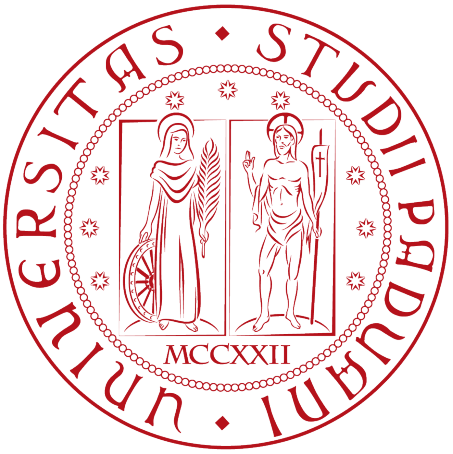
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Trasformare i device di firma in console portatili tramite l'ENGaming

Tesi di laurea

Relatore

Prof. Claudio Enrico Palazzi

Laureando

Nicola Baesso

Anno Accademico 2022-2023

Sommario

Il presente documento espone il lavoro svolto dal laureando Nicola Baesso, presso l'azienda ESignWorld S.R.L., durante il tirocinio della durata di circa trecento ore.

Il tirocinio prevedeva innanzitutto lo sviluppo di un applicativo desktop da utilizzare in un device di firma, unendo tecnologie frontend e backend. In particolare, era richiesto lo sviluppo dell'applicazione, utilizzando Electron e il framework Angular. Inoltre, volendo recuperare i dati dal device di firma, l'applicazione necessitava l'utilizzo di moduli compilati per NodeJS, e scritti in C++.

Lo scopo di tale progetto era dimostrare che il device utilizzato non si limitasse alla semplice firma, ma che potesse essere un ottimo strumento multimediale, in particolare un ottimo strumento per l'utilizzo di videogiochi.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Claudio Enrico Palazzi, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro. Inoltre, ringrazio profondamente il signor Matteo Gnoato, mio tutor aziendale, per avermi affiancato durante il lavoro. Desidero ringraziare con affetto i miei genitori e la mia ragazza, per il sostegno e per essermi stati vicini in ogni momento durante gli anni di studio. Desiderio ringraziare anche i miei amici, sia quelli conosciuti durante il mio percorso sia chi conoscevo già da prima, per tutti i momenti passati insieme. Infine, ringrazio anche chi, senza tanti complimenti, si è allontanato dalla mia vita durante questi anni. Essere qui senza di voi ha un sapore ancora più buono.

Sant'Angelo di Piove, Settembre 2023 Nicola Baesso

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questa sezione vengono descritte l'azienda ospitante, il device utilizzato durante il tirocinio e gli strumenti di sviluppo e comunicazione utilizzati.

1.1 Euronovate Group

Figura 1.1: Logo Euronovate Group

Euronovate Group è un gruppo multinazionale con oltre 150 dipendenti, leader nel mercato nell'implementazione e commercializzazione di soluzioni innovative per la trasformazione digitale, Certification Authority eIDAS compliant, e produttrice di oltre 50 prodotti proprietari HW e SW. Ha sede centrale a Mendrisio (Svizzera) e controllate con presenza diretta in quattro paesi:

- Italia (Padova, Reggio Emilia, Milano);
- Spagna (Barcellona, Madrid, Bilbao);
- Romania (Bucarest);
- Cina (Shanghai).

Euronovate Group inoltre si suddivide in:

- Euronovate SA;
- eSignWorld;
- Víntegris.

1.1.1 Euronovate SA

Fondata nel 2012 e con sede a Lugano (CH), è una società svizzera innovativa, leader in soluzioni di Digital Trasformation con approccio end-to-end, combinando soluzioni software, hardware e servizi di consulenza. L'obiettivo principale è aiutare ogni tipo di azienda a eliminare tutti i processi e i documenti cartacei passando completamente al digitale, garantendo la stessa validità legale.

1.1.2 eSignWorld

Società che opera nel settore della consulenza IT, processi e sistemi avanzati di firme elettroniche, fornitura, esercizio e manutenzione di sistemi informativi hardware e software, specializzata nel campo della produzione a filiera corta di tecnologia grafometrica. In particolare, eSignWorld fornisce soluzioni personalizzate e proprietarie nel campo della dematerializzazione documentale e dell'Information Communication Technology, garantendo la possibilità di visualizzare, elaborare e firmare elettronicamente qualsiasi tipo di documento. eSignworld offre soluzioni paperless con utilizzo di firma elettronica semplice e firma elettronica avanzata, un sistema di composizione e successiva conservazione di documenti in formato elettronico, nonché di firma dei documenti, attraverso un innovativo dispositivo di firma.

1.1.3 Víntegris

Víntegris progetta, implementa e gestisce infrastrutture di sicurezza per istituzioni finanziarie e grandi società. Fornisce solide soluzioni su misura per ogni esigenza di business, integrando tecnologie ad alte prestazioni e soddisfacendo le esigenze di ogni azienda. La società definisce e realizza progetti ad alto valore aggiunto per la protezione delle informazioni, la sicurezza web, il controllo e la gestione degli accessi. Vengono utilizzate tecnologie tra le più robuste sul mercato (Docker, Kubernetes, AWS, HSM), integrandole nell'ambiente aziendale di ciascun cliente. L'essere consulenti e integratori, pone Víntegris in una posizione privilegiata nello sviluppo di nuovi prodotti: la conoscenza delle esigenze e delle criticità delle grandi aziende in materia di sicurezza delle informazioni, guida Víntegris verso la progettazione di prodotti che coprono il divario tra le soluzioni di sicurezza dei grandi produttori e le reali esigenze aziendali. In questo modo, l'azienda è in grado di anticipare le esigenze di mercato in segmenti critici come la gestione, il controllo e gli audit di certificati digitali e autenticazione dell'utente. Tutti i consulenti di Víntegris hanno una vasta esperienza nel campo della tecnologia dell'informazione e sono esperti nella progettazione, implementazione e gestione d'infrastrutture di sicurezza delle informazioni. Inoltre, la maggior parte dei consulenti è in possesso di certificazioni riconosciute a livello internazionale, come CISA, CISSP e CISM, che garantiscono la loro esperienza e conoscenza nell'ambito della sicurezza delle informazioni.

1.2 ENSign 11

Figura 1.2: ENSign 11

L'ENSign 11[1] è un tablet versatile per firme grafometriche. Con le sue differenti applicazioni, è lo strumento perfetto per la digitalizzazione delle firme a mano. L'ENSign è dotato delle seguenti caratteristiche:

- pannello multi-tocco con isolamento nativo dello schermo;
- connessione diretta a un computer;
- altamente compatto, con grande stabilità, alti livelli di sicurezza e design elegante;
- cattura di parametri biometrici come pressione, accelerazione, velocità, ritmo e movimento in aria;
- sistema proprietario di criptazione per ogni tipo di transazione.

Inoltre, la versatilità dell'ENSign 11 permette di trasformarlo in un secondo schermo, semplicemente collegandolo tramite USB a un computer e installando l'apposita applicazione ENSIGN 11 trayApp, in modo da avere tutte le funzionalità multimediali. Con lo schermo multi-touch di ENSign 11 è possibile creare, scrivere, disegnare e mostrare contenuti multimediali personalizzati in tempo reale, che possono essere automaticamente condivisi attraverso videoconferenze o in presentazioni d'affari.

1.3 Strumenti di comunicazione utilizzati

Durante il tirocinio, sono stati utilizzati i seguenti strumenti di comunicazione:

- **Gmail**[2], per le comunicazioni formali, quali la validazione dei documenti prodotti;
- **Google Meet**[3], per gli allineamenti durante le varie fasi del progetto;

[1] *ENSign 11 Signature Pad.* URL: https://www.euronovategroup.com/solutions/product-map/ensign-11-ensign-nfc-hardware.
[2] *GMail.* URL: https://www.google.com/intl/it/gmail/about/.
[3] *Google Meet.* URL: https://meet.google.com/?pli=1.

- **Telegram**[4], per le comunicazioni informali;
- **Trello**[5], per la pianificazione delle attività settimanali (o per fase, se la durata è inferiore a sette giorni).

1.4 Strumenti di sviluppo

1.4.1 Sistema operativo

L'azienda non ha imposto un determinato sistema operativo per lo sviluppo del progetto, in quanto non determinante per la buona riuscita dello stesso, dunque la mia scelta è ricaduta su Windows 11[6], in quanto è la versione più recente del famoso sistema operativo di casa Microsoft. Inoltre, essendo utilizzato da circa il 28% degli utenti mondiali[7], ha un ampio supporto sia ufficiale che non per l'installazione e la risoluzione di problemi all'interno dello stesso.

1.4.2 Ambiente di sviluppo integrato

Anche in questo caso, veniva lasciato al tirocinante la scelta di un Integrated Development Environment, ovvero di un ambiente di sviluppo integrato. La mia scelta è ricaduta su Visual Studio Code[8], sia per motivi di famigliarità sia per la varietà di tecnologie che il progetto richiedeva di utilizzare. Infatti, grazie alle innumerevoli estensioni, Visual Studio Code è adatto per la programmazione con qualunque linguaggio esistente.

1.4.3 Strumento per la creazione di documenti

Dovendo creare dei documenti, era inoltre necessario scegliere uno strumento che ne permettesse la creazione. In questo caso, la mia scelta è caduta su LaTeX, in quanto avendolo utilizzato in ambito accademico è un linguaggio che ho apprezzato. Per l'installazione, ho utilizzato MiKTex[9], che ne permette l'installazione sui sistemi Windows, e ho utilizzato Visual Studio Code per l'utilizzo di questo linguaggio.

1.5 Organizzazione del testo

Il documento è strutturato nella seguente forma:

Questo capitolo ha fornito una descrizione generale sull'azienda, sul dispositivo oggetto del tirocinio e ha citato gli strumenti di sviluppo e comunicazione utilizzati.

Il secondo capitolo approfondisce in dettaglio in cosa consiste l'ENGaming.

Il terzo capitolo illustra i requisiti che l'ENGaming affronta.

Il quarto capitolo mostra come l'ENGaming garantisce la soddisfazione dei requisiti.

[4] *Telegram*. URL: https://telegram.org/.
[5] *Trello*. URL: https://trello.com/it.
[6] *Windows 11*. URL: https://www.microsoft.com/it-it/windows/windows-11?r=1.
[7] *Market Share dei Sistemi operativi*. URL: https://gs.statcounter.com/os-market-share.
[8] *Visual Studio Code*. URL: https://code.visualstudio.com/.
[9] *MiKTeX*. URL: https://miktex.org/.

Il quinto capitolo indica come si garantisce che il prodotto sviluppato, in questo caso l'ENGaming, rispetti i requisiti definiti.

Nel sesto capitolo si conclude, dando una visione di massima di questo progetto.

Capitolo 2

Descrizione dello stage

In questo capitolo, viene fornita una spiegazione più dettagliata del progetto e della relativa organizzazione.

2.1 Introduzione al progetto

Uno dei principali mercati dove opera Euronovate è il mercato dei device di firma grafometrica dotati di un monitor da 10 pollici multi-touch. Questi dispositivi USB possono essere utilizzati per molteplici scopi oltre all'inserimento di una firma in un documento, nel tempo l'azienda ha visto realizzare sistemi di presentazioni slideshow, digital signage, questionari di valutazione e form per l'aggiornamento delle anagrafiche. Oggi Euronovate vuole spingersi ulteriormente nel mondo dei multimedia e utilizzare il device di firma ENSign 11 per l'intrattenimento videoludico.

2.2 Requisiti e obiettivi

L'obiettivo dello stage è apprendere le nozioni dello sviluppo di applicazioni desktop e d'integrazioni di dispositivi fisici, seguendo una pianificazione settimanale delle attività. Tale obiettivo viene raggiunto attraverso la realizzazione di un prodotto software, dall'analisi dello stesso alla sua validazione, passando per progettazione, codifica e collaudo. In particolare, viene richiesto che il prodotto funzioni su un ambiente desktop e tramite il device di firma collegato allo stesso. Il device stesso è l'unico strumento d'interazione con il software. Viene quindi richiesto che il prodotto da sviluppare, ovvero ENGaming:

- permetta il caricamento di giochi;
- permetta l'interazione con i giochi attraverso un controller virtuale, visibile nel device di firma;
- permetta la realizzazione, il salvataggio e la visione, dei punteggi effettuati nei singoli giochi.

Inoltre, per motivi aziendali, si vuole che il prodotto richiesto possa essere eseguito indipendentemente dalla piattaforma, ovvero si vuole che il prodotto possa essere definito multipiattaforma. Ulteriori feature di prodotto possono essere proposte, con valutazione da parte del tutor.

2.3 Pianificazione

2.3.1 Fasi e dettagli

In accordo con il tutor aziendale, si è deciso di suddividere il tirocinio nelle seguenti fasi:

- Conoscenze generali, dal 2 Maggio al 5 Maggio;
- Formazione personale, dal 8 Maggio al 19 Maggio;
- Analisi dei Requisiti, dal 22 Maggio al 25 Maggio;
- Progettazione Tecnica, dal 25 Maggio al 2 Giugno;
- Codifica, dal 5 Giugno al 6 Luglio;
- Documentazione, dal 7 Luglio al 10 Luglio;
- Demo, l'11 Luglio.

Conoscenze generali

Questa fase, della durata di quattro giorni, prevede l'installazione degli ambienti di sviluppo e di versionamento, nonché un approfondimento degli stessi. Inoltre, vengono abilitati gli strumenti aziendali (account git, AWS e G-Suite) necessari per il tirocinio.

Formazione personale

In questa fase, durata dieci giorni, l'obiettivo è l'approfondimento dello sviluppo attraverso le tecnologie necessarie per il progetto. In particolare, l'approfondimento riguarda Electron per lo sviluppo dell'applicazione desktop, comprese le tecniche di comunicazione tra processi per il passaggio dei dati tra C++ e TypeScript. Inoltre, è necessario approfondire il funzionamento del framework Angular per la parte frontend dell'applicativo. L'approfondimento di questa fase produce una piccola applicazione che unisce tutte le parti appena citate.

Analisi dei Requisiti

Durante questa fase, della durata di quattro giorni, è necessaria per definire gli utenti che utilizzano l'applicativo e i casi d'uso che l'applicazione deve prevedere. Inoltre, vengono elencati i requisiti che l'applicativo deve rispettare (dicasi requisiti obbligatori) e i requisiti che l'applicativo può rispettare (detti requisiti opzionali). Tale analisi è alla base dell'intero sviluppo del prodotto, poiché indica **cosa** deve fare il prodotto, perciò tale fase viene ritenuta conclusa all'approvazione del documento da parte del tutor.

Progettazione tecnica

Questa fase, con durata di sette giorni, indica gli elementi che andranno a creare l'applicativo. Più concretamente, questa fase rivela:

- le classi che verranno utilizzate;
- l'architettura che il prodotto dovrà avere;

- gli elementi di design che il prodotto utilizzerà;
- eventuali file di configurazione, dove si deve elencarne anche la struttura.

La progettazione è cruciale, in quanto permette di non cambiare (o più realisticamente, di cambiare il meno possibile) gli elementi che compongono l'applicativo. Per tal motivo, anche questa fase viene ritenuta conclusa all'approvazione del documento da parte del tutor.

