


Dino Run and Jump

1	Introduzione	4
1.1	Informazioni sul progetto	4
1.2	Abstract	4
1.3	Scopo	4
2	Analisi	5
2.1	Analisi del dominio	5
2.2	Analisi e specifica dei requisiti	5
2.3	Use case	8
2.4	Pianificazione	9
2.5	Analisi dei mezzi	10
3	Progettazione	11
3.1	Design dell'architettura del sistema	11
3.2	Design dei dati e database	11
3.3	Design delle interfacce	12
3.3.1	Design interfacce telefono:	12
3.3.2	Design interfacce computer	14
3.4	Design procedurale	16
4	Implementazione	17
4.1	Refactoring	17
4.2	Struttura cartelle del progetto	17
4.3	docs/css/bootstrap.css	17
4.4	docs/css/game.css	17
4.5	docs/js/Bootstrap.bundle.min.js	17
4.6	docs/js/game.js	17
4.6.1	blockInput	17
4.6.2	getOS	Errore. Il segnalibro non è definito.
4.6.3	handleMotion	18
4.6.4	requestPermission	18
4.7	docs/js/index.js	18
4.7.1	writeMedals	18
4.7.2	connectToGame	18
4.7.3	generateGuestId	18
4.7.4	jump	18
4.7.5	registerNewUser	18
4.7.6	loginUser	18
4.7.7	logoutUser	18
4.7.8	openUserInformation	19
4.7.9	generateSession	19
4.7.10	changeDinoColor	19
4.7.11	saveDinoColor	19
4.7.12	showUserInformation	19
4.7.13	watchGame	19
4.7.14	checkLoggedUser	19
4.7.15	getIsTouchingDown	19
4.7.16	firebase.auth().onAuthStateChanged((user) => {	19
4.8	docs/GUI/bacheca.html	19
4.9	docs/GUI/collegamentoPartita.html	19
4.10	docs/GUI/game.html	19
4.11	docs/GUI/login.html	19
4.12	docs/GUI/paginaUtente.html	19
4.13	docs/GUI/personalizzaDino.html	20
4.14	docs/Game/js/game.js	20
4.15	docs/Game/js/medaglie.js	20
4.16	docs/Game/js/phaser.js	20
4.17	docs/Game/js/phaser-arcade-physics.js	20
4.18	docs/Game/index.html	20
5	Test	21
5.1	Protocollo di test	21

	SAMT – Sezione Informatica	Pagina 3 di 32
	Esempio di documentazione	

5.2	Risultati test	28
5.3	Mancanze/limitazioni conosciute	28
6	Consuntivo	29
7	Conclusioni.....	30
7.1	Sviluppi futuri	30
7.2	Considerazioni personali.....	31
7.2.1	Michea	31
7.2.2	Nadia	31
7.2.3	Thomas	31
8	Sitografia	32
9	Allegati	32

1 Introduzione

1.1 Informazioni sul progetto

- Allievi coinvolti nel progetto: Michea Colautti, Nadia Fasani, Thomas Sartini.
- Classe: I3AA/BB/BC Scuola Arti e Mestieri Trevano.
- Docenti responsabili: Geo Petrini
- Data inizio: 27 gennaio 2022.
- Data di fine: 05 maggio 2022.
- Linguaggio: JavaScript

1.2 Abstract

We all know the famous Chrome Dino, that little black dinosaur that jumps over a lot of cactuses endlessly and tells us that we are not connected to the internet. We've all hated him and loved him at some point in our lives. With this project the Chrome Dino is taken to another level: starting from a previously created project, we have improved the user experience and added a new interesting game mechanic. With this project, which is even multiplayer, the players jump over cactuses, this time not by pushing a button, but by their own real movements. Using the phone's sensors the player's dinosaur will jump over obstacles. New features have also been added, such as the possibility to customize the dinosaur and earn rewards. But the project is also available to those who cannot, for one reason or another, jump. For this reason, the phone can also be used: by pressing a button they control their own dinosaur.

1.3 Scopo

Lo scopo del progetto è quello di creare una versione multiplayer del famoso *Chrome Dino* dove diversi utenti si possono connettere ad una partita e possono giocare tutti insieme. Il numero di giocatori è quindi variabile, da un minimo di uno ad un massimo di dieci. Ci sarà dunque un utente "host" che si occupa di creare la partita e di mostrarla agli altri utenti, idealmente su uno schermo sufficientemente grande. Man mano che si aggiungono giocatori i loro dinosauri appariranno sullo schermo principale.

L'idea del progetto è il fatto che i giocatori fanno saltare il proprio dinosauro muovendosi essi stessi con un salto. Grazie ai sensori di movimento del telefono e alla struttura a sessioni del programma, i salti vengono trasmessi al server, che comunica alla pagina l'evento. Tuttavia, per quelle persone che non possiedono un telefono con i giusti sensori, oppure sono impossibilitate nel saltare, bisogna introdurre una modalità di gioco basilare, che permette di far saltare il dinosauro cliccando su un tasto.

Gli utenti devono poter inoltre creare un account tramite il quale potranno personalizzare l'aspetto del proprio dino e, una volta giocata una o più partite, guadagnarsi delle medaglie che poi saranno visibili in una pagina dedicata. Per coloro che non vogliono effettuare il login, saranno creati degli utenti ospiti che verranno eliminati quando l'utente esce dal sito.

Ci deve essere infine la possibilità di vedere la partita come spettatori, senza interagire con il gioco vero e proprio.

2 Analisi

2.1 Analisi del dominio

Non dovremo sviluppare questo progetto da zero. Come base avremo infatti il *Chrome Dino* realizzato da Manuel Grosso (vedi sitografia), nel corso del primo semestre dell'anno scolastico 2021/2022. Questo progetto e quello precedente, hanno in comune l'aspetto multiplayer, anche se per la versione sviluppata da Manuel, il numero di giocatori era impostato a 4. Noi dovremmo rendere questo aspetto dinamico, in modo da permettere maggiore flessibilità. Inoltre, per questo progetto la meccanica di salto cambia notevolmente: infatti, se nella precedente versione del gioco il dinosauro saltava in seguito al comando dato premendo su un tasto, ora i giocatori per far saltare il dinosauro dovranno anch'essi saltare veramente con il loro corpo. Attualmente non ci sono progetti simili al nostro. Esso non risolve un problema vero e proprio, ma non per questo è poco rilevante; troviamo infatti molto interessante questa meccanica di gioco, in quanto rappresenta un'evoluzione di un gioco di per sé semplice. Il progetto si rivolge a tutti coloro che hanno voglia di provare qualcosa di nuovo e -avendo introdotto una meccanica di salto alternativa- non esclude coloro che non possono fisicamente saltare o hanno un telefono privo di sensori adeguati.

