

PROGETTO INGEGNERIA DEL SOFTWARE – FAKE INVADERS

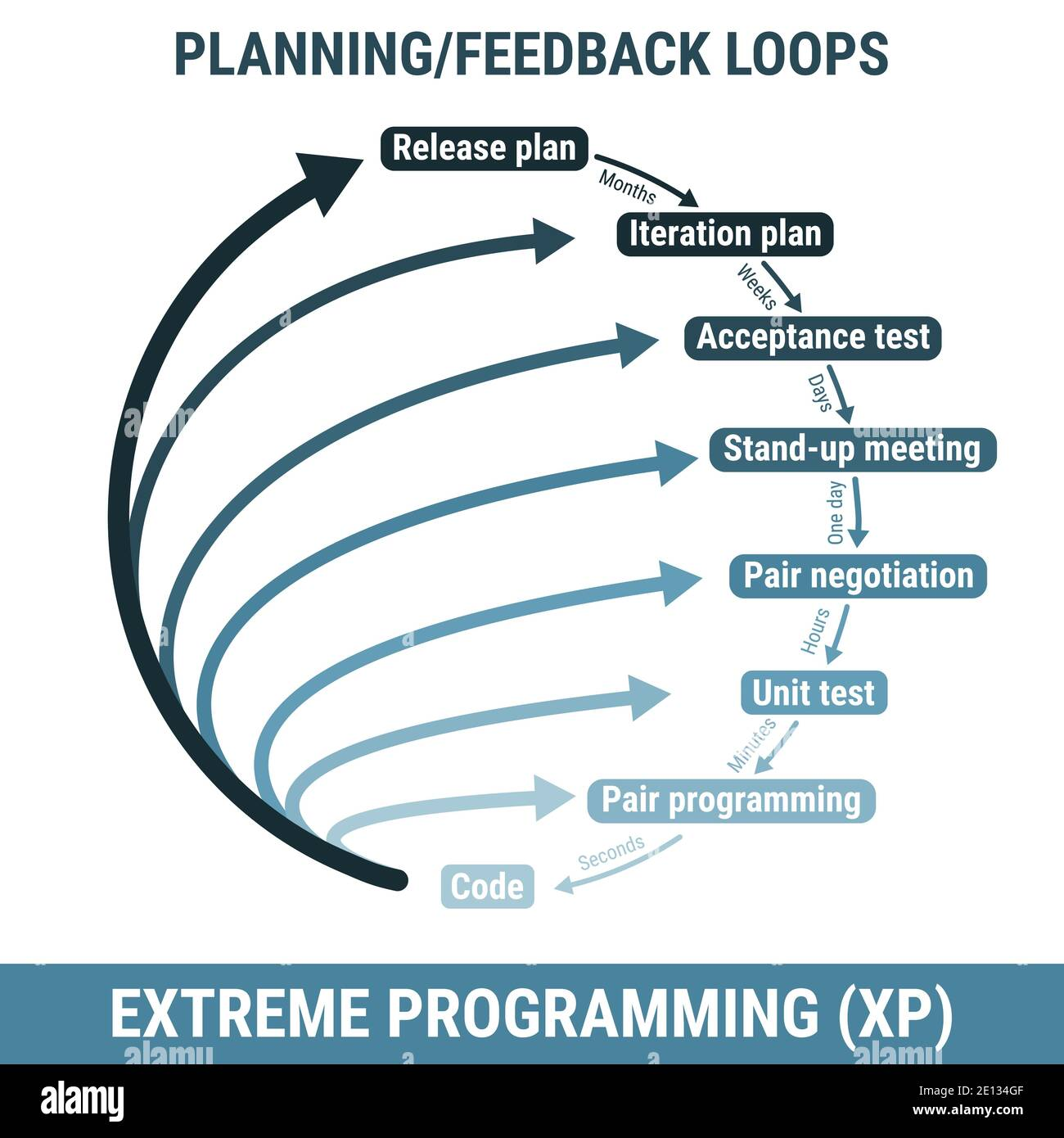
Tomasoni Francesco matr. 1080980

Maccari Luca matr. 1081951

1. Software Life Cycle

Il metodo di sviluppo che è stato adottato è il metodo agile: Extreme Programming (XP). Questa scelta è stata motivata dalla dimensione molto ridotta del nostro gruppo, poiché XP si adatta alla perfezione con una “coppia di programmatori che si guarda le spalle”.

