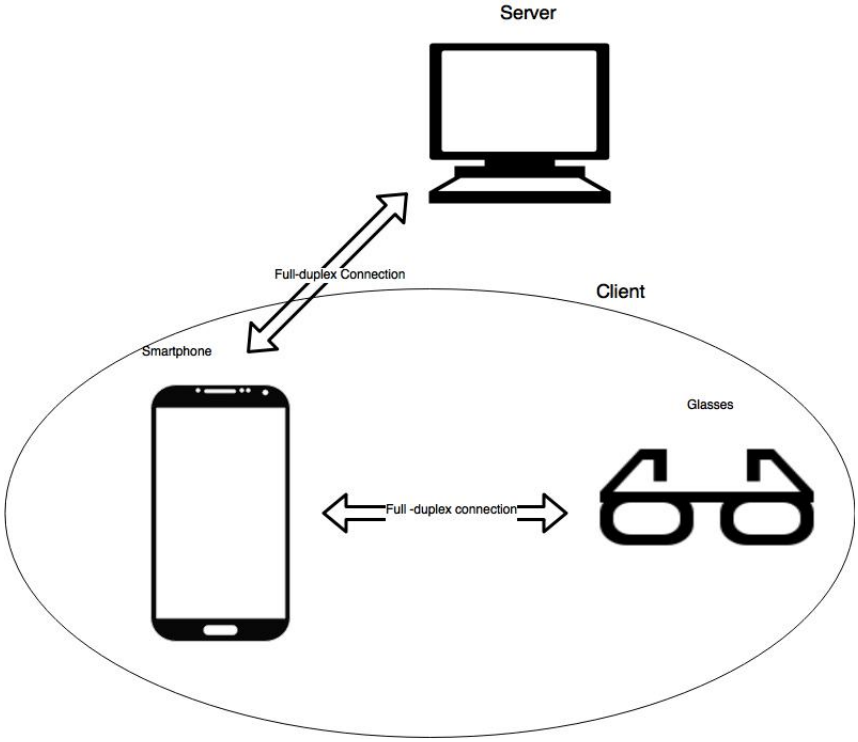


Figura 4.1: Architettura generale del sistema

I software presenti su Server e Client saranno strutturati su tre livelli, dall'alto al basso:

- User Interface Layer
- Cooperation Layer
- Communication Layer

Ognuno di tali strati fornirà API ai livelli superiori. In seguito vengono presentati i due schemi (lato Server e lato Client) che mostrano le principali API fornite dai livelli di Comunicazione e Cooperazione.