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	I giocatori devono poter creare una partita con URL o sessione
Priorità	1
Versione	1.0
Note	

ID: REQ-02	
Nome	È possibile registrarsi
Priorità	1
Versione	1.0
Note	

ID: REQ-03	
Nome	È possibile eseguire il login
Priorità	1
Versione	1.0
Note	

ID: REQ-04	
Nome	I giocatori devono potersi unire alla partita
Priorità	1
Versione	1.0
Note	

ID: REQ-05	
Nome	Funzioni aggiuntive di login
Priorità	2
Versione	1.0
Note	
Sotto Requisiti	
001	Personalizzazione del dinosauro salvata
002	Punteggio salvato nel profilo
003	Bacheca per visualizzare le medaglie

ID: REQ-06	
Nome	La GUI deve essere responsive
Priorità	2
Versione	1.0
Note	

ID: REQ-07	
Nome	Possibilità di giocare come ospite
Priorità	2
Versione	1.0
Note	

ID: REQ-08	
Nome	Il dinosauro deve essere personalizzabile prima della partita
Priorità	3
Versione	1.0
Note	

ID: REQ-09	
Nome	Il dinosauro deve saltare sfruttando i sensori del telefono
Priorità	2
Versione	1.0
Note	
Sotto Requisiti	

001	Alternativa di gioco in caso di handicap o assenza di sensori.
------------	--

ID: REQ-10	
Nome	Il numero di giocatori deve essere dinamico: da uno a dieci.
Priorità	1
Versione	1.0
Note	

ID: REQ-11	
Nome	Il nome dei giocatori deve apparire a schermo
Priorità	3
Versione	1.0
Note	

ID: REQ-12	
Nome	Alla fine del gioco deve essere visualizzata una classifica
Priorità	2
Versione	1.0
Note	
Sotto Requisiti	
001	Punteggi in ordine decrementale
002	Visualizzare la/le medaglie direttamente nella classifica

ID: REQ-13	
Nome	Al vincitore viene assegnata una medaglia
Priorità	3
Versione	1.0
Note	
Sotto Requisiti	
001	Medaglia generata con un algoritmo per rendere il più univoche possibili

ID: REQ-14	
Nome	Possibilità di vedere la partita da remoto
Priorità	3
Versione	1.0

Note	
------	--

2.3 Use case

Ecco lo *use case* da noi definito:

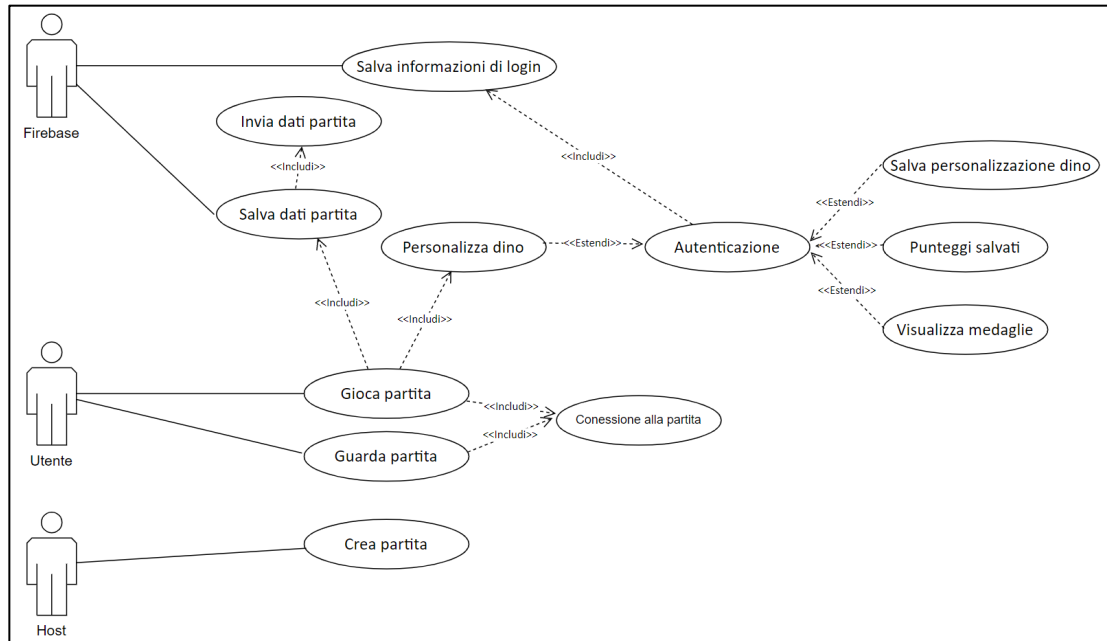


Figura 1 Use case

2.4 Pianificazione

Per quanto riguarda la pianificazione alleghiamo il diagramma di Gantt iniziale. Per lo sviluppo del progetto abbiamo deciso di utilizzare un approccio *Waterfall*.

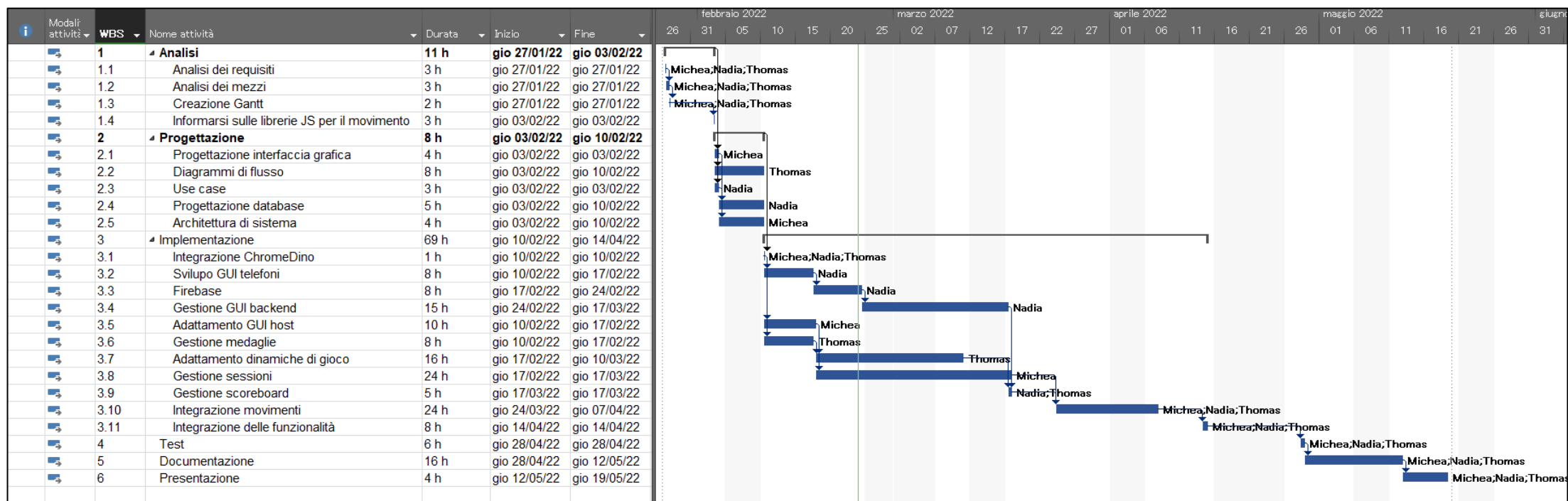


Figura 2 Gantt preventivo

2.5 Analisi dei mezzi

Software

- Firebase 8.2.1
- MockFlow 1.4.7
- Draw.io
- Visual Studio Code 1.65.2
- GIMP 2.10.24
- GitHub
- GitHub Desktop 2.9.12

Hardware

- Laptop personali
- PC scolastici

Il progetto è scritto in JavaScript, sarà quindi eseguibile da tutti i sistemi operativi. Per quanto riguarda la possibilità di saltare tramite i sensori di movimento, occorre specificare che sui terminali che eseguono IOS sarà possibile usufruire di questa funzione ma, invece di un salto vero e proprio, occorrerà far compiere al telefono un movimento dal basso in alto tramite movimento del polso. Ciò a causa delle diverse impostazioni dei sensori implementate da Apple.