Durante tutta la fase di progettazione del software è stato seguito il ciclo dell’XP di seguito riportato.



2. Configuration Management

Tutto il lavoro svolto, che si tratti di documentazione o di codice, viene salvato in una repository su GitHub in condivisione con tutti i membri del team; le comunicazioni urgenti sono state gestite mediante l’uso di Discord.

Struttura del progetto:  
All’interno della repository sono presenti 2 cartelle:

* **Codice**: in cui è presente sia il codice che le risorse necessarie a farlo funzionare
* **Docs**: vi è un file contenente la documentazione completa del progetto, e una cartella contenente i diagrammi UML.

Branch utilizzati:

* **Main**: main principale.
* **Branch** individuali: Francesco e Luca.

Entrambi i membri del gruppo hanno la facoltà di caricare o apportare modifiche agli elementi presenti nella repository Main in qualsiasi momento.

In particolare, per modifiche di rilevanza, specialmente nel codice, sono stati seguiti tutti i passaggi delineati nel paragrafo "Software Life Cycle".

La repository è stata mantenuta costantemente aggiornata, includendo ogni documentazione prodotta.

3. People Management and Team Organization

Nella realizzazione del Progetto è stata adottata una struttura organizzativa non gerarchica per migliorare la flessibilità e l'adattabilità alle esigenze del progetto, ponendo il focus anche sulle disponibilità dei singoli individui coinvolti.

In questa configurazione, non vi è una suddivisione prefissata del lavoro, è compito del team decidere autonomamente come suddividere e assegnare le diverse attività. In situazioni di difficoltà, entrambi i membri del team hanno la possibilità di collaborare, affiancandosi reciprocamente per risolvere le problematiche in modo collettivo.

L'approccio adottato, noto come *adhocrazia* è caratterizzato da grande capacità di adattamento, offre ulteriori vantaggi, contribuendo a creare un ambiente più sereno e informale, favorevole al benessere dei membri del team.

4. Software Quality

Il team si è prefissato di sviluppare un'applicazione che rispetti i parametri e gli attributi di qualità definiti da *McCall-Richards-Walters* nel documento redatto nel 1977. Di seguito sono elencati suddivisi in categorie.

### Operatività del software

* **Correttezza:** L'applicazione soddisfa i requisiti e le specifiche definite all'inizio del progetto. Eventuali funzionalità non ancora implementate verranno aggiunte mediante futuri aggiornamenti del software.
* **Affidabilità:** L'applicazione è stata sottoposta a diversi test, risultando con il minor numero possibile di errori.
* **Efficienza:** Il consumo di risorse è ottimizzato, eliminando la necessità di hardware potente o una grande quantità di RAM. Il peso dell'applicazione è limitato a pochi MB.
* **Integrità:** il software è sicuro, in quanto è necessaria un’autenticazione con username e password.
* **Usabilità:** Il prodotto è progettato per essere intuitivo e non richiede competenze particolari per essere utilizzato. L'applicazione è ottimizzata su tutte le piattaforme desktop grazie all'utilizzo di librerie compatibili con ciascun sistema operativo, garantendo una grafica uniforme.

Revisione del software

* **Manutenibilità**: Il team si è impegnato per organizzare il codice al meglio, per poterlo mantenere facilmente
* **Testabilità**: Tutte le funzionalità incluse sono testabili in qualunque momento tramite test.
* **Flessibilità**: Grazie all' organizzazione del codice, il software è flessibile, garantendo semplificazioni in eventuali ulteriori modifiche e personalizzazioni al programma.