### 4.3.1 Architettura logica del sistema lato Server

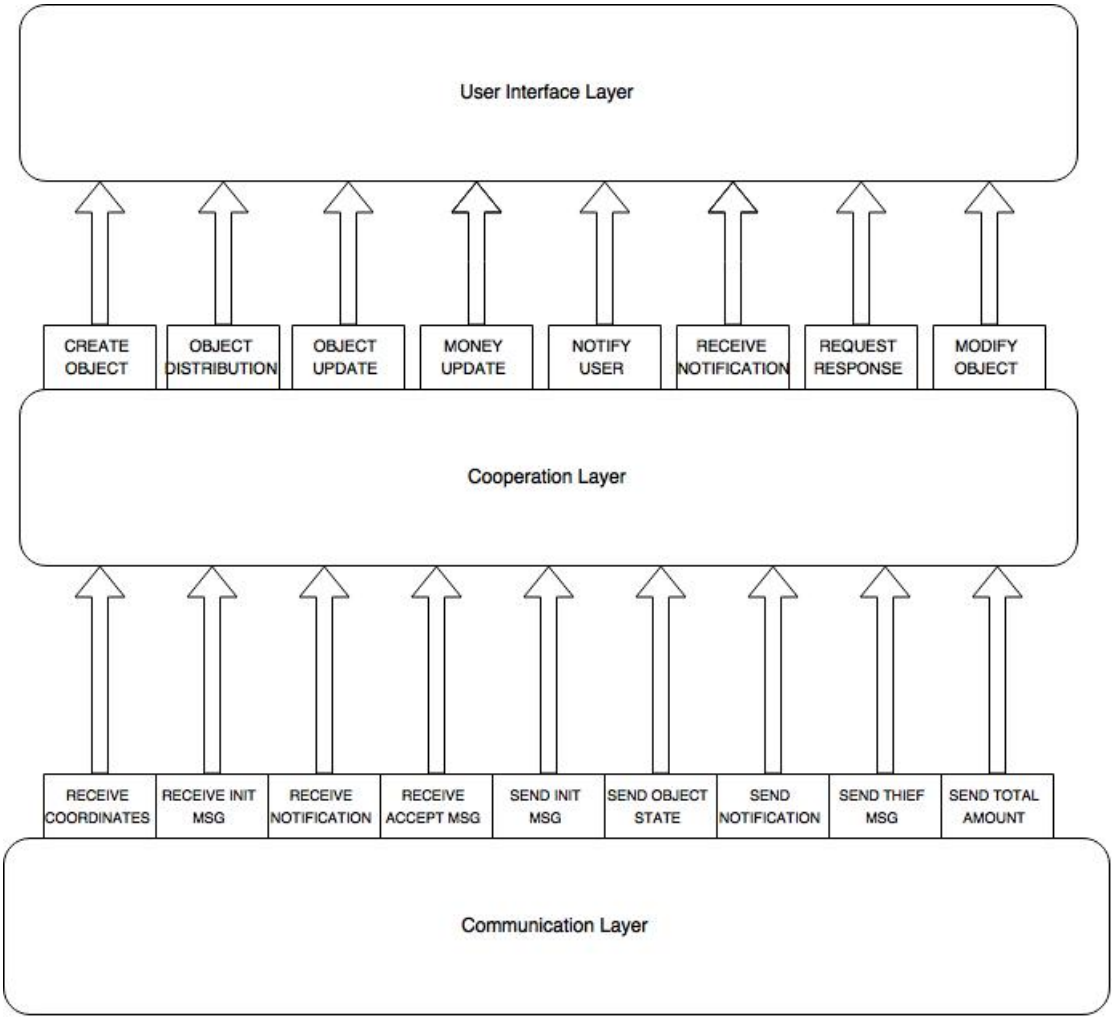


Figura 4.2: Architettura generale del sistema

### 4.3.2 Architettura logica del sistema lato Client

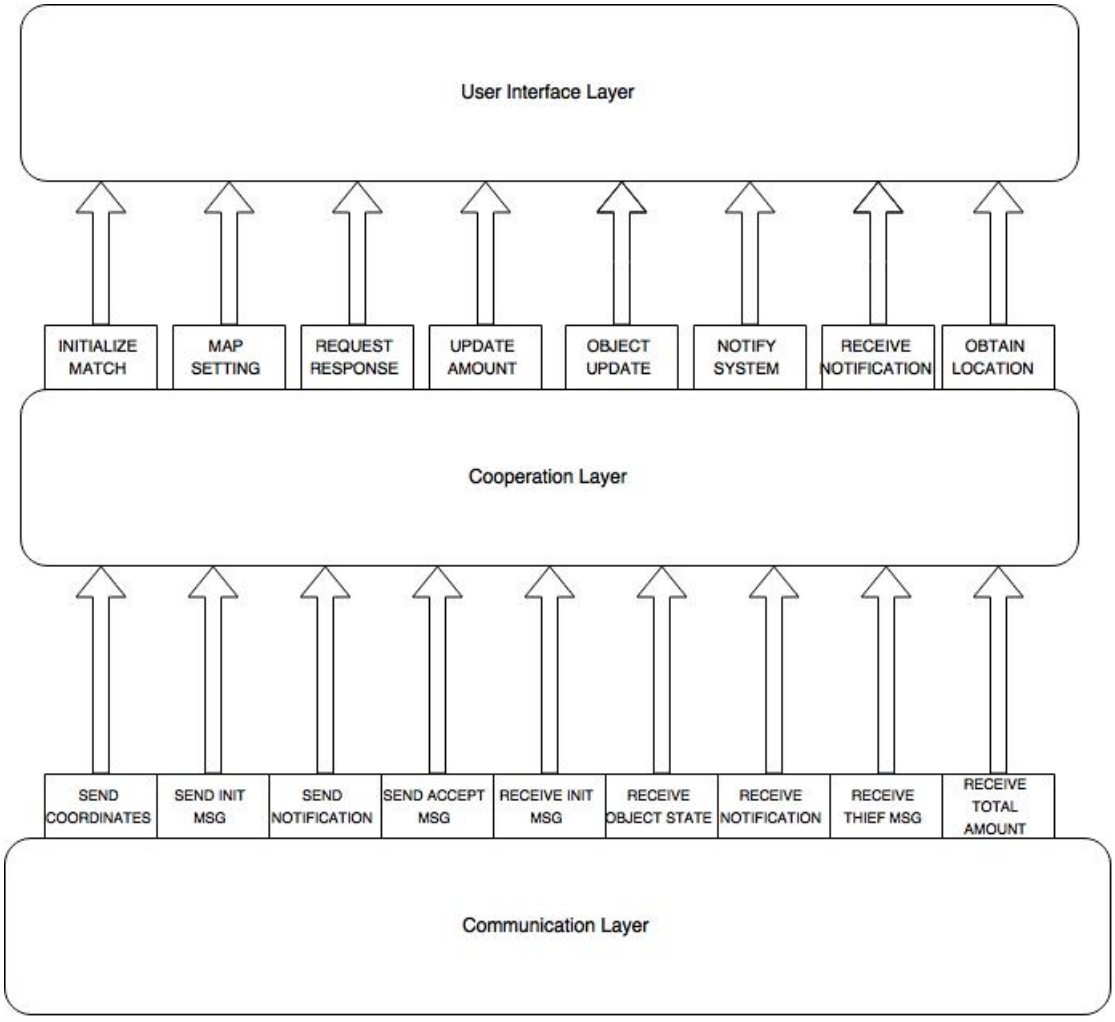


Figura 4.3: Architettura generale del sistema

## 4.4 API fornite allo strato superiore

### 4.4.1 Lato Server

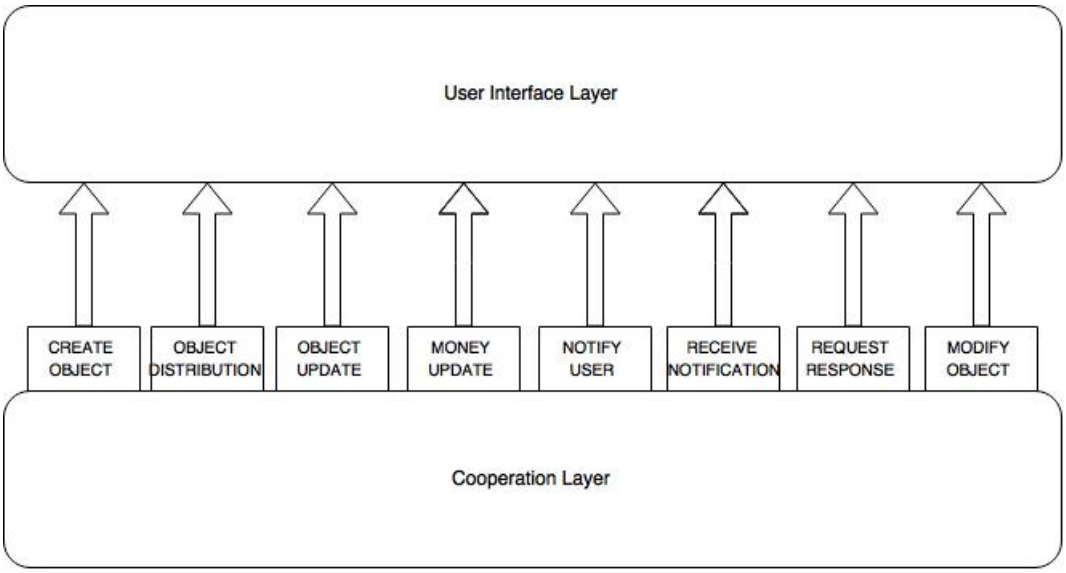


Figura 4.4: Architettura logica lato server

Lo strato di collaborazione fornisce API allo strato di User Interface a supporto della cooperazione e delle interazioni tra i vari utenti

**CREATE OBJECT:** questa chiamata viene effettuata nella fase di inizializzazione in cui il server crea il gioco e prima ancora crea gli oggetti che sono l'entità primaria della nostra applicazione

**OBJECT DISTRIBUTION:** questa chiamata viene effettuata dopo che è stata ricevuta una richiesta di partecipazione al gioco e viene utilizzata per mandare le informazioni iniziali indispensabili per l'inizializzazione

**OBJECT UPDATE:** questa chiamata viene effettuata tutte le volte che è necessario cambiare lo stato di un oggetto

**MONEY UPDATE:** questa chiamata viene effettuata ogni qual volta va aggiornato o il quantitativo massimo di tesori trovabili o il quantitativo trovato

**NOTIFY USER:** questa chiamata viene effettuata dopo aver ricevuto una notifica dall'utente nel momento in cui l'altro utente deve essere notificato

**RECEIVE NOTIFICATION:** questa chiamata viene effettuata nel momento in cui il server riceve una notifica e la registra per poi notificarla all'altro utente

**REQUEST RESPONSE:** questa chiamata viene effettuata quando il server ha bisogno di avere una risposta da un utente, per esempio nel caso di richiesta collaborazione

**MODIFY OBJECT:** questa chiamata viene effettuata nel momento in cui il server modifica il contenuto degli oggetti

### 4.4.2 Lato Client

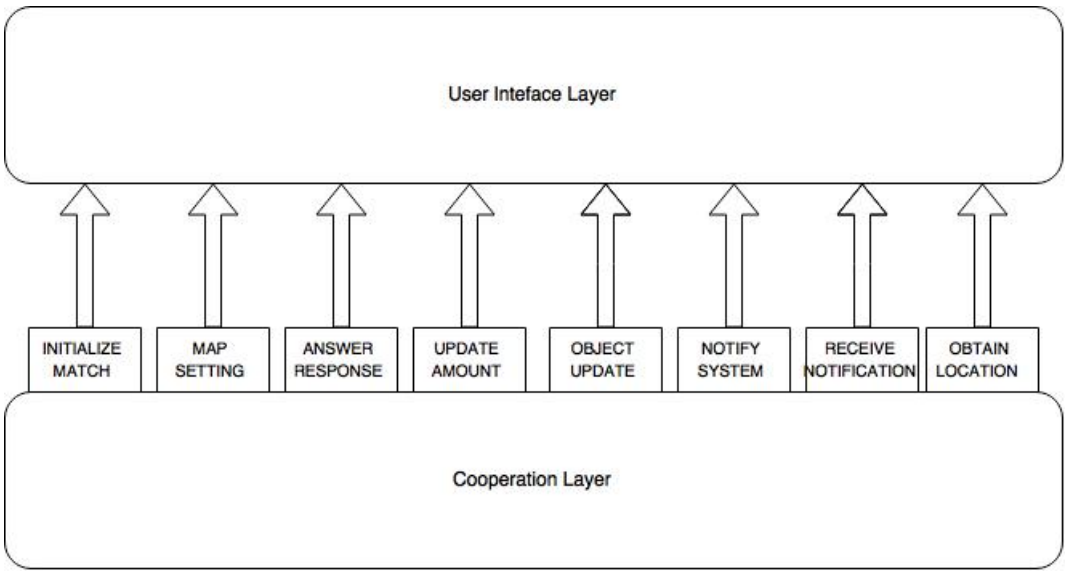


Figura 4.5: Architettura logica lato client

**INITIALIZE MATCH:** questa chiamata viene effettuata dall'utente nel momento in cui richiede di far parte della sessione di gioco per poi inizializzare tutti gli oggetti e la mappa inizializzando così la partita

**MAP SETTING:** questa chiamata viene effettuata dall'utente per inizializzare la mappa e tutti gli scrigni

**ANSWER RESPONSE:** questa chiamata viene effettuata nel momento in cui il server ha fatto una richiesta all'utente

**UPDATE AMOUNT:** questa chiamata viene effettuata quando il server manda l'aggiornamento dell'ammontare di tesori o trovati o trovabili

**OBJECT UPDATE:** questa chiamata viene effettuata quando avvenuta un'interazione con un oggetto il server manda l'aggiornamento del suo stato.

**NOTIFY SYSTEM:** questa chiamata viene effettuata nel momento in cui l'utente vuole rilasciare una notifica nel sistema

**RECEIVE NOTIFICATION:** questa chiamata viene effettuata quando il server manda una notifica all'utente

**OBTAIN LOCATION:** questa chiamata viene effettuata ogni qual volta l'utente ha bisogno di conoscere la sua posizione

## 4.5 Moduli principali lato Client

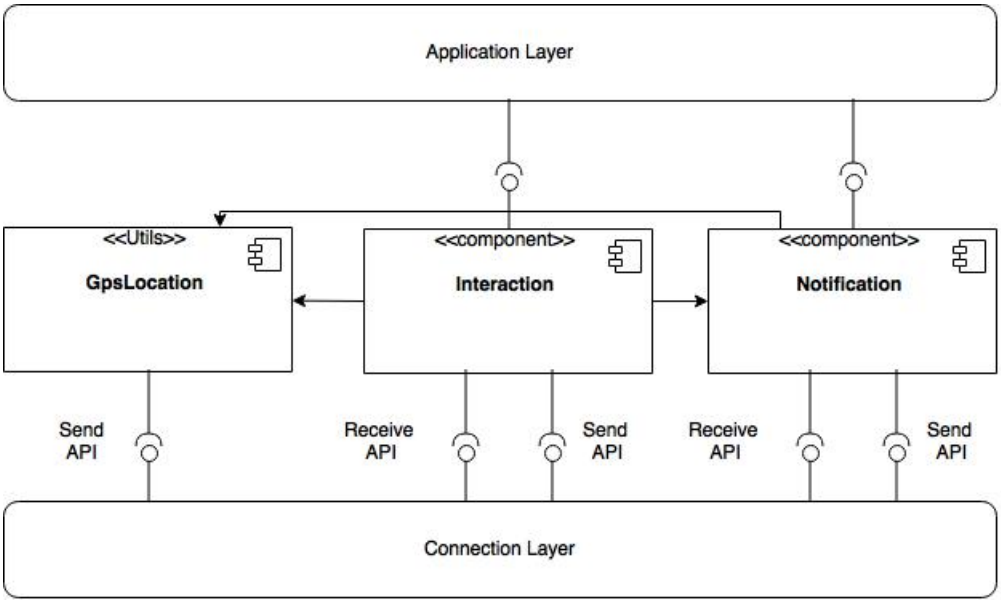


Figura 4.6: Diagramma dei componenti lato client

Per quanto riguarda lo strato di collaborazione lato Client possiamo trovare tre macro moduli, ovvero GPSLocation, Interaction e Notification che ora andrò a spiegare.

### 4.5.1 GPSLocation

Si tratta del componente che si occupa della localizzazione e della notificazione della latitudine e della longitudine al Server.

A questo componente viene richiesto ad intervalli di tempo regolari di inviare la sua posizione. GPSLocation è quindi un componente che ha un suo flusso di controllo continuo che a step prefissati invia la posizione.

Dovendo affrontare il problema della localizzazione continua, ovvero che il Server necessita di sapere la posizione degli utenti con una buona precisione in qualsiasi momento, questo componente invia con una periodicità elevata i messaggi.

Il sistema verrà aggiornato della posizione ogni volta che l'utente fa un certo spostamento o ogni volta che scade un timer in questo modo si previene in buona parte il problema della localizzazione. Chiaramente questo ha senso dando per certo che l'infrastruttura di connessione fornisca un servizio sicuro, ovvero che tutti i messaggi arrivino e siano consegnati in ordine.