3 Progettazione

Essendo consapevoli che la progettazione è una fase importante di ogni progetto, abbiamo voluto dedicare il tempo necessario ad essa, definendo tutti gli aspetti ai quali abbiamo pensato.

3.1 Design dell'architettura del sistema

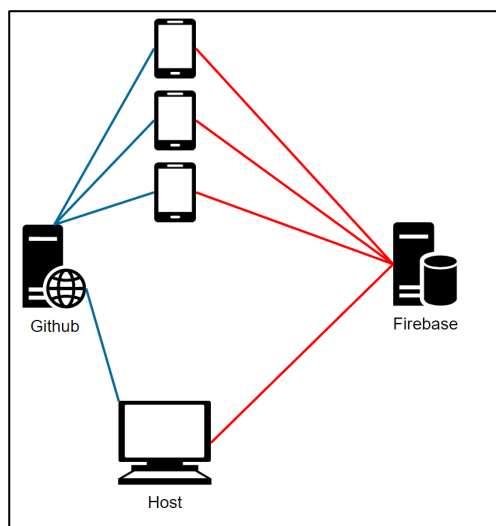


Figura 3 Architettura di sistema

Per l'architettura di sistema abbiamo voluto adottare una struttura abbastanza semplice ma funzionale. Il primo passo della comunicazione, in ordine cronologico, è l'host che crea una partita e la mostra agli utenti. Tutta la parte di gestione del server e delle connessioni, è gestita grazie ai server di GitHub. Infatti questa piattaforma offre un servizio chiamato "GitHub Pages" che svolge la funzione di Web server, togliendo così l'incombenza all'utente.

Contemporaneamente alla creazione della partita, l'host comunica al server Firebase le istruzioni necessarie per il buon funzionamento della stessa. Una volta che la partita è stata creata, gli utenti si collegano alla pagina, collegandosi quindi ai server GitHub, ma instaurano anche una comunicazione con il server Firebase.

3.2 Design dei dati e database

3.3 Design delle interfacce

Per la progettazione delle interfacce abbiamo deciso di dividere i *mockups* in 2 famiglie: quelle pensate per il telefono e quelle per il computer. Per le interfacce che dovevano essere visualizzate da entrambi i terminali abbiamo realizzato entrambe le versioni.