### Transizione verso un nuovo ambiente

* **Portabilità**: Il gioco è nativamente eseguibile sui più diffusi sistemi operativi desktop (Windows, Linux, MacOS) dotati di un hardware relativamente recente e una versione di Java installata al loro interno.
* **Riusabilità**: Il codice è stato scritto in modo modulare, per poter essere riutilizzabile in futuro.

Requisiti base del software:

* Avere un pc con sistema operativo Windows, Linux o MacOS
* Avere una scheda video (anche integrata)

5. Requirement Engineering

La prima fase dello sviluppo è stata focalizzata sulla definizione dei requisiti necessari per la realizzazione software. Poiché la creazione del gioco non è stata commissionata, ma è stata un'idea spontanea del team, i requisiti sono stati determinati in base a ciò che il team considerava essenziale, tenendo conto della conoscenza del videogioco arcade da cui il software ha preso ispirazione, cercando di non snaturarlo ma rinnovarlo ed implementare nuove funzionalità.

I requisiti sono stati delineati seguendo il modello *MoSCoW*:

**Must Have:**

* Sviluppo di una versione base del gioco funzionante.
* Interfaccia grafica intuitiva e adatta a tutti i tipi di giocatori.
* Stabilità e sicurezza del software.

**Should Have:**

* Impostazioni di gioco.
* Modalità multigiocatore

**Could have:**

* Effetti sonori.
* Traduzione del gioco in altre lingue.

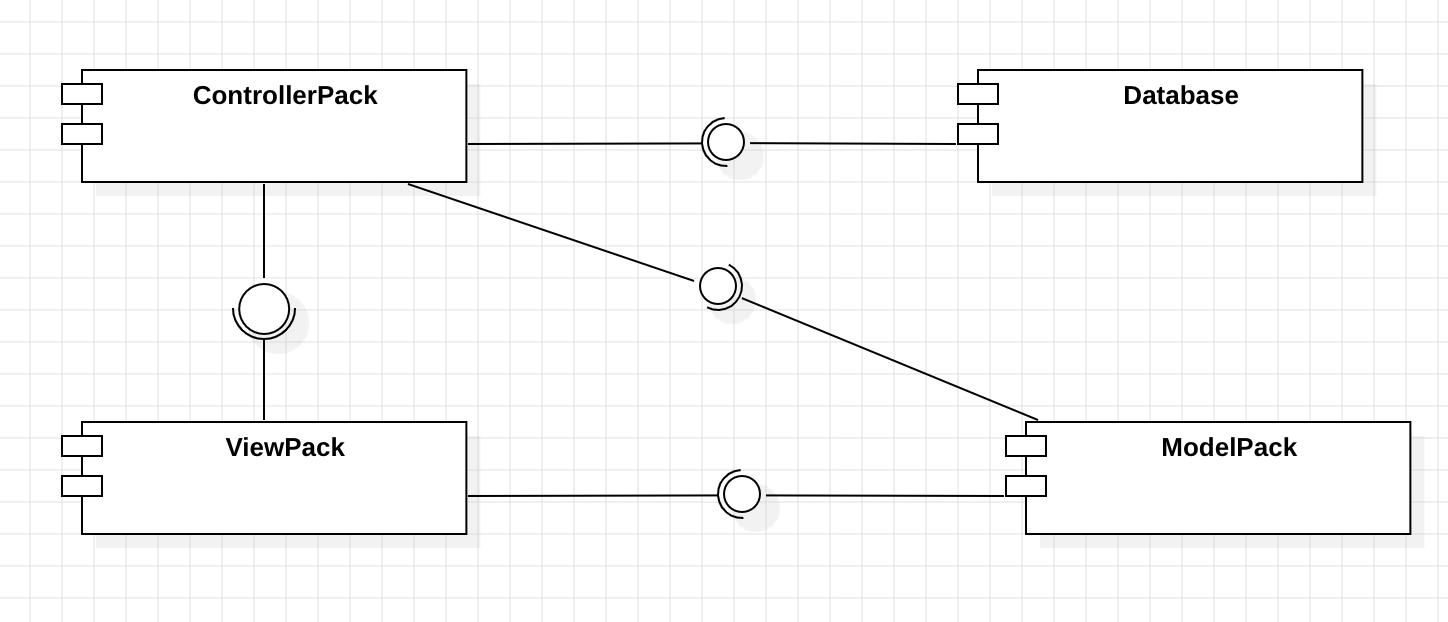
**Won't Have:**

* Impostazioni per modificare la grafica.

