



ASTRO ALLIES

Ingegneria del Software a.a. 2024/25

TEAM: Marco Anghinetti, Andrea Cagnolati



Panoramica del Progetto

- Creazione di un gioco arcade
 - Space Invaders, Asteroid e simili
- Multigiocatore cooperativo
 - Un giocatore per il movimento
 - Altri giocatori per l'uso dei cannoni
- Applicativo Web
 - PHP, HTML, CSS, JavaScript
 - Container Docker



INDICE

1. PROGETTAZIONE
 - 1.1. Requisiti Utente
 - 1.2. Software Requirements Specification
 - 1.3. Story Cards
2. SVILUPPO
 - 2.1. Sistema
 - 2.2. Implementazione Classi
 - 2.3. Visualizzazione Client
 - 2.4. Input Client
3. DEBUGGING

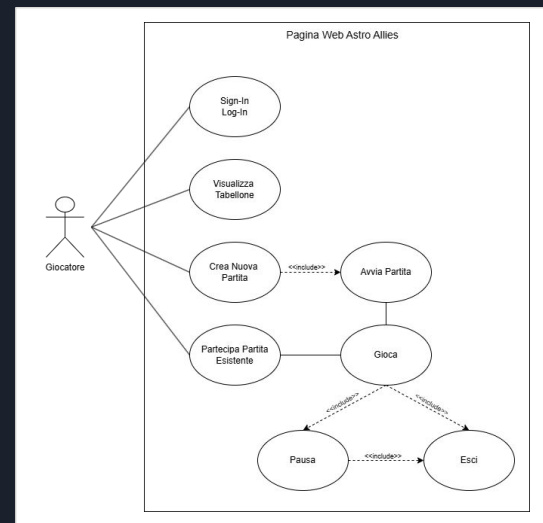
PROGETTAZIONE



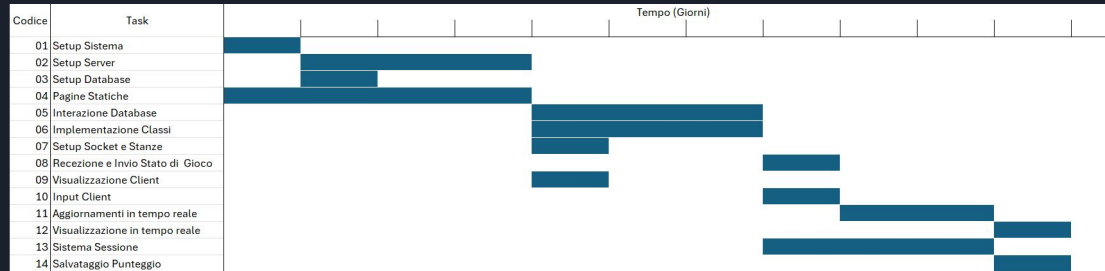
REQUISITI UTENTE

Descrizione formale delle richieste del “cliente”:

- Componenti del gioco
- Interfaccia Utente
- Esperienza di gioco



SOFTWARE REQUIREMENTS SPECIFICATION



Analisi tecnica dei requisiti utente:

- MoSCoW
- Vincoli, Ipotesi, Dipendenze
- Analisi Implementazione
- Pianificazione Sviluppo e Testing



STORY CARDS

Il codice univoco “AA_##” consente una ricerca rapida e netta distinzione del lavoro svolto.

Facile monitoraggio tempistiche durante lo sviluppo.

Story Card		
Codice _____	Priorità _____	Tempo Stimato _____
Descrizione		
Requisiti	Attività Svolta	
Note		
Sviluppatore _____	Data Inizio _____	Tempo effettivo _____

SVILUPPO



SISTEMA

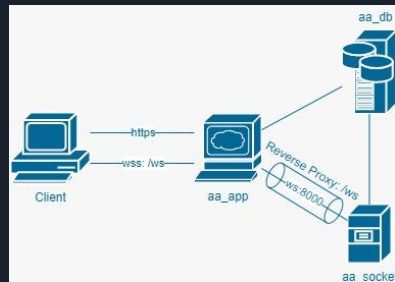
Container Docker:

- Database: mariadb 10.5
- Web Server: apache php 8.1
- Socket Server: alpine php 8.1

Garantisce portabilità ed indipendenza tra le parti del software.

