**“Ingegneria del Software”**

**2021-2022**

**Docente: Prof. Angelo Furfaro**

**<nome progetto>**

|  |  |
| --- | --- |
| **Data** | <gg/mm/aaaa> |
| **Documento** | Documento Finale – D3 |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Nome e Cognome** | **Matricola** | **E-mail address** |
| **De Marco Luca** | **234373** | **Dmrlcu01d24a053x@studenti.unical.it** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Sommario

**Non è stata trovata alcuna voce d'indice.**

## List of Challenging/Risky Requirements or Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

A. Stato dell’Arte

Per cercare di comprendere meglio il dominio applicativo del gioco KenKen, è stata condotta un’analisi di alcune implementazioni di giochi KenKen disponibili online, sotto forma di applicazione web e mobile.

Questa analisi ha rilevato una varietà di approcci per quanto riguarda l’interfaccia utente (UI), le modalità di interazione, come l’inserimento dei numeri all’interno delle celle, la presentazione delle gabbie e delle regole, e le funzionalità offerte, come il controllo degli errori, i suggerimenti o la risoluzione automatica del gioco.

Dall’esame di queste soluzioni esistenti, sono emersi alcuni elementi particolarmente interessanti per la loro chiarezza e facilità d’uso, che si ritiene possano contribuire positivamente all’esperienza utente nell’uso del gioco. L’obiettivo non è replicare una specifica implementazione, ma trarre ispirazione dagli aspetti più riusciti osservati, combinandoli in modo coerente con i requisiti del progetto.

In particolare, si è notato come alcune interfacce web, come il sito [www.kenkenpuzzle.com](http://www.kenkenpuzzle.com) presentino la griglia e le gabbie in modo chiaro e leggibile. Questo approccio grafico sarà preso come riferimento per la progettazione dell’UI del progetto, in quanto permette all’utente di comprendere rapidamente lo schema di gioco. Anche per quanto riguarda l’inserimento dei valori numerici offre un metodo abbastanza intuitivo, si cercherà di implementare una soluzione simile, così da avere una maggiore facilità di gioco ed evitare l’inserimento di valori non concessi all’interno delle celle.

B. Raffinamento dei Requisiti

*A partire dai servizi minimali richiesti, raffinate la descrizione dei servizi offerti dal vostro applicativo. Descrivete anche I requisiti non funzionali*