Codifica

Questa fase, la più sostanziosa con la sua durata di ventiquattro giorni, prevede la stesura del codice dell'applicativo, partendo da quanto prodotto nella fase precedente. Durante la stesura del codice, le parti prodotte vengono verificate attraverso dei test, che possono essere sia manuali che automatizzati. Verso la conclusione di questa fase, il prodotto viene collaudato e validato.

Documentazione

In questa fase, della durata di due giorni, si va a creare la documentazione necessaria per l'utilizzo e la manutenzione del prodotto. Per questo progetto, si devono produrre due documenti:

- **Manuale dello Sviluppatore**, dove si elencano i passaggi per la preparazione dell'ambiente di sviluppo, i problemi noti e i possibili miglioramenti;
- **Manuale Utente**, dove si elencano i requisiti di sistema e le attività che si possono effettuare con l'applicativo.

Demo

In questa ultima fase, della durata di un giorno, viene mostrato all'azienda quanto prodotto. Questa esposizione permette al tirocinante di mostrare quanto prodotto, e all'azienda di valutarne possibili riscontri nei prodotti attuali o nei prodotti futuri.

2.3.2 Supervisione e controllo

Le prime fasi dello stage sono di preparazione alle competenze necessarie al corretto svolgimento del progetto. Gli allineamenti con il referente aziendale vengono svolti a ogni termine di attività. Durante la fase di analisi, progettazione e codifica, gli allineamenti sono quotidiani, mentre gli incontri per gli sprint Agile e per la pianificazione delle attività sono a cadenza settimanale.

2.3.3 Diagramma di Gantt

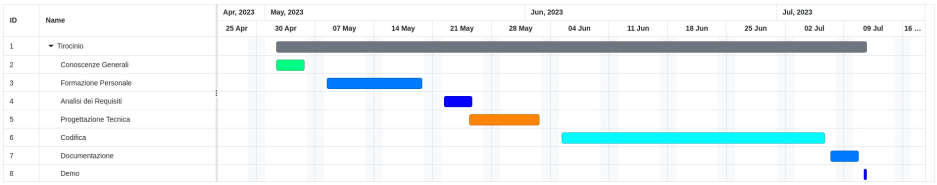


Figura 2.1: Diagramma di Gantt del tirocinio

Capitolo 3

Analisi dei requisiti

3.1 Casi d'uso

In questo capitolo si illustrano i vari casi d'uso per il progetto ENGaming.
Per lo studio dei casi d'uso vengono utilizzati dei diagrammi, chiamati diagrammi dei casi d'uso (in inglese *Use Case Diagram*). Questi diagrammi sono diagrammi UML, dedicati alla descrizione delle funzioni offerte da un applicativo, descrivendoli come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

3.1.1 Attori

Essendo questo prodotto rivolto agli utenti finali, l'unico attore presente nei casi d'uso è l'utente. Non viene fatta una distinzione tra un utente generico e un utente autenticato poichè non viene previsto che l'applicativo implementi un sistema di autenticazione.

Figura 3.1: Attore presente nei casi d'uso

3.1.2 UC01 - Visualizzazione schermata iniziale

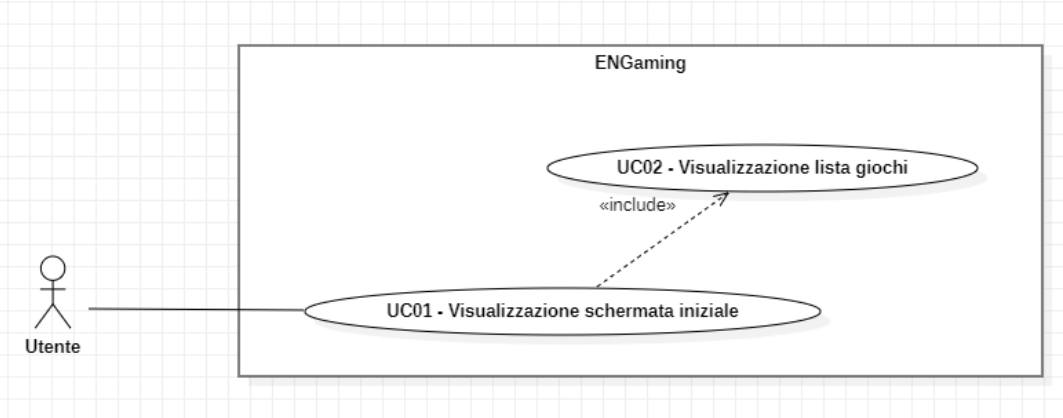


Figura 3.2: Diagramma dei casi d'uso per la schermata iniziale

- Attore principale: Utente
- Pre-condizioni: l'utente non visualizza la schermata iniziale.
- Post-condizioni: l'utente visualizza la schermata iniziale.
- Scenario principale:
  - Il sistema carica l'applicazione sul device ENSign11.
  - L'utente arriva nella postazione in cui è presente il device.
  - L'utente visualizza, sullo schermo del device, una lista di giochi selezionabile e un pulsante relativo ai record.
- Include:
  - UC02 - Visualizzazione lista giochi

3.1.3 UC02 - Visualizzazione lista giochi

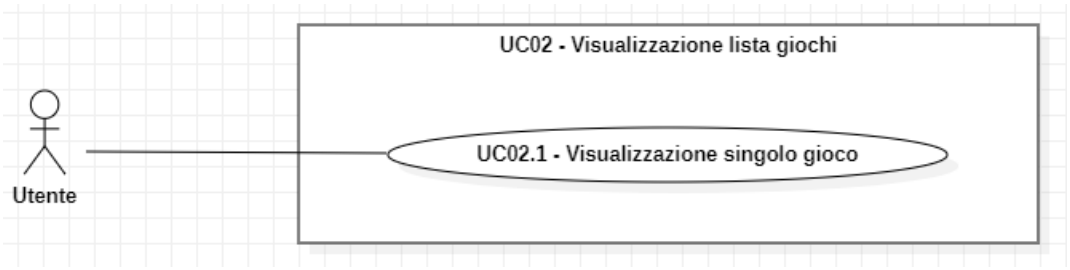


Figura 3.3: Diagramma dei casi d'uso per la visualizzazione della lista dei giochi

- Attore principale: Utente
- Pre-condizioni: l'utente non visualizza la lista dei giochi disponibili.
- Post-condizioni: l'utente visualizza la lista dei giochi disponibili.
- Scenario principale:
  - L'utente visualizza una lista di giochi selezionabili. La lista è composta da icone che possono essere selezionate tramite il tocco. Essendo visualizzabili al massimo dieci icone relative ai giochi, l'utente può utilizzare gli appositi pulsanti per visualizzare le icone non visibili.
- Include:
  - UC02.1 - Visualizzazione singolo gioco

3.1.4 UC02.1 - Visualizzazione singolo gioco

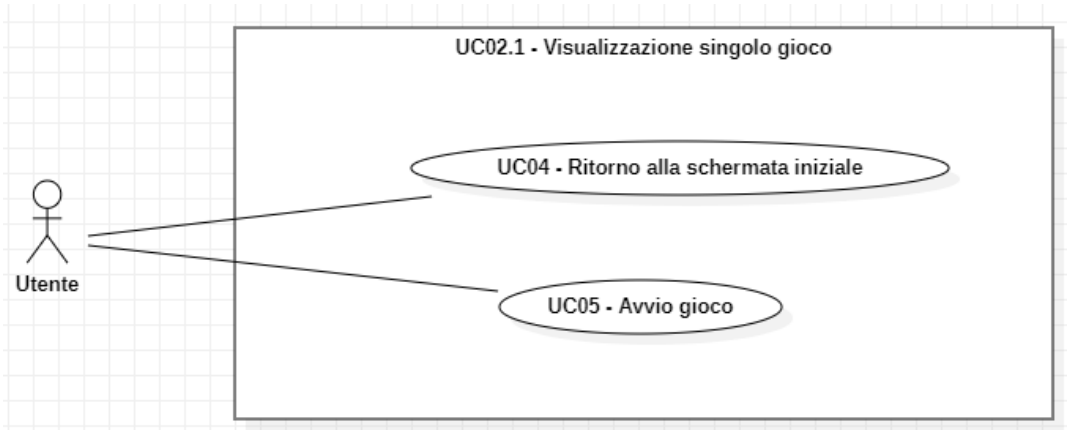


Figura 3.4: Diagramma dei casi d'uso per la visualizzazione di un singolo gioco

- Attore principale: Utente

- Pre-condizioni: l'utente sta visualizzando la lista dei giochi disponibili.
- Post-condizioni: l'utente visualizza le caratteristiche del singolo gioco.
- Scenario principale:
  - L'utente, dalla lista di giochi selezionabili, seleziona un gioco.
  - L'utente viene reindirizzato nella pagina del gioco scelto. Ogni gioco contiene, oltre all'apposito pulsante per iniziare a giocare, le seguenti informazioni:
    * Nome del gioco
    * Tipo di gioco (es. arcade, racing, action...)
    * Descrizione del gioco, se presente
    * Tipologia d'input, tra controller e touch/digitalizer
    * Mappatura dei comandi, se il gioco richiede l'uso del controller

    Inoltre, la pagina contiene anche un pulsante per tornare alla schermata iniziale.
- Include:
  - UC04 - Ritorno alla schermata iniziale
  - UC05 - Avvio gioco

3.1.5 UC03 - Visualizzazione record

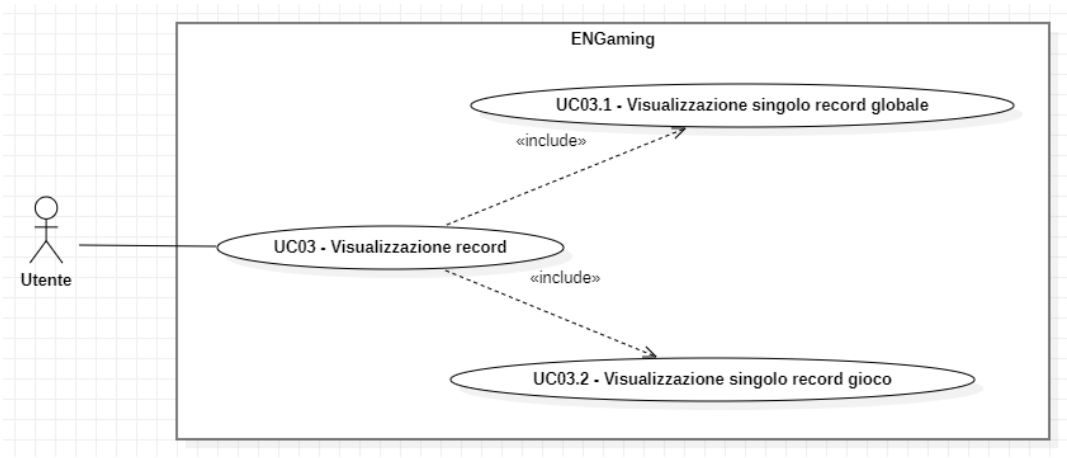


Figura 3.5: Diagramma dei casi d'uso per la visualizzazione dei record

- Attore principale: Utente
- Pre-condizioni: l'utente si trova nella schermata principale, e non sta visualizzando i record.
- Post-condizioni: l'utente visualizza i record.
- Scenario principale:
  - L'utente preme il pulsante relativo ai record, posto nell'area principale.
  - L'utente, in una nuova pagina, visualizza la lista dei record globali, ovvero di tutti i record presenti nel device, ordinati per punteggio. In caso di punteggio uguale, i record vengono ordinati per data ed eventualmente per nome del gioco ascendente.
  - L'utente, tramite gli appositi pulsanti, rimane nella stessa pagina e può passare alle liste per gioco, visualizzando i record presenti per i singoli giochi, ordinati per punteggio. In caso di punteggio uguale, i record vengono ordinati per data.
- Include:
  - UC03.1 - Visualizzazione singolo record globale
  - UC03.2 - Visualizzazione singolo record gioco

3.1.6 UC03.1 - Visualizzazione singolo record globale

- Attore principale: Utente
- Pre-condizioni: l'utente sta visualizzando la lista dei record globali.
- Post-condizioni: l'utente visualizza le caratteristiche del singolo record.

- Scenario principale:
  - L'utente, dalla lista dei record presenti, ne vede i dettagli. Ogni record nella lista globale contiene le seguenti informazioni:
    * Posizione in classifica
    * Nome dell' utente che ha effettuato il record, composto da tre lettere
    * Punteggio conseguito
    * Nome del gioco in cui si è fatto il record

3.1.7 UC03.2 - Visualizzazione singolo record gioco

- Attore principale: Utente
- Pre-condizioni: l'utente sta visualizzando la lista dei record per un determinato gioco.
- Post-condizioni: l'utente visualizza le caratteristiche del singolo record.
- Scenario principale:
  - L'utente, dalla lista dei record presenti, ne vede i dettagli. Ogni record nella lista di un determinato gioco contiene le seguenti informazioni:
    * Posizione in classifica
    * Nome dell' utente che ha effettuato il record, composto da tre lettere
    * Punteggio conseguito

3.1.8 UC04 - Ritorno alla schermata iniziale

- Attore principale: Utente
- Pre-condizioni: l'utente vuole tornare alla schermata iniziale.
- Post-condizioni: l'utente si trova nella schermata iniziale.
- Scenario principale:
  - L'utente preme l'apposito pulsante per ritornare nella schermata iniziale.
  - Il sistema riceve l'input dal device e reindirizza l'utente verso la schermata iniziale.

3.1.9 UC05 - Avvio gioco

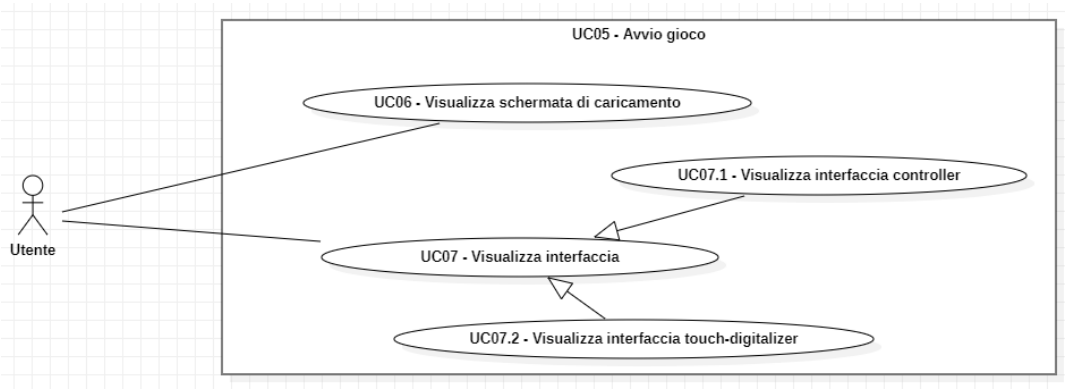


Figura 3.6: Diagramma dei casi d'uso per l'avvio di un gioco

- Attore principale: Utente
- Pre-condizioni: l'utente sta visualizzando le caratteristiche del singolo gioco.
- Post-condizioni: l'utente è pronto per giocare.
- Scenario principale:
  - L'utente, che si trova nella schermata di un gioco, avvia il gioco tramite l'apposito pulsante.
  - L'utente visualizza una pagina di caricamento.
  - Al termine del caricamento, l'utente visualizza una pagina con il gioco e l'interfaccia necessaria.
- Include:
  - UC06 - Visualizza schermata di caricamento
  - UC07 - Visualizza interfaccia

3.1.10 UC06 - Visualizza schermata di caricamento

- Attore principale: Utente
- Pre-condizioni: l'utente ha iniziato l'avvio del gioco.
- Post-condizioni: il caricamento termina e l'utente è pronto per giocare.
- Scenario principale:
  - L'utente, che si trova nella schermata di un gioco, avvia il gioco tramite l'apposito pulsante.
  - L'utente visualizza una pagina di caricamento. Questa pagina, completamente statica, mostra semplicemente una scritta che avvisa l'utente del caricamento, ad esempio "caricamento in corso". La schermata deve rimanere su schermo per quattro secondi, indipendentemente dall'effettivo caricamento del gioco stesso.

