Titolo del progetto:

PC-Assembler

Gruppo formato da:

- Feltrin Emanuele, numero di matricola: 2034314;
- Feltrin Alessandro, numero di matricola: 2042331.

Introduzione al software sviluppato:

Il programma da noi sviluppato è chiamato PC-Assembler. Esso è un software idealizzato per soddisfare il bisogno di utenti inesperti di PC e hardware: costruirsi un computer personale in base alle proprie necessità, casi d'uso e disponibilità economica. Il software è molto semplice da usare e richiede all'utente di inserire come input: il budget massimo (rappresenta i dollari disponibili al programma per creare una build, quindi il programma non potrà consigliare una build che costa di più del budget massimo) e la tipologia di computer che vuole assemblare in base ai propri casi d'uso. Il programma, tramite svariati algoritmi, seleziona l'hardware con il rapporto qualità-prezzo migliore.

Essenzialmente l'algoritmo per la scelta funziona in questo modo: un componente hardware è considerato *"migliore"* rispetto ad un altro, solamente se sia il prezzo che il rating risultino migliori oppure uguale all'altro componente. Con "migliori" si intende che il prezzo deve essere più vicino al prezzo che il particolare tipo di build vuole allocare a quel componente, mentre per il rating si intende che deve avere un valore più alto o uguale (il rating è un valore intero che va da 0 a 5).

Le diverse tipologie di computer possono essere implementate attraverso il polimorfismo, sfruttando le caratteristiche comuni e le differenze tra le varie classi dell'hardware. Si è voluto creare un software utile e necessario, che potesse facilitare la vita a chi volesse costruirsi un computer "su misura" senza dover spendere oltre al proprio budget o affidarsi a negozi specializzati di terze parti.

Il software è scritto in C++ e utilizza il framework Qt per creare una interfaccia utente basata su widget.

PC-Assembler è costruito e implementato su una serie di dati raccolti da vari siti web ottenuti con tecniche di *scraping*.

Il modello:

Il modello inizia con una classe astratta ***Pc*** che rappresenta informazioni comuni a tutti i tipi di computer e contiene tutti gli elementi hardware che compongono qualsiasi calcolatore (RAM, CPU, Motherboard, ecc.).

Questa classe ha alcuni metodi virtuali che sono la base per l'esecuzione del programma, questi metodi sono assemble() e allocateBudget():

allocateBudget() è un metodo virtuale che alloca il budget in base al tipo di macchina selezionato. Diversi tipi di Pc richiedono segmentazioni di budget molto diversi fra di loro.

assemble() è un metodo virtuale da implementare nella classi concrete, questo metodo popola il campo component con componenti adatti al tipo di build e al budget massimo previsto;

Il modello viene quindi suddiviso in due parti, la parte ***Full_Setup*** e la parte ***Server***.

Server deriva da ***Pc*** e aggiunge un nuovo componente hardware, in seguito, altri due componenti concreti sono derivati da questa classe astratta, chiamati ***Standard*** e ***Artificial_Intelligence***.

Artificial_Intelligence richiede due schede video come minimo, con una certa quantità minima di memoria VRAM ciascuna, mentre ***Standard*** ha un vincolo e richiede un numero minimo di dischi rigidi.

Full_Setup è una classe astratta derivata da ***Pc***, che rappresenta un calcolatore dotato di tutte le periferiche che devono essere utilizzate. ***Full_Setup*** è in derivato virtualmente nella classe "***Gaming***" e la classe "***Streaming***", per creare la classe ***Gaming & Streaming*** (in ereditarietà multipla a diamante).

La classe ***Streaming*** deve aggiungere una webcam con messa a fuoco automatica per consentire agli utenti di trasmettere in live streaming (su piattaforme come Twitch e Youtube) il loro viso.

La classe ***Gaming*** rappresenta i PC che verranno utilizzati per giocare a vari giochi, il che richiede lo spostamento del budget verso componenti come la scheda video.