***A.1 Servizi (con prioritizzazione)***

* **ID:** RF01
  + **Nome Servizio:** Definizione Parametri Griglia KenKen
  + **Descrizione Dettagliata:** L’applicazione deve consentire all’utente di iniziare la creazione di un nuovo puzzle KenKen specificando la dimensione della griglia che deve essere necessariamente N (dove N indica la dimensione delle righe e delle colonne, in quanto il puzzle deve essere necessariamente quadrato), da un minimo di 3 ad un massimo di 6, tramite un elemento dell’UI. Questa selezione determinerà la creazione di una griglia di dimensione N su cui verranno poi create le gabbie. Per quanto riguarda la scelta della modalità di gioco l’utente potrà selezionare tra le diverse tipologie, che consistono nei vari tipi di operazioni matematiche (ossia +, -, x, ÷), ci saranno 3 tipi di modalità. Nella modalità easy saranno presenti solo operazioni di addizione e sottrazione, nella modalità media saranno presenti operazioni di moltiplicazione e divisione, mentre nella modalità hard saranno presenti tutti i tipi di operazioni. La scelta di dimensione e di vincolo delle operazioni determineranno la difficoltà del puzzle.
  + **Soluzione Concettuale:** È necessaria una funzione che presenti all’utente l’opzione per determinare la dimensione della griglia e della scelta dei vincoli. La scelta dell’utente viene memorizzata e utilizzata per inizializzare la struttura dati principale che rappresenta la griglia di gioco.
  + **Priorità:** 
    - Importanza: Alta (È il primo passo indispensabile per la creazione del puzzle)
    - Complessità: Bassa (Input facile da gestire)
* **ID:** RF02
  + **Nome Servizio:** Generazione Automatica Puzzle KenKen
  + **Descrizione Dettagliata:** L’applicazione deve avere la capacità di generare proceduralmente un nuovo Puzzle KenKen valido. L’utente dovrà poter richiedere la generazione specificando la dimensione N della griglia e la modalità di gioco. L’algoritmo dovrà creare una disposizione valida di gabbie, con i relativi target numerici e utilizzando operatori aritmetici conformi alla modalità scelta (vedi **RF01**), assicurando che il puzzle abbia almeno una soluzione disponibile (idealmente una). La configurazione viene quindi rappresentata all’utente così che possa iniziare a giocare
  + **Soluzione Concettuale:** È necessario sviluppare un algoritmo di generazione del Puzzle KenKen. Questo algoritmo parte da una griglia latina completa (una griglia già risolta, quindi una potenziale soluzione), la suddivide in modo casuale, calcola i vincoli per ciascuna gabbia, e verifica le proprietà del puzzle risultante.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Alta
* **ID:** RF03 (Pattern Memento)
  + **Nome Servizio:** Salvataggio Stato Partita Corrente
  + **Descrizione Dettagliata:** Il sistema deve fornire all’utente la funzionalità di salvare lo stato corrente della partita, così che possa interromperla e riprenderla in un momento successivo dallo stesso punto. L’operazione di salvataggio verrà avviata dall’utente tramite un comando esplicito nell’interfaccia grafica. Il salvataggio deve catturare e memorizzare in maniera persistente su un repository le informazioni necessarie a ricostruire la sessione di gioco. Obbligatoriamente deve includere la dimensione N del puzzle, la modalità selezionata, la struttura dettagliata di ogni gabbia (le celle che la compongono, l’operatore e il numero target), i vari numeri immessi dall’utente nelle varie celle se presenti, specificandone la riga e la colonna.
  + **Soluzione Concettuale:** Il servizio richiede una funzione capace di fare uno snapshot all’intero stato della partita al momento della richiesta di salvataggio. Questa istantanea deve includere sia la configurazione statica del puzzle ma anche lo stato dinamico ossia gli input dell’utente. Questo snapshot deve essere serializzato in un formato dati standardizzato e persistente. I dati serializzati vengono scritti sul supporto di memorizzazione prescelto.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Alta
* **ID:** RF04 ( Pattern Observer)
  + **Nome Servizio:** Inserimento Numeri Utente
  + **Descrizione Dettagliata:** Una volta visualizzato il puzzle KenKen (generato tramite RF02 o caricato da un salvataggio), il sistema deve permettere all’utente di interagire con esso per tentare di risolverlo. Una volta selezionata una cella l’utente deve poter inserire in essa un valore numerico. Il sistema deve effettuare una validazione di base dell’input: sono accettati solo valori interi compresi nell’intervallo da 1 ad N, dove N è la dimensione della griglia. Qualsiasi input non valido deve essere segnalato come errore. Se il numero è valido deve essere visualizzato all’interno della cella. L’utente deve inoltre poter cancellare un numero precedentemente inserito in una cella riportandolo allo stato di vuoto.