Tutti i messaggi inviati dalla localizzazione GPS vengono simultaneamente consegnati al Server e al modulo di interazione come mostrato dal diagramma di sequenza riportato sotto:

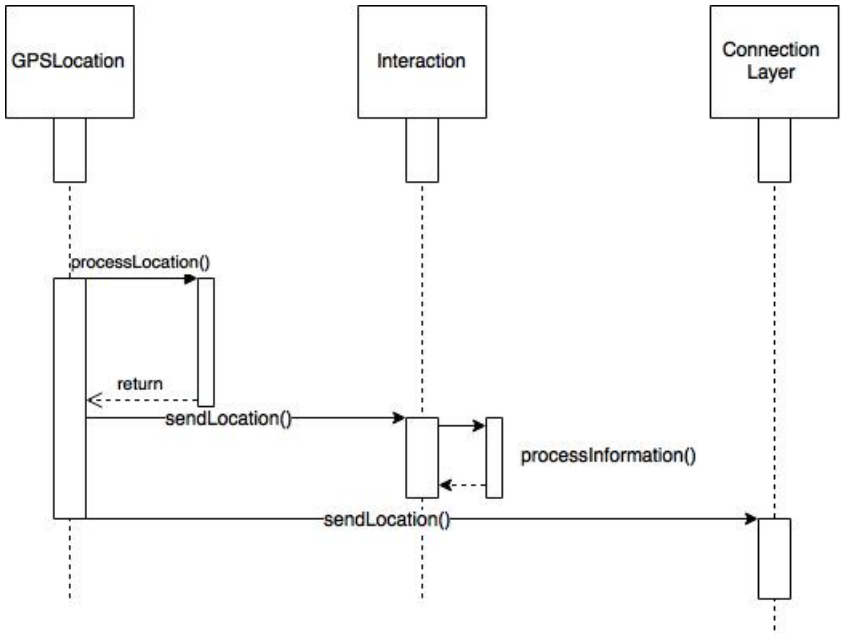


Figura 4.7: Diagramma di sequenza GPSLocation

### 4.5.2 Interaction

Si tratta del componente a cui è affidata la gestione di tutte le interazioni. In particolare è il componente che deve processare le informazioni ricevute dallo stato di connessione relative agli oggetti e agire di conseguenza. Al suo interno si distinguono due moduli: Update Model e Object Interaction

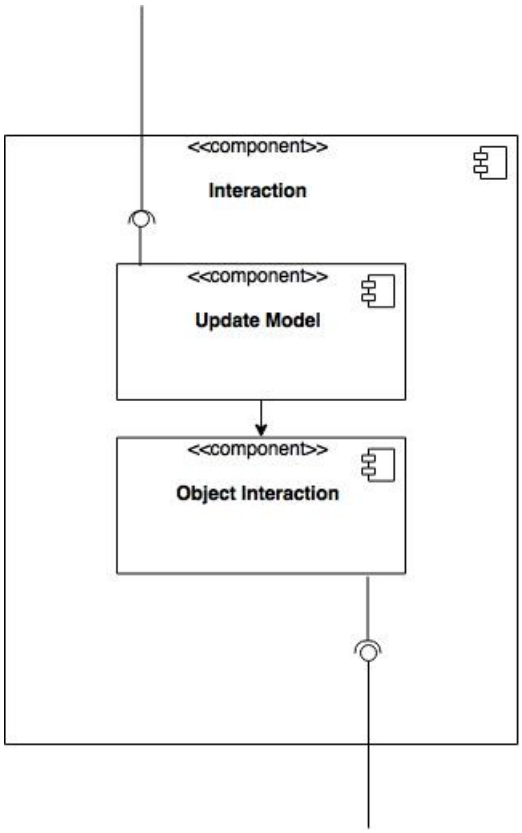


Figura 4.8: Diagramma dei componenti di Interaction

**Object Interaction** è il componente che si occupa della gestione delle interazioni tra utente e oggetto in particolare quelle gestite in locale. Ovvero ogni volta che riceve informazioni riguardanti la propria posizione, conoscendo a priori la posizione di tutti gli scrigni effettua un controllo di prossimità nei confronti dello scrigno e nel caso sia entro le distanze prestabilite dall'oggetto manda allo strato di User Interface questa informazione. In più questo componente al compito di ricevere tutti i cambiamenti degli oggetti dal server e gestire la comunicazione con Update Model. I diagrammi di sequenza sottostanti presentano queste dinamiche:

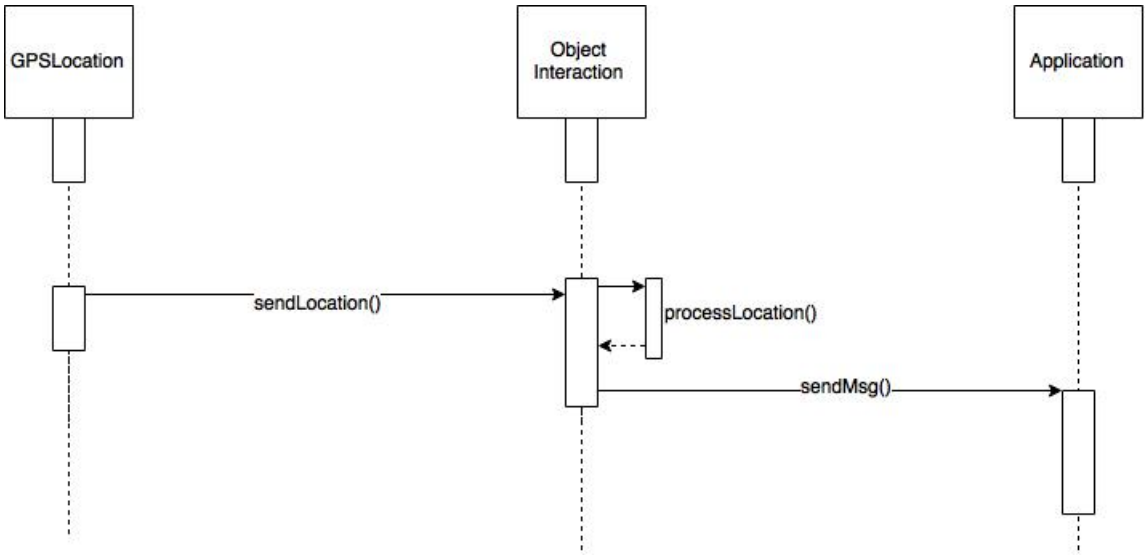


Figura 4.9: Diagramma di sequenza gestione locale

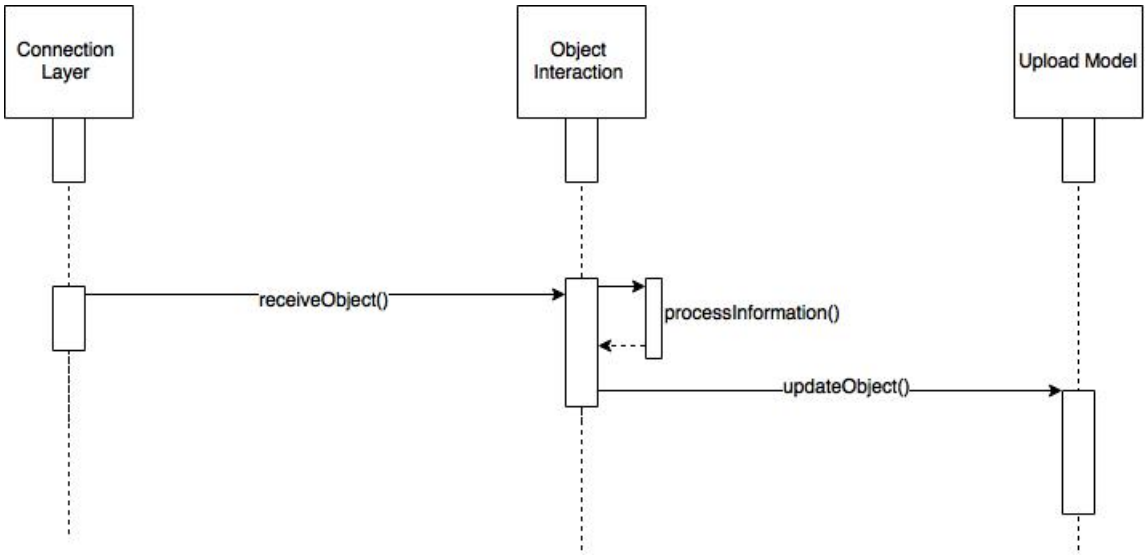


Figura 4.10: Diagramma di sequenza ricezione da Server

**Update Model** è il componente che si occupa di aggiornare il model appena riceve informazioni Object Interaction. Il suo compito è proprio quello di gestire il dominio ogni qual volta avvengano dei cambiamenti. Update Model ha il compito di interfacciarsi con lo strato User Interface.

### 4.5.3 Notification

Si tratta del componente che fornisce tutte le API relative allo scambio di messaggi e di notifiche. Notification ha anche il compito di gestire in parte la problematica di disconnessione, ma questo verrà approfondito nei componenti che lo compongono. Al suo interno vi sono: Notification Gesture e Recovery.

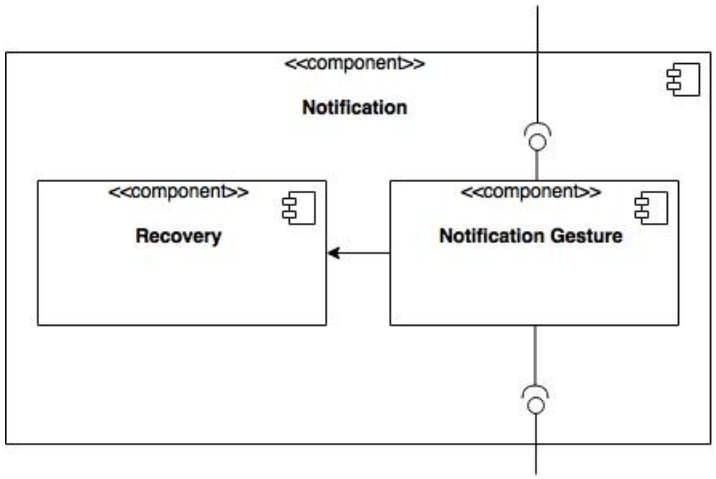


Figura 4.11: Diagramma dei componenti di Notification

**Notification Gesture** è lo strato che riceve dallo strato di connessione i messaggi e le notifiche e processa queste informazioni. E' questo componente che si deve occupare però anche di informare lo strato di User Interface di tutte le notifiche che arrivano dal sistema o dagli altri utenti.

I casi più importanti di cui si occupa questo componente sono la creazione di una notifica nel sistema, ovvero un informazione localizzata che può essere percepita da chi è in una determinata posizione, la ricezione di una notifica, la gestione dei messaggi. I messaggi sono richieste da parte del sistema, per esempio la richiesta di collaborazione o la segnalazione dell'avvenuta scoperta di uno scrigno di cui si possiede la chiave.

Sotto riportati due scenari fondamentali attraverso diagrammi di sequenza.

