Game Hub

Bedeschi Federica, Delja Alesja,

Furegato Silvia

A.A. 2021/2022

Indice

1 Analisi 2

1.1 Requisiti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.2 Analisi e modello del dominio . . . . . . . . . . . . . . . . . . 4

2 Design 7

2.1 Architettura . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.2 Design dettagliato . . . . . . . . . . . . . . . . . . . . . . . . . 9

3 Sviluppo 19

3.1 Testing automatizzato . . . . . . . . . . . . . . . . . . . . . . 19

3.2 Metodologia di lavoro . . . . . . . . . . . . . . . . . . . . . . . 20

3.3 Note di sviluppo . . . . . . . . . . . . . . . . . . . . . . . . . 21

4 Commenti finali 25

4.1 Autovalutazione e lavori futuri . . . . . . . . . . . . . . . . . . 25

4.2 Difficoltà incontrate e commenti per i docenti . . . . . . . . . 25

A Guida utente 27

B Esercitazioni di laboratorio 28

B.0.1 Bedeschi Federica . . . . . . . . . . . . . . . . . . . . . 28

B.0.2 Delja Alesja . . . . . . . . . . . . . . . . . . . 28

B.0.3 Furegato Silvia . . . . . . . . . . . . . . . . . 28

B.0.4 Savarese Marco . . . . . . . . . . . . . . . . . 28

Capitolo 1

Analisi

* 1. Requisiti

L’applicazione GameHub si occupa di raccogliere diversi giochi. I giochi proposti sono giocabili in singolo e sono: Flood it, Minefield e Numerical bond.

Requisiti funzionali:

* L’utente potrà decidere a quale gioco giocare tramite il menu principale
* Una volta scelto il gioco, l’utente potrà impostare certi parametri per ottenere varianti diverse dello stesso gioco
* Se il gioco scelto è Flood it, …
* Se il gioco scelto è Minefield, …
* Se il gioco scelto è Numerical bond, l’utente potrà vedere una griglia quadrata di blocchi, ognuno con un certo numero all’interno. A questo punto:
  + L’utente potrà collegare tra loro qualunque coppia di blocchi adiacenti (orizzontalmente o verticalmente), fino a un massimo di due collegamenti per coppia
  + Il numero all’interno del blocco rappresenta il numero di collegamenti che il blocco deve ancora avere, di conseguenza esso si aggiornerà ad ogni collegamento che interessa il blocco stesso
  + Se l’utente, arrivato al massimo di due collegamenti tra due blocchi, si renderà conto che non sono necessari, potrà rimuoverli ripartendo di nuovo dai blocchi scollegati
  + Sarà inoltre concesso di fare più collegamenti del necessario, in tal caso il numero del blocco sarà negativo, a segnalare il fatto che ci sono dei collegamenti di troppo
  + L’utente vincerà la partita nel momento in cui tutti i blocchi della griglia segneranno 0
* Da ogni gioco sarà possibile aprire un menu di pausa, che consentirà all’utente di incominciare una diversa partita dello stesso gioco, di riprendere la partita stessa, oppure di tornare al menu principale interrompendo la partita
* Al termine di ogni gioco verrà visualizzata una schermata che indicherà se il gioco è stato vinto o meno, e permetterà di cominciare una nuova partita oppure di tornare al menu principale per scegliere un altro gioco

Requisiti non funzionali:

* ~~Salvataggio su file di record e statistiche dei giochi, identificando ciascun giocatore tramite un username univoco~~
* Fluidità e intuitività
  1. Analisi e modello del dominio

Descrizione generica dello schema UML (generico, interfacce, model) del proprio gioco.

Il gioco Numerical bond si comporrà di una griglia quadrata di blocchi. Essi possono essere collegati tra loro, con un massimo di due collegamenti per lato. Ogni blocco dovrà essere collegato a un certo numero di blocchi. La griglia li racchiude, occupandosi di sapere quali blocchi sono presenti e dove, ne saprà determinare la possibilità di collegamento e potrà riconoscere il suo completamento.

Capitolo 2

Design

2.1 Architettura

L’architettura di GameHub segue il pattern architetturale MVC (Model – View – Controller).

Il controller dell’applicazione è il MainController, che gestisce i menu (la view) e il cambio di contesto tra i vari GameController. Il MainController è di fatto un Observer dei GameController, dai quali riceve comunicazioni circa il loro stato (i.e. gioco in pausa, gioco finito); in base alle informazioni ricevute gestisce la view. Essa, invece, notifica al MainController le scelte dell’utente, il quale agisce di conseguenza notificando i GameController o gestendo la view stessa.

L’architettura è progettata in modo tale da permettere l’implementazione di un gioco a partire dal GameController, rendendolo responsabile del controllo del flusso di gioco.

In ogni gioco, presente o futuro, si è però liberi di organizzare la relativa architettura come meglio si crede, fermo restando che la scelta consigliata è il pattern MVC, utilizzando GameController come controller del gioco, GameView come view, e il proprio model. Infatti, l’interfaccia GameController è progettata per essere utilizzata per interfacciare la GameView con il model, senza alcun riferimento esplicito a determinate implementazioni degli stessi.

I vantaggi nell’adottare questa architettura sono i seguenti:

* Facilità nel modificare o estendere le interazioni tra i GameController e la view dei menu, grazie al fatto che i GameController interagiscono con essa solo attraverso il MainController
* In caso di sostituzione in blocco dell’implementazione della view dei menu, nessuna modifica andrà fatta al MainController, né tanto meno ai GameController. Diverso è il caso della GameView, la cui gestione dell’implementazione rientra nel pattern architetturale scelto nel singolo gioco