3.3.1 Design interfacce telefono:

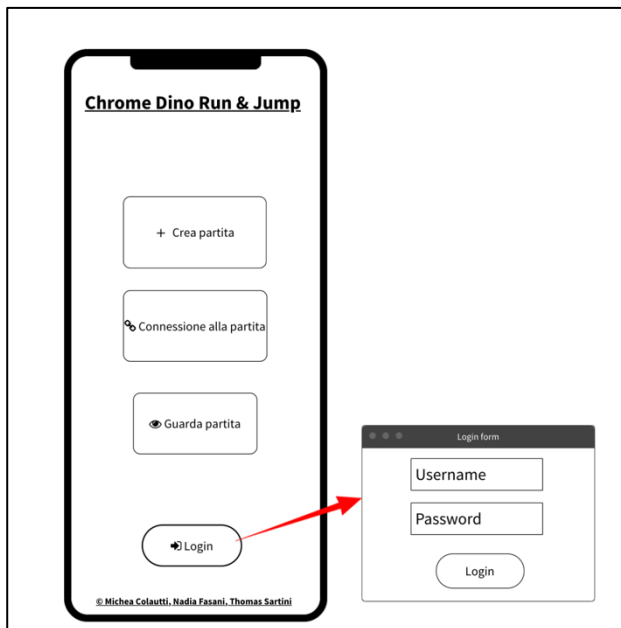


Figura 4 Home page con pop-up di login

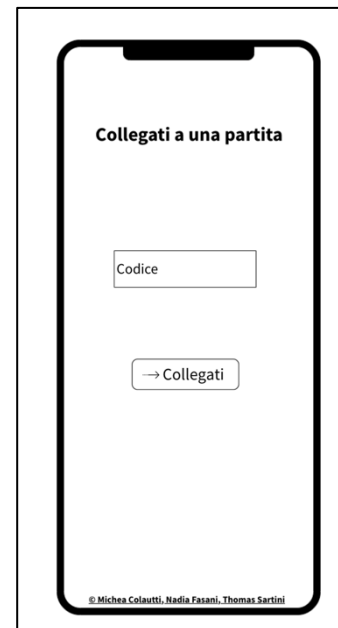


Figura 6 Pagina per collegamento ad una partita



Figura 5 Home page con utente loggato

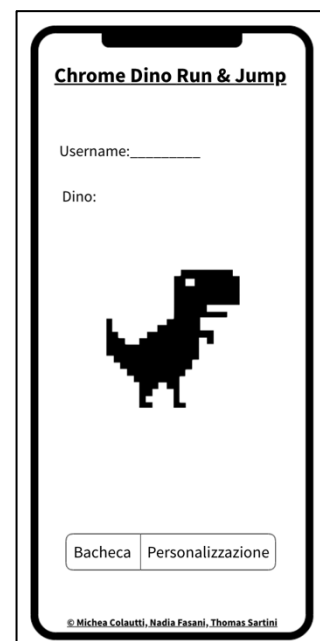


Figura 7 Pagina utente

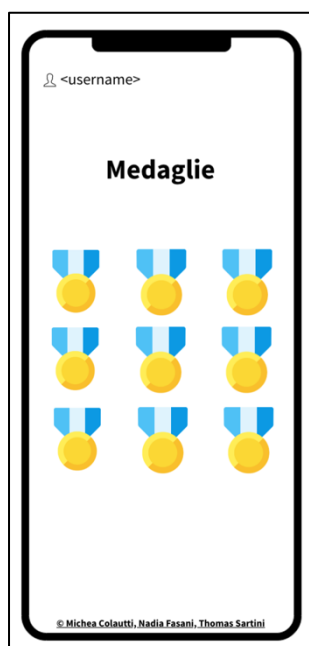


Figura 8 Bacheca medaglie

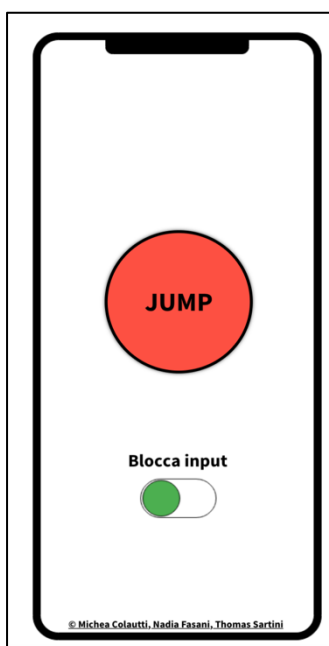


Figura 9 Pagina personalizzazione utente

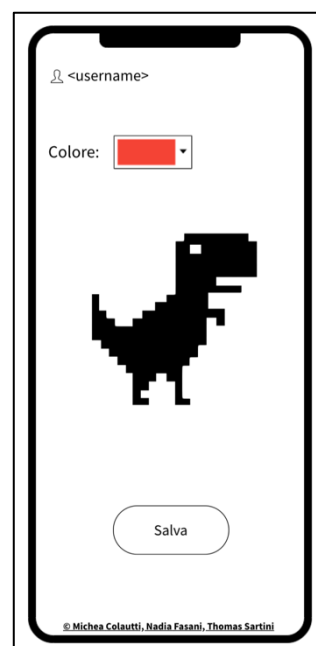


Figura 10 Pagina di gioco

3.3.2 Design interfacce computer

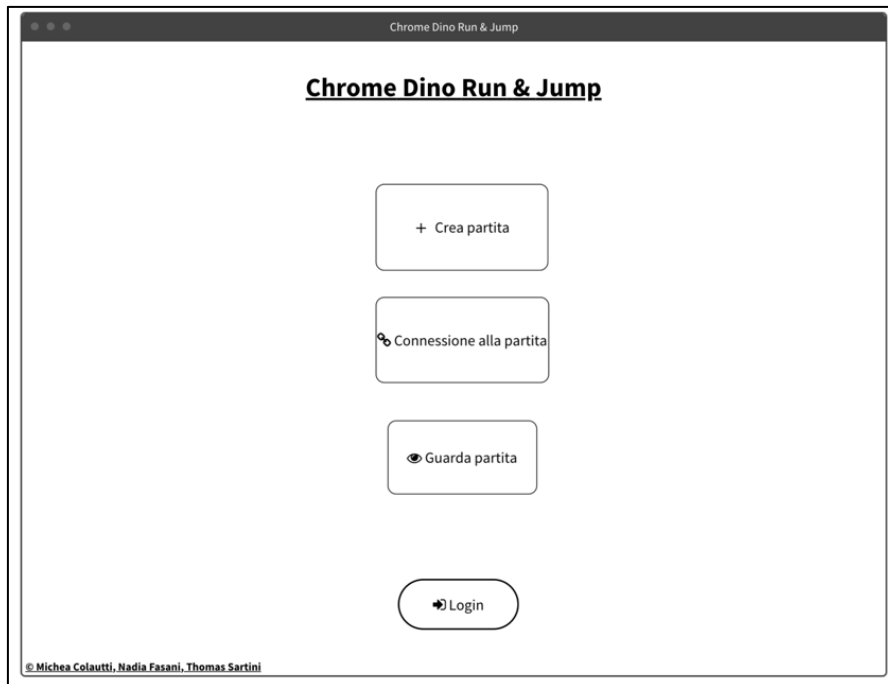


Figura 11 Home page

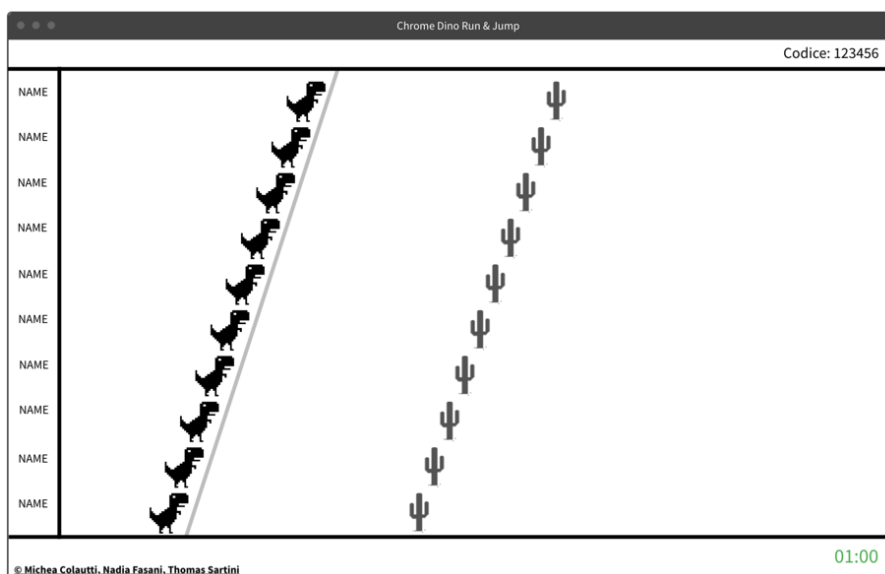


Figura 12 Creazione di una partita

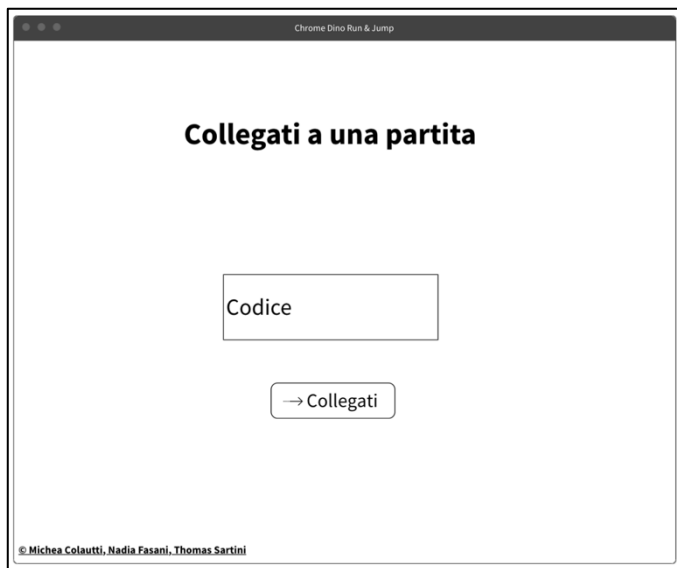


Figura 13 Collegamento ad una partita

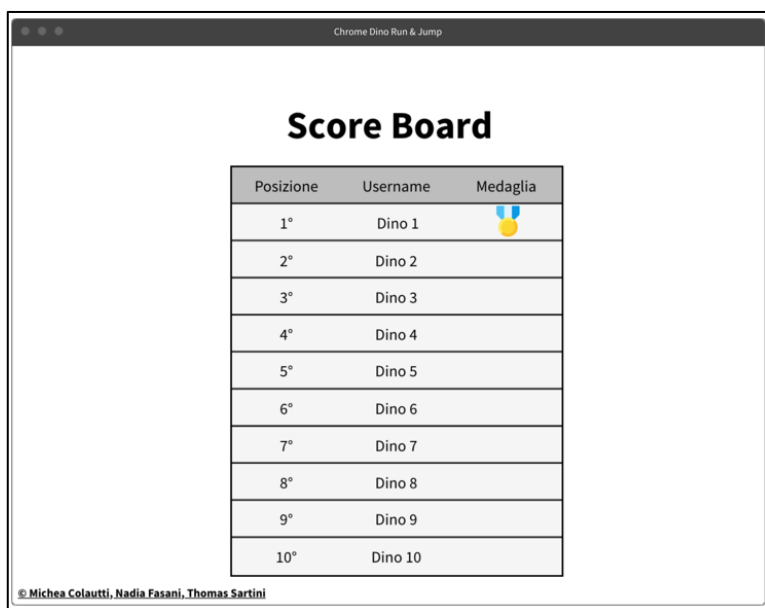


Figura 14 Classifica

3.4 Design procedurale

Per quanto riguarda il design procedurale, alleghiamo il diagramma di flusso da noi pensato.