  + **Soluzione Concettuale:** È necessario implementare un meccanismo nell’interfaccia utente che permetta all’utente di indicare univocamente la cella bersaglio della sua azione. Seve poi un gestore di eventi per catturare l’input numerico proveniente dalla fonte di scelta. Una logica di validazione deve verificare che l’input sia intero consentito. Lo stato interno del gioco che rappresenta i valori correnti in ogni cella, deve essere aggiornato in base all’input validato ricevuto.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Media
* **ID:** RF05
  + **Nome Servizio:** Controllo dei vincoli in tempo reale
  + **Descrizione Dettagliata:** Il sistema deve fornire all’utente la possibilità di abilitare/disabilitare un controllo automatico dei vincoli KenKen durante la fase di inserimento dei numeri (vedi RF04). Questa opzione deve essere controllabile tramite un elemento dell’UI. Quando è attiva la funzione deve verificare immediatamente, se il valore che l’utente inserisce o modifica, viola qualche vincolo (l’unicità dello stesso numero in riga e in colonna, se inserendo il numero la gabbia viene completata e il risultato dei numeri inseriti corrisponda al target assegnato), in caso di violazione di almeno uno di questi vincoli, il sistema deve fornire un feedback visivo immediato all’utente, senza impedire di continuare ma provare. Mentre se questa opzione non è attiva nessuno di questi controlli viene effettuati, ma deve essere cura dell’utente la responsabilità del soddisfacimento dei vincoli.
  + **Soluzione Concettuale:** Serve una variabile booleana che memorizzi se la modalità di controllo è attiva o meno. Deve essere implementata una logica di validazione che, prendendo in input lo stato attuale della griglia e la posizione della cella modificata, esegua i controlli di unicità su riga e colonna, e il controllo sulla gabbia. Questa logica di validazione deve essere invocata dopo ogni inserimento valido, ma solo se la modalità di controllo è attiva. Serve inoltre un meccanismo nell'interfaccia utente per applicare il feedback visivo richiesto in caso di violazione
  + **Priorità:** 
    - Importanza: Media
    - Complessità: Media
* **ID:** RF06
  + **Nome Servizio:** Calcolo e Visualizzazione Soluzione Puzzle
  + **Descrizione Dettagliata:** Il sistema deve mettere a disposizione dell’utente una funzione per calcolare automaticamente almeno una soluzione valida per il puzzle KenKen corrente. Questa funzione viene attivata su richiesta esplicita dell’utente. Il sistema dovrà implementare un algoritmo di risoluzione in grado di trovare le configurazioni complete della griglia che soddisfino simultaneamente tutti i vincoli del gioco, ossia unicità dei numeri per ogni riga e colonna, e rispetto del risultato e operatore specificato per ogni gabbia. L’algoritmo deve essere capace di trovare soluzioni multiple, se il puzzle ne ammette più di una. Per gestire questo e per limitare i tempi di calcolo si trovano le prime X possibili soluzioni esistenti scelte dall’utente tramite UI. L’algoritmo cercherà al massimo le prime X soluzioni valide esistenti e quelle trovate dovranno essere memorizzate. Se l’utente non inserirà nessun valore nel calcolo delle soluzioni il sistema di default calcolerà solo 3 soluzioni, ma allo stesso tempo se l’utente inserirà un numero maggiore di 100 il sistema non permetterà l’inserimento. Al termine del calcolo delle soluzioni l’utente visualizzerà un messaggio che informa il numero delle soluzioni trovate, e la possibilità di visualizzarle e navigare tra di esse.
  + **Soluzione Concettuale:** è richiesto un algoritmo risolutore per problemi e soddisfacimento dei vincoli, con la tecnica del backtracking ricorsivo.L’algoritmo dovrà esplorare lo spazio delle possibili assegnazioni numeriche alle celle vuote, verificando ad ogni passo la coerenza dei vincoli e tornando indietro in caso di violazione. Per trovare le soluzioni l’algoritmo si fermerà solo quando non trova più soluzioni possibili, o altrimenti quando avrà raggiunto il limite di soluzioni scelte dall’utente. Per la visualizzazione servirà una logica che gestisca l’indice della soluzione correntemente mostrata e aggiorni la griglia nell’interfaccia utente in base alla soluzione selezionata tramite dei comandi di navigazione. La logica deve gestire l’abilitazione dei pulsanti successivo/precedente in base alla soluzione mostrata.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Alta
* **ID:** RF07
  + **Nome Servizio:** Caricamento Partita Salvata
  + **Descrizione Dettagliata:** Il sistema deve consentire all’utente di caricare uno stato di partita precedentemente salvato (vedi RF03), permettendo all’utente di riprendere il gioco esattamente dal punto in cui era stato interrotto. È possibile attivare questa operazione tramite l’UI. Il sistema deve tentare di leggere i dati di salvataggio dal repository designato. Devono essere gestiti anche i casi in cui il salvataggio non esista o sia corrotto, informando l’utente con un messaggio appropriato. Se i dati vengono reperiti correttamente il sistema dovrà deserializzarli e ripristinare la partita, questo include la ricostruzione della definizione del puzzle così com’era al momento del salvataggio, ossia la dimensione, le gabbie, i target, operatori e il ripristino dei numeri inseriti dall’utente. Infine, l'interfaccia utente deve essere aggiornata per visualizzare fedelmente lo stato della partita caricata