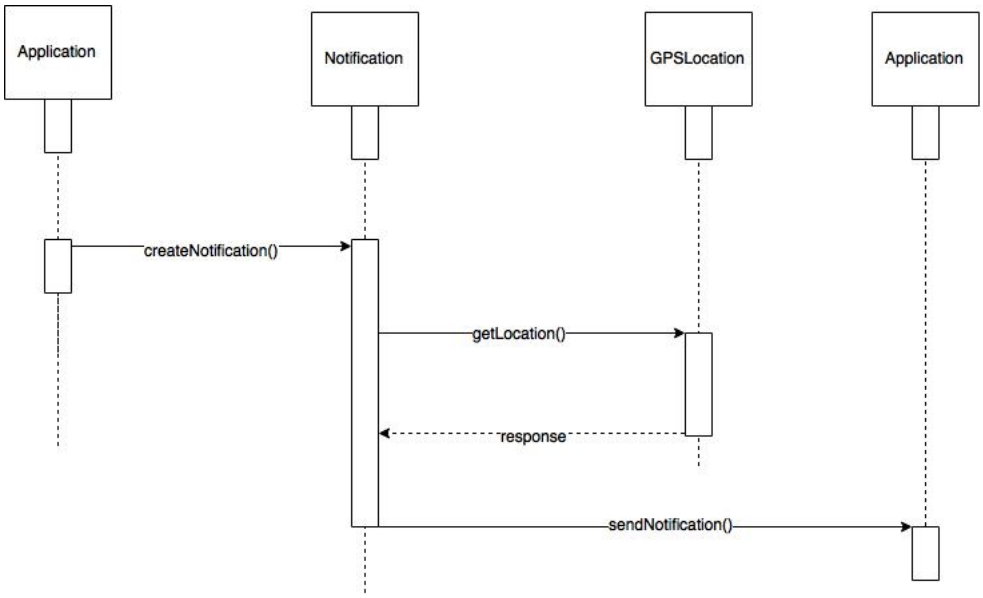


Figura 4.12: Diagramma di sequenza creazione notifica

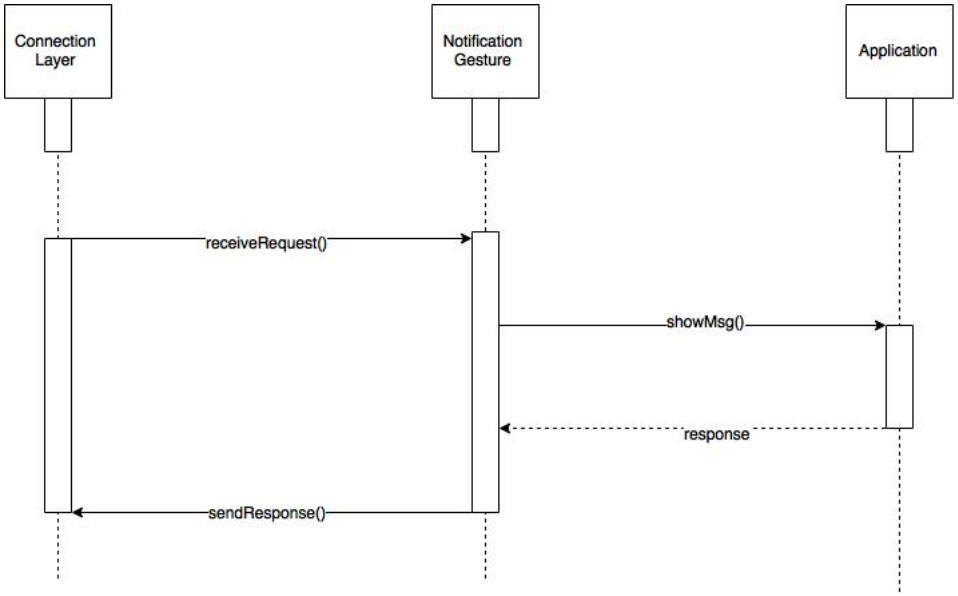


Figura 4.13: Diagramma di sequenza ricezione richiesta

**Recovery** è il componente che si propone di gestire la problematica della disconnessione per quanto riguarda lato client. In particolare c'è da sottolineare che in caso di disconnessione lato client è lo strato di connessione che si rende conto che il dispositivo è disconnesso.

Recovery è un componente, dotato di un flusso proprio, utilizzato per memorizzare i messaggi che si vuole inviare ma che non si è sicuri di avere inviato.

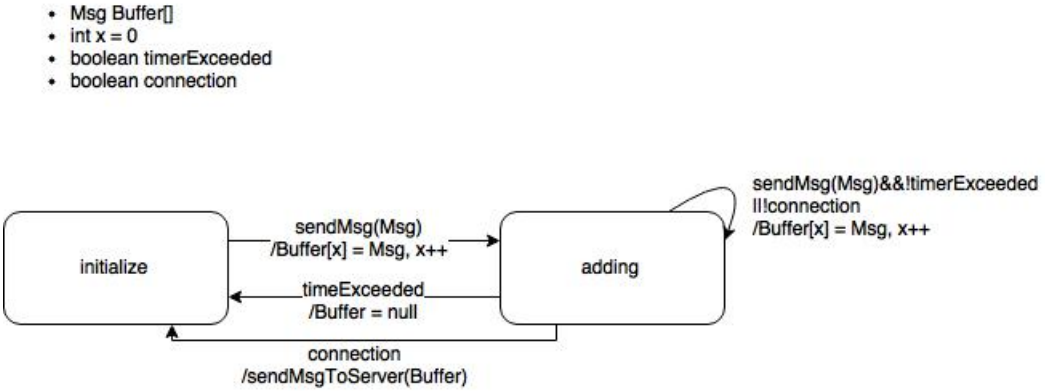


Figura 4.14: Diagramma a stati del flusso di recovery

Il sistema di controllo funzionerà in questo modo: ogni volta che il componente Notification Gesture invia un messaggio al server recovery memorizza il messaggio in un buffer. Questi messaggi vengono tenuti nel buffer fintanto che non sì è sicuri di averli inviati. A questo punto viene svuotato il buffer di quei messaggi. Ma quando si è sicuri che la connessione non sia presente, il sistema raccoglie nel buffer tutti i messaggi e non appena la connessione è ristabilita viene effettuato il riinvio.

## 4.6 Moduli principali lato Server

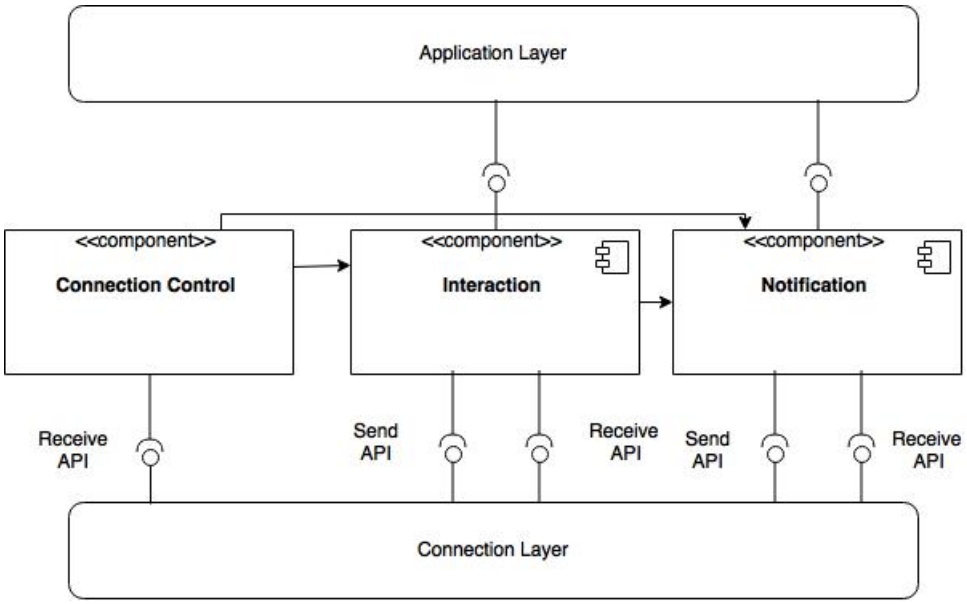


Figura 4.15: Moduli di base lato Server

I moduli principali lato server per quanto riguarda lo strato di collaborazione sono tre: Interaction, Notification e Connection Control. Nei prossimi paragrafi verranno spiegati nel dettaglio in base alle loro funzionalità principali.

### 4.6.1 Interaction

A questo componente è affidata la gestione delle interazioni tra utente e oggetto, le interazioni tra più utenti, la gestione e l'aggiornamento del model e la creazione e gestione dell'entità nemica. E' il modulo che si occupa principalmente della gestione del dominio distribuito, ma poi vedremo più in dettaglio dove e come agisce per evitare incoerenze nel sistema.

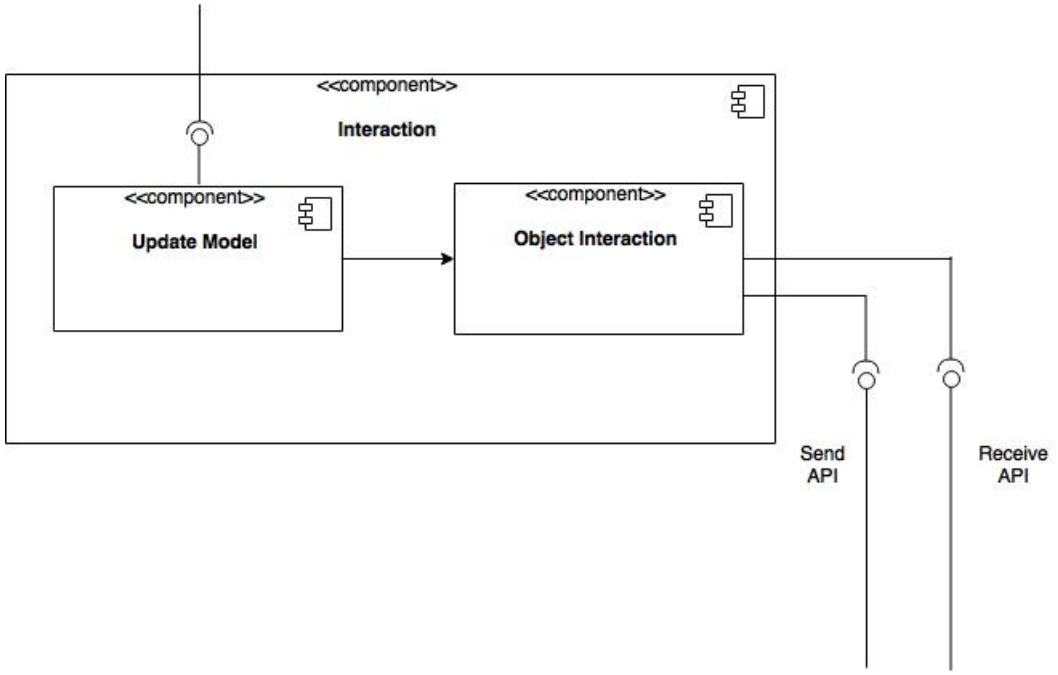


Figura 4.16: Componenti del modulo Interaction

**Object Interaction** è il componente che ha il compito di gestire tutte le interazioni tra utente e oggetto. E' il centro nevralgico delle azioni sugli oggetti. La funzione base di questo componente è ricevere la posizione dell'utente, valutarla e di conseguenza agire sugli oggetti passando dal model. Infatti necessità di poter vedere il model in quanto le azioni sugli oggetti dipendono anche da ciò che possiedono i vari utenti. Altra funzione fondamentale è quella di modifica degli oggetti. Al server, che in questo caso svolge la funzione di centrale operativa, può anche modificare direttamente il contenuto degli oggetti, per esempio diminuendo il valore del tesoro contenuto in uno scrigno.