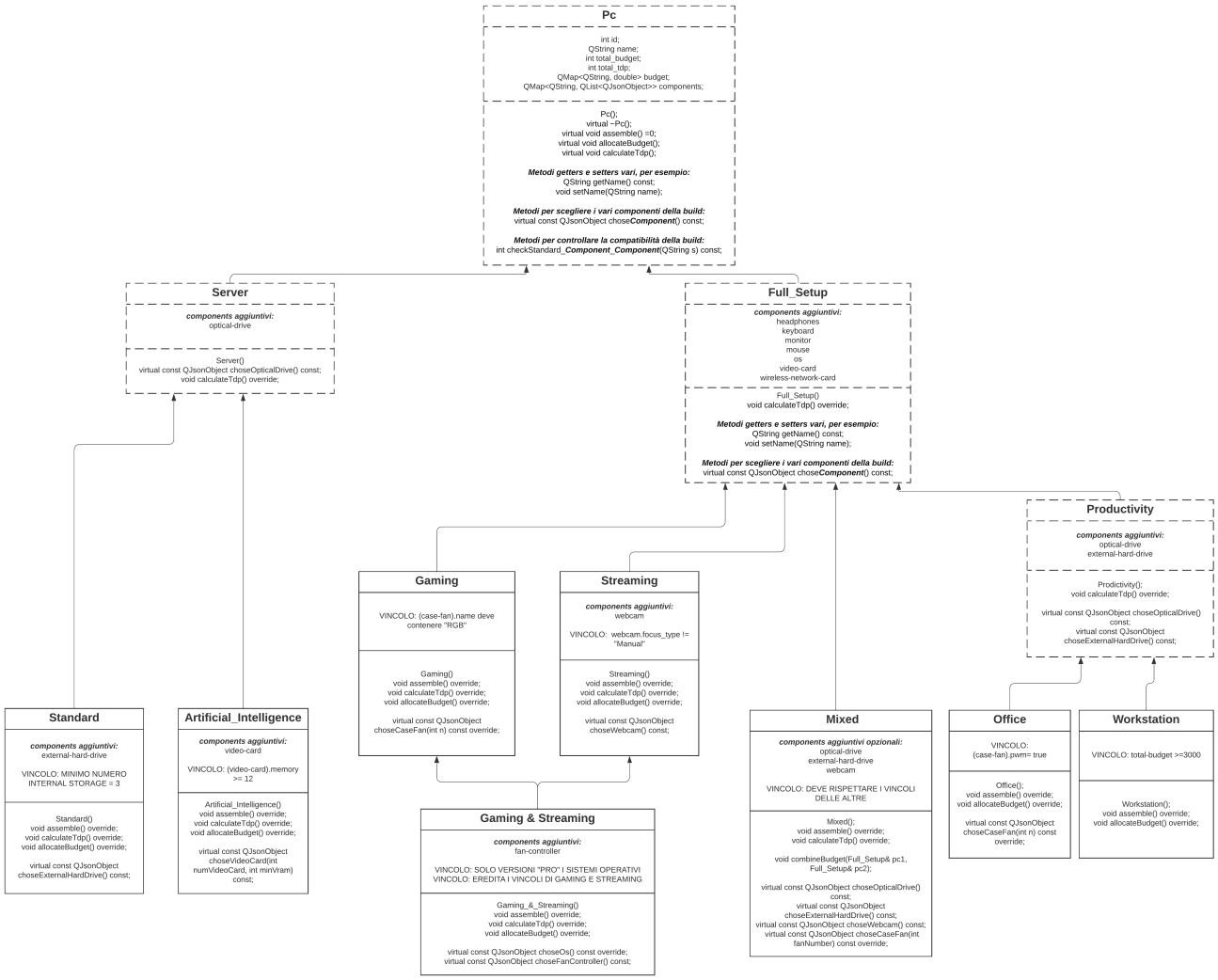
La classe ***Gaming & Streaming*** riunisce i concetti delle classi "***Gaming***" e "***Streaming***", creando computer adatti a entrambi i casi d'uso.

La classe ***Mixed*** eredita da ***Full_Setup*** e permette all'utente di creare un PC con le caratteristiche di due classi diverse (ad esempio può essere utilizzata per costruire un ibrido tra un computer da "***Streaming***" e una "***Workstation***").

Il modello è completato da ***Productivity***, con le sottoclassi ***Office*** e ***Workstation***.

Productivity è una classe astratta che aggiunge componenti tipicamente utili sul posto di lavoro.

Workstation garantisce una maggiore potenza di calcolo per carichi di lavoro più importanti. ***Office*** aggiunge vincoli alle ventole del case.



Polimorfismo non banale

L'uso del polimorfismo in questo progetto è fondamentale per il funzionamento della gerarchia Pc. Esistono diversi metodi virtuali per far funzionare correttamente la gerarchia: “chooseComponent()”, “calculateTdp()”, “allocateBudget()” e “assemble()”.

Spiegazione dei metodi:

- I vari metodi chooseComponent() permettono di individuare i componenti adatti alle caratteristiche della build in questione.
- calculateTdp() calcola il wattaggio del nostro computer, permettendoci di scegliere con precisione il nostro alimentatore.
- allocateBudget() alloca un budget in base al tipo di computer che è stato selezionato.
- assemble() chiama i metodi precedenti per "assemblare" la nostra build e popolare il campo component dell'oggetto pc.