  + **Soluzione Concettuale:** quando l’utente avvia la richiesta di caricamento, la prima fase consisterà nel leggere i dati grezzi dal supporto di memorizzazione persistente. È necessaria una fase di validazione e deserializzazione, in cui i dati vengono convertiti dal formato di salvataggio alla rappresentazione interna dello stato di gioco, questo processo deve gestire i potenziali errori. La logica applicativa usa quindi questi dati per ricostruire lo stato interno della partita. Infine, è necessario notificare o comandare all'interfaccia utente di aggiornarsi per riflettere completamente lo stato caricato.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Media
* **ID:** RF08
  + **Nome Servizio:** Avvio Nuova Partita
  + **Descrizione Dettagliata:** Il sistema deve fornire all’utente un modo per iniziare una nuova partita, questa funzionalità deve essere attivata tramite un comando esplicito. Se l’utente ha una partita in corso dovrà essere avvisato se desidera effettuare un salvataggio oppure no. Una volta effettuata questa operazione si dovrà procedere ad avviare una nuova partita, seguendo in sequenza i passi dettagliati nei requisiti RF01 ed RF02. Al termine della generazione il nuovo puzzle viene visualizzato nell’UI, pronto per essere risolto dal giocatore, interagendoci come spiegato in RF04 e RF05.
  + **Soluzione Concettuale:** è necessario un punto di attivazione dall’UI, questo comando dovrebbe prima verificare se ci sono modifiche non salvate nella partita corrente e, in caso, chiedere all'utente come procedere, poi deve invocare la logica di RF01 per permettere all’utente di inserire i parametri desiderati, una volta ottenuti i parametri si passa alla logica di RF02, dopo di che si passa ad abilitare le funzioni di gioco come da RF04
  + **Priorità:** 
    - Importanza: Alta
    - Complessità: Media
* **ID:** RF09
  + **Nome Servizio:** Gestione Fine Partita
  + **Descrizione Dettagliata:** Una volta riempita la griglia il sistema deve verificare automaticamente la correttezza del puzzle. L’attivazione di questa verifica è l’inserimento di un numero nell’ultima cella vuota, una volta attivo verificherà che la configurazione inserita dall’utente soddisfi i vincoli contemporaneamente, quindi l’unicità della riga e colonna e il vincolo della gabbia. In base all’esito della verifica il sistema notificherà all’utente un messaggio positivo se il puzzle è corretto, altrimenti un messaggio negativo se il puzzle risulta non coerente con i vincoli.In caso di successo, l'interazione con la griglia risolta potrebbe essere disabilitata e/o potrebbe essere proposta una nuova partita (RF08). In caso di errore, l'utente potrà continuare a modificare la griglia (RF04).
  + **Soluzione Concettuale:** è necessario un meccanismo che rilevi in automatico l’inserimento di un numero nell’ultima cella vuota e che verifichi la correttezza del puzzle. Tale funzione deve implementare la logica per iterare su tutte le righe, colonne e gabbie della griglia, verificando il rispetto di tutti i vincoli KenKen. La funzione restituirà un risultato indicante la correttezza o meno della soluzione e, opzionalmente, informazioni sugli errori specifici riscontrati. In base al risultato della funzione di validazione, una logica di presentazione mostrerà il feedback appropriato all'utente attraverso l'interfaccia (messaggio di successo o errore). Se è prevista l'evidenziazione degli errori, questa logica dovrà mappare gli errori identificati dalla funzione di validazione agli specifici elementi della UI da modificare visivamente. Dovrà anche gestire lo stato dell'applicazione post-successo, come la proposta di una nuova partita e bloccare l’input sul puzzle appena risolto, o permettere la continuazione del gioco in caso di errore.
  + **Priorità:**
    - Importanza: Alta
    - Complessità: Media
* **ID:** RF10
  + **Nome Servizio:** Reset Griglia
  + **Descrizione Dettagliata:** Il sistema deve fornire all'utente un comando per resettare lo stato del puzzle corrente, cancellando tutti i numeri inseriti dal giocatore fino a quel momento. Questo permette all'utente di ricominciare il tentativo di soluzione dello stesso puzzle senza doverne generare o caricare uno nuovo. Questa funzione viene attivata tramite un comando esplicito dell'utente. Alla ricezione di questo comando si deve identificare ogni cella in cui l’utente abbia inserito un numero, rimuovere il valore inserito riportando la cella allo stato di vuoto, aggiornando l’UI.
  + **Soluzione Concettuale:** è necessario un comando attivabile dall'utente nell'interfaccia. La logica associata a questo comando deve accedere allo stato interno della griglia. Per ogni cella, deve verificare se essa contiene un numero inserito dall'utente, e resettarlo al suo stato iniziale.
  + **Priorità:**
    - Importanza: Media
    - Complessità: Bassa