Nel diagramma di sequenza di seguito si mostra come avviene la gestione delle interazioni tra oggetti e utenti.

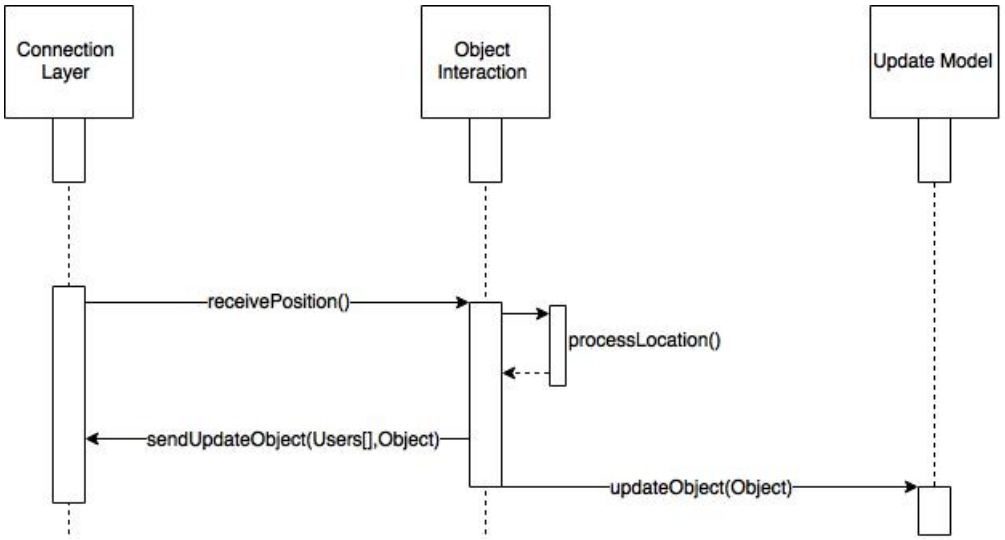


Figura 4.17: Diagramma di sequenza ricezione posizione

Infine funzione fondamentale di questo componente è quella di garantire che il dominio condiviso non abbia incoerenze. Per questo tipo di problema la soluzione che abbiamo progettato è di un server garante del sistema condiviso. Ovvero non ci sono cambi di informazioni nei domini locali delle applicazioni finché il server non ha confermato il cambio. Per esempio se un utente ha una disconnessione, ma in locale continua a vedere la posizione degli oggetti e si avvicina ad uno di questi, finché non si è riconnesso al server e non ha avuto la conferma dell'interazione con l'oggetto nemmeno il suo dominio cambia.

Questa situazione è semplificata da due aspetti fondamentali: la prima è che le informazioni del modello del Client non sono dettagliate e complete come quelle del Server. Questo rende il Client dipendente dal server nell'aggiornamento degli stati degli oggetti. La seconda è che nel caso di disconnessione il server, una volta ristabilita la connessione, rieffettua l'inizializzazione del gioco ma con i dati aggiornati allo stato attuale.

Qui di seguito un diagramma di sequenza che mostra quest'ultimo processo.

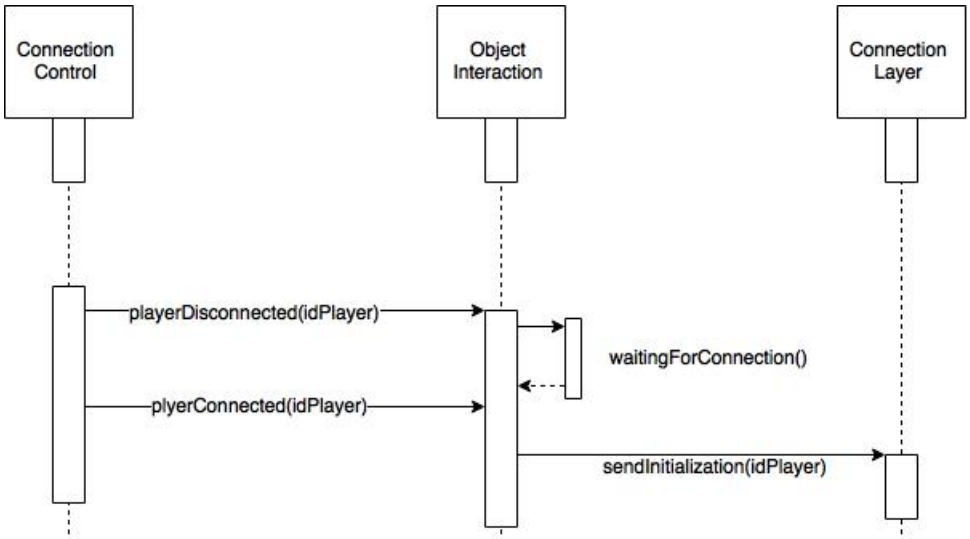


Figura 4.18: Diagramma di sequenza reinizializzazione dopo una disconnessione

**Update Model** è gestito nella medesima maniera del lato Client.

### 4.6.2 Notification

Il modulo Notification ha la funzione di gestire tutti i sistemi di notifica, dall'invio di un messaggio, come una richiesta di cooperazione, alla gestione delle notifiche localizzate. Anche in questo strato si utilizzano elementi per gestire il problema di disconnessione, che poi vedremo nel dettaglio nei prossimi paragrafi. Questo modulo è composto di tre elementi principali: Notification Gesture, Notification Sender, Recovery.

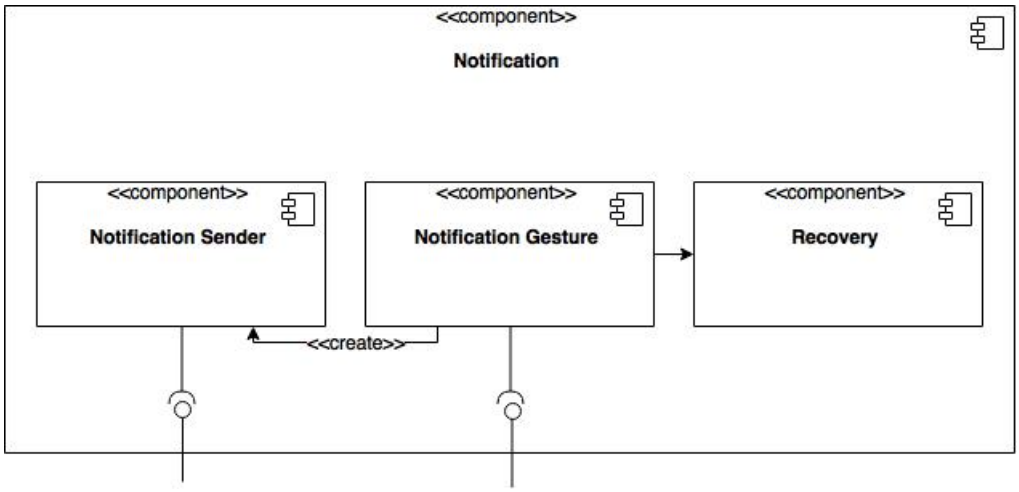


Figura 4.19: Componenti del modulo Notification

**Notification Gesture** è il componente che ha la funzione vera e propria di controllo delle notifiche e dei messaggi. Notification Gesture è il componente a cui fa riferimento interaction nel momento in cui deve richiedere ad un utente un'azione, o ancora è il componente attraverso la quale viene gestita l'invio e la ricezione di notifiche.

E' questo componente che attiva Notification Sender, che poi vedremo in dettaglio, per il controllo della posizione.

Qui di seguito vediamo uno scenario dove dopo l'interazione Interaction fa una richiesta all'utente mostrato con un diagramma di sequenza.

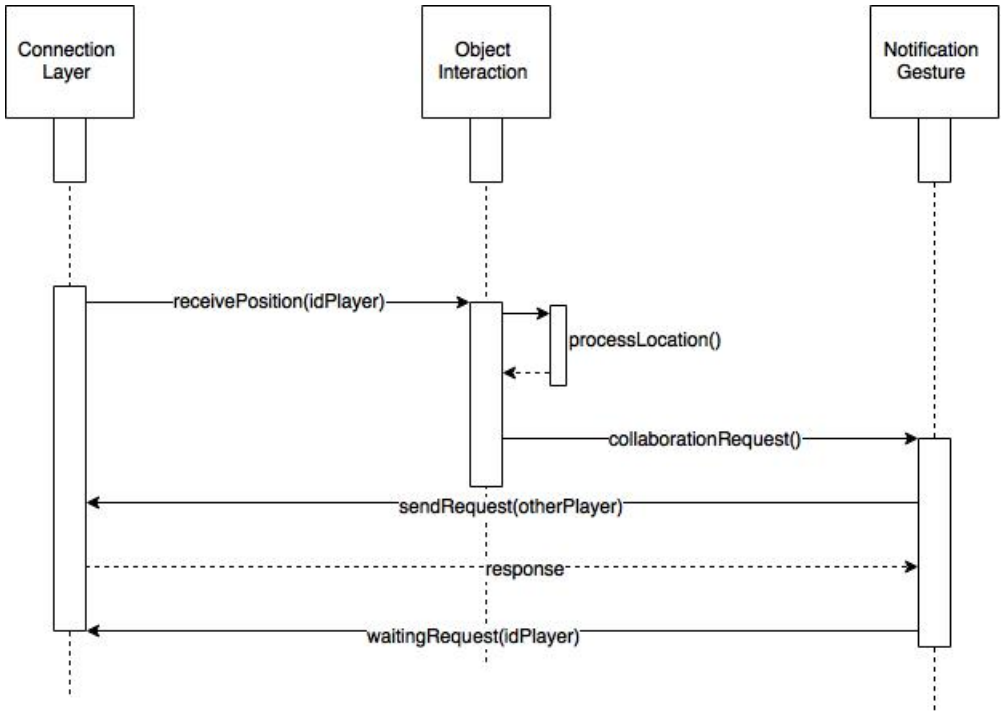


Figura 4.20: Diagramma sequenza invio request

**Notification Sender** è un componente attivo che ha il compito di intercettare il momento in cui inviare la notifica ad un Utente. Ovvero Notification sender viene creato da Notification Gesture a seguito di una richiesta di notifica. A questo punto il flusso di controllo di Notification Sender inizia in attesa di ricevere una posizione entro al range della sua notifica. Quando riceve la posizione richiama Notification Gesture per l'invio della notifica.

Qui sotto riportato il diagramma di sequenza che evidenzia come avviene questa interazione.