3.1.11 UC07 - Visualizza interfaccia

- Attore principale: Utente
- Pre-condizioni: l'utente ha iniziato l'avvio del gioco ed ha appena superato la schermata di caricamento.
- Post-condizioni: il controller viene caricato e l'utente è pronto per giocare.
- Scenario principale:
  - L'utente esce dalla schermata di caricamento.
  - L'utente visualizza la pagina del gioco con l'interfaccia adatta al gioco stesso.

3.1.12 UC07.1 - Visualizza interfaccia controller

- Attore principale: Utente
- Pre-condizioni: l'utente ha iniziato l'avvio del gioco ed ha appena superato la schermata di caricamento.
- Post-condizioni: il controller viene caricato e l'utente è pronto per giocare.
- Scenario principale:
  - L'utente esce dalla schermata di caricamento.
  - L'utente visualizza la pagina del gioco con il controller. Il controller è formato da quattro tasti direzionali, quattro tasti per le azioni e un tasto di pausa. Il posizionamento dei tasti deve permettere l'utilizzo del gioco senza prendere in mano il device utilizzato.

3.1.13 UC07.2 - Visualizza interfaccia touch-digitalizer

- Attore principale: Utente
- Pre-condizioni: l'utente ha iniziato l'avvio del gioco ed ha appena superato la schermata di caricamento.
- Post-condizioni: l'interfaccia viene caricata e l'utente è pronto per giocare.
- Scenario principale:
  - L'utente esce dalla schermata di caricamento.
  - L'utente visualizza la pagina del gioco con l'interfaccia per i comandi touch, o per l'utilizzo del digitalizer. L'interfaccia è semplicemente formata dal tasto di pausa. Il posizionamento del tasto deve permettere l'utilizzo del gioco senza occupare lo spazio necessario per lo stesso, od occupandone il meno possibile.

3.1.14 UC08 - Interazione gioco

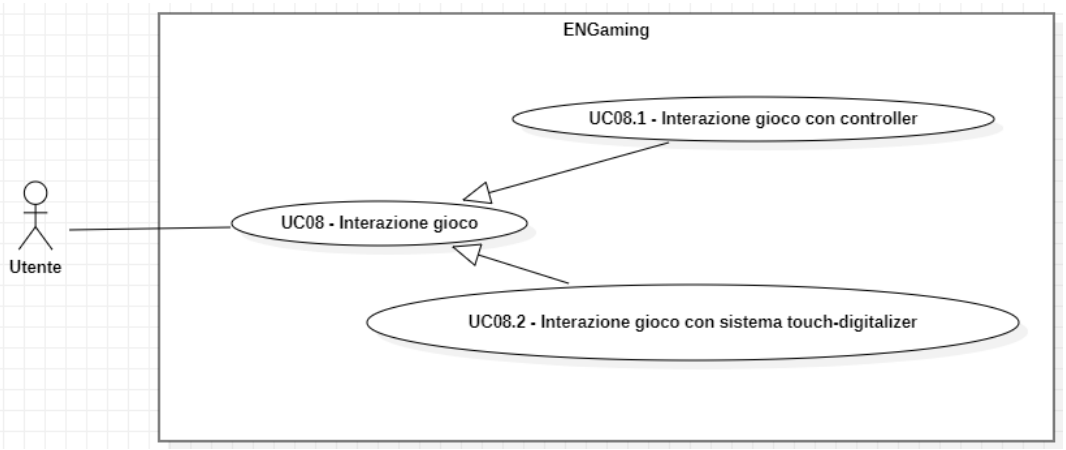


Figura 3.7: Diagramma dei casi d'uso per l'interazione con un gioco

- Attore principale: Utente
- Pre-condizioni: l'utente vuole eseguire una determinata azione nel gioco.
- Post-condizioni: l'utente ha eseguito l'azione nel gioco.
- Scenario principale:
  - L'utente esegue un azione attraverso l'interfaccia del gioco.
  - Il sistema riceve l'input dell'interfaccia dal device e lo elabora.
  - Il sistema prende il dato elaborato e lo manda al gioco.

3.1.15 UC08.1 - Interazione gioco con controller

- Attore principale: Utente
- Pre-condizioni: l'utente vuole eseguire una determinata azione nel gioco.
- Post-condizioni: l'utente ha eseguito l'azione nel gioco.
- Scenario principale:
  - L'utente preme un pulsante nel controller.
  - Il sistema riceve l'input del controller dal device e lo elabora.
  - Il sistema prende il dato elaborato e lo manda al gioco.

3.1.16 UC08.2 - Interazione gioco con sistema touch-digitalizer

- Attore principale: Utente
- Pre-condizioni: l'utente vuole eseguire una determinata azione nel gioco.
- Post-condizioni: l'utente ha eseguito l'azione nel gioco.

- Scenario principale:
  - L'utente esegue una gesture, ovvero un gesto, nel gioco.
  - Il sistema riceve l'input del gesto dal device e lo elabora.
  - Il sistema prende il dato elaborato e lo manda al gioco.

3.1.17 UC09 - Pausa del gioco

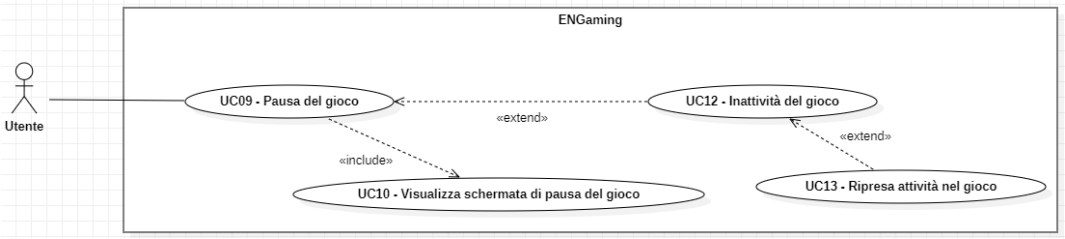


Figura 3.8: Diagramma dei casi d'uso per la pausa di un gioco

- Attore principale: Utente
- Pre-condizioni: l'utente vuole mettere in pausa il gioco.
- Post-condizioni: il gioco è in stato di pausa.
- Scenario principale:
  - L'utente preme l'apposito pulsante di pausa.
  - Il sistema riceve l'input dal device e mette in pausa il gioco.
  - L'utente visualizza una pagina che informa dello stato di pausa del gioco.
- Include:
  - UC10 - Visualizza schermata di pausa del gioco
- Estensioni:
  - UC12 - Inattività del gioco

3.1.18 UC10 - Visualizza schermata di pausa del gioco

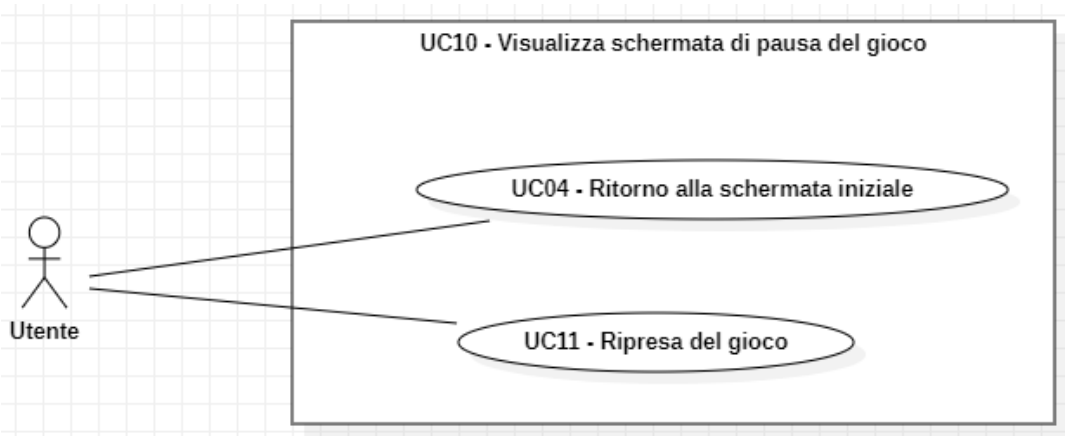


Figura 3.9: Diagramma dei casi d'uso per la schermata di pausa

- Attore principale: Utente
- Pre-condizioni: L'utente ha messo in pausa il gioco.
- Post-condizioni: L'utente è informato dello stato di pausa del gioco.
- Scenario principale:
  - L'utente visualizza una pagina che informa dello stato di pausa del gioco. La pagina contiene una scritta che avvisa l'utente dello stato di pausa del gioco e due pulsanti. Il primo pulsante permette di ricominciare a giocare, togliendo il gioco dallo stato di pausa, mentre il secondo permette di chiudere il gioco e tornare alla schermata iniziale.
- Include:
  - UC04 - Ritorno alla schermata iniziale
  - UC11 - Ripresa del gioco

3.1.19 UC11 - Ripresa del gioco

- Attore principale: Utente
- Pre-condizioni: il gioco è in stato di pausa.
- Post-condizioni: il gioco è uscito dallo stato di pausa e l'utente può continuare a giocare.
- Scenario principale:
  - L'utente preme l'apposito pulsante di ripresa del gioco.
  - Il sistema riceve l'input dal device e toglie lo stato di pausa dal gioco.

3.1.20 UC12 - Inattività nel gioco

- Attore principale: Utente
- Pre-condizioni: l'utente non sta eseguendo alcuna attività sul gioco.
- Post-condizioni: il sistema visualizza la schermata iniziale.
- Scenario principale:
  - Dopo 10 secondi dall'ultimo input, il sistema avvia un cronometro di quattro minuti.
  - Al termine del cronometro, il sistema chiude il gioco e torna alla schermata iniziale.
- Estensioni:
  - UC13 - Ripresa attività nel gioco

3.1.21 UC13 - Ripresa attività nel gioco

- Attore principale: Utente
- Pre-condizioni: l'utente non sta eseguendo alcuna attività sul gioco.
- Post-condizioni: l'utente ha eseguendo un'attività sul gioco.
- Scenario principale:
  - Dopo 10 secondi dall'ultimo input, il sistema avvia un cronometro di quattro minuti.
  - L'utente esegue un'attività sul gioco prima dello scadere del cronometro.
  - Il sistema interrompe il cronometro e lo riporta alla durata iniziale.

3.1.22 UC14 - Inserimento nuovo record

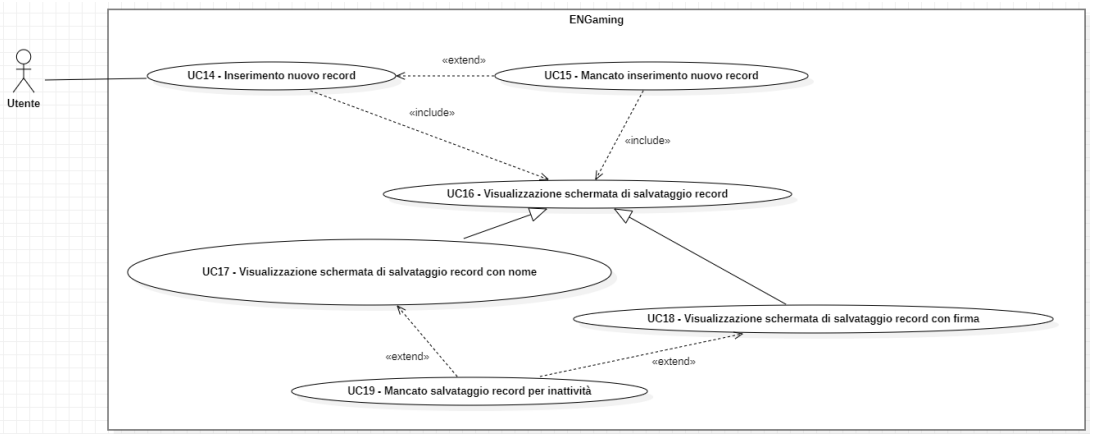


Figura 3.10: Diagramma dei casi d'uso per l'inserimento di un nuovo record

- Attore principale: Utente
- Pre-condizioni: l'utente chiude il gioco, ed ha effettuato un nuovo record.
- Post-condizioni: il sistema ha memorizzato il record effettuato dall'utente.
- Scenario principale:
  - L'utente visualizza una schermata per informarlo della creazione di un nuovo record.
  - Tramite la schermata, l'utente seleziona il metodo di salvataggio del record che più preferisce.
  - Il sistema riceve il dato inserito dall'utente e memorizza il record.
- Include:
  - UC16 - Visualizzazione schermata di salvataggio record
- Estensioni:
  - UC15 - Mancato inserimento nuovo record

3.1.23 UC15 - Mancato inserimento nuovo record

- Attore principale: Utente
- Pre-condizioni: l'utente ha effettuato un nuovo record in un gioco, ma non vuole salvarlo.
- Post-condizioni: il sistema non ha memorizzato il record effettuato dall'utente.
- Scenario principale:
  - L'utente visualizza una schermata per informarlo della creazione di un nuovo record.

  - Tramite la schermata, l'utente seleziona la volontà di non salvare il record effettuato.
  - Il sistema scarta i dati relativi al record e non lo memorizza.
- Include:
  - UC16 - Visualizzazione schermata di salvataggio record

3.1.24 UC16 - Visualizzazione schermata di salvataggio record

- Attore principale: Utente
- Pre-condizioni: l'utente ha effettuato un nuovo record in un gioco.
- Post-condizioni: l'utente sta visualizzando la schermata relativa al salvataggio del record.
- Scenario principale:
  - L'utente visualizza la schermata che lo informa della creazione di un nuovo record, e ne propone il salvataggio. Questa schermata, oltre a riportare il punteggio del record, propone tre opzioni: il salvataggio del record tramite nome, il salvataggio del record tramite firma e il non salvataggio del record.

3.1.25 UC17 - Visualizzazione schermata di salvataggio record con nome

- Attore principale: Utente
- Pre-condizioni: l'utente vuole salvare il record con il proprio nome.
- Post-condizioni: l'utente ha inserito il proprio nome correttamente.
- Scenario principale:
  - L'utente visualizza la schermata di salvataggio tramite nome. Il nome può essere inserito tramite due pulsanti per carattere, ognuno dei quali passa al carattere precedente o al successivo. Una volta che l'utente è soddisfatto, può premere l'apposito pulsante per il salvataggio del nome.
- Estensioni:
  - UC19 - Mancato salvataggio record per inattività

3.1.26 UC18 - Visualizzazione schermata di salvataggio record con firma

- Attore principale: Utente
- Pre-condizioni: l'utente vuole salvare il record con la propria firma.
- Post-condizioni: l'utente ha inserito la propria firma correttamente.
- Scenario principale:

  - L'utente visualizza la schermata di salvataggio tramite firma. La firma può essere inserita sia tramite l'utilizzo del touch screen, quindi firmando con il dito, sia tramite l'utilizzo del digitalizer. Una volta che l'utente è soddisfatto, può premere l'apposito pulsante per il salvataggio della firma.