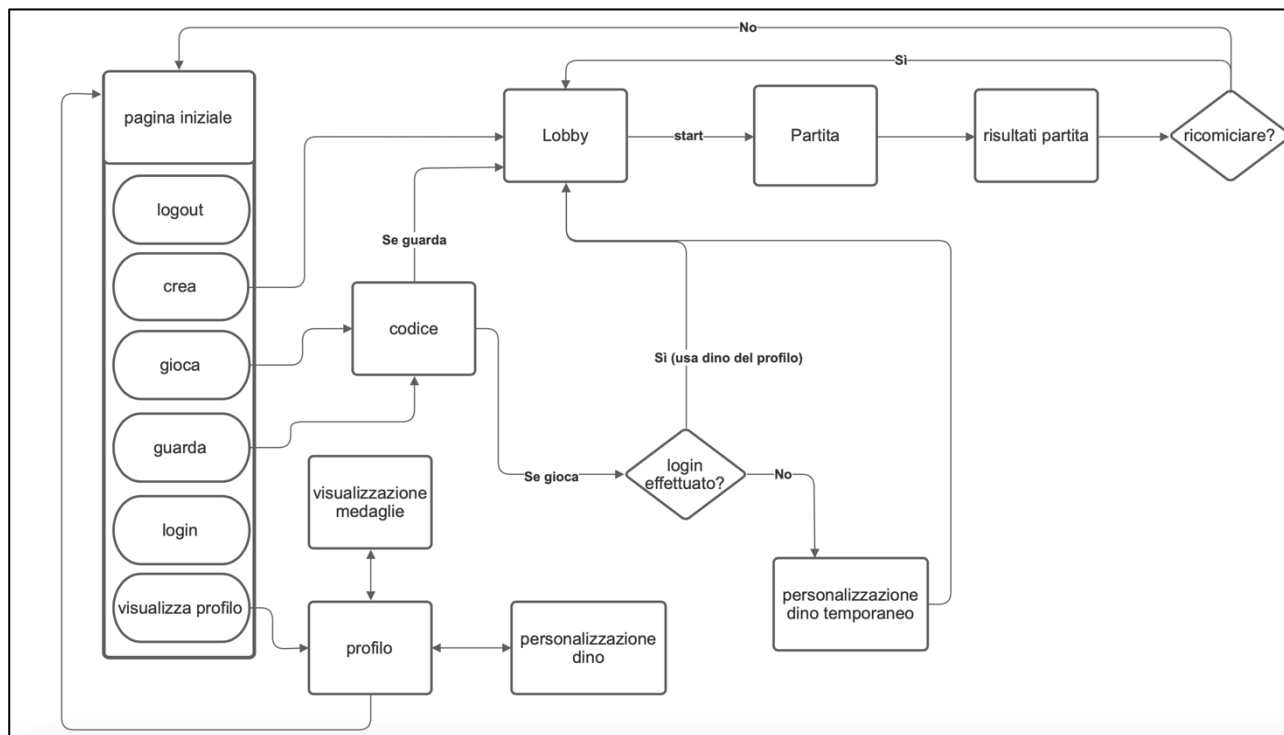


Figura 15 Diagramma di flusso

Lo schema del design è di per sè abbastanza chiaro. Tuttavia sarà maggiormente comprensibile con le seguenti precisazioni. Questo design ci è stato molto utile in fase di implementazione, poiché ci ha permesso di strutturare meglio il codice. Il punto di partenza è la *pagina iniziale*: da qui si può arrivare alla pagina di login oppure fare il logout. Nella *pagina iniziale* sono inoltre presenti altre funzionalità, che sono descritte di seguito:

- Creare una partita
 - Questo collegamento apre la partita nella lobby dove gli utenti si possono connettere.
 - In seguito, quando viene fatta partire la partita, il gioco comincia.
 - Una volta finito il gioco viene mostrata la classifica, e si può ricominciare a giocare o terminare.
- Giocare una partita
 - Questa parte del gioco prevede come prima cosa l'inserimento del codice della partita.
 - Se a giocare è un *guest* si viene indirizzati su una pagina di personalizzazione del dinosauro, altrimenti appare la lobby in attesa di iniziare la partita.
- Visualizzare il proprio profilo.
 - Questa funzione invece permette all'utente di vedere il profilo, e contiene due sotto-funzioni:
 1. Visualizzare la bacheca con le medaglie
 2. Personalizzare il proprio dinosauro.

4 Implementazione

4.1 Refactoring

4.2 Struttura cartelle del progetto

```
docs/
├── css/
│   ├── bootstrap.css
│   └── game.css
├── js/
│   ├── bootstrap.bundle.min.js
│   ├── game.js
│   └── index.js
├── GUI/
│   ├── bacheca.html
│   ├── collegamentoPartita.html
│   ├── dino.png
│   ├── game.html
│   ├── login.html
│   ├── paginaUtente.html
│   └── personalizzaDino.html
├── Game/
│   ├── img
│   ├── js/
│   │   ├── game.js
│   │   ├── medaglie.js
│   │   ├── phaser.js
│   │   └── phaser-arcade-physics.js
│   └── index.html
```

4.3 docs/css/bootstrap.css

È la libreria di bootstrap per la gestione del CSS. Questa libreria ci è servita nelle pagine html per la gestione della grafica e per realizzare un sito responsive.

4.4 docs/css/game.css

Questo file contiene il codice per la grafica dello switch per l'interfaccia di gioco dal telefono.

4.5 docs/js/Bootstrap.bundle.min.js

È la libreria di bootstrap per la gestione degli elementi tramite JavaScript, serve per esempio per l'apertura dinamica dei modal di bootstrap.

4.6 docs/js/game.js

4.6.1 blockInput

Metodo per disabilitare il bottone “jump” per permettere all'utente di saltare con il telefono in tasca senza che venga cliccato per sbaglio.

4.6.2 handleMotion

Il metodo viene richiamato da un listener al movimento del dispositivo. Controlla se il dino dell'utente sta toccando terra e nel caso in cui l'accelerazione rilevata dal sensore fosse maggiore di 10 richiama il metodo jump che si trova nel file **docs/js/index.js**

4.6.3 requestPermission

Mick??

4.7 docs/js/index.js

4.7.1 writeMedals

Metodo per mostrare da firebase tutte le medaglie ottenute dall'utente con un account. Fa riferimento al percorso **user/<user UID>/medals** e per ogni medaglia presente mostra gli svg aggiungendoli ad una tabella.

4.7.2 connectToGame

Questa funzione legge il valore della sessione inserita dell'input dall'utente nel file **docs/GUI/collegamentoPartita.html**. Poi viene creata una nuova variabile nel local storage per salvare il numero della sessione a cui l'utente vuole accedere. **Riga 49??**

Se l'utente non ha eseguito il login viene eseguito il metodo generateGuestId per poter avere un identificativo univoco per tutti gli utenti. In seguito l'utente viene aggiunto su Firebase all'interno della sessione chiesta dall'utente se esiste e viene aperta la pagina **docs/GUI/game.html**. Se invece l'utente ha eseguito il login, viene automaticamente controllato se la sessione esiste e in seguito l'utente viene aggiunto alla sessione tramite l'uid e viene aperta la pagina **docs/GUI/game.html**.

4.7.3 generateGuestId

Il metodo genera un id randomico per gli utenti guest nel seguente formato: "guest_XXXXXX" e in seguito apre la pagina docs/GUI/personalizzaDino.html per permettere anche ai guest la possibilità di scegliere il colore del dino. Viene anche creata una nuova variabile local storage per salvare localmente l'id appena creato.