La procedura per l'installazione, l'avvio e la gestione dei container è presente nel documento "Manuale Installazione".

Il Database istanzia lo schema al primo avvio del container tramite il file init.sql.

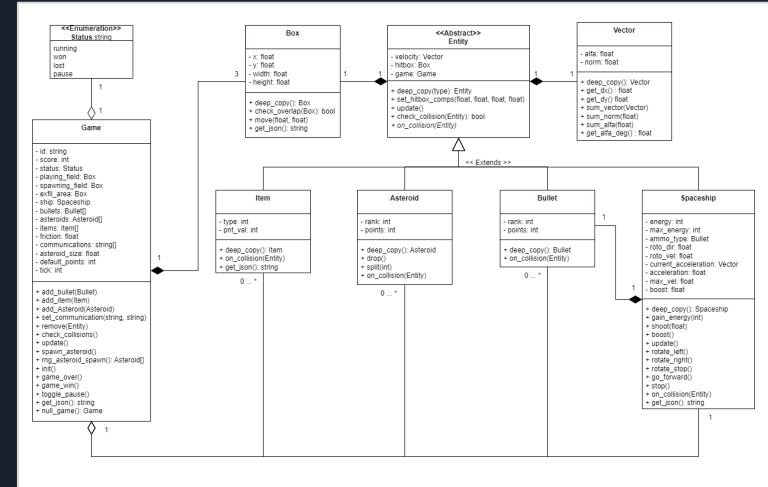


IMPLEMENTAZIONE CLASSI

Suddivisione in due fasi:

1. Sviluppo diagramma
2. Refactoring codice

Significativo il testing coordinato del software per il debugging.



VISUALIZZAZIONE CLIENT

Per la grafica è stato utilizzato l'elemento grafico HTML <canvas> ed aggiornato tramite procedure js.

Tipologie di Rendering :

- Cannone
- Oggetti e HUD

L'utilizzo degli sprite è stato gestito tramite dizionario e caching lato client.

capitano



cannoniere

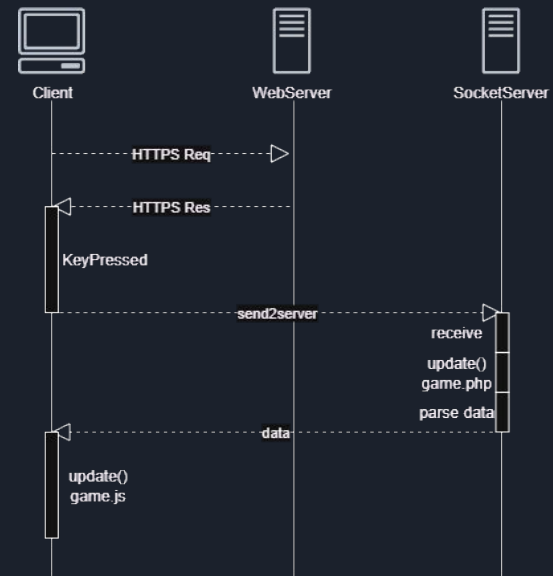


```
/**
 *
 * Pulizia canvas e redraw oggetti
 * @param {object} data = { asteroids: [{x, y, w, h, a}, ...],
 *                           items: [{x, y, w, h, type}, ...],
 *                           bullets: [{x, y, w, h}, ...],
 *                           ship: {x, y, w, h, a},
 *                           energy: int,
 *                           maxenergy: int,
 *                           score: int,
 *                           comms: {string:int , ...},
 *                           status: string
 *                           };
 */
```

INPUT CLIENT

Creazione dei controller js per gli input del client e l'invio al server.

- Distinzione capitano / cannoniere
- Pool di tasti accettati dal server



DEBUGGING





DEBUGGING

Due fasi di testing:

1. Completamento del task
2. Coordinato al merge

Il testing coordinato ha permesso il rilevamento di bug significativi.

<u>CAUSA</u>	<u>EFFETTO</u>
Sistema di riferimento del canvas	Spawn e comportamento asteroidi anomalo
	Dimensioni campo di gioco e zona di spawn traslati
Stima errata dei valori	Movimenti anomali
Analisi errata	Accelerazione anomala spaceship