- Estensioni:
  - UC19 - Mancato salvataggio record per inattività

3.1.27 UC19 - Mancato salvataggio record per inattività

- Attore principale: Utente
- Pre-condizioni: l'utente non sta eseguendo alcuna attività sulla schermata di salvataggio record.
- Post-condizioni: il record non viene salvato nel sistema.
- Scenario principale:
  - Dopo dieci secondi dall'ultima interazione, il sistema avvia un cronometro di quattro minuti.
  - Al termine del cronometro, il sistema chiude la schermata di salvataggio record, non salva il record e torna alla schermata iniziale.

3.2 Requisiti

In questa sezione vengono elencati i requisiti analizzati per l'applicativo. Le tabelle, suddivide per tipologia di requisito, forniscono le seguenti informazioni:

- **Codice**, che indica la tipologia di requisito (tra F=funzionale, Q=qualità e V=vincolo) e il numero del requisito;
- **Requisito**, dove si indica il requisito vero e proprio;
- **Tipologia**, ovvero se il requisito indicato è obbligatorio oppure opzionale;
- **Caso d'uso** di riferimento (solo per i requisiti funzionali).

3.2.1 Requisiti funzionali

Codice	Requisito	Tipologia	Caso d'uso
RF01	Dev'essere possibile visualizzare la lista di giochi disponibili	Obbligatorio	UC01,UC02
RF02	Dev'essere possibile utilizzare dei pulsanti per visualizzare le icone nascoste	Obbligatorio	UC02
RF03	Dev'essere possibile toccare le icone presenti nella schermata iniziale	Obbligatorio	UC01,UC02,UC03
RF04	Dev'essere possibile visualizzare le informazioni relative a un singolo gioco	Obbligatorio	UC02.1
RF05	Dev'essere possibile visualizzare la lista dei record globali	Obbligatorio	UC03
RF06	Dev'essere possibile visualizzare la lista dei record per ogni gioco presente	Obbligatorio	UC03
RF07	Dev'essere possibile utilizzare dei pulsanti per visualizzare i record nascosti	Obbligatorio	UC03
RF08	Dev'essere possibile visualizzare le informazioni relative a un singolo record	Obbligatorio	UC03.1,UC03.2
RF09	Dev'essere possibile ritornare alla schermata iniziale	Obbligatorio	UC04
RF10	Dev'essere possibile avviare un gioco	Obbligatorio	UC05
RF11	Dev'essere possibile visualizzare una pagina di caricamento	Obbligatorio	UC06

RF12	La pagina di caricamento dev'essere visibile per almeno quattro secondi	Obbligatorio	UC06
RF13	Dev'essere possibile l'utilizzo di un controller virtuale	Obbligatorio	UC07.1
RF14	Il controller deve avere nove tasti	Obbligatorio	UC07.1
RF15	Il controller non deve richiede l'impugnatura del device utilizzato	Obbligatorio	UC07.1
RF16	Dev'essere possibile giocare utilizzando le dita delle mani	Obbligatorio	UC07.2
RF17	Dev'essere possibile giocare utilizzando il digitalizer	Obbligatorio	UC07.2
RF18	Il pulsante di pausa deve occupare il meno possibile l'area di gioco	Obbligatorio	UC07.2
RF19	Dev'essere possibile interagire con un gioco attraverso il controller	Obbligatorio	UC08.1
RF20	Dev'essere possibile interagire con un gioco attraverso le dita delle mani	Obbligatorio	UC8.2
RF21	Dev'essere possibile interagire con un gioco attraverso il digitalizer	Obbligatorio	UC8.2
RF22	Dev'essere possibile mettere in pausa un gioco	Obbligatorio	UC09
RF23	Il sistema deve avvisare quando un gioco è in pausa	Obbligatorio	UC09,UC10
RF24	Dev'essere possibile far ripartire un gioco dopo la pausa	Obbligatorio	UC10,11
RF25	Dev'essere possibile chiudere un gioco e tornare alla schermata iniziale	Obbligatorio	UC10,UC04
RF26	Il sistema deve tornare alla schermata iniziale dopo quattro minuti d'inattività	Obbligatorio	UC12,UC19
RF27	Il sistema deve far partire il cronometro d'inattività dopo dieci secondi dall'ultimo input	Obbligatorio	UC12,UC19
RF28	Il sistema deve interrompere il cronometro d'inattività all'azione dell'utente, prima del suo scadere	Obbligatorio	UC13

RF29	Dev'essere possibile memorizzare un nuovo record	Obbligatorio	UC14
RF30	L'utente deve poter scegliere se salvare o no il record effettuato	Obbligatorio	UC14,UC15,UC16
RF31	L'utente deve visualizzare la schermata dove decidere se salvare un record	Obbligatorio	UC16
RF32	L'utente deve poter salvare il record tramite il suo nome	Obbligatorio	UC17
RF33	L'utente deve poter inserire il suo nome	Obbligatorio	UC17
RF34	L'utente deve poter salvare il record tramite la sua firma	Opzionale	UC18
RF35	L'utente deve poter inserire la sua firma	Opzionale	UC18
RF36	Il sistema non deve salvare il record in caso d'inattività	Obbligatorio	UC19

Tabella 3.1: Tabella dei requisiti funzionali

3.2.2 Requisiti di qualità

Codice	Requisito	Tipologia
RQ01	Dev'essere fornita la documentazione necessaria per l'utilizzo del prodotto	Obbligatorio
RQ02	Dev'essere fornita la documentazione necessaria per la manutenzione del prodotto	Obbligatorio
RQ03	Il prodotto dev'essere versionato e pubblicato sulle piattaforme aziendali	Obbligatorio

Tabella 3.2: Tabella dei requisiti di qualità

3.2.3 Requisiti di vincolo

Codice	Requisito	Tipologia
RV01	L'interfaccia grafica del prodotto dev'essere sviluppata utilizzando i linguaggi HTML, CSS, JavaScript e TypeScript	Obbligatorio
RV02	L'interfaccia grafica del prodotto dev'essere sviluppata utilizzando il framework Angular	Obbligatorio

RV03	L'interfaccia grafica del prodotto dev'essere sviluppata utilizzando il framework Electron	Obbligatorio
RV04	Il prodotto dev'essere utilizzabile attraverso il device ENSign 11	Obbligatorio
RV05	Il prodotto deve utilizzare il linguaggio C++ per la ricezione dei dati dal device	Obbligatorio
RV06	Il prodotto dev'essere utilizzabile senza impugnare il device	Obbligatorio
RV07	Il prodotto dev'essere utilizzabile attraverso il sistema touch screen del device	Obbligatorio
RV08	Il prodotto dev'essere utilizzabile attraverso il digitalizer presente nel device	Obbligatorio

Tabella 3.3: Tabella dei requisiti di vincolo

Capitolo 4

Progettazione e codifica

4.1 Tecnologie e strumenti

In questa sezione, si descrivono le tecnologie utilizzate sia per ENGaming sia per i giochi. Infatti, nonostante i giochi non necessitino di modifiche per il funzionamento in ENGaming, trovo comunque opportuno citarne le tecnologie che utilizzano, che ritengo possano essere interessanti.

4.1.1 Tecnologie e strumenti del prodotto

Electron

Figura 4.1: Logo di Electron

Electron viene utilizzato per creare l'applicazione desktop, visualizzandola nello schermo dell'ENSign11. Inoltre, si occupa della comunicazione tra il backend e il frontend dell'applicazione, attraverso canali di Inter-Process Communication.

Angular

Figura 4.2: Logo di Angular

Angular è il framework scelto per la realizzazione della parte visiva dell'applicazione. Angular permette la creazione di Componenti, ovvero di elementi o pagine, dove sono presenti:

- un file HTML, per la visualizzazione del componente;
- un foglio di stile (CSS/SASS/LESS) per gli effetti grafici;
- un file TypeScript, per definire il comportamento del componente.

Inoltre, permette l'utilizzo di Servizi per il trasferimento di dati tra componenti.

TypeScript

Figura 4.3: Logo di TypeScript

TypeScript viene utilizzato sia da Electron che da Angular (da quest'ultimo per la definizione del comportamento di componenti e servizi). Ciò che caratterizza TypeScript da JavaScript è la tipizzazione delle variabili, elemento che trovo molto utile sopratutto se si è in precedenza lavorato con linguaggi di programmazione come Java e C++,

come nel mio caso. I sorgenti scritti in questo linguaggio devono poi essere compilati per generare il file JavaScript effettivamente letto dal browser.

C++

Figura 4.4: Logo di C++

C++ è utilizzato per la logica relativa al device. In particolare, attraverso un driver (fornito dall'azienda), si possono rilevare le interazioni con il device (sia tramite tocco che tramite digitalizer).

Node-API

Figura 4.5: Logo delle Node-API

Le Node-API sono utilizzate per rendere la parte in C++ "usabile" dalla parte sviluppata in Electron. Infatti, senza di esse, non sarebbe possibile utilizzare i metodi per l'interazione con il device.

Electron-Forge

Electron Forge viene utilizzato per creare degli installer, o dei pacchetti, a seconda del sistema operativo in cui si va a compilare il progetto. Tale strumento permette di compilare l'applicazione per Windows, Linux e MacOS, e ciò permette di definire ENGaming come un applicativo multipiattaforma.

4.1.2 Tecnologie e strumenti dei giochi

HTML-CSS-JavaScript

Figura 4.6: Loghi di HTML, CSS e JavaScript

I giochi che ENGaming esegue sono semplici giochi eseguibili anche su browser. Per tale motivo, sono scritti con HTML per la struttura (generalmente composta da canvas), CSS per lo stile e JavaScript per il gioco vero e proprio. Per usufruire al meglio di tutte le funzionalità che l'applicativo offre, in particolare per poter salvare i record relativi ai giochi, è necessario che i giochi stessi implementino una piccola modifica, che consiste nell'inviare un messaggio all'applicativo. Infatti, essendo il gioco visualizzato all'interno di una pagina (attraverso un elemento iframe), è necessario che il gioco mandi il punteggio "in alto", utilizzando la seguente riga di codice:

```
window.top.postMessage({type:'gameIsOver',data:score});
```

Questo comando manda un messaggio con il punteggio conseguito. Il tipo (in questo caso, "gameIsOver") indica la tipologia del messaggio, in modo da poterlo distinguere rispetto ad altri messaggi.

WebAssembly

Figura 4.7: Logo di WebAssembly

Nel caso si volesse eseguire un classico nell'ENGaming (ad esempio, Doom), bisogna prima passare per WebAssembly, in modo da renderlo eseguibile su un browser. Le modifiche da fare per il funzionamento dipendono dal singolo gioco, è in ogni caso indispensabile essere in possesso dei sorgenti. Tale gioco viene poi letto da un normale file JavaScript attraverso il file con estensione .wasm, che non permette di effettuare la modifica per la ricezione del punteggio, quindi per tali giochi non si possono avere record salvati.

4.2 Progettazione

4.2.1 Classi utilizzate

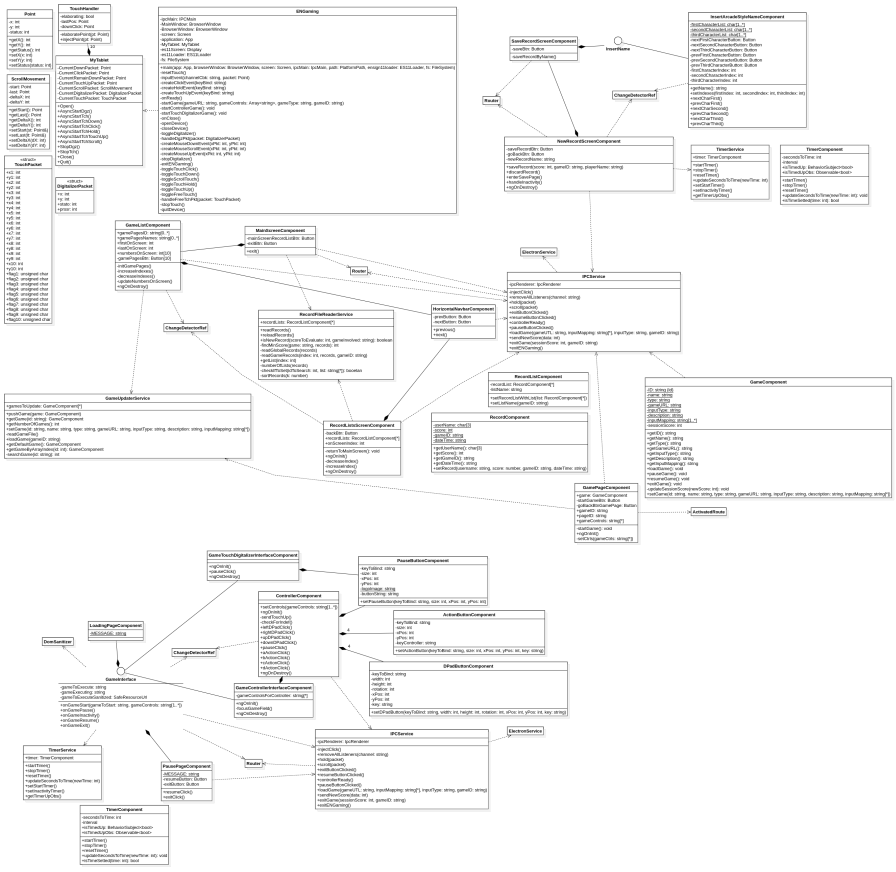


Figura 4.8: Diagramma completo delle classi utilizzate

Le classi utilizzate possono essere classificate tramite il linguaggio d'implementazione:

- Electron:
  - ENGaming.
- C++:
  - MyTablet;
  - TouchHandler;
  - Point;
  - ScrollMovement;
  - TouchPacket;
  - DigitalizerPacket.
- Angular:

- Tutte le classi "Component";
- Tutte le classi "Service";
- GameInterface;
- InsertName;
- Router;
- ChangeDetectorRef;
- DomSanitizer;
- ActivatedRoute.

Di seguito si analizzano le vari componenti con maggior dettaglio.