6. Modelling

la modellazione è avvenuta tramite l’utilizzo dei seguenti diagrammi UML:

* use case diagram
* class diagram
* state machine diagram
* sequence diagram
* activity diagram
* component diagram

Di seguito riportato il diagramma delle componenti che rappresenta l’architettura del software:

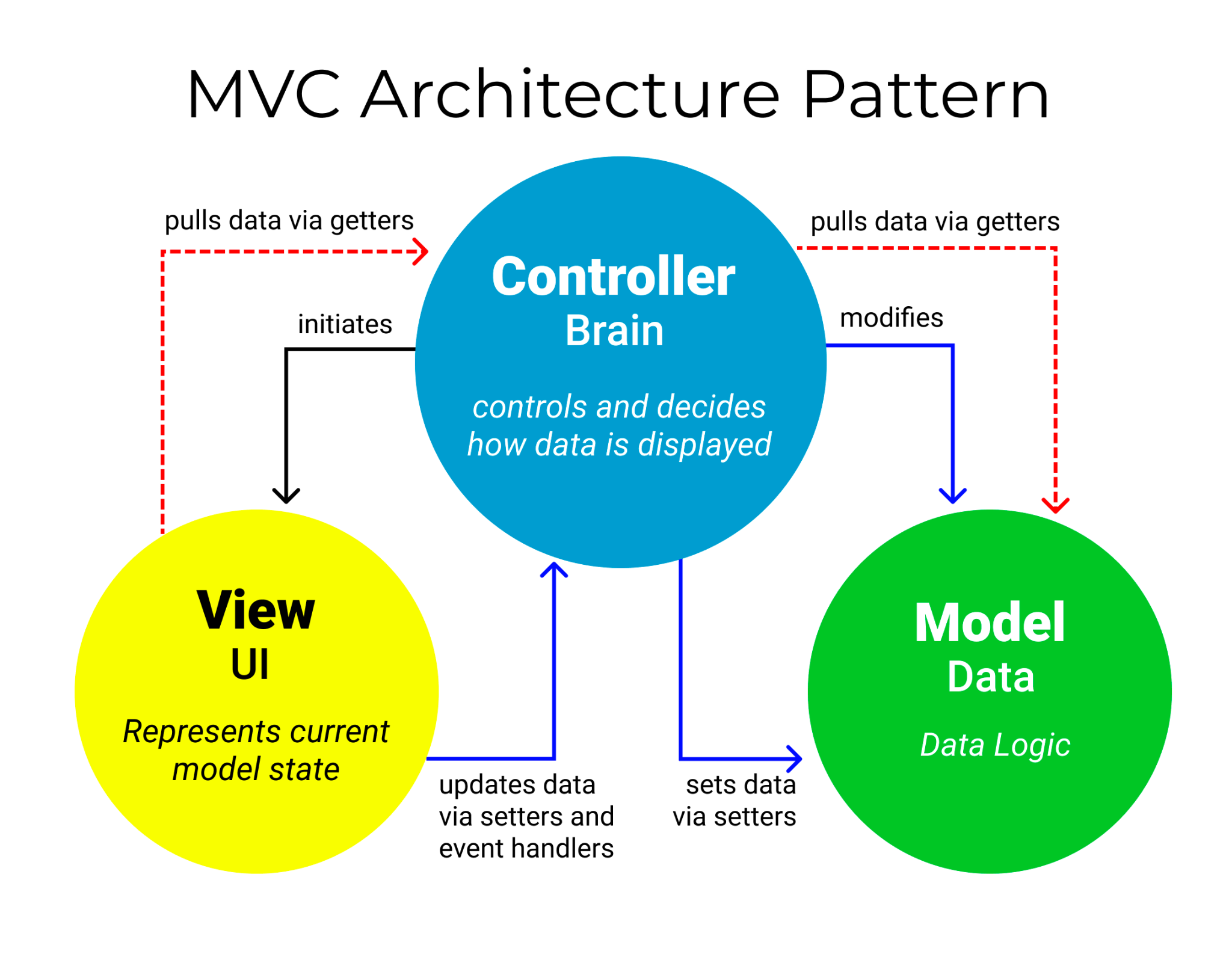
7. Software Architecture

Il software è stato progettato mediante l’ausilio del pattern *Model-View-controller* (*MVC)*, dividendo il codice in package distinti, tra cui: **ModelPack**, **ViewPack**, **ControllerPack**.

**Model**: Nel package ModelPack risiedono le classi che definiscono la logica di business e gestiscono i dati. Il Model interagisce con il database (database package) per recuperare, aggiornare o salvare informazioni.

**View**: Il package Viewpack contiene le classi responsabili della GUI che si interfaccia con l’utente. e interagiscono con il Controller per la gestione delle azioni dell’utente. La separazione della logica di presentazione in questo package favorisce la modularità e la riusabilità del codice.

**Controller**: Il package ControllerPack contiene le classi Controller, che fungono da intermediari tra il Model e la View. I Controller gestiscono gli input utente, coordinando le azioni richieste e aggiornando il Model e la View di conseguenza.



8. Software Design

La descrizione del design è svolta mediante l’utilizzo di diagrammi UML, di seguito si puo trovare:

Diagramma delle attività

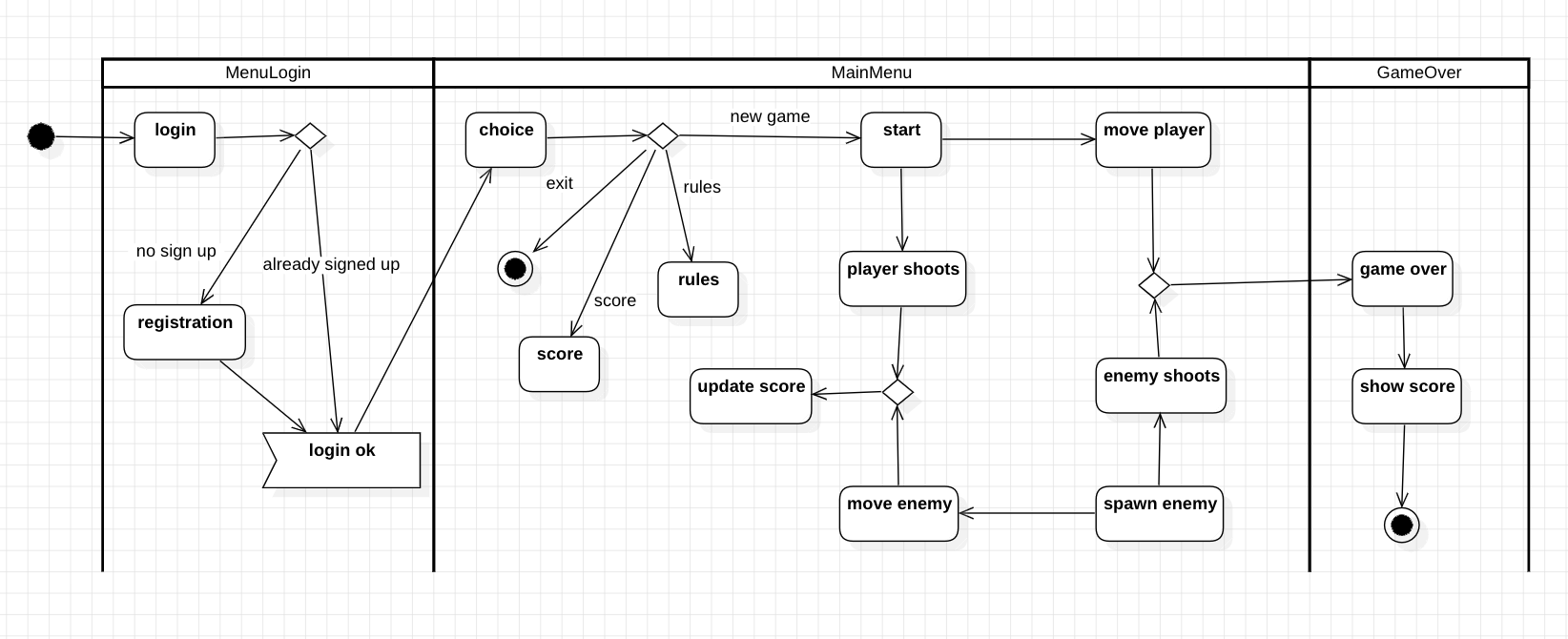


Diagramma delle sequenze che mostra il dialogo tra Player, Menu e Database

//foto diagramma

Metriche

//foto dei diagrammi di coderm

9. Software Testing

Per quanto riguarda la fase di testing del codice, fin dalle prime versioni del programma, anche non stabili o completamente funzionanti, si è fatto affidamento sulla fase di debug di Eclipse, impostando diversi breakpoint sul codice per visualizzare lo stato delle variabili durante l’esecuzione del programma.

Per testare le funzionalità che richiedevano una certa bravura del giocatore, o diversi minuti di gioco, sono stati inseriti all'interno del codice un gran numero di KeyListener collegati a diversi tasti per guadagnare punti istantaneamente, rendendo le prime operazioni di testing più rapide (per esempio nelle prime versioni del gioco, premendo il tasto “g” venivano eliminati tutti gli alieni in un colpo solo (Utile per il superamento di livelli senza perdere tempo nel gioco).

Ogni KeyListener è stato poi rimosso per evitare di inserire all’interno del gioco delle scorciatoie ai giocatori per superare i livelli più velocemente.

Per gestire le possibili problematiche di sicurezza del database, sono state inserite istruzioni SQL nelle apposite text-box per visualizzare l’effettiva sicurezza del database, portando quindi ad avere un’ulteriore attenzione alle righe di codice dedicate alla gestione del database embedded e garantendone la sicurezza dei dati.

Arrivati al termine dello sviluppo del codice, l’applicazione di test utilizzando JUnit è stata considerata troppo complessa a fronte di una deadline per la consegna del codice imminente. La fase di testing con JUnit è stata quindi riprogrammata al futuro, per poter testare appieno la robustezza del codice.