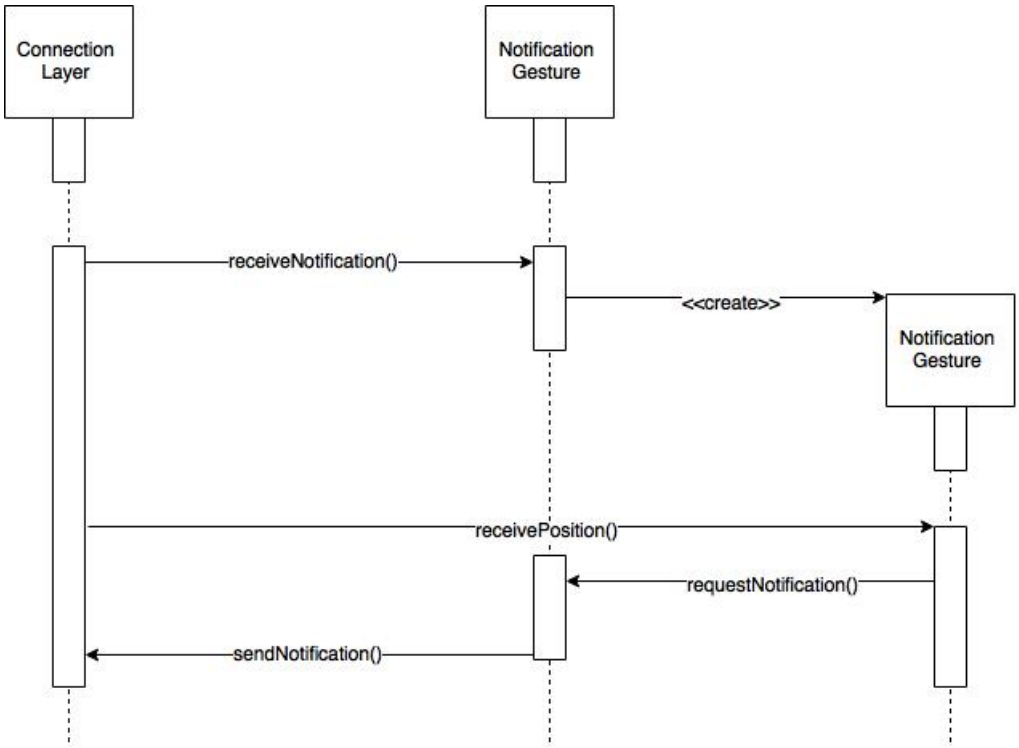


Figura 4.21: Diagramma sequenza richiesta notifica e invio

**Recovery** è il componente che si occupa di tenere in memoria i messaggi che devono essere ancora inviati all'utente in caso di disconnessione. Il funzionamento è identico a quello del Client ma l'informazione di connessione e disconnessione è data da Connection Control, e non dallo strato di connessione. Infatti la macchina a stati diviene:

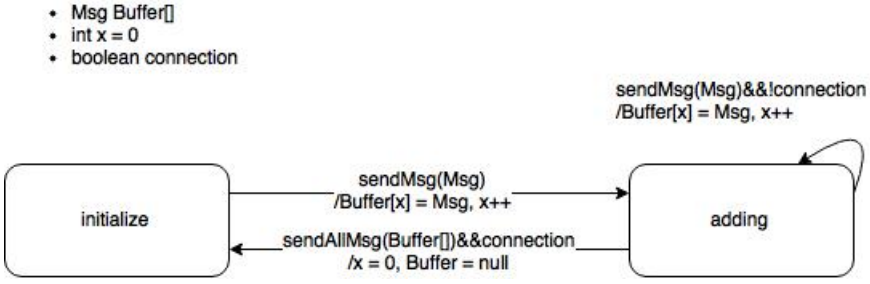


Figura 4.22: Diagramma stati Recovery

### 4.6.3 Connection Control

Connection Control è un modulo fondamentale all'interno del server, in quanto è il componente che controlla ed è consapevole dello stato di connessione di tutti gli utenti. Il suo funzionamento è basato sulla ricezione della posizione degli utenti. Il perché di questa scelta è dato da due aspetti cruciali: il primo è che essendo un software location aware, la posizione è un dato che viene inviato con periodicità elevate, e soprattutto che il server può conoscere a priori. Il secondo è che per evitare di avere latenze elevate, aspetto fondamentale in un sistema real time, non è conveniente appesantire la rete con messaggi che possono non essere indispensabili. In poche parole la ricezione della localizzazione fa da acknowledgement di connessione. Infatti il server è a conoscenza del fatto che l'utente invia o con una periodicità prefissata, o dopo una certa distanza percorsa, per cui può stimare il tempo di ricezione di un messaggio a priori. Scaduto un timer abbastanza ampio per considerare anche congestioni della rete questo componente considera disconnesso l'utente. Come descritto in figura 4.15 gestisce anche la riconnessione ela notifica all'Object Interaction.

## 4.7 Progettazione User Interface Layer

Lo strato applicativo si occuperà di fornire ai giocatori e ai gestori del gioco tutte le funzionalità necessarie al suo svolgimento.

Lato server, dovrà poter gestire la creazione del gioco con tutte le informazioni relative agli oggetti da posizionare nella mappa di gioco (con relative latitudine e longitudine, contenuto ed eventali informazioni aggiuntive).

Oltretutto, dovrà permettere l'inserimento delle entità nemiche e le eventuali riduzioni dell'ammontare di denaro disponibile.

Lato client, lo strato applicativo si occuperà di fornire all'utente tutte le informazioni relative allo svolgimento del gioco, permettendone la visualizzazione sullo schermo dei glasses. Dovrà inoltre intercettare gli eventi causati dall'utente, come il rilascio di notifiche e le risposte alle richieste di cooperazione.

### 4.7.1 Lato Server

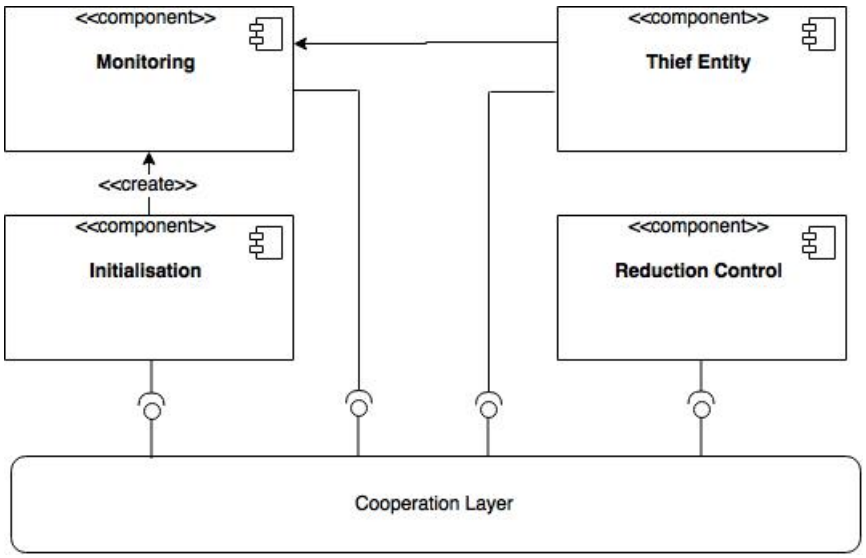


Figura 4.23: Diagramma componenti strato applicativo lato server

I componenti principali dello strato applicativo sono i seguenti:

- Initialisation
- Thief Entity
- Monitoring
- Reduction Control

**Initialisation:** è il componente che si occupa dell'inizializzazione del gioco. I gestori della partita potranno impostare tutte le informazioni attraverso un'apposita schermata.

**Thief Entity:** è il componente che rappresenta l'entità nemica del gioco. Tale entità cambia la sua posizione entro un raggio predefinito dai gestori del gioco in fase di inizializzazione.

**Monitoring:** è il componente che si occupa del monitoring del gioco. Permette ai gestori del gioco di visualizzare in tempo reale tutte le informazioni sulla posizione dei giocatori e delle entità nemiche e

**Reduction Control:** è l'entità che gestisce la riduzione del premio totale disponibile. Tale riduzione può essere temporizzata (dopo un certo periodo avviene automaticamente) oppure pilotata direttamente dai gestori del gioco.

### 4.7.2 Lato Client

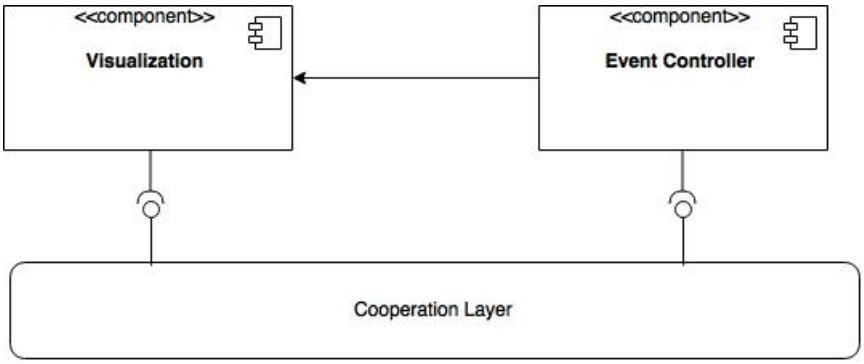


Figura 4.24: Diagramma componenti strato applicativo lato server

I componenti principali dello strato applicativo sono i seguenti:

- Visualization
- Event Controller

**Visualization:** è l'entità che gestisce la visualizzazione delle informazioni sullo schermo dei glasses dei giocatori. In base alle informazioni ricevute dallo strato di cooperazione le schermate saranno aggiornate.

**Event Controller:** intercetta gli eventi degli utenti, passandoli allo strato di cooperazione.

# Capitolo 5

# Sviluppo

## 5.1 Tecnologie utilizzate

Il nostro sistema come visto in precedenza è basata su una architettura Client-Server, in cui il Client è a sua volta diviso in due sotto parti: uno smartphone e un paio di smartglasses.

### 5.1.1 Moverio BT-200

La tecnologia usata come supporto per la realtà aumentata sono stati gli smartglasses Epson Moverio Bt-200. Epson aveva già messo sul mercato Epson Moverio BT-100 nel Novembre del 2011 in giappone e poi ad Aprile del 2012 in molte altre parti del mondo, per poi decidere di migliorare il proprio prodotto e mettere sul mercato la seconda generazione dal Marzo del 2014.

I Moverio sono visori binoculari con lenti trasparenti. Questa loro peculiarità permette di sovrappore i contenuti, visualizzati da entrambi gli occhi, alla realtà circostante così fornendo una nuova forma di realtà aumentata. In più questa sua caratteristica permette la "renderizzazione" di oggetti 3D con un maggiore coinvolgimento dell'utente.