***A.2 Requisiti non Funzionali***

* **ID:** RNF-US-01
  + **Nome Requisito:** Intuitività Interfaccia Utente
  + **Descrizione Dettagliata:** L’interfaccia utente deve essere progettata in modo da risultare intuitiva e facile da apprendere per le funzioni principali, come l’avvio della partita (vedi RF08), l’inserimento dei numeri (vedi RF04) o anche la richiesta della soluzione (vedi RF06), anche per gli utenti che non conoscono il gioco KenKen.
* **ID:** RNF-US-02
  + **Nome Requisito:** Chiarezza Feedback Utente
  + **Descrizione Dettagliata:** Il feedback fornito dal sistema all’utente, come i messaggi di errore o di successo (vedi RF07 e RF09) o l’evidenziazione dei vincoli violati (vedi RF05), deve essere tempestivo, non ambiguo e facilmente visibile all’interno dell’interfaccia.
* **ID:** RNF-PE-01
  + **Nome Requisito:** Prestazioni Calcolo Soluzione
  + **Descrizione Dettagliata:** Il tempo richiesto per il calcolo delle prime 5 soluzioni (vedi RF06) per una griglia di dimensione massima (6x6) non deve superare i 15 secondi.
* **ID:** RNF-PE-02
  + **Nome Requisito:** Reattività Interfaccia Utente
  + **Descrizione Dettagliata:** L’interfaccia utente deve rispondere alle azioni comuni dell’utente, come la selezione di una cella o l’inserimento di un numero (vedi RF04) senza ritardi percepibili, entro 0.5 secondi.
* **ID:** RNF-AF-01
  + **Nome Requisito:** Correttezza Algoritmi
  + **Descrizione Dettagliata:** Gli algoritmi implementati per la generazione dei puzzle (vedi RF02), il calcolo delle soluzioni (vedi RF06) e la verifica della soluzione finale (vedi RF09) devono essere corretti rispetto alle regole matematiche e logiche del KenKen.
* **ID:** RNF-AF-02
  + **Nome Requisito:** Affidabilità Persistenza Dati
  + **Descrizione Dettagliata:** Il salvataggio (vedi RF03) e il caricamento dello stato della partita (vedi RF07) non devono introdurre corruzione o perdita dei dati dell’utente