ENGaming

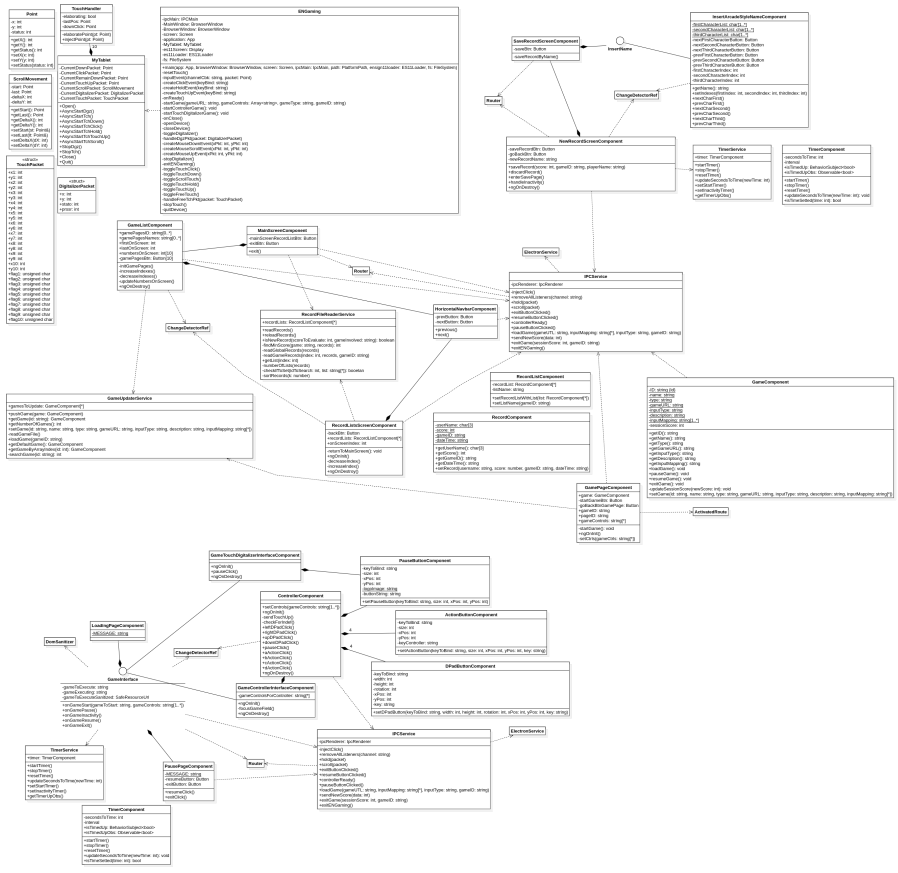


Figura 4.9: Classe ENGaming

La classe *ENGaming* si occupa dell'avvio dell'applicazione. Sviluppata in Electron, contiene i componenti necessari per la creazione della finestra nell'ENSign 11 (Screen, BrowserWindow) e per la comunicazione con la parte in Angular (IPCMain). Inoltre, contiene anche i moduli per la comunicazione con ENSign11.
Contiene:

- i metodi per interagire con l'ENSign11:
  - *openDevice*, per aprire il device;
  - *toggleTouchClick*, per ricevere un click;
  - *toggleTouchDown*, per ricevere un down, ovvero l'evento di un dito appoggiato sullo schermo del device;
  - *toggleTouchHold*, per ricevere un hold, ovvero l'evento di un dito ancora appoggiato sullo schermo del device;
  - *toggleTouchUp*, per ricevere un touch up, ovvero l'evento di un dito sollevato dallo schermo del device;

  - *toggleScrollTouch*, per ricevere un evento di scroll, ovvero un evento di movimento di un dito sullo schermo del device;
  - *toggleFreeTouch*, per ricevere i pacchetti definiti nella struttura *TouchPacket*;
  - *stopTouch*, per fermare le interazioni con lo schermo del device;
  - *toggleDigitalizer*, per ricevere i pacchetti definiti nella struttura *DigitalizerPacket*;
  - *stopDigitalizer*, per fermare le interazioni con il digitalizer;
  - *closeDevice*, per chiudere il device;
  - *quitDevice*, per "liberare" il device e renderlo disponibile ad altri applicativi (o a una nuova istanza di ENGaming).

- i metodi per l'avvio dei giochi:
  - *startGame*: identifica la tipologia del gioco e lo inizializza;
  - *startControllerGame*: inizializza un gioco che richiede l'uso del controller;
  - *startTouchDigitalizerGame*: inizializza un gioco che richiede l'uso del dito o del digitalizer.

- i metodi per interagire con un gioco che utilizza il controller:
  - *inputEvent*, che permette l'interazione con il controller, nell'apposito componente;
  - *createClickEvent*, che crea l'evento "click" nella pagina del gioco;
  - *createHoldEvent*, che crea l'evento "hold" nella pagina del gioco;
  - *createTouchUpEvent*, che crea l'evento "touch up" nella pagina del gioco.

- i metodi per interagire con un gioco che utilizza il dito o il digitalizer:
  - *handleFreeTchPkt*: in base allo stato del pacchetto, si decide che evento sollevare;
  - *handleDgzPkt*: come per *handleFreeTchPkt*, per il digitalizer;
  - *createMouseDownEvent*: crea l'evento di appoggio del dito/digitalizer nel gioco;
  - *createMouseScrollEvent*, crea l'evento di movimento del dito/digitalizer nel gioco;
  - *createMouseUpEvent*, crea l'evento di sollevamento del dito/digitalizer nel gioco.

- *resetTouch*: ripristina il touch del device alla fine di un gioco;
- *exitENGaming*: chiude l'applicazione.

Schermata iniziale

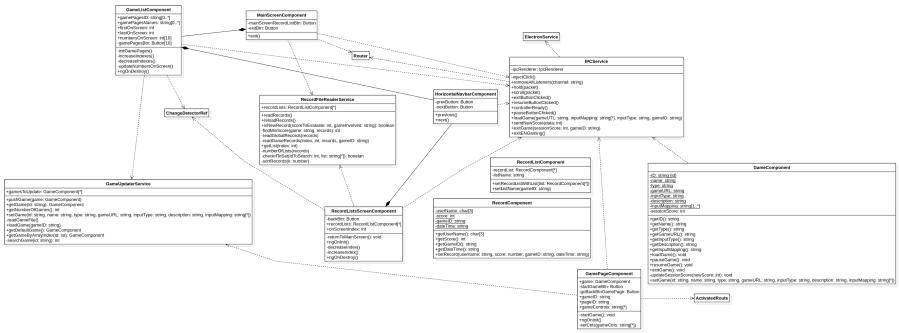


Figura 4.10: Classi della schermata iniziale

All'avvio dell'applicazione, la schermata iniziale presenta tre elementi:

- L'elenco dei giochi disponibili, ottenuto da *GameListComponent*;
- Il pulsante per accedere alla pagina di *RecordListScreenComponent*;
- Il pulsante per uscire dall'applicazione.

Inoltre, il componente che gestisce questa schermata, ovvero *MainScreenComponent*, si serve del service *IPCService* per le comunicazioni attraverso le altre componenti dell'applicazione, utilizzando i canali IPC. A sua volta:

- *RecordListScreenComponent* è formato da almeno un *RecordListComponent*, poichè la schermata ha almeno la lista globale dei record. Inoltre, utilizza il *HorizontalNavbarComponent* per navigare tra una lista e l'altra;
- *RecordListComponent* contiene il nome della lista stessa e utilizza *IPCService* per la comunicazione con il resto dell'applicazione. Ovviamente, contiene anche vari *RecordComponent*;
- *GameListComponent* è formato sia dall' *HorizontalNavbarComponent*, poichè gestisce la visualizzazione della lista di giochi disponibili, sia da almeno una *GamePageComponent*, ossia da almeno una pagina relativa a un gioco;
- Ogni *GamePageComponent* è relativa a un solo *GameComponent*, le cui informazioni sono caricate da GameUpdaterService. Contiene sia il *GameComponent*, sia i pulsanti per avviare il gioco e per tornare alla schermata principale.

GameComponent e *RecordComponent* utilizzano anche determinate proprietà, che si elencano negli appositi paragrafi rispettivamente in GameComponent e RecordComponent.

GameComponent

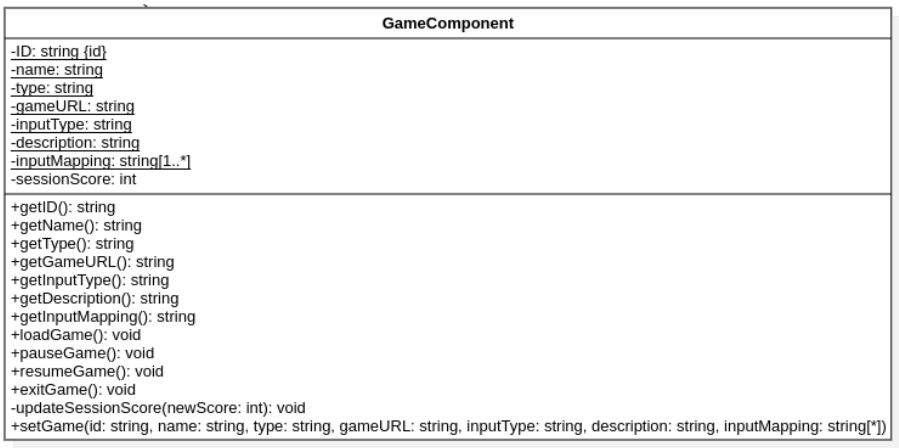


Figura 4.11: Dettaglio della classe GameComponent

GameComponent rappresenta un gioco all'interno dell'applicativo. Contiene tutte le informazioni necessarie per l'avvio e il controllo dello stesso, oltre alle classiche informazioni "anagrafiche".
Un gioco viene rappresentato come:

- ID: l'identificativo del gioco, formato da una stringa;
- name: nome del gioco;
- type: la categoria del gioco;
- gameURL: l'indirizzo dove reperire la finestra di gioco, sia locale che remoto;
- inputType: la tipologia d'input, tra "controller" e "touchDigit" (ovvero touch/digitalizer);
- description: opzionale. Descrizione del gioco, con informazioni generali in merito. Se non sono disponibili, il campo assume valore "not_available";
- inputMapping: la mappatura dei controlli da utilizzare per giocare.

Inoltre, utilizza *IPCService* per inviare e ricevere eventi inerenti al gioco in esecuzione, e utilizza la variabile sessionScore per registrare il punteggio di gioco più alto durante la sessione. Infine, i giochi sono memorizzati nel file *games.json*, della cui struttura si parlerà in Struttura dei file utilizzati.

RecordComponent

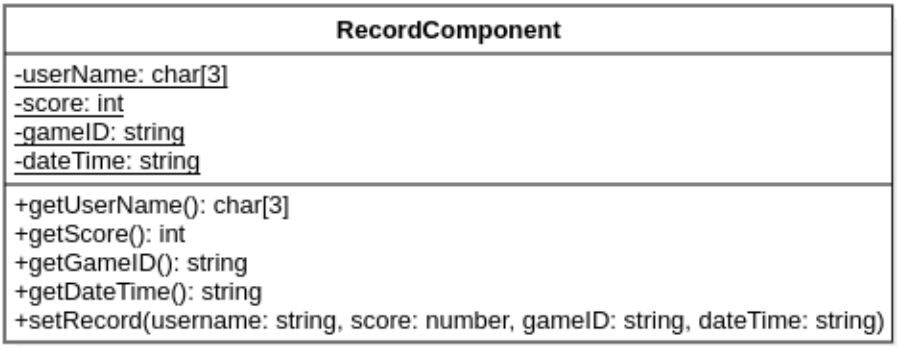


Figura 4.12: Dettaglio della classe RecordComponent

RecordComponent rappresenta un record memorizzato dall'applicativo. Contiene le informazioni relative a un punteggio, effettuato da un determinato utente in un determinato gioco.
Un record viene rappresentato come:

- userName: il nome dell'utente che ha effettuato un record, formato da un array di tre caratteri;
- score: il punteggio eseguito dall'utente;
- gameID: l'ID del gioco dove si è effettuato il record;
- dateTime: la data e ora di conseguimento del record.

I record sono memorizzati nel file *records.json*, della cui struttura si parlerà sempre in Struttura dei file utilizzati.

Interfaccia di gioco

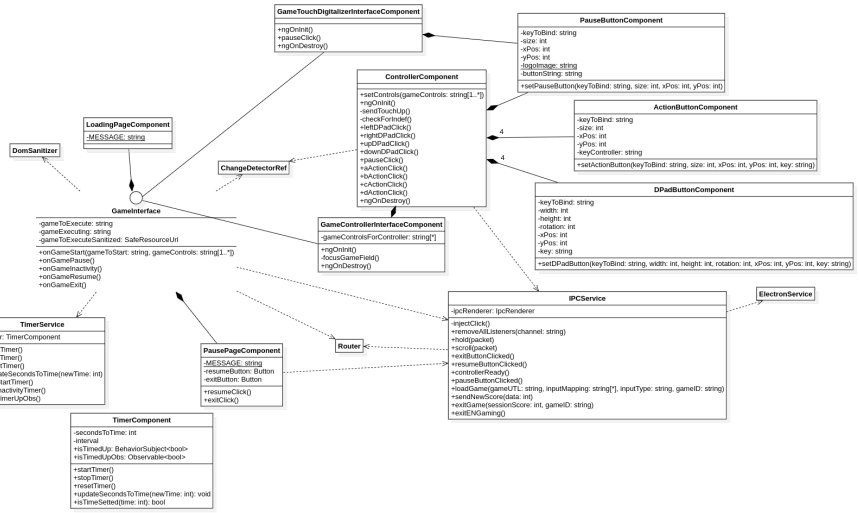


Figura 4.13: Diagramma delle classi per l'interfaccia di gioco

Dovendo implementare due tipologie di gioco, dev'essere necessario poter creare due interfacce separate, che abbiano però elementi in comune. Per questo motivo, viene utilizzata l'interfaccia *GameInterface*, che contiene il necessario per la corretta visualizzazione del gioco.
In particolare, utilizza:

- *TimerComponent*, attraverso *TimerService*, per la gestione della schermata di caricamento e lo stato d'inattività;
- *LoadingPageComponent* per la visualizzazione della schermata di caricamento;
- *PausePageComponent* per visualizzare l'informazione dello stato di pausa del gioco, con le possibilità di riprenderlo o di uscire dallo stesso;
- *IPCService* per le comunicazioni tra componenti;
- *Router*, *DomSanitizer* e *ChangeDetectorRef*, per i motivi elencati in Servizi di Angular.

Questa interfaccia viene implementata da due classi: *GameControllerInterfaceComponent* e *GameTouchDigitalizerInterfaceComponent*.

GameControllerInterfaceComponent *GameControllerInterfaceComponent* rappresenta l'interfaccia di gioco per giochi che richiedono l'utilizzo del controller. Questa classe, oltre a essere formata dagli elementi che implementa da *GameInterface*, è formata da un controller. Il controller è composto da:

- Quattro *DPadButtonComponent*, che rappresentano i pulsanti per il movimento all'interno del gioco.
- Quattro *ActionButtonComponent*, che rappresentano i pulsanti per le azioni all'interno del gioco.
- Un *PauseButtonComponent* per mettere in pausa il gioco.