4.7.4 jump


La funzione accede all'id della sessione presente nel local storage e percorre tutti i nodi presenti e trova il nodo che corrisponde al guest o all'utente che ha eseguito l'accesso. Poi esegue un update sull'attributo is_jumping che imposta a true.

4.7.5 registerNewUser

Il metodo legge le informazioni degli input inseriti dagli utenti e crea un nuovo account con email e password. Viene chiesto un nickname all'utente; in seguito viene aggiunto all'input dell'utente un dominio per far accettare a Firebase il nuovo account. Il formato della stringa che viene inviata a Firebase è "<nickname>@dino.ch". In caso di errori, c'è un elemento html che mostra il messaggio d'errore restituito da Firebase.

4.7.6 loginUser

Il metodo legge le informazioni degli input inseriti dagli utenti e autentica l'utente con email e password come per il metodo precedente. Poi mostra alcuni elementi html non visibili per gli utenti guest. In caso di errori nel login c'è un elemento html che mostra il messaggio d'errore ritornato da Firebase.

	SAMT – Sezione Informatica	Pagina 19 di 32
	Esempio di documentazione	

4.7.7 logoutUser

La funzione disconnette l'utente corrente e ricarica la pagina.

4.7.8 openUserInformation

Il metodo apre la pagina **docs/GUI/paginaUtente.html**.

4.7.9 generateSession

La funzione crea un numero randomico di 6 cifre e crea su Firebase un nuovo child sotto il ramo session. Poi apre la pagina **docs/Game/index.html**.

4.7.10 changeDinoColor

4.7.11 saveDinoColor

4.7.12 showUserInformation

4.7.13 watchGame

4.7.14 checkLoggedUser

4.7.15 getIsTouchingDown

4.7.16 firebase.auth().onAuthStateChanged((user) => {

4.8 docs/GUI/bacheca.html

La pagina mostra agli utenti con un account le medaglie ottenute.

4.9 docs/GUI/collegamentoPartita.html

La pagina contiene un form per inserire il codice di una partita per poi potersi collegare.

4.10 docs/GUI/game.html

È la pagina principale di gioco. Contiene un bottone da utilizzare in caso di problemi di mobilità o quando il sito non ha accesso alle informazioni sui movimenti del dispositivo.

4.11 docs/GUI/login.html

È la pagina principale che permette di creare un nuovo account o accedere a uno già esistente. Permette inoltre di creare o connettersi ad una partita.

4.12 docs/GUI/paginaUtente.html

La pagina mostra le varie informazioni dell'utente se si è precedentemente autenticato.

4.13 docs/GUI/personalizzaDino.html

4.14 docs/Game/js/game.js

4.15 docs/Game/js/medaglie.js

4.16 docs/Game/js/phaser.js

4.17 docs/Game/js/phaser-arcade-physics.js

4.18 docs/Game/index.html

5 Test

5.1 Protocollo di test

Test Case:	TC-01	Nome:	Creazione di una sessione
Riferimento:	REQ-01		
Descrizione:	L'utente raggiunge il sito e crea una partita		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere il pulsante "Crea una partita" 		
Risultati attesi:	<ol style="list-style-type: none"> 1. La pagina deve mostrare la schermata di gioco 2. La console di Firebase deve mostrare la sessione creata 		

Test Case:	TC-02	Nome:	Registrazione utente
Riferimento:	REQ-02		
Descrizione:	L'utente deve potersi registrare nel sito		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Sign Up" 3. Immettere username e password 4. Confermare 		
Risultati attesi:	<ol style="list-style-type: none"> 1. L'utente deve essere loggato 2. Nella console di Firebase deve essere presente un nuovo utente con l'username corretto (la password è cifrata) 		

Test Case:	TC-03	Nome:	Login
Riferimento:	REQ-03		
Descrizione:	L'utente deve potersi autenticare		
Prerequisiti:	Un utente creato		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Sign In" 3. Immettere username e password 4. Confermare 		
Risultati attesi:	L'utente deve essere loggato		

Test Case:	TC-04	Nome:	Unione ad una partita
Riferimento:	REQ-04		
Descrizione:	Un utente deve poter giocare una partita		
Prerequisiti:	<ol style="list-style-type: none"> 1. Un utente creato 2. Una partita creata ed aperta su uno schermo 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Effettuare il login 3. Premere sul pulsante "Connessione alla partita" 4. Immettere il codice della partita 5. Confermare 		
Risultati attesi:	L'utente deve apparire nella partita nella pagina dell'host		

Test Case:	TC-05	Nome:	Funzione aggiuntive login 1
Riferimento:	REQ-05		
Descrizione:	L'utente deve poter personalizzare il suo dinosauro		
Prerequisiti:	Un utente creato		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Effettuare il login 3. Recarsi nella pagina utente premendo sull'icona dello stesso 4. Premere su "Personalizza" 5. Scegliere un colore per il dinosauro 6. Premere "Salva" 		
Risultati attesi:	Il colore del dinosauro deve essere cambiato nella pagina utente		

Test Case:	TC-06	Nome:	Funzione aggiuntive login 2
Riferimento:	REQ-05		
Descrizione:	Il punteggio dell'utente deve essere salvato		
Prerequisiti:	<ol style="list-style-type: none"> 1. Un utente creato che non ha mai giocato una partita 2. Una partita creata ed aperta su uno schermo 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Effettuare il login 3. Premere sul pulsante "Connessione alla partita" 4. Immettere il codice della partita 5. Confermare 6. Avviare la partita dall'host 7. Giocare 		

	8. Quando la partita finisce recarsi nella pagina dell'utente
Risultati attesi:	Il punteggio fatto deve essere visibile

Test Case:	TC-07	Nome:	Funzione aggiuntive login 3
Riferimento:	REQ-05		
Descrizione:	Le medaglie devono essere presenti nella bacheca		
Prerequisiti:	<ol style="list-style-type: none"> 1. Un utente creato con almeno una medaglia 2. Una partita creata ed aperta su uno schermo 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Aprire la pagina utente 3. Premere su "Bacheca" 		
Risultati attesi:	Le medaglie dell'utente devono visibili		

Test Case:	TC-08	Nome:	GUI responsive
Riferimento:	REQ-06		
Descrizione:	La GUI del sito deve essere responsive		
Prerequisiti:	Un utente loggato		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Effettuare il login 3. Provare a ridimensionare la pagina e verificare che nessun elemento venga nascosto 4. Ripetere il test per tutte le pagine eccetto quella di gioco 		
Risultati attesi:	Tutti gli elementi nella pagina si spostano/ridimensionano correttamente		

Test Case:	TC-09	Nome:	Pagina di gioco delle dimensioni massime
Riferimento:	REQ-06		
Descrizione:	La pagina di gioco deve essere sempre delle dimensioni massime della pagina		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Creare una nuova partita 3. Verificare che la pagina occupi tutto lo spazio disponibile 4. Provare a ridimensionare la pagina e poi refreshare 		
Risultati attesi:	La pagina assume sempre le dimensioni massime consentite		

Test Case:	TC-10	Nome:	Gioco come guest
Riferimento:	REQ-07		
Descrizione:	Si deve poter giocare anche senza aver effettuato il login		
Prerequisiti:	Una partita creata ed aperta su uno schermo		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Confermare 5. Avviare la partita dall'host 6. Giocare 		
Risultati attesi:	L'utente deve poter giocare normalmente		

Test Case:	TC-11	Nome:	Personalizzazione dinosauro guest
Riferimento:	REQ-08		
Descrizione:	Si deve poter personalizzare il proprio dinosauro anche senza aver effettuato il login, appena prima di unirsi alla partita		
Prerequisiti:	Una partita creata ed aperta su uno schermo		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Selezionare il colore del dinosauro 5. Confermare 		
Risultati attesi:	Il dinosauro deve apparire nella pagina dell'host con il colore prescelto.		