10. Software Maintenance

La manutenzione del codice è una fase essenziale nel ciclo di vita di esso e si divide in quattro tipologie differenti.

* Manutenzione **correttiva**
* Manutenzione **adattiva**
* Manutenzione **evolutiva**
* Manutenzione **preventiva**

In particolar modo, la manutenzione preventiva viene curata soprattutto nella prima parte dello sviluppo del codice, andando a simulare ogni comportamento possibile del singolo utente per prevenire eventuali bug ancora prima che vengano individuati dall’utente finale.

Una volta rilasciato il software, le manutenzioni correttive ed evolutive sono le due tipologie in primo piano. La manutenzione evolutiva è una fase avviata fin dal giorno 0 della release finale, per cercare di migliorare continuamente il gioco offrendo nuove funzionalità e rendendolo più accattivante. Viene sospesa quando si ha necessità di eseguire manutenzione correttiva dopo un feedback dell’utente.

Una volta ricevuto feedback dagli utenti tramite diversi canali di comunicazione più o meno informali, avviene la manutenzione correttiva, dove il problema indicato viene registrato, classificato in base alla sua priorità ed all’impatto che può avere sul gioco. Una volta individuata la gravità ed individuate le possibili soluzioni, viene avviata una fase di test per validare le correzioni al codice trovate dai membri del team. Questa operazione si ripete in modo iterativo per correggere ogni singolo problema riscontrato dagli utenti.

La fase di manutenzione adattiva non viene direttamente controllata o segnalata dagli utenti, bensì è il sistema operativo su cui viene fatto eseguire il programma a dettarne la necessità.