Il controller viene configurato, prima dell'avvio, secondo i comandi previsti per il singolo gioco. Se uno o più pulsanti non vengono configurati per un determinato gioco, essi vengono rimossi dalla visualizzazione, al fine di non creare confusione durante l'utilizzo del gioco. Infine, essendo il device target posizionato in orizzontale su una superficie piana, il layout è configurato in maniera da facilitarne l'utilizzo senza prendere in mano il device. In particolare, gli *ActionButtonComponent* presenti sono posizionati in modo da risultare il più possibilmente comodi per questa posizione, ispirandosi alla posizione della mano destra di un pianista. Un layout di esempio, adatto per questi vincoli, è nell'immagine seguente:

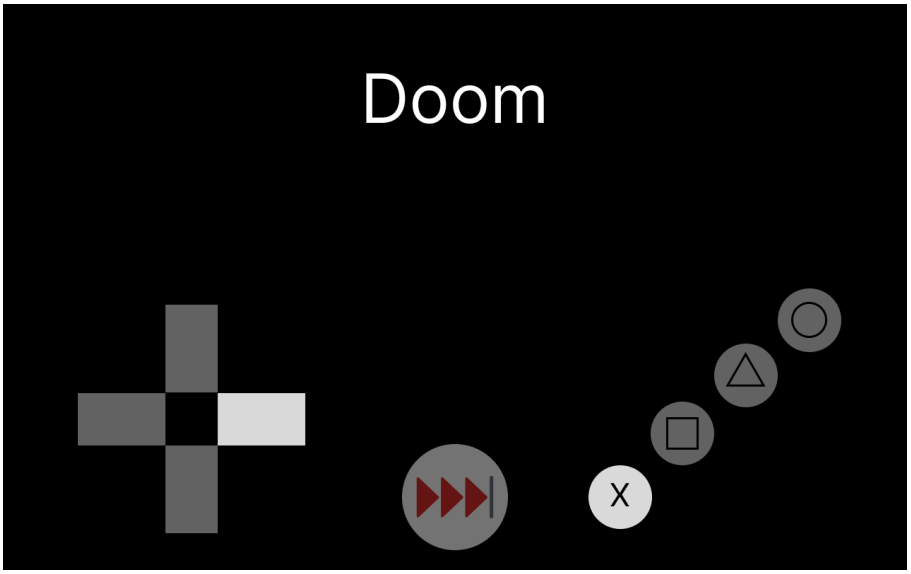


Figura 4.14: Prototipo dell'interfaccia per giochi che richiedono l'uso del controller

GameTouchDigitalizerInterfaceComponent *GameTouchDigitalizerInterfaceComponent* rappresenta l'interfaccia di gioco per giochi che richiedono l'utilizzo del touch, oppure l'utilizzo del digitalizer. Si noti come in questa classe non si richiede all'utente se scegliere un input tramite il sistema touch o tramite il digitalizer. Questa scelta permette all'utente di utilizzare la modalità che più preferisce, senza vincolarlo. Oltre a essere formata dagli elementi che implementa da *GameInterface*, la classe è formata da un *PauseButtonComponent* per mettere in pausa il gioco. Il *PauseButtonComponent*

viene posizionato in alto a destra, poichè l'obiettivo è d'invadere il meno possibile lo spazio di gioco, in modo da non arrecare disturbo durante il gioco.
La visualizzazione di tale layout è presente nella seguente figura:

Figura 4.15: Prototipo dell'interfaccia per giochi che richiedono l'uso di touch/digitalizer

Schermata di salvataggio record

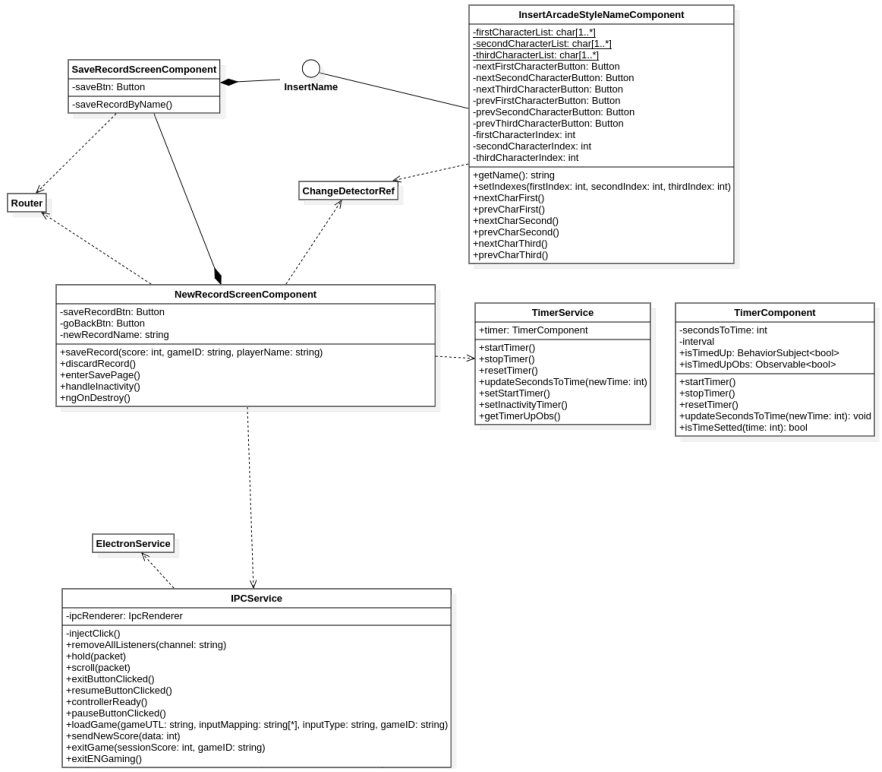


Figura 4.16: Diagramma delle classi per il salvataggio dei record

Alla notifica di un nuovo record, viene visualizzata la pagina *NewRecordScreenComponent*, dove viene chiesto all'utente se desidera salvare il record appena effettuato o non salvarlo. Nel caso di salvataggio del record, si passa alla pagina *SaveRecordScreenComponent*, utilizzando un *Router*, che si occupa di ricevere il nome dell'utente. Per far ciò, si avvale della classe *InsertArcadeStyleNameComponent*. Tale classe, che implementa l'interfaccia *InsertName*, prevede l'inserimento dei tre caratteri che compongono il nome utente scegliendo tra i caratteri disponibili, utilizzando gli appositi bottoni. *SaveRecordScreenComponent* inoltre utilizza anche *TimerComponent* attraverso *TimerService* per la gestione dell'inattività in quella determinata schermata.

TimerComponent

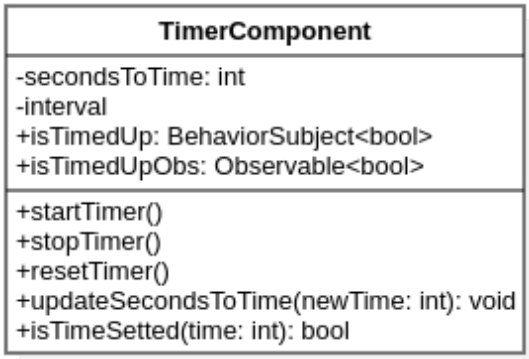


Figura 4.17: Dettaglio della classe TimerComponent

TimerComponent, servito in altri componenti attraverso *TimerService*, si occupa della gestione del tempo. Il suo scopo è, dato un tempo espresso in millisecondi, d'iniziare il conteggio, e notificare quando il tempo ha raggiunto la fine. Per far questo, imposta un intervallo della durata memorizzata, notificando tramite un *Observable* quando scade.

Gestione ENSign11

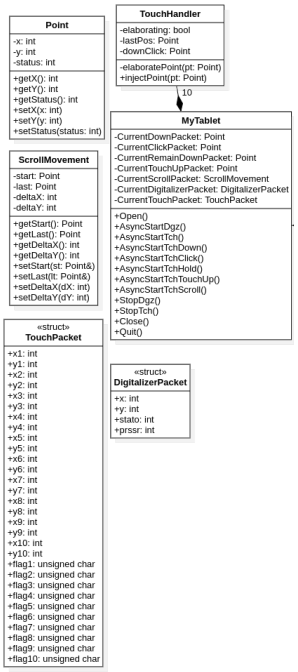


Figura 4.18: Diagramma per la gestione dell'input da ENSign11

Il device di firma, ovvero l'ENSign11, fornisce le funzioni per interagire con lo stesso attraverso una classe, chiamata *MyTablet*. Questa classe permette l'interazione con il device, come l'apertura e la chiusura dello stesso. Inoltre, offre le funzioni per ottenere gli input dallo stesso. Per il recupero dell'input dal device ENSign11, *MyTablet* si serve della classe *TouchHandler* per il recupero di eventi click, hold, scroll...
Questa classe, della quale *MyTablet* ha dieci istanze (una per ogni possibile iterazione, per un massimo di dieci tocchi in contemporanea), rileva se quanto avviene sullo schermo rientra tra i seguenti eventi:

- **Down**: il primo pacchetto che il driver "emette", indica che il dito è stato appena appoggiato sullo schermo;
- **Click**: indica che il dito è stato appoggiato sullo schermo per un breve periodo di tempo;
- **Scroll**: indica che il dito è ancora appoggiato sullo schermo, ma le sue coordinate hanno superato la tolleranza (impostata per evitare falsi movimenti);
- **Hold**: indica che il dito è ancora appoggiato sullo schermo;
- **Touch Up**: l'ultimo pacchetto che il driver "emette", indica che il dito è stato sollevato dallo schermo.

I punti, rappresentati da oggetti, che *MyTablet* manda attraverso gli appositi metodi sono:

- **Point**: oggetto che contiene le coordinate del punto e lo stato. Questo oggetto viene mandato per gli eventi elencati prima, a eccezione dell'evento **Scroll**, durante i giochi che richiedono l'uso del controller e nell'utilizzo generale dell'applicativo;
- **ScrollMovement**: oggetto che contiene i punti (attraverso oggetti *Point*) d'inizio e fine del movimento, e le variazioni delle coordinate x e y. Questo oggetto viene mandato per l'evento **Scroll**;
- **TouchPacket**: oggetto che contiene le coordinate e lo stato di tutti e dieci i tocchi. L'oggetto viene inviato durante l'esecuzione di giochi che non richiedono l'uso del controller;
- **DigitalizerPacket**: oggetto che contiene le coordinate, lo stato del digitalizer e la pressione applicata dall'utente. L'oggetto viene inviato durante l'esecuzione di giochi che non richiedono l'uso del controller.

Servizi di Angular

ENGaming utilizza i seguenti servizi, offerti da Angular, per l'esecuzione del programma:

- **Router**: permette di essere reindirizzati da un componente a un altro, aggiornando la vista, tramite il metodo *navigate*;
- **ChangeDetectorRef**: aggiorna la vista tramite il metodo *detectChanges*;
- **ActivatedRoute**: permette di recuperare l'URL della pagina in cui ci si trova;
- **ElectronService**: permette l'utilizzo dei canali IPC e delle funzioni generalmente presenti in Electron;
- **DomSanitizer**: effettua il processo di "pulizia" di un URL, permettendo la visualizzazione del suo contenuto in un elemento *iframe* oppure *object*.

4.2.2 Struttura dei file utilizzati

Come indicato in GameComponent e RecordComponent, vengono utilizzati due file json, il primo contiene le informazioni sui giochi (*games.json*), il secondo contiene i record effettuati (*records.json*).

Games.json

Il file, che viene aperto da ENGaming solamente in lettura, contiene le informazioni necessarie dei giochi e delle loro relative pagine. Ne consegue che la modifica di questo file debba essere fatta manualmente. Un esempio di giochi presenti nel file è il seguente:

```
[
    {
        "id": "catMario",
        "name": "Cat Mario",
        "type": "Azione",
        "gameURL": "https://www.cat-mario.com/",
        "inputType": "controller",
        "description": "Clone di Super Mario, con un gatto.
            Per nulla stressante!",
        "inputMapping": [
            "salto:A:Space",
            "muovi a sinistra:DPad Left:Left",
            "muovi a destra:DPad Right:Right"
        ]
    },
    {
        "id": "subwaySurfers",
        "name": "Subway Surfers",
        "type": "Endless",
        "gameURL": "src_angular/assets/games/subwaySurfers/
            index.html",
        "inputType": "touchDigit",
        "description": "not_available",
        "inputMapping": [
            "inizia: tap",
            "muovi a sinistra: swipe a sinistra",
            "muovi a destra: swipe a destra",
            "salto: swipe in alto",
            "rotolata:swipe in basso"
        ]
    }
]
```

Dove i giochi con il controller richiedono anche la conoscenza dei tasti che il gioco richiede tramite l'utilizzo della tastiera. Inoltre, i giochi possono essere caricati con i sorgenti tramite il percorso relativo degli stessi.

Records.json

Il file contiene tutti i record effettuati nell'applicativo. Tale file non è condiviso in rete ed è unico per ogni installazione. Quindi, i record possono variare da una macchina a un'altra. Questo è l'unico file che viene caricato da ENGaming sia per la lettura che per la scrittura, non richiedendo una scrittura manuale dei dati. Un esempio di record che il file contiene è il seguente:

```
[
    {
        "userName": "AAA",
        "score": "250",
        "gameID": "dukeNukem",
        "dateTime":"2024-02-15 17:23"
    },
    {
        "userName": "RIC",
        "score": "420",
        "gameID": "angryBirds",
        "dateTime":"2024-05-19 08:23"
    }
]
```

4.2.3 Scelte progettuali

Oltre a quanto già visto, ho adoperato diverse scelte durante questa fase.
In particolare:

- architetturalmente, ho scelto di utilizzare il modello Model-View-ViewModel (abbreviato MVVM), in quanto tra le possibili architetture l'ho ritenuta la più adatta per il progetto;
- l'utilizzo di Angular mi consente di ridurre l'accoppiamento per componenti attraverso l'utilizzo della *Dependency Injection* per l'utilizzo dei servizi;
- l'utilizzo degli Inter-Process Communication mi permettono di far comunicare gli elementi che compongono l'applicazione, in particolare tra le classi implementate in Angular ed Electron, attraverso lo scambio di messaggi in appositi canali.Tale comportamento è simile al comportamento di un *Observer*;
- per rendere minore la dipendenza tra le componenti, ho scelto di non utilizzare alcun tipo di gerarchia. Questa scelta è dovuta, oltre alla non reale necessità del suo utilizzo, alla limitatezza a livello applicativo, in quanto tra le tecnologie da usare solo C++ permette l'uso di una gerarchia;
- invece di ricorrere all'utilizzo delle gerarchie, ho scelto di definire le parti comuni attraverso un'interfaccia, che viene poi implementata dalle singole classi che la necessitano.

4.3 Codifica

4.3.1 Strumenti utilizzati

Per lo sviluppo di ENGaming, oltre a quanto già citato in Strumenti di sviluppo, ho utilizzato:

- NodeJS, versione 18.16.0;
- Npm, versione 9.5.1;
- TypeScript, versione 4.9.5;
- C++, versione 17;
- AngularCLI, versione 15.2.8;
- Electron, versione 25.0.1;
- Git, versione 2.41.0 .

Inoltre, il codice versionato è stato caricato nella repository apposita, presente nello spazio Amazon Web Services (AWS) aziendale.

4.3.2 Ambiente di sviluppo

Per tenere una suddivisione chiara delle parti da sviluppare, ho impostato la cartella di lavoro secondo questa struttura:

- src_angular
  - app
    - COMPONENTI
    - «servizi»
    - «interfacce»
  - assets
    - games
      - GIOCHI
    - previews
      - «immagini»
    - games.json
    - records.json
    - «altri file»
  - index.html
  - main.ts
  - «altri file»
- src_electron
  - App.ts
  - ENGaming.ts

- src_module
  - ES11LIB
  - ES11LOADER
  - LIBUSB
  - «altri file»

Dove:

- COMPONENTI indica le cartelle di tutti i componenti, di cui la struttura si è precedentemente illustrata in Angular;
- «servizi» indica i file dei servizi, che sono dotati solo di un file TypeScript per la definizione dei comportamenti;
- «interfacce» indica i file delle interfacce, dotati solo di un file TypeScript per la definizione delle stesse;
- GIOCHI indica le cartelle dove sono presenti i giochi, con i relativi sorgenti;
- «immagini» indica le immagini utilizzate dall'applicativo;
- ES11LIB indica la cartella che contiene i driver per il funzionamento dell'ENSign11, ricevuti direttamente dall'azienda;
- ES11LOADER indica la cartella dove risiedono i file per la gestione dell'ENSign11, ai quali ho personalmente lavorato;
- LIBUSB indica la cartella dove risiede la libreria usata dai file contenuti in ES11LIB;
- «altri file» indica altri file, presenti nelle cartelle ma di minore interesse (ad esempio, file .gitignore).