Gli occhiali sono formati di due parti distinte: il visore e l'unità di controllo. Quest ultima è dotata di un pannello touch, di comandi integrati, della batteria e di un ricevitore GPS. Sul visore invece sono presenti diversi sensori tra cui accelerometri e bussola digitale. Altro componente fondamentale di questa tecnologia e la videocamera posta sugli occhiali che permette di riprendere, e nel caso anche identificare, ciò che l'utente sta guardando.

La visione che si ha utilizzando questo tipo di smart glasses è quella di uno schermo virtuale di 50 pollici alla distanza di 5 metri approssimativamente. Lo schermo visualizzato di base ha l'aspetto di un qualsiasi device che utiliz-

za come sistema operativo Android in particolare la versione 4.0.4 in forma landscape.

Tramite l'unità di controllo l'utente ha la possibilità di muovere all'interno dello schermo virtuale il cursore. Questo permette di utilizzare tutte le funzionalità e intraprendere le varie azioni che sono fornite dal sistema operativo Android.

La decisione di utilizzare i Moverio BT-200 viene dal fatto che si tratta di tecnologia wearable all'avanguardia e alla portata di tutti. In più per le sue peculiarità che la rendono tra le più adatta per la realtà aumentata.

### 5.1.2 Linguaggi utilizzati

I due linguaggi utilizzati per sviluppare il nostro software sono stati Java e Android.

Lato Server è stata sviluppata un'applicazione in Java mediante l'utilizzo dell'IDE Eclipse. Lato Client invece abbiamo due parti distinte ma entrambe sviluppate su Android mediante AndroidStudio.

## 5.2 Implementazione

L'implementazione del nostro progetto è stata fatta in maniera prototipale per riuscire a verificare e testare le funzionalità di base del nostro sistema distribuito. Nella fase di sviluppo infatti la nostra concentrazione è stata posta sul dare una forma ai componenti emersi durante la parte di progettazione al fine di renderli capaci di adempiere i compiti richiesti dall'applicazione.

### 5.2.1 Lato Server

Il server è impostato come un'applicazione basata su pattern MVC. All'interno delle tre parti fondamentali si distinguono i componenti ideati e progettati come mostrato sopra.

Prima di tutto bisogna spiegare qual'è la metodologia di scambio di informazioni tra lo strato di connessione e lo strato di cooperazione. Il controller principale, che è colui che media tra view e model, funge da Observer dello strato di connessione. In questo modo, essendo un componente attivo, infatti Observer estende Thread, il controller osserva i cambiamenti dello strato di connessione e in particolare è in grado di vedere quando un messaggio è in ricezione. L'Observer poi demandera i compiti sui messaggi ricevuti allo strato di Cooperazione.

Vediamo ora come sono stati implementati i vari componenti andando a esaminare le classi che ho deciso di sviluppare.

**Interaction** come già visto in precedenza è il componente che deve gestire tutte le interazioni tra oggetti e utenti. Le classi che lo caratterizzano sono: ObjectInteraction, PositionControl, UpdateModel.

- ObjectInteraction: è la classe di gestione di tutte le interazioni. Dentro a questa classe vi sono i metodi che permettono di cambiare gli oggetti, i metodi che permettono di controllarne lo stato e modificarlo. E' la classe che crea PositionControl nella sua inizializzazione.

- PositionControl: è un thread di controllo sulla posizione attuale dell'utente. Ovvero una volta ricevuta la posizione fa un controllo su tutti i punti interessati e ne verifica la distanza dall'utente. Se questa rientra in un certo raggio richiama ObjectInteraction per fare il cambiamento di stato.

- UpdateModel: è la classe che si interfaccia al model. UpdateModel possedendo un istanza del model al suo interno ne modifica lo stato richiamando i metodi del model. Questa classe funge da separatore tra i due livelli UserInterface e CooperationLayer creando il tramite tra i due.

**Notification** è il componente che si occupa di tutta la gestione delle notifiche. Al suo interno presenta tre classi fondamentali: NotificationGesture, NotificationSender, Recovery.

- NotificationGesture: è la classe di gestione delle notifiche. Al suo interno ha tutti i metodi riguardanti le notiche, gli Alert del nostro dominio. Nel momento in cui riceve un messaggio di notifica crea il thread NotificationSender, mentre quando riceve da NotificationSender una segnalazione si occupa di inviare all'utente interessato l'alert. E' la classe utilizzata anche per la gestione di tutti i messaggi, per esempio le richieste, e in questo caso viene invocata da ObjectInteraction.

- NotificationSender: è un thread di controllo della posizione. Una volta creato controlla a loop se la posizione dell'utente ricevuta è dentro al range prestabilito della notifica; se l'utente si trova all'interno del range, questo thread invoca NotificationGesture per l'invio dell'alert all'utente.

- Recovery: è un thread che riceve tutti i messaggi inviati, tutte le segnalazioni connessione assente e tutte i messaggi di posizione relativi agli utenti. Recovery inserisce nei suoi buffer i messaggi relativi ai vari utenti e controlla a loop se la connessione è presente o se ha ricevuto un messaggio di posizione, nel caso entrambe siano vere toglie dal buffer i messaggi inviati prima della ricezione del messaggio di posizione.

**Connection Control** è il componente di controllo della connessione. All'interno del nostro software è stato strutturato come un thread che riceve e controlla se gli utenti inviano la loro posizione. Dopo un intervallo, che da noi è stato scelto in maniera euristica per garantirne nella maggior parte dei casi il rilevamento di disconnessione e non solo di rallentamento della connessione, in cui non riceve più la posizione di un utente si occupa di segnalare a recovery la disconnessione e richiede la distruzione della socket relativa a quel utente.

### 5.2.2 Lato Client

Il Client è sviluppato come un'applicazione Android dotato di un'unica Activity principale in cui non viene visualizzato nulla, in quanto la visualizzazione viene fatta attraverso gli smartglasses, i quali a loro volta hanno un'applicazione Android con un Activity principale per la mappa e alcune activity di gestione come la visualizzazione delle chiavi possedute dagli altri utenti.

Anche in questo caso è fondamentale individuare l'elemento che fa da tramite tra lo strato di connessione e quello di cooperazione. In questo caso la tecnologia utilizzata è quella dell'handler che ottiene dal connection layer i messaggi e li consegna al cooperation layer.

Andiamo ora a vedere come sono stati implementati i componenti principali dello strato di cooperazione.

**GPSLocation** è il componente che si occupa della localizzazione come abbiamo visto nella parte di progettazione. A livello implementativo è stato organizzato in due classi: GPSUtils e GPSListener.

- GPSUtils: è il componente che sfrutta il service di Google per la localizzazione. In questa classe viene impostato la frequenza di controllo dello spostamento, ovvero ogni 500 millisecondi e ogni 3 metri. La localizzazione non viene fatta solamente attraverso il sensore GPS, ma è anche impostata all'interno di questa classe la localizzazione attraverso la rete.

- GPSListener: è una classe che implementa LocationListener ed è quella che si occupa di rilevare lo spostamento.

**Notification** Questo componente è colui che si occupa della gestione delle notifiche e dei messaggi. E' stato sviluppato in due classi: NotificationGesture e Recovery.

- NotificationGesture: è la classe di gestione delle notifiche. Mediante i metodi implementati si possono creare e gestire la ricezione di Alert. E' l'implementazione semplificata della stessa classe a lato server.

- Recovery: è un thread di controllo per gestire i messaggi da reinviare. Recovery viene invocato ad ogni invio di messaggio, e salva il messaggio nel suo buffer. Il controllo all'interno del thread è sullo stato di connessione che viene gestito in questo caso dal connection layer. Se scade il timer impostato in base all'intervallo di invio della posizione senza che sia segnalata la disconnessione rimuove i messaggi inseriti nel buffer prima dell'ultimo invio sicuro.

**Interaction** Questo componente è gestito esattamente come lato server semplicemente è semplificata la sua implementazione in quanto lato client necessità di meno controlli e soprattutto deve gestire meno interazioni. Le classi infatti in cui è sviluppato sono ObjectInteraction, PositionControl, UpdateModel

# Capitolo 6

# Valutazioni

La prototipazione del sistema che abbiamo progettato è stata svolta seguendo le fasi di analisi, progettazione e sviluppo che ho mostrato fino a questo momento. Il software, pur essendo un prototipo, ha messo in luce le funzionalità di base che il nostro caso di studio richiedeva , le quali sono state implementate in modo da poterne verificare l'efficienza e l'utilità all'interno del nostro sistema. Da questo prototipo è emerso che i requisiti sono stati soddisfatti, in particolare testando sugli scenari precedentemente descritti.

In fase di progettazione erano state evidenziate le problematiche principali del nostro sistema che in breve sono: la coerenza dei dati, la localizzazione, le disconnessioni. Andando ad analizzare come sono state affrontate cercherò di dare una valutazione al lavoro svolto.

Un sistema distribuito in cui le varie sue parti possono interagire con il dominio, come nel nostro caso, deve avere come prerogativa fondamentale la coerenza delle informazioni, ancora di più se il dominio è condiviso. Infatti al fine di poter operare tutti sulle stesse informazioni, ogni parte deve essere certa che il dominio su cui interagisce sia lo stesso anche per tutte le altre parti. Questa problematica, che come appena detto è cruciale all'interno della nostra applicazione, è stata affrontata rendendo il server garante della coerenza. Tutti i cambiamenti del dominio vengono fatti prima dal server e poi aggiornati sui client. Questo ha reso solido in tutti gli scenari il sistema distribuito. I casi più critici sono dati dall'alta latenza. Infatti se per caso avviene un cambiamento nel dominio, ma un utente avendo la connessione appesantita o comunque rallentata per qualche ragione non venisse a conoscenza di questo cambiamento prima di poter interagire con un altro oggetto deve aggiornare il suo dominio in quanto l'azione sullo scrigno viene svolta dal server nel momento in cui riconosce la presenza dell'utente nei pressi dell'oggetto. Possiamo ritenere quindi soddisfacente la gestione del dominio condiviso e funzionale per un sistema distribuito strutturato come quello della nostra applicazione.