Test Case:	TC-12	Nome:	Giocare con i sensori del telefono
Riferimento:	REQ-09		
Descrizione:	Si deve poter giocare sfruttando i sensori del telefono		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Un telefono con i sensori di movimento appropriati (NO IOS) 3. Un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Confermare 5. Provare a saltare con il telefono in mano o in tasca 6. Ripetere il test con un utente loggato 		

Risultati attesi:	Il dinosauro deve saltare nella pagina dell'host.
-------------------	---

Test Case:	TC-13	Nome:	Giocare con il pulsante
Riferimento:	REQ-09		
Descrizione:	Si deve poter giocare sfruttando i sensori del telefono.		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Unisciti ad una partita" 3. Immettere il codice della partita 4. Confermare 5. Provare a premere il pulsante "Jump" 6. Ripetere il test con un utente loggato 		
Risultati attesi:	Il dinosauro deve saltare nella pagina dell'host.		

Test Case:	TC-14	Nome:	Numero di giocatori dinamico
Riferimento:	REQ-10		
Descrizione:	Si deve poter giocare in più persone.		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Almeno un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Confermare 5. Ripetere più volte, fino ad un massimo di 11 6. Utilizzare anche almeno un utente loggato 7. Provare a saltare/premere jump su i vari dispositivi 8. Avviare la partita 		
Risultati attesi:	<ol style="list-style-type: none"> 1. Ad ogni connessione deve apparire un nuovo dinosauro 2. Ogni dinosauro deve saltare quando il dispositivo che lo controlla ordina un salto 3. Ci possono essere un massimo di 10 giocatori 4. Quando la partita viene avviata tutti i dinosauri devono ancora poter saltare 		

Test Case:	TC-15	Nome:	Nome dei giocatori a schermo
Riferimento:	REQ-11		
Descrizione:	Quando ci si unisce ad una partita, oltre al dinosauro, appare anche il nickname		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Almeno un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo con 2 dispositivi 2. Effettuare il login 3. Premere sul pulsante "Connessione alla partita" 4. Immettere il codice della partita 5. Confermare 6. Ripetere il test con un utente loggato e con più utenti contemporaneamente 		
Risultati attesi:	I nickname degli utenti devono apparire correttamente		

Test Case:	TC-16	Nome:	Classifica finale
Riferimento:	REQ-12		
Descrizione:	Quando una partita finisce deve apparire una classifica		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Almeno un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo con 2 dispositivi 2. Effettuare il login 3. Premere sul pulsante "Connessione alla partita" 4. Immettere il codice della partita 5. Confermare 6. Giocare 		
Risultati attesi:	La classifica appare alla fine della partita		

Test Case:	TC-17	Nome:	Classifica finale ordinata
Riferimento:	REQ-12		
Descrizione:	Quando una partita finisce deve apparire una classifica ordinata per punteggio		
Prerequisiti:	<ol style="list-style-type: none"> 1. Una partita creata ed aperta su uno schermo 2. Almeno un utente creato 		
Procedura:	<ol style="list-style-type: none"> 1. Raggiungere il sito dell'applicativo con 2 dispositivi 2. Effettuare il login 3. Premere sul pulsante "Connessione alla partita" 4. Immettere il codice della partita 5. Confermare 		

	6. Unirsi con un secondo utente (loggato o normale) 7. Giocare
Risultati attesi:	La classifica appare alla fine della partita ed è ordinata

Test Case:	TC-18	Nome:	Medaglie nella classifica
Riferimento:	REQ-12		
Descrizione:	Quando una partita finisce deve apparire una classifica con una medaglia per il primo classificato		
Prerequisiti:	1. Una partita creata ed aperta su uno schermo		
Procedura:	1. Raggiungere il sito dell'applicativo con 2 dispositivi 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Confermare 5. Unirsi con un secondo utente (loggato o normale) 6. Giocare		
Risultati attesi:	La classifica appare alla fine e solo il primo classificato ha una medaglia		

Test Case:	TC-19	Nome:	Medaglie assegnate al vincitore
Riferimento:	REQ-12		
Descrizione:	Quando una partita finisce la medaglia viene assegnata al vincitore		
Prerequisiti:	1. Un utente creato con almeno una medaglia 2. Una partita creata ed aperta su uno schermo		
Procedura:	1. Raggiungere il sito dell'applicativo 2. Premere sul pulsante "Connessione alla partita" 3. Immettere il codice della partita 4. Confermare 5. Avviare la partita dall'host 6. Giocare 7. Quando la partita finisce verificare la presenza della medaglia nella classifica e recarsi nella pagina dell'utente 8. Premere su "Bacheca"		
Risultati attesi:	La medaglia appena vinta deve essere nella bacheca		

5.2 Risultati test

Test Case	Risultato	Commenti
TC-01	Passato	
TC-02	Passato	
TC-03	Passato	
TC-04	Passato	
TC-05	Passato	
TC-06	FALLITO	Punteggi assenti
TC-07	Passato	
TC-08	Passato	
TC-09	Passato	
TC-10	Passato	
TC-11	Passato	
TC-12	FALLITO	
TC-13	Passato	Perfezionare utente loggato
TC-14	FORSE	Max 10?
TC-15	Passato	
TC-16	Passato	
TC-17	Passato	
TC-18	Passato	
TC-19	Passato	

5.3 Mancanze/limitazioni conosciute

Pensiamo che la mancanza principale del nostro progetto sia la struttura del codice. Pur essendo stato compiuto un buon lavoro di refactoring, che ci ha permesso di portare a termine questo progetto con successo, abbiamo purtroppo individuato, cammin facendo, una mancanza riguardante non tanto il refactor in sé, quanto nella modalità dello stesso.

Un approccio a classi, infatti, dove ogni parte del gioco sarebbe stata rappresentata da un oggetto e non da un array, avrebbe semplificato e abbellito ulteriormente il codice.


Inoltre, un simile approccio avrebbe forse risolto anche il problema del ghost. Problema che abbiamo peraltro già esplicitato in maniera esauriente nel capitolo 4. Ciò, pur non essendo a conti fatti, una vera e propria limitazione, è sicuramente un punto a sfavore di questo progetto.

A livello di organizzazione procedurale, con il senno di poi, ci siamo resi conto che la stima del tempo occorrente per implementare alcune parti era sbagliata, per eccesso o per difetto. Quando è capitato, il buon clima collaborativo che si è creato nel gruppo, ci ha permesso di aiutarci a vicenda.



6 Consuntivo

Per questo capitolo alleghiamo il nostro Gantt consuntivo.

