ANALISI, PROGETTAZIONE E MODELLAZIONE

I diagrammi sviluppati ci permettono di descrivere ed affrontare tutte le fasi della modellazione, dall'elicitazione e analisi dei requisiti, che permettono di formulare e analizzare il problema e costruire il modello del dominio applicativo, passando poi a suddividere il problema in blocchi al fine di trovare strategie e soluzioni per ogni blocco e quindi per progettare l'intero sistema.

RAD: Requirements Analysis Document

Requisiti

Funzionali

La piattaforma RdF deve essere in grado di supportare tre tipi di utenti:

- L'amministratore deve essere in grado di gestire le frasi misteriose e relativi temi, in particolare la loro aggiunta, fondamentali ai fini del gioco.
- Il concorrente di gioco deve poter organizzare e partecipare alle partite.
- L'osservatore che deve poter monitorare l'andamento di una partita in corso osservando le mosse dei concorrenti oppure di una partita in attesa registrandosi e venendo poi avvisato.

Gli utenti devono essere in grado di registrarsi ed effettuare il login sulla piattaforma. Per entrambe le funzionalità sono richiesti dei dati personali, modificabili, ma in particolare per la prima bisogna supportare la creazione, l'invio via email e il controllo di codici temporanei di durata di dieci minuti.

Per quanto riguarda le partite, gli utenti devono poter accedere allo storico delle statistiche basato sulle manches di gioco, utile inoltre per selezionare cinque frasi mai viste dai concorrenti al fine di iniziare una nuova partita.

Non funzionali

- Performance: il sistema deve supportare l'inizio contemporaneo di numerose partite, ognuna composta esattamente tre giocatori e da un numero indefinito di osservatori. Tutto questo deve quindi avvenire in concorrenza.
- Implementazione: la piattaforma RdF dovrebbe poter essere eseguita su un qualsiasi sistema operativo.
- Sicurezza: le funzioni di amministrazione come la gestione delle frasi misteriose, non dovrebbero essere accessibili attraverso la piattaforma.

System Models

Object Model

Con i Class Diagram abbiamo presentato la composizione della nostra piattaforma. Partendo dal diagramma RdF, vengono mostrati i moduli presenti che in aggiunta ai richiesti PlayerRdf, AdminRdf e ServerRdf, vede anche RdfUtil, il quale raggruppa tutte le funzionalità del sistema utili a parti del progetto, come mostrato nell'omonimo Class Diagram.
Ci sono poi i Class Diagram relativi ad altri modelli del progetto, in particolare per il modulo ServerRdf abbiamo identificato diversi managers con il compito di occuparsi di una suddivisione di ruoli, ad esempio il PhraseManager si occupa della gestione e della manipolazione delle frasi, il ProfileManager per la gestione del profilo degli utenti, l'AuthenticationManager per il login, il RegistrationManager per la registrazione, il MonitoringManager per tutto ciò che concerne le statistiche di gioco globali e personali ed infine il MatchManager per la gestione delle partite in attesa di giocatori e in corso. Ognuno di essi, come anche la classe MatchVisualizer con il compito di visualizzare le partite, sfruttano il design pattern Singleton che impedisce la creazione di ulteriori oggetti dello stesso tipo.
L'utilizzo di manager per suddividere i contenuti della piattaforma risulta fondamentale per l'implementazione del Server che sfrutta il design pattern Façade fornendo un'interfaccia che comprende tutte le funzionalità disponibili al client. Quindi al fine di diminuire la complessità, l'implementazione dell'interfaccia del server sfrutterà i managers precedentemente citati senza doversi interamente occupare della messa a punto.
La comunicazione del server con il database è gestita attraverso l'uso del Data Access Object pattern, in particolare DBManager funge da DAO factory andando ad istanziare le altre interfacce. Inoltre esistono diverse interfacce DAO, come ad esempio UsersDAO, che gestiscono l'accesso alle tabelle e utilizzano dei Data Transfer Objects, come UsersDTO, contenenti i campi relativi agli attributi della tabella.
Per quanto riguarda la comunicazione tra Client e Server questa è gestita tramite oggetti remoti e un sistema di callback per notificare successi e fallimenti. In particolare dal punto di vista del Client, la classe Player è una delega dell'oggetto remoto Client che permette, oltre che inviare notifiche, anche di gestire le informazioni dei giocatori in-game. Dal punto di vista del Server invece, ogni partita è gestita da un proprio oggetto remoto RemoteMatch e la comunicazione tra Match e Client sfrutta il design pattern Observer dove il Match è l'oggetto osservato, Client e Player sono gli osservatori e Match mantiene una lista di entrambi.

In aggiunta ai requisiti non funzionali richiesti, individuati durante l'analisi, tale progettazione permette future aggiunte di funzionalità al sistema. Dal punto di vista del server, essendo la comunicazione del database gestita da un'unica interfaccia, è sufficiente aggiungere nuove classi di controllo. Inoltre dato che la comunicazione tra client e server è gestita da due interfacce, di conseguenza è possibile aggiungere dal lato server nuove funzionalità e dal lato client nuovi controller con il minimo sforzo.

Dynamic Model

Abbiamo sviluppato uno State Diagram per la partita, in quanto è il soggetto che nel corso del gioco assume diversi stati, gli Activity Diagram per la gestione di una manche e diversi Sequence Diagram che concernono l'autenticazione, l'avvio del Server, la gestione delle frasi misteriose, la registrazione, la richiesta di partecipazione a una partita da parte di un concorrente e il funzionamento della manche di una partita.
In relazione ai Class Diagram, abbiamo deciso che fosse opportuno raggruppare tutti i managers indicati nel sequence in un'unica interfaccia utilizzabile dal Client (Pattern Façade) e che non fosse opportuno far gestire tutte le partite solo a MatchManager. Abbiamo quindi stabilito di aggiungere degli oggetti remoti RemoteMatch ognuno dei quali si occupa della gestione di una singola partita.

Modifiche durante l'implementazione

- Per la registrazione sono state aggiunte delle classi per gestire l'inserimento dell'OTP entro dieci minuti (OTPHelper e WaitingThread).
- Per le partite è stata aggiunta la classe MoveTimer per la gestione del timer.
- Aggiunte le classi MatchData per il trasferimento dei dati relativi alle partite e AdminChecker per gestire e il funzionamento dell'interfaccia in funzione dei permessi dell'utente autenticato.
- Nell'implementazione finale, le classi che devono inviare delle mail non sfruttano direttamente EmailSender ma una classe intermedia EmailManager.