***A.3 Scenari d’uso dettagliati***

*Descrivere gli scenari più comuni, più interessanti, o più complicati d’uso dei vostri servizi.*

**Utente 🡪 U, Sistema 🡪 S.**

* Scenario 1: Avvio Nuova Partita

1. **U:** Avvia l’applicazione Puzzle KenKen Solver
2. **S:** Visualizza home applicazione
3. **U:** Seleziona il comando “New Game” (RF08)
4. **S:** Presenta le opzioni per i parametri (RF01)
5. **U:** Selezione dimensione “4x4” e modalità “Easy”.Conferma selezione
6. **S:** Esegue la generazione del puzzle (RF02) con i parametri scelti. Visualizza la nuova griglia 4x4 con le gabbie e i vincoli (+ e -), pronta per l’inserimento dei numeri.
7. **U:** Selezione una cella modificabile (RF04), e inserisce il numero “2”
8. **S:** Visualizza “2” nella ella selezionata. Se il controllo dei vincoli è attivo (RF05), verifica che i vinicoli siano rispettati e fornisce eventuale feedback

* Scenario 2: Risoluzione Partita e Navigazione Soluzione

1. **U:** Ha caricato un puzzle di dimensione 5x5 in modalità “Hard”, con alcuni numeri già inseriti. È rimasto bloccato, non riesce a proseguire con il gioco. Quindi seleziona il comando “Show Solution” (RF06), specificando 7 soluzioni.
2. **S­:** Avvia l’algoritmo di calcolo (RF06). Trova solo 3 soluzioni ammissibili, le memorizza. Visualizza un messaggio “Find 3 solutions”. Mostra la prima soluzione completa sulla griglia. Attiva il pulsante di navigazione “Next”
3. **U:** Clicca sul pulsante “Next”
4. **S:** Aggiorna la griglia visualizzando la soluzione trovata. Aggiorna lo stato dei pulsanti di navigazione attivando sia “Next” che “Previous”
5. **U:** Clicca nuovamente su “Next”
6. **S:** Mostra la terza e ultima soluzione, disabilitando il pulsante “Next”

***A.4 Excluded Requirements***

*Descrivere i servizi eventualmente i esclusi, e spiegare il perchè*

1. **Generazione Puzzle Avanzata:** 
   * Descrizione: l’algoritmo di generazione (RF02) non garantirà proprietà avanzate come l’esistenza di una soluzione unica per ogni puzzle generato, ne fornirà livelli di difficoltà calibrati in modo fine.
   * Motivazione: implementare algoritmi che garantiscano soluzione unica è difficoltà precisa è significativamente più complesso
2. **Funzionalità di Gioco Avanzate:** 
   * Descrizione: Non saranno implementate funzionalità come le “Pencil Marks”, ossia la funzionalità di annotare numeri nelle celle, o funzioni come “Undo/Redo” per le mosse
   * Motivazione: Aggiungono complessità all’interfaccia utente e alla gestione dello stato
3. **Caricamento di Puzzle da Fonti Esterne:** 
   * Descrizione: L’applicazione permetterà solo di caricare lo stato di partite salvate internamente (RF07). Non sarà possibile caricare definizioni di puzzle KenKen da file esterni, ad esempio scaricati da internet o creati da altri programmi.
   * Motivazione: la gestione di formati esterni e relativa validazione complicherebbero lo sviluppo, l’applicazione si basa sui puzzle generati internamente.
4. **Funzionalità UI aggiuntive:**
   * Descrizione: Non sono previste funzionalità di personalizzazione dell’aspetto grafico, come i temi o i colori.
   * Motivazione: Considerate come abbellimenti non sono necessari al funzionamento dell’applicazione
5. **Editor Manuale di Puzzle:**
   * Descrizione: Non sarà implementata una modalità “Editor” che permetta all’utente di creare manualmente i propri puzzle KenKen disegnando gabbie o specificando numeri/operatori.
   * Motivazione: Il requisito “Specificare la configurazione” è stato interpretato e soddisfatto tramite la selezione dei parametri (RF01) e la generazione automatica (RF02), per focalizzare il progetto sul gioco e risoluzione, piuttosto che sulla creazione manuale, ritenuta troppo complessa