Nello stesso livello delle cartelle principali, sono presenti anche i file di configurazione, in particolare i file *package.json*, *angular.json*, *tsconfig.json* e *forge.config.js*. Il progetto, durante il suo sviluppo, è stato versionato e caricato su una repository, presente in AWS.

4.3.3 Utilizzo dell'applicativo

Alla fine della fase di codifica, assistito dal tutor, ho creato gli eseguibili (installers oppure pacchetti, a seconda della piattaforma), attraverso Electron Forge. Allo stato da me raggiunto, ENGaming può essere utilizzato con:

- un PC con installato il driver DisplayLink e uno dei seguenti sistemi operativi:
  - Windows 11[1] (non necessario installare il driver citato in precedenza, in quanto già presente);
  - MacOS Ventura[2].
- un ENSign11[3] da collegare a suddetto PC.

Purtroppo non ho potuto testare l'applicativo in altre versioni di Windows o MacOS, a causa della mancanza di tempo nel farlo. Non sono invece stati testati altri device di firma poichè non sono stati considerati come device target, a differenza dell'ENSign11. Su sistemi operativi con kernel Linux, l'applicativo veniva eseguito ma non riconosceva il tablet, costringendo un utilizzo tramite tastiera e mouse e in generale con errori dovuti proprio al non riconoscimento del device.

[1] *Windows 11*.
[2] *MacOS Ventura*. URL: https://www.apple.com/it/macos/ventura/.
[3] *ENSign 11 Signature Pad*.

4.3.4 Panoramica del prodotto

Per concludere questa sezione, voglio mostrare il frutto della fase di codifica, partendo dalla schermata iniziale e concludendo con la gestione dell'inattività.

Schermata iniziale

La schermata iniziale dell'applicazione comprende:

- Il nome dell'applicazione stessa;
- L'elenco dei giochi disponibili, sotto forma d'icone;
- Un'icona per visualizzare i record effettuati;
- Un'icona per uscire dall'applicazione.

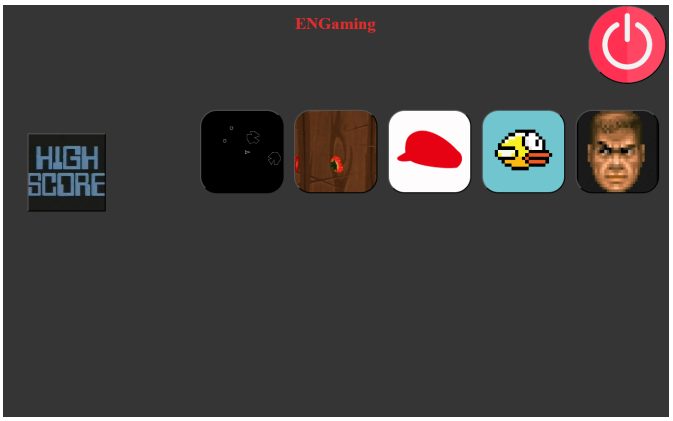


Figura 4.19: Schermata iniziale di ENGaming

Ho scelto una visualizzazione il più possibile vicina alle icone per portare un senso di famigliarità, in quanto ENGaming viene usato attraverso uno schermo touch.

Visualizzazione record

Attraverso l'icona "High Score", è possibile passare alla visualizzazione dei record. La pagina apposita mostra la lista dei record globale, ovvero mostra tutti i record effettuati senza distinzione di gioco.
In particolare, ogni record mostra:

- l'utente che ha effettuato il record;
- il punteggio totalizzato;
- la data in cui è stato effettuato il record;
- il gioco in cui è stato effettuato il record.

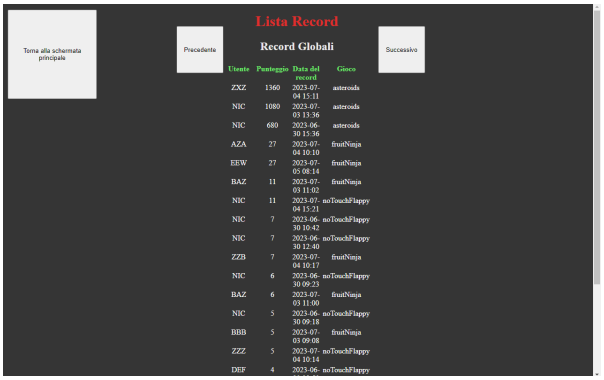


Figura 4.20: Lista dei record globali

Utilizzando i pulsanti "attorno" alla lista, si può passare alla lista dei record per i giochi specifici. Tali liste sono tante quanti i giochi in cui sia stato fatto almeno un record. La prima e l'ultima lista di questo genere, alla pressione degli appositi pulsante, riportano alla lista globale.

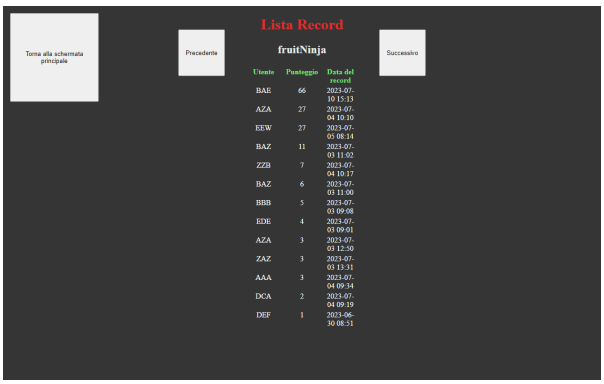


Figura 4.21: Esempio di lista per un singolo gioco

Pagina di gioco

Ritornando alla schermata iniziale, si può selezionare un gioco attraverso la propria icona. Tale operazione porta alla pagina del gioco stesso, che contiene le informazioni sul gioco stesso, ovvero:

- il nome del gioco;
- il genere del gioco;
- il tipo d'input, ovvero se viene utilizzato il controller o il touch/digitalizer;
- la descrizione del gioco;
- L'elenco degli input, o delle gesture, che il gioco ha.

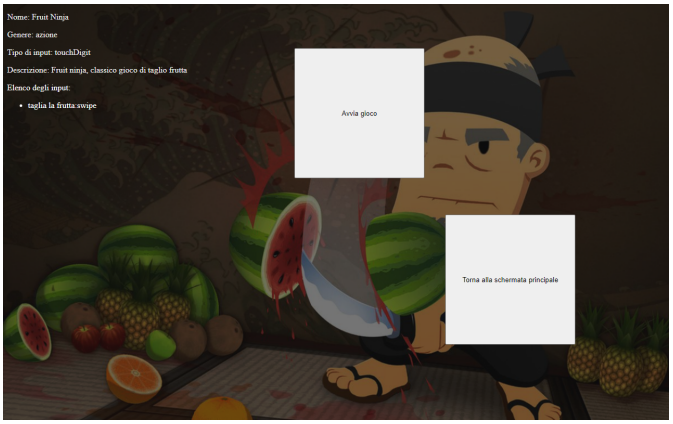


Figura 4.22: Esempio di pagina di un gioco

Ovviamente, da questa pagina si può decidere se avviare il gioco o tornare alla schermata principale.

Schermata di caricamento

All'avvio del gioco, si passa attraverso una pagina di caricamento. La pagina in questione è spoglia, in quanto informa semplicemente l'utente dello stato di caricamento. Questa pagina viene visualizzata per quattro secondi, per poi passare all'interfaccia di gioco.

Figura 4.23: Schermata di caricamento

Schermate di gioco

La differenza tra le due tipologie di gioco entra adesso. Infatti, se il gioco prevede l'utilizzo del controller, oltre al gioco stesso si visualizza anche il controller, che dalla struttura di default (spiegata in GameControllerInterfaceComponent), viene configurato attraverso gli input che il gioco richiede.

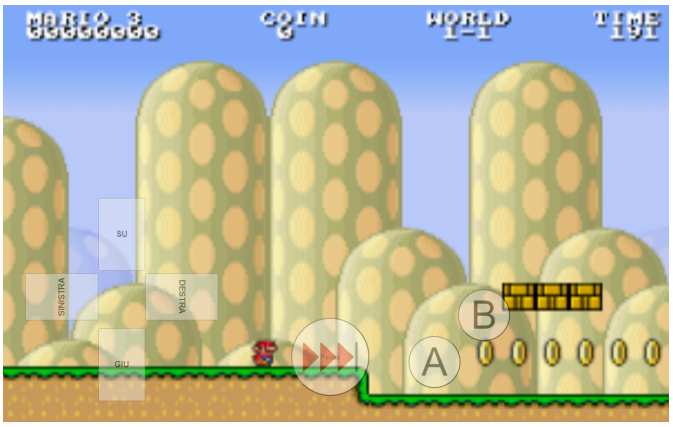


Figura 4.24: Esempio di schermata per un gioco con il controller

Mentre, se il gioco prevede l'utilizzo del touch o del digitalizer, ciò che viene visualizzato sono solo il gioco e il pulsante di pausa, essendo gli input del gioco dati dalle gestures.

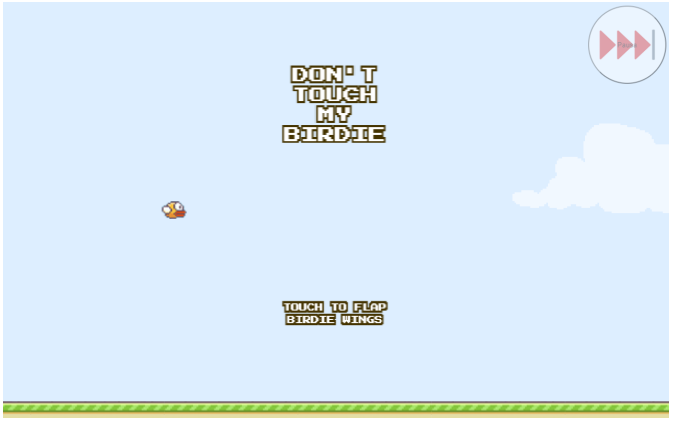


Figura 4.25: Esempio di schermata per un gioco con il touch/digitalizer

Schermata di pausa

In ogni gioco, a prescindere dal tipo, premendo sul pulsante di pausa si va all'apposita schermata di pausa. La schermata di pausa informa l'utente dello stato di pausa del gioco, e ne permette la ripresa oppure l'uscita dallo stesso.

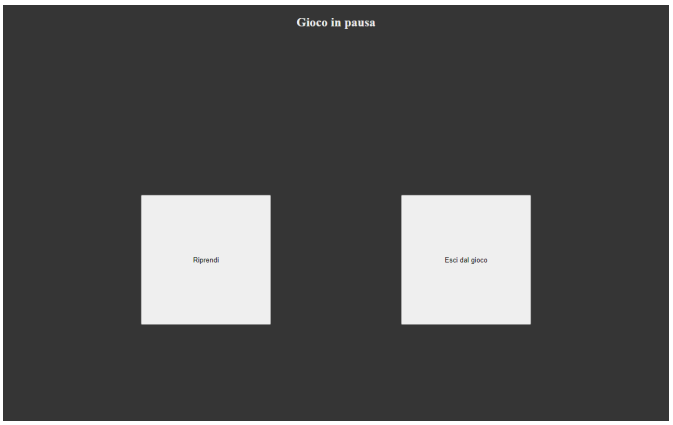


Figura 4.26: Schermata di pausa

Salvataggio record

Ovviamente una parte importante dell'applicazione è il salvataggio dei record effettuati nei giochi. Per far questo, si passa attraverso due schermate. La prima informa semplicemente l'utente che ha effettuato un nuovo record nel gioco appena concluso. Oltre a questo, permette allo stesso di decidere se salvarlo o meno.

Figura 4.27: Schermata di avviso per un nuovo record

Nel caso non si voglia salvare il record, l'applicativo semplicemente ritorna nel menù principale, perdendo il punteggio da salvare. Se invece l'utente vuole salvare il record, viene portato alla seconda schermata, dedicata all'inserimento del nome.
Il nome viene inserito attraverso dei pulsanti, che permettono di selezionare, una per volta, le tre lettere che compongono il nome a cui associare il record.

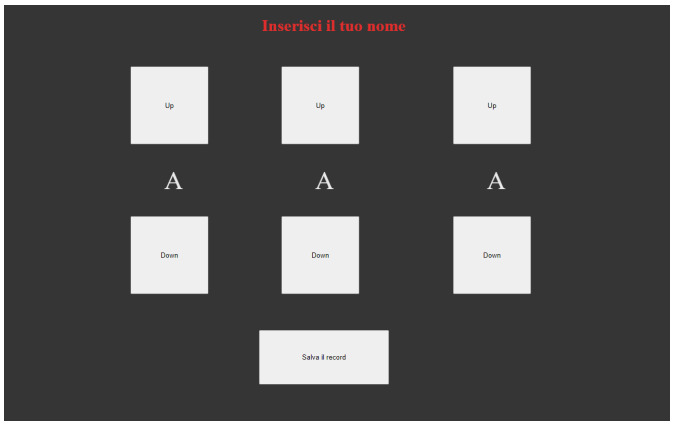


Figura 4.28: Schermata per l'inserimento del nome

Ho scelto lo stile dei giochi arcade per un principio specifico, ovvero l'inserimento d'input senza dover ricorrere a periferiche esterne a quelle presenti. Infatti, grazie a questa modalità, si può utilizzare lo schermo del device senza incorre in tastiere virtuali o fisiche.

4.3.5 Gestione degli errori

A livello di comportamento, ENGaming prevede pochi casi di errore, essendo un sistema che non prevede configurazioni da parte dell'utente. Voglio comunque elencare delle piccole accortezze che ho implementato, in quanto si sono rivelate utili anche alla prevenzione di altri errori.

Gestione del non collegamento del device

ENGaming, durante l'avvio, cerca d'individuare un ENSign11, in modo da collocarci l'applicazione al suo interno. Nel caso l'utente non colleghi un device prima dell'avvio, l'applicazione riporta un errore all'utente, terminando l'esecuzione del programma.

Gestione dell'inattività

L'applicazione, se non rileva alcun input per dieci secondi, fa partire un timer d'inattività della durata massima di quattro minuti. Tale timer si azzera se durante questo lasso di tempo riceve un input. Nel caso scada il tempo, il sistema chiude l'attività in esecuzione (gioco o salvataggio del record) e ritorna alla schermata principale. Tale azione comporta la perdita di un eventuale record effettuato e non ancora salvato, in quanto l'attività precedentemente in esecuzione viene chiusa forzatamente.

Capitolo 5

Verifica e validazione

Verifica e validazione sono due parti fondamentali dello sviluppo di un software. Il processo di verifica permette di controllare che quanto è prodotto sia effettivamente congruo con i requisiti, attraverso test automatici o manuali. Il processo di validazione permette la convalida di quanto prodotto, secondo quanto precedentemente verificato.

Per la fase di verifica, per questo progetto, non sono ricorso all'utilizzo di test automatizzati, poiché i casi d'uso e i comportamenti che l'applicativo esegue sono definiti e limitati in quantità. Dunque, ho effettuato la verifica sulle singole parti in maniera manuale, anche attraverso il debug quando necessario, tramite gli strumenti da me scelti.

La fase di validazione è stata fatta assieme al tutor, in due momenti distinti. Infatti, la prima validazione è stata effettuata ancor prima della fase di Analisi, che ha decretato quanto prodotto durante la fase di Formazione Personale come Proof of Concept[1]. La seconda (e vera) validazione ha coinvolto l'applicativo vero e proprio, collaudandolo e accettandolo come prodotto finale.