	SAMT – Sezione Informatica	Pagina 30 di 32
	Esempio di documentazione	

7 Conclusioni

A bocce ferme, possiamo dirci moderatamente soddisfatti del nostro progetto. Non solo perché ci siamo impegnati a rispettare il più fedelmente possibile le specifiche, ma anche perché pensiamo di aver realizzato, nel complesso, un buon gioco. Per due di noi era la prima esperienza di lavoro in gruppo su un progetto semestrale: si trattava di evolvere da piccoli progetti svolti a coppie o in gruppo in passato, ad un progetto maggiormente complesso ed articolato, che avesse uno sviluppo puntuale sotto più aspetti, come progettazione, codice, documentazione, ecc.

Per quanto riguarda la modalità e il clima di lavoro nel gruppo, riteniamo che sia stato molto positivo. Non abbiamo avuto momenti di tensione e gli scambi di idee sono stati costruttivi e orientati al raggiungimento dell'obiettivo comune. Cosa non scontata perché non avevamo mai lavorato insieme. Il progetto ci ha dunque permesso di affinare le nostre abilità di collaborazione in modo piuttosto significativo.

Il progetto ci ha sicuramente permesso di apprendere e mettere in pratica nuove importanti nozioni. Questo non solo per quanto riguarda JavaScript. Abbiamo imparato anche ad usare il framework Phaser e due di noi hanno avuto anche la possibilità di conoscere Firebase. L'utilizzo di strumenti come il Gantt e GitHub ci ha aiutato a dividerci i compiti e a condividere il nostro lavoro cammin facendo.

Un altro aspetto che riteniamo sia stato importante per la buona riuscita del progetto è l'aver messo in comune le nostre conoscenze, in modo da aiutarci l'un l'altro e sopperire così a eventuali lacune che potevano sorgere. Per esempio, Nadia nel progetto ha aiutato per la parte di comunicazione con il database Firebase: avendo lei già sviluppato un progetto con questo sistema è stato più semplice implementare il codice e apprendere nuove nozioni. Thomas ha portato un'ottima conoscenza di JavaScript che, unita a molta pazienza, ci ha permesso di migliorare e convertire il codice prodotto in precedenza da Manuel Grosso. Michea, per contro, ha sicuramente aiutato a rendere tutta la parte di documentazione più chiara e efficace, grazie alla maggiore esperienza accumulata lo scorso anno.

Un aspetto che ha molto interessato tutti è il fatto di non essersi dovuti confrontare con uno sviluppo di progetto dall'inizio, bensì di aver potuto usare come base il progetto di Manuel, anche se prima di svilupparlo si è comprensibilmente presentata una fase di comprensione e di ristrutturazione del codice. Questa fase è stata una delle più complesse, in quanto comprendere del codice scritto da un'altra persona, con una documentazione poco consistente, è stato molto difficile. In particolare, perché l'interesse del codice era stata sviluppata con un approccio *Hard coded*. Questo ha aumentato la difficoltà.

Come già accennato, sarebbe stato bello e utile essere più performanti nella fase di progettazione, in modo da non incappare nel problema del ghost. Ma lo riteniamo comunque un errore "utile", nel senso che eviteremo di compierlo ancora in futuro.

Infine, esprimiamo il nostro sincero ringraziamento al prof. Petrini, che lungo tutto l'arco del progetto ha vegliato su di noi, fornendoci utili spunti di riflessione quando incontravamo dei problemi e spingendoci a cercare delle soluzioni efficaci. Senza la sua paziente supervisione il progetto non avrebbe avuto uno sbocco così positivo. Grazie di cuore.

7.1 Sviluppi futuri

Per quanto riguarda gli sviluppi futuri di questo progetto, abbiamo identificato alcune opzioni:

La prima non riguarda una funzione o aggiunta al progetto in sé, bensì il codice di base. Come detto, nel refactor abbiamo deciso di utilizzare degli array per rendere l'applicazione multiplayer, ma un migliore approccio sarebbe stato quello di utilizzare le classi. È quindi questa la prima miglioria che può essere fatta al progetto; ciò permetterebbe non solo di avere un codice molto più comprensibile, ma semplificherebbe anche future modifiche.

Un altro sviluppo è da ricercare nell'implementazione della funzione *ghost*. Ci è dispiaciuto molto infatti non poterla realizzare, in quanto ritenevamo molto interessante la possibilità di assistere da remoto ad una partita come spettatore. Tuttavia, se ciò non risultasse comunque possibile tramite un'implementazione ad oggetti o tramite migliorie al codice, sarebbe necessario cambiare la piattaforma di base del database, ovvero Firebase. Nel caso si prendesse questa strada bisognerebbe ripensare l'applicazione quasi da zero, modificando tutte le logiche per il passaggio e l'ottenimento dei dati.

Ulteriori sviluppi futuri ipotizzabili, e che riguardano maggiormente il progetto nel suo aspetto principale, sono la possibilità di personalizzare il proprio dinosauro, implementando l'inserimento di *skin* o la possibilità di utilizzare più colori per il proprio personaggio. Inoltre si potrebbe pensare alla creazione di oggetti esterni da aggiungere, come cappelli o occhiali, fruibili solo da utenti registrati che hanno raggiunto un determinato punteggio.

Ispirandoci invece al vero Chrome Dino, sono molteplici le modifiche/aggiunte che si potrebbero applicare: ad esempio diverse tipologie di cactus, diversi in altezza e che compaiono a coppie. Si potrebbero inserire anche gli uccelli che sono presenti nel gioco originale, sviluppando la possibilità di potersi abbassare, in modo da variare i movimenti che l'utente deve fare. Infine, dal punto di vista temporale, si potrebbe implementare lo scorrimento del tempo, in modo che ogni tanto il gioco passi dal giorno alla notte.

7.2 Considerazioni personali

7.2.1 Michea


Personalmente la cosa che più ho apprezzato nel corso del progetto è stata la sfida, per me praticamente nuova, di creare un gioco. Pur avendo fatto ormai -nel bene e nel male- ormai 3 progetti, non mi era mai capitato di dover sviluppare un gioco. Sono sempre stato convinto che lo sviluppo dei giochi fosse una branca piuttosto complessa e a me non vicina, dato che non sono mai stato un videogiocatore, ma sviluppare il progetto mi ha molto appassionato. Sono abbastanza soddisfatto del risultato finale e anche del percorso fatto con i miei compagni. La buona armonia presente nel gruppo ci ha aiutato a pianificare le varie fasi senza troppe difficoltà, così come a dividerci il lavoro da fare in modo piuttosto efficace, rispettando al contempo le varie individualità e punti di forza all'interno del gruppo.

Spero che questo progetto, prima o poi, venga riproposto ad altri allievi, sia in ottica di un possibile miglioramento, sia come progetto da sviluppare da zero.

Un ulteriore motivo per cui ho apprezzato molto questo progetto, è stato il fatto che dopo un anno e mezzo sono tornato ad occuparmi di un progetto in JavaScript, linguaggio che mi è sempre piaciuto e che nell'ultimo periodo avevo purtroppo dovuto abbandonare. Spero quindi che questo possa essere, oltre che un arricchimento per le mie conoscenze, un punto di ripartenza per tornare a masticare JavaScript e dintorni.

7.2.2 Nadia

7.2.3 Thomas

	SAMT – Sezione Informatica	Pagina 32 di 32
	Esempio di documentazione	

8 Sitografia

- <http://standards.ieee.org/guides/style/section7.html>, *IEEE Standards Style Manual*, 07-06-2008.

9 Allegati

- Diari di lavoro
- QdC
- Prodotto