***A.5 Assunzioni***

*<Briefly document, in this section, the most relevant requirement assumptions/decisions you had to made during your project>*

Le assunzioni e decisioni più rilevanti prese durante il raffinamento dei requisiti sono:

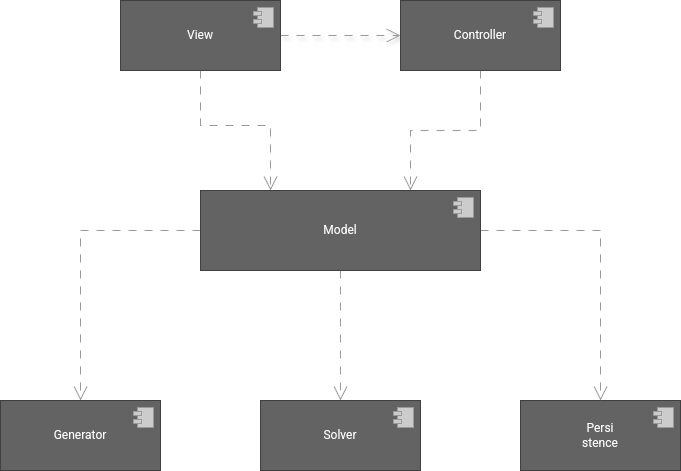
* Si assume che l'utente conosca le regole base del gioco KenKen.
* Il requisito di "specificare la configurazione" è stato interpretato come selezione di parametri (RF01) per la generazione automatica (RF02), escludendo un editor manuale.
* Il requisito di "salvare/caricare” è stato interpretato come salvataggio/caricamento dello stato completo della partita (RF03/RF07).
* L'algoritmo di generazione (RF02) non garantisce l'unicità della soluzione.
* L'utente può richiedere al massimo 100 soluzioni (RF06), con un default di 3, per motivi di performance.
* La tecnologia GUI scelta è Java Swing
* La persistenza (RF03/RF07) avverrà su Data Base MySQL.
* La dimensione della griglia (RF01) è limitata a N compreso tra 3 e 6

***.6 Use Case Diagrams***

Immagine che contiene testo, schermata, diagramma, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.C. Architettura Software   
*<IF RELEVANT, Report here both the static and the dynamic view of your system design, in terms of a Component Diagram, and their related Sequence Diagrams >*

***C.1 The static view of the system: Component Diagram***

**

***C.2 The dynamic view of the software architecture: Sequence Diagram***

Salvataggio Partita:

Immagine che contiene schermata, design

Il contenuto generato dall'IA potrebbe non essere corretto.

Avvio Nuova Partita:

Immagine che contiene schermata, diagramma, linea, design

Il contenuto generato dall'IA potrebbe non essere corretto.

Calcolo Soluzione e Navigazione:

Immagine che contiene schermata, design

Il contenuto generato dall'IA potrebbe non essere corretto.

D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS)

*Definite le sorgenti di dati a voi necessarie per realizzare I servizi di cui sopra. Modellate tali dati tramite un ER o similari. Specificate se e quali di tali dati sono gia’ forniti da applicativi esistenti.*

E. Scelte Progettuali (Design Decisions)   
<Document here the **5** most important design decisions you had to take. You can use both a textual or a diagrammatic specification.>

F. Progettazione di Basso Livello

G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs) *<Report in this section how the architectural and low level design you produced satisfies the FRs and the NFRs>*

Appendix. Prototype   
*<Provide a brief report on your prototype, and especially: information on what you have implemented, how the implementation covers the FR and NFR, how the prototypes demonstrates your project correctness with respect to the FR and NFR. You may add some screenshots to describe what required above. Be ready to show your prototype during the oral examination>* 