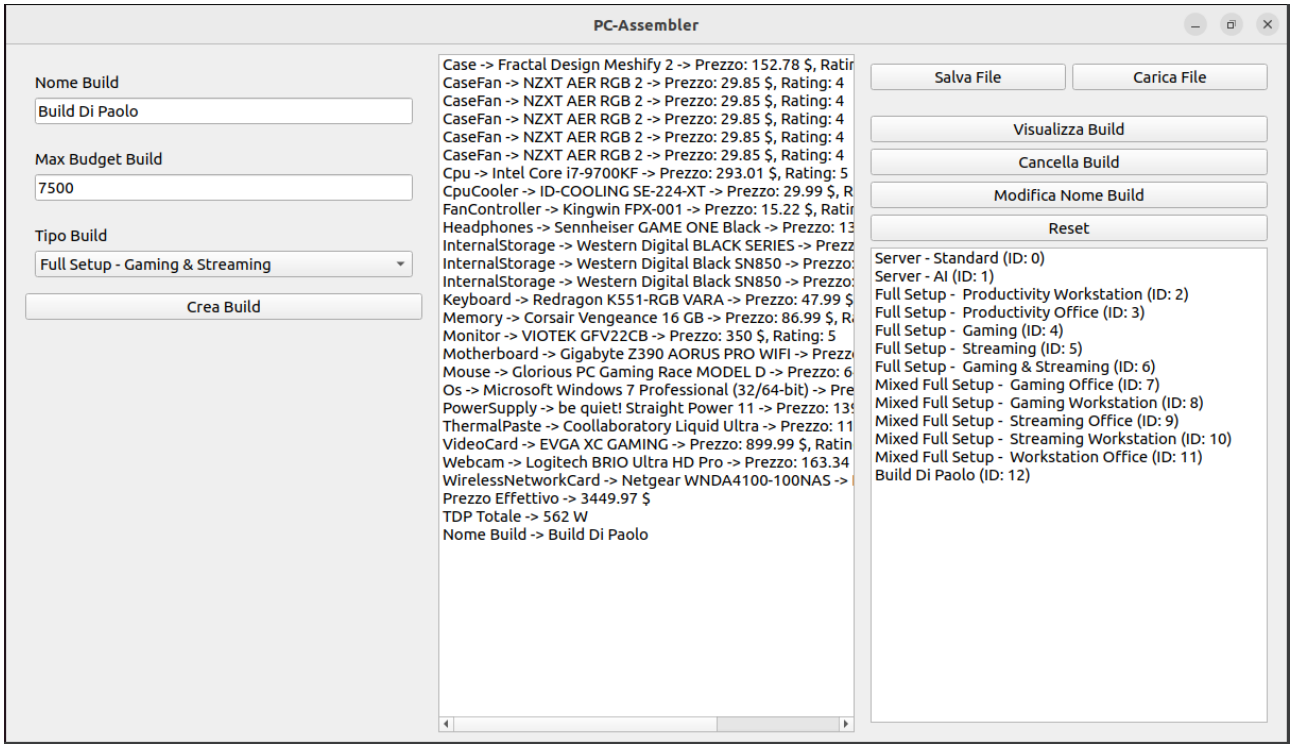
Persistenza dati

Per la persistenza dei dati viene utilizzato il formato JSON, i file JSON contengono una lista di oggetti di tipo pc.

Un esempio di file JSON viene fornito assieme al codice e contiene una lista di ognuna delle build che il programma supporta.

Per essere identificati viene aggiunto ad ogni pc un campo identificatore chiamato “type” che specifica il tipo.

Funzionalità implementate al software



La *GUI* è stata suddivisa in tre colonne:

1. La prima colonna permette all'utente di dare un nome alla build, fornire un budget e scegliere, attraverso una *ComboBox*, il tipo della build che si vuole creare.
2. La seconda colonna è composta unicamente da una lista che verrà popolata dalle componenti della build creata.
3. La terza colonna presenta diverse funzionalità:
   - salvare e caricare file JSON contenenti una o più build;
   - visualizzare una build (bisogna selezionare una build specifica nella lista presente nella seconda parte della colonna);
   - cancellare una build selezionata;
   - modificare il nome di una build;
   - eliminare tutte le build salvate nella lista tramite il pulsante reset.

Spartizione delle attività

La parte della progettazione è stata svolta insieme a tutte le persone appartenenti al gruppo, durante questa fase è stato delineato il modello del progetto.

In seguito alla realizzazione e all'implementazione della classe *Pc,* c'è stata la suddivisione degli incarichi:

- Feltrin Alessandro:
  1. Sviluppo del codice per la classe *Server*;
  2. Sviluppo del codice per la classe *Standard;*
  3. Sviluppo del codice per la classe *Artificial_Intelligence*;
  4. Idealizzazione e sviluppo della GUI del software.

- Feltrin Emanuele:
  1. Sviluppo di tutto il codice restante del modello;
  2. Sviluppo del codice e progettazione per il salvataggio e caricamento dati JSON;
  3. Sviluppo del codice della classe "linked_list".

Il testing e debugging del programma è stato fatto in collaborazione con tutti i membri del gruppo.

Rendicontazione delle ore

Attività	Ore Previste	Ore Effettive
Studio e progettazione	12	13
Sviluppo del codice del modello	15	16
Sviluppo del codice della GUI	12	11
Test e debug	5	6
Stesura della relazione	6	6
totale	**50**	**52**