[1] Con Proof of Concept si intende una realizzazione incompleta o abbozzata di un progetto, allo scopo di dimostrarne la fattibilità.

Capitolo 6

Conclusioni

6.1 Raggiungimento degli obiettivi e analisi del prodotto

Con questo tirocinio, ho soddisfatto il 95% dei requisiti totali, soddisfacendo al 100% i requisiti obbligatori, come riportato in tabella.

Codice	Stato
RF01	Soddisfatto
RF02	Soddisfatto
RF03	Soddisfatto
RF04	Soddisfatto
RF05	Soddisfatto
RF06	Soddisfatto
RF07	Soddisfatto
RF08	Soddisfatto
RF09	Soddisfatto
RF10	Soddisfatto
RF11	Soddisfatto
RF12	Soddisfatto
RF13	Soddisfatto
RF14	Soddisfatto
RF15	Soddisfatto
RF16	Soddisfatto
RF17	Soddisfatto
RF18	Soddisfatto
RF19	Soddisfatto
RF20	Soddisfatto
RF21	Soddisfatto
RF22	Soddisfatto
RF23	Soddisfatto
RF24	Soddisfatto
RF25	Soddisfatto
RF26	Soddisfatto
RF27	Soddisfatto

RF28	Soddisfatto
RF29	Soddisfatto
RF30	Soddisfatto
RF31	Soddisfatto
RF32	Soddisfatto
RF33	Soddisfatto
RF34	Non Soddisfatto
RF35	Non Soddisfatto
RF36	Soddisfatto

Tabella 6.1: Tabella di soddisfacimento dei requisiti

Non ho potuto soddisfare i requisiti opzionali (R34 e R35) poiché il tempo necessario per il loro completamento era superiore al tempo rimastomi. Quanto sviluppato corrisponde dunque ai requisiti analizzati e al comportamento aspettato.
In particolare, come anche già visto in Panoramica del prodotto, ENGaming permette di:

- visualizzare le informazioni relativa a un gioco;
- avviare il gioco e interagirci attraverso l'interfaccia presente;
- mettere in pausa, riprendere e uscire dal gioco;
- salvare i record effettuati;
- visualizzare i record effettuati.

6.2 Conoscenze acquisite

In primis, ho imparato molto sulla creazione di elementi web tramite Angular. Venendo da una conoscenza "base", ovvero con solo l'utilizzo di HTML, CSS e JavaScript, ho trovato interessanti i contenuti che Angular stesso propone. Ho sicuramente apprezzato l'utilizzo di TypeScript, che grazie alla tipizzazione è stato concettualmente semplice da utilizzare per me, avendo un background con linguaggi tipizzati (come Java e C++). Inoltre, l'utilizzo delle Node-API per poter eseguire codice C++ in ambito web è stato molto interessante, sopratutto a livello comunicativo, venendo utilizzate tecnologie ideate per ambiti diversi.

6.3 Valutazione personale

Questo tirocinio ha sicuramente aiutato la mia crescita sia professionale che personale. Gli strumenti che ho utilizzato si sono rivelati adeguati allo scopo, e ciò ha reso nettamente più facile l'intero processo di sviluppo. Le fasi, che inizialmente vedevo come limitate temporalmente, si sono dimostrate adatte e mi hanno permesso di non avere ritardi e completare quanto poi definito. Effettuare analisi e progettazione mi hanno permesso di non avere intoppi durante lo sviluppo. Inoltre, mi hanno aiutato a migliorare le conoscenze in merito allo sviluppo software, rispetto a quanto visto durante il percorso accademico. Le tecnologie utilizzate sono state per me molto interessanti, a prescindere dalla conoscenza pregressa che avevo. L'ambiente di lavoro

è stato positivo e stimolante, e ha sicuramente aiutato nella realizzazione di questo progetto. Nel complesso, sono soddisfatto di come è andato il tirocinio, delle conoscenze apprese e del risultato ottenuto.

Acronimi e abbreviazioni

AWS Amazon Web Services. 49, 50

Glossario

Angular *Angular* (conosciuto anche come *"Angular 2+"*) è un *framework* per applicazioni web, gratuito e *open source*, basato su *TypeScript* e mantenuto sia dal *Team* di *Angular* in *Google* sia dalla *community*, da singoli e da altre aziende. Completa riscrittura dallo stesso team che ha sviluppato *AngularJS*, *Angular* è un *framework* per applicazioni a pagina singola, utilizzato per la creazioni di applicazioni web veloci. 7, 30, 36, 48, 61

C++ *C++* è un linguaggio di programmazione orientato a oggetti, cross-platform, che può essere utilizzato per lo sviluppo di applicazioni per alte prestazioni. Essendo uno dei linguaggi di programmazione più popolari, C++ può essere trovato nei sistemi operativi moderni, in interfacce grafiche e sistemi *embedded*. C++ nasce come estensione del linguaggio C, supportando la creazione di classi e oggetti. 7, 31, 48, 61

CSS Sigla di *Cascading Style Sheet*, ovvero foglio di stile a cascata, con CSS si indica un complesso d'istruzioni o regole che serve a determinare l'aspetto grafico di un documento generato con un linguaggio a marcatori come *HTML* o *XML*. Un foglio di stile permette di separare le istruzioni che riguardano caratteristiche come colori, caratteri, spazi e *layout* dal contenuto del documento. L'aspetto dei caratteri di un testo può essere regolato simultaneamente da due fogli differenti, uno creato dall'autore, l'altro dall'utente (per es. un ipovedente che desideri ingrandire il testo). La sovrapposizione è possibile perché le regole funzionano a cascata, secondo gerarchie che fanno prevalere una sull'altra. Viceversa, il medesimo foglio può formattare più documenti: agendo su di esso, si può definire in blocco il loro aspetto. I *browser* sono dotati di propri fogli "di default". Le specifiche *CSS* 1 sono state messe a punto per la prima volta nel 1996 dal *W3C* (*World Wide Web Consortium*); nel 1998 sono state emanate le *CSS* 2 e nel 2004 le *CSS* 2.1, con nuove funzionalità, un linguaggio ben supportato e la possibilità di creare fogli di stile separati per dispositivi portatili. Le specifiche *CSS* 3, oltre a presentare nuove funzionalità, correggono alcuni bug, ovvero errori di programma o di sistema, presenti nelle versioni precedenti. 30, 32, 61

DisplayLink *DisplayLink* è un driver, sviluppato da *Synaptics*, per l'utilizzo di device usb come monitor e docking stations. L'azienda offre i driver per Windows, MacOS, Android, ChromeOS e Ubuntu. 51

Electron *Electron* è un *framework open source* gestito e ospitato da *GitHub*. *Electron* consente lo sviluppo della *GUI* di applicazioni *desktop* utilizzando tecnologie Web: combina il motore di *rendering Chromium* e il *runtime Node.js*. Le applicazioni

Electron sono composte da più processi: il processo *"browser"* e diversi processi *"renderer"*. Il processo *browser* esegue la logica dell'applicazione e può quindi avviare più processi di *rendering*, restituendo le finestre che appaiono sullo schermo di un utente processando *HTML* e *CSS*. Entrambi i processi *browser* e *renderer* possono essere eseguiti con l'integrazione di *Node.js*. 7, 29, 36, 48

Electron Forge *Electron Forge* è un tool all-in-one per pacchettizzare e distribuire applicazioni in Electron. Unisce molti pacchetti single-purpose per creare una pipeline completa che funziona out of the box, completa con code signing, installers, e pubblicazione degli artefatti. 31, 51

HTML Sigla di *Hyper text markup language*, HTML è un linguaggio di *markup* utilizzato per la creazione di pagine web. Creato da *Timothy J. Berners-Lee* all'inizio degli anni Novanta del secolo scorso, si basa su file di testo in cui alcuni marcatori, chiamati *tags*, racchiusi tra simboli particolari, specificano proprietà e comportamenti di testi e immagini inseriti nella pagina. I tag HTML sono racchiusi tra parentesi angolari (< e >), scritti indifferentemente in lettere maiuscole o minuscole. Una pagina *HTML* è formata da una sezione di un'intestazione, ovvero il *tag head*, e una di corpo, ovvero il *tag body*, a loro volta racchiuse dai *tag* <*html*> e </*html*>: questi sono preceduti da una dichiarazione iniziale, definita con *DOCTYPE*, che indica al *browser* la versione *HTML* utilizzata. L'evoluzione del linguaggio, da *HTML* 1.0 a *HTML* 4.01, ha permesso d'inserire nella struttura delle pagine *web* elementi sempre più complessi, come le tabelle, alcune funzioni di accessibilità per utenti diversamente abili, l'incorporamento di oggetti e la gestione dei moduli (detti *form*, che rendono bidirezionale il flusso delle informazioni fra *provider* del servizio e utente). Un'evoluzione importante è l'*HTML5* che, forzando la separazione fra struttura della pagina, gli stili e i contenuti, consente una più agevole realizzazione di applicazioni web, un miglior controllo dei contenuti multimediali e una maggior integrazione con i dispositivi mobili (per es. con funzioni di geolocalizzazione). Inoltre, le funzioni di memorizzazione in locale dei contenuti web consentono l'utilizzo delle applicazioni *HTML* anche in presenza di una connessione Internet poco stabile o performante. 30, 32, 61

Integrated Development Environment Un *IDE*, Integrated Development Environment,, o ambiente di sviluppo integrato, è un software progettato per la realizzazione di applicazioni che aggrega strumenti di sviluppo comuni in un'unica interfaccia utente grafica. In genere è costituito da un editor di testo che agevola la scrittura di codice software grazie a utili funzionalità come l'evidenziazione della sintassi con suggerimenti visivi, il completamento automatico specifico del linguaggio e l'individuazione di bug durante la scrittura. Inoltre, generalmente offre anche l'automazione della build locale e un debugger per risolvere problemi (conosciuti come *bug*) nei programmi sviluppati. 4

Inter-Process Communication L'*IPC, Inter-Process Communication*, è un sistema di comunicazione tra processi offerto da *Electron*. La comunicazione avviene dal processo *main* al processo *render* attraverso appositi canali, dove si possono inviare anche dati semplici. Le comunicazioni possono essere in forma sincrona o asincrona. 29, 48

JavaScript *JavaScript* è un linguaggio di scripting orientato agli oggetti, interpretato e debolmente tipizzato, comunemente utilizzato nelle pagine *HTML* dei siti *web* per arricchirle di funzionalità e aspetti dinamici. Uno script *JavaScript* viene solitamente ospitato da un altro programma, nella maggior parte dei casi da un browser. I browser più comuni incorporano un interprete *JavaScript* e quando viene visitata una pagina *HTML* che contiene il codice di uno script *JavaScript*, quest'ultimo è eseguito dall'interprete contenuto nel browser stesso. Se ospitato da un browser, *JavaScript* offre la possibilità di manipolarne gli elementi mediante gli oggetti del *Document Object Model*, o *DOM*: a titolo esemplificativo, sono oggetti del *DOM* l'oggetto *window*, che rappresenta una finestra del *browser*, e l'oggetto *document*, che rappresenta la pagina *HTML* vera e propria. *JavaScript* offre inoltre la possibilità di catturare alcuni eventi che si possono verificare durante la navigazione di una pagina *HTML*, come il passaggio del *mouse* su un'immagine o la pressione del tasto sinistro del *mouse* su un'ancora, e abbinarli all'esecuzione di uno *script JavaScript*. 30–33, 61

LaTeX *LaTeX* è un sistema di composizione tipografica di alta qualità, che include funzionalità progettate per la produzione di documentazione tecnica e scientifica. LaTeX è lo standard de facto per la comunicazione e la pubblicazione di documenti scientifici. Inoltre, LaTeX è disponibile come software libero. 4

Node-API *Node-API* (conosciute anche come *N-API*) sono delle API per creare Addons nativi. Sono indipendenti dal runtime JavaScript e sono mantenute come parte di Node.js. Il loro intento è d'isolare gli addon dal runtime, permettendo agli stessi di girare per versioni successive di Node.js rispetto alla versione di compilazione, senza ricompilarli nuovamente. 31, 61

TypeScript *TypeScript* è un linguaggio di programmazione *open source* sviluppato da *Microsoft*. Si tratta di un'estensione di *JavaScript* che basa le sue caratteristiche su *ECMAScript 6*. Il linguaggio estende la sintassi di *JavaScript* in modo che qualunque programma scritto in *JavaScript* sia anche in grado di funzionare con *TypeScript* senza nessuna modifica. È stato progettato per lo sviluppo di grandi applicazioni e dev'essere compilato in *JavaScript* per poter essere interpretato da qualunque web browser o app. 7, 30, 50, 61

Unified Modeling Language In ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 10

WebAssembly *WebAssembly* è uno standard web che definisce un formato binario e un corrispondente formato testuale per la scrittura di codice eseguibile nelle pagine web. Ha lo scopo di abilitare l'esecuzione del codice quasi alla stessa velocità con cui esegue il codice macchina nativo. È stato progettato come integrazione di JavaScript per accelerare le prestazioni delle parti critiche delle applicazioni Web e in seguito per consentire lo sviluppo web in altri linguaggi

oltre a JavaScript. È sviluppato dal World Wide Web Consortium (W3C) con ingegneri provenienti da Mozilla, Microsoft, Google e Apple. Viene eseguito in una sandbox nel browser Web dopo una fase di verifica formale. I programmi possono essere compilati da linguaggi di alto livello in moduli Wasm e caricati come librerie dalle applet JavaScript. 33

Bibliografia

Siti web consultati

Ambiente di Sviluppo Integrato. URL: https://www.redhat.com/it/topics/middleware/what-is-ide.

AWS. URL: https://aws.amazon.com/it/what-is-aws/?nc1=f_cc.

C++. URL: https://www.w3schools.com/cpp/cpp_intro.asp.

CSS. URL: https://www.treccani.it/enciclopedia/css_%28Lessico-del-XXI-Secolo%29/.

DisplayLink. URL: https://www.synaptics.com/products/displaylink-graphics.

Electron Forge. URL: https://www.electronforge.io/.

ENSign 11 Signature Pad. URL: https://www.euronovategroup.com/solutions/product-map/ensign-11-ensign-nfc-hardware (cit. alle pp. 3, 51).

GMail. URL: https://www.google.com/intl/it/gmail/about/ (cit. a p. 3).

Google Meet. URL: https://meet.google.com/?pli=1 (cit. a p. 3).

HTML. URL: https://www.treccani.it/enciclopedia/html_%28Lessico-del-XXI-Secolo%29/.

Javascript. URL: https://www.treccani.it/enciclopedia/javascript_%28Lessico-del-XXI-Secolo%29/.

LaTeX. URL: https://www.latex-project.org/.

MacOS Ventura. URL: https://www.apple.com/it/macos/ventura/ (cit. a p. 51).

Market Share dei Sistemi operativi. URL: https://gs.statcounter.com/os-market-share (cit. a p. 4).

MiKTeX. URL: https://miktex.org/ (cit. a p. 4).

Node-API. URL: https://nodejs.org/api/n-api.html#node-api.

Telegram. URL: https://telegram.org/ (cit. a p. 4).

Trello. URL: https://trello.com/it (cit. a p. 4).

Verdaccio. URL: https://verdaccio.org/it-it/.

Visual Studio Code. URL: https://code.visualstudio.com/ (cit. a p. 4).

WebAssembly. URL: https://webassembly.org/.

Windows 11. URL: https://www.microsoft.com/it-it/windows/windows-11?r=1 (cit. alle pp. 4, 51).