La nostra applicazione si inserisce nel panorama dei location based game,o comunque all'interno delle applicazioni location aware. Ai fini del nostro progetto quindi è necessario che il sistema di localizzazione sia funzionale ed efficiente. Per fare questo il nostro sistema si basa su un invio costante e continuo della propria posizione in base o ad un timer scaduto o ad una quantità di spazio percorsa. Questo sistema in sé si è dimostrato efficiente e funzionale rispetto ai nostri requisiti. La problematica principale però della localizzazione è data dal fatto che le tecnologie che ci sono fornite oggi sono ancora limitate pur essendoci stato un grande sviluppo in questo ambito. L'utilizzo del sistema A-GPS migliora e rende più precisa la localizzazione, ma sempre mantenendo un'errore di qualche metro, dai 3 agli 8 metri più o meno. In più nei così detti canyon urbani il segnale diventa difficile da mantenere stabile e preciso. Con l'integrazione di tecniche di triangolazione mediante l'utilizzo di sensori, magari posti nei pressi dei punti di interesse, si andrebbe ad aumentare la precisione però perdendo la sua essenza di pervasive game, ovvero la possibilità di svolgere questa applicazione dovunque e non in un luogo a priori prestabilito.

Infine l'ultima problematica affrontata è la gestione delle disconnessioni. Questa problematica è stata affrontata su più livelli io mi concentrerò a dare una valutazione su quello di cooperazione, ovvero come io ho affrontato questo punto critico della nostra applicazione.

In generale anche la gestione di questa problematica è stata sviluppata con successo, ovvero le soluzioni fornite si sono dimostrate funzionali ma mettendo in mostra alcune criticità. Rivedendo le soluzioni fornite cercherò di dare una visione generale di quali sono le criticità incontrate.

La gestione delle disconnessioni è stata gestita su due versanti: lato server e lato client.

Lato client sul mio livello mi sono limitato a fornire un sistema di recovery per i messaggi da inviare al sistema, in particolar modo l'invio di notifiche. Ovvero il sistema memorizza in un buffer i messaggi inviati fino a che non scada un timer basato sull'intervallo di tempo minimo fra due invii della propria posizione. Chiaramente in questa funzione di recovery come già detto in precedenza viene tenuto conto del feedback di connessione che fornisce il livello sottostante. Lato client la criticità maggiore è se il client tenta di inviare un informazione importante ai fini del gioco, come una notifica, e durante la sua disconnessione un'altro client va nel posto da lui segnalato. In questo caso l'utente non può conoscere il pericolo a cui sta andando incontro. Lato server invece il sistema di acknowledgement dato dal costante invio della posizione ha dimostrato qualche problematica. La più forte è il caso di invio di un messaggio importante ad un utente ed una successiva ricezione di posizione di un utente che però era stata inviata prima dell'invio del messaggio. Questo può

venire a causa di latenze elevate. In questo caso il nostro sistema considera il messaggio inviato e consegnato anche se non è vero. Quindi si può rischiare di avere perdite di informazione. Si posso avere metodologie di risoluzione di questa problematica, ognuna però con una sua criticità. Per esempio si potrebbe aggiungere una conferma di ricezione ad ogni messaggio mandato dal server, anche se questa soluzione appesantirebbe la rete. Si possono mandare messaggi replicati che vengono poi scartati dall'utente, anche questo è un appesantimento della rete anche se meno rilevante del primo caso.

Ultimo aspetto rilevante del nostro studio è la possibile astrazione e generalizzazione dei livelli che abbiamo fornito noi alla nostra applicazione. Infatti nel nostro sistema individuiamo tre livelli: connection, cooperation e Ueser Interface. All'interno del nostro progetto non si può parlare di un vero e proprio middleware in quanto il livello cooperation è stato pensato mirato al caso di studio. Bensì offre spunti importanti per la creazione di un layer dedicato alla cooperazione. Si possono facilmente individuare API che siano funzionali ai sistemi distribuiti situati a supporto della collaborazione basati su realtà aumentata. Infatti ci sono molte funzionalità che possono essere utilizzate nella creazione di nuove applicazioni di questo genere.

La localizzazione per esempio in questi tipi di sistemi è necessaria e può essere un API funzionale ad altre applicazioni. La gestione delle interazioni con gli oggetti e tutte le funzionalità che rendono il nostro sistema distribuito un supporto alla cooperazione possono essere quindi astratte e generalizzate in modo da creare un livello utilizzabile anche per applicazioni differenti ma sempre inerenti all'ambito studiato.

Questo tipo di studio che abbiamo approfondito in definitiva va veramente ad aggiungere funzionalità che espandono il concetto e l'utilizzo che fino ad oggi si è fatto della realtà aumentata. Attraverso la progettazione di questa applicazione abbiamo infatti mostrato, in piccola parte, che potenziale può avere la realtà aumentata sfruttando oggetti computazionali e in grado di interagire con la realtà che li circonda non limitando la realtà aumentata al concetto puramente grafico che ha oggi.

A mio avviso lo studio da me svolto apre veramente le porte a nuovi mondi all'interno della realtà aumentata, a veri e propri Augmented World.

# Conclusioni

Il sistema che abbiamo ideato, analizzato, progettato e sviluppato può dirsi a mio parere un successo. In primis l'approcio con cui abbiamo svolto tutte le fasi lavorative si è rivelato funzionale ed efficiente. Ovvero pur scontrandoci con i limiti delle nostre conoscenze su questi ambiti, infatti la prima fase corposa della tesi è stata di studio delle tecnologie che non conoscevamo, come la realtà aumentata, la strutturazione di sistemi distribuiti, è stato un lavoro con dei buoni risultati. Infatti il software sviluppato ha presentato aspetti innovativi in grado di contribuire allo studio della realtà aumentata.

Questo studio iniziale mi ha permesso di ampliare il mio bagaglio culturale, infatti le nozioni imparate sulla realtà aumentata e sui sistemi distribuiti sono tutte novità per me. In più la possibilità di approfondire aspetti che già avevo esplorato ma non così in profondità, per esempio lo sviluppo di applicazioni Android ed in particolare di software location aware, è stata di certo una ricchezza.

Infine è stato molto positivo il lavoro in team che ci ha permesso di lavorare su un progetto più ampio e complesso, studiando aspetti differenti, ma che hanno reso il nostro studio più modulare. Infatti la suddivisione del nostro lavoro in layer ci ha permesso di approfondire con più precisione le problematiche che ci si sono presentate.

## Sviluppi Futuri

Gli sviluppi futuri che questa applicazione può avere sono riguardanti allo sviluppo di User Interface più sostanzioso e mirato.

Durante il nostro lavoro ci siamo concentrati più sullo sviluppo degli strati sottostanti rendendo disponibili allo strato superiore delle API. L'applicazione però può di certo essere ampliata aggiungendo uno strato di User Interface integrando magari framework come Wikitude o Layar. Ora quello che noi mostriamo è semplicemente un messaggio ma in casi come quello del ladro, o comunque di entità nemiche, il rendering del nemico renderebbe più usabile l'applicazione. E' chiaro però che apportando queste modifiche non si deve an-

dare ad intaccare la scalabilità del nostro progetto. Quindi non andrà ampliato con marker.

Altri sviluppi futuri che possono affinare questo studio sono relativi alle tecnologie utilizzate. Come visto nella fase di valutazione abbiamo riscontrato limiti tecnologici. Bensì se verranno sviluppati tecnologie di localizzazione più precise rispetto all'attuale sistema GPS, potremmo arrivare a identificare un punto con un margine di errore di qualche centimetro. Questo diminuirebbe per esempio il limite di utilizzare applicazioni di questo tipo solo in ambiente outdoor.

Un altro sviluppo futuro importante è quello di generalizzare le funzionalita e le API che abbiamo sviluppato per creare un middleware a supporto della collaborazione. Questo può portare ad un ampliamento prima di tutto delle funzionalità che i toolkit di realtà aumentata forniscono, in secondo luogo ad un ampliamento dell'utilizzo di tecnologie che supportino la realtà aumentata e di applicazioni di questo genere. Sono tanti gli ambiti in cui può essere utile un'applicazione collaborativa basata su realtà aumentata per esempio il learning o i soccorsi. Questo è lo sviluppo che può avere più importanza o per lo meno che rende questo studio valido.

# Ringraziamenti

Devo prima di tutto porre un grande ringraziamento alla mia famiglia che mi ha sostenuto nello studio di questi anni e soprattutto nell'ultimo periodo in cui il mio lavoro era diretto all'ideazione di questa tesi. Un'altro ringraziamento particolare va ad Angelo Croatti, Pietro Brunetti e il Prof. Alessandro Ricci che ci hanno seguito con pazienza dandoci preziosi consigli e puntualizzazioni sul nostro operato. Infine un ringraziamento sentito va ai miei compagni di studio di questi anni Brando e Matteo coi quali ho affrontato tutti i progetti che l'università ci ha richiesto concludendo con l'ideazione in team di questa tesi.

# Bibliografia


[1] J. Carmigniani, B. Furht,M. Anisetti, P. Ceravolo, E. Damiani, M. Ivkovic *Augmented reality technologies, systems and applications*

[2] T. Jebara, C. Eyster, J. Weaver, T. Starner, and A. Pentland. Stochasticks *Augmenting the billiards experience with probabilistic vision and wearable computers.*

[3] D.W.F. van Krevelen and R. Poelman *A Survey of Augmented Reality Technologies, Applications and Limitations*

[4] Wolfgang Broll, Jan Ohlenburg, Irma Lindt, Iris Herbst, Anne-Kathrin Braun *Meeting Technology Challenges of Pervasive Augmented Reality Games*

[5] Juan Rodrıguez-Covili , SergioF.Ochoa, José A. Pino, RocMesseguer, EsunlyMedina, DolorsRoyo *A communication infrastructure to ease the development of mobile collaborative applications*

[6] Carstensen, P.H, Schmidt, K. *Computer supported cooperative work: new challenges to systems design*

[7] Wilson, P. *Computer Supported Cooperative Work: An Introduction*

[8] Mark Billinghurst, Hirokazu Kato *Collaborative Augmented Reality*

[9] Mark Billinghurst , Ivan Poupyrev , Hirokazu Kato , Richard May *Mixing Realities in Shared Space: An Augmented Reality Interface for Collaborative Computing*

[10] Z. Szalavàri, D. Schmalstieg, A. Fuhrmann, M. Gervautz *Studierstube ': An Environment for Collaboration in Augmented Reality*

[11] M. Biilinghurst, S. Weghorst, Furness III *Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work*

[12] Nikolaos Avouris, Nikoleta Yiannoutsou *A Review of Mobile Location-based Games for Learning across Physical and Virtual Spaces*

[13] Wolfgang Broll, Jan Ohlenburg, Irma Lindt, Iris Herbst, Anne-Kathrin Braun *Meeting Technology Challenges of Pervasive Augmented Reality Games*

[14] Alessandro Ricci, Luca Tummolini, Cristiano Castelfranchi, Michele Piunti *Mirror Worlds as Agent Societies Situated in Mixed Reality Environments*

# Elenco delle figure