. Matteo Calvanico

+39-347-385-2782 matteo.calvanico03@gmail.com Matteo Calvanico MatteoCalvanico

Competenze

- **Linguaggi**: Java, C# (.NET 8.0), C++, C89, Python, Kotlin, Javascript, Typescript, SQL, GDScript, LaTex.
- **Framework & Librerie**: Vue, Bootstrap, TensorFlow, Scikit Learn, PyTorch, CUDA, OpenCV, Pandas, Numpy, Gradle.
- **Tools**: Visual Studio, VS Code, Andorid Studio, Arduino, Google Cloud Console, Git, GitHub, Godot, Docker, Jenkins, Vagrant, Dockerhub.

Formazione

- **"Alma Mater Studiorum" - Università di Bologna** Cesena (FC), Italia
  Laurea triennale in Tecnologie dei Sistemi Informatici" *2022 – Ongoing*
  - **Competenze**: Object-Oriented Programming (Java, C#), Programmazione di sistemi mobili (Kotlin, Android Studio, Room, Gradle, Retrofit), Basi di dati (SQL, MySQL), Ingegneria dei sistemi web (JavaScript, TypeScript, CSS, SCSS/SASS, NodeJS, Vue.js, Bootstrap), Basi di Reti Neurali, Machine Learning, Elaborazione di immagini (OpenCV, CUDA, Python, PyTorch, TensorFlow, Scikit Learn, Pandas), Sistemi Embedded e Internet of Things (Arduino, ESP32, Raspberry Pi, MQTT), Piattaforme di sviluppo per automazione e virtualizzazione (Docker, Vagrant, Kubernetes, Active Directory), Sistemi di rete, Sviluppo in Asp.NET
- **I.T.T. "Blaise pascale"** Cesena (FC), Italia
  Diploma da perito informatico *Settembre 2017 – Giugno 2022*

Esperienze lavorative

- **Maggioli S.P.A.** Santarcangelo di Romagna (FC), Italia
  Intern Software Engineer *Novembre 2024 – Current*
  - **Tecnologie usate**: .
  - **Descrizione**: Azienda italiana specializzata in sviluppo di applicazioni per la pubblica amministrazione.

Progetti

- **Game Vault**: Android app per il tracking della propria collezione di giochi, con la possibilità di raiting e ricerca.
  - **Play Store:**: l'applicazione è stata testata da 20 persone e approvata da Google, ora è disponibile per il download direttamente dal Google Play Store
  - **API:**: la ricerca viene fatta tramite api pubbliche, che forniscono tutte le informazioni necessarie sul gioco per poi essere mostrate all'utente; i file Json ritornati venivano gestiti tramite Moshi e Retrofit.
  - **DB:**: per salvare le informazioni in maniera persistente si è utilizzato Room.
- **GetEat**: Web app per la gestione di un ristorante, con la possibilità di acquistare prodotti per gli utenti e di gestirli da parte dell'admin.
  - **Multi-tecnologia**: la web app è stata creata tramite il framework Vue.js, utilizzando typeScript e Scss con l'utilizzo di Express.js per gestire le chiamate al DB relazionale creato tramite SQL.
  - **Modularità**: : l'applicazione è stata divisa prima in Front-end/Back-end per poi passare ad una suddivisione ancora maggiore in file modulari all'interno di ciascuna parte.
  - **Autenticazione**: l'applicazione prevede una parte di registrazione e login con controllo dei vari permessi, andando così a nascondere agli utenti base la parte dell'amministratore
  - **Accessibilità**: l'obiettivo non era solo quello di avere una applicazione funzionante ma di avere anche una web app responsive e soprattuto accessibile.
- **Virtual Casinò**: Applicativo Desktop in Java che simuli alcuni giochi di un casinò, come Blackjack, Roulette e i dadi.
  - **MVC**: per rendere il progetto meno complesso e favorire una facile modifica e testabilità si è deciso di usare il Pattern MVC, andando a dividere la parte di presentazione da quella dei dati garantendo la comunicazione tra i due tramite il controller

- **Collaborazione**: il progetto è sviluppato insieme ad altri due colleghi, questo ci ha permesso di avere un primo assaggio sul lavoro in team, portando a suddividere il lavoro in maniera equa, sentirci e discutere sul da farsi in maniera regolare tramite call virtuali e aiutarci in caso di difficoltà.
- **Test-Driven Development**: una parte cruciale del progetto era quella di effettuare test periodici e automatizzati per far si che ogni parte funzioni correttamente, questo è stato possibile grazie a JUnit.
- **Gradle**: per automatizzare la compilazione e l'esecuzione di test abbiamo utilizzato Gradle, che ci è stato molto utile anche per la gestione delle dipendenze.

- **MultiClass Classification**: Rete neurale per la creazione e test di un modello di machine learning per la classificazione multiclasse di diversi tipi di frutta. La rete è una classica CNN con una funzione di attivazione non lineare (ReLU) con anche un ottimizzatore di tipo SGD.
  - **CONDA:**: Anaconda è stato fondamentale per creare un enviroment con tutto il necessario per lo sviluppo e la prova della rete.
  - **Multi-libreria**: Per creare, addestrare e testare la rete è stato necessario l'utilizzo di diverse librerie come Pytorch o OpenCV che ha permesso di modificare le immagini del dataset per aggiustare certi aspetti.
  - **Kaggle**: per trovare un dataset che fosse perfetto per le mie esigenze ho effettuato diverse ricerche per poi finire su Kaggle, che mi ha permesso anche di utilizzare le sue API per il dowload delle immagini.

- **DoomClone**: Videogioco sviluppato in C++ simile ai vecchi classici come Doom o Wolfenstein 3D.
  - **Basi di computer grafica:**: per creare un effetto 3D tramite una mappa 2D mi sono cimentato nella tecnica del Raycasting, dove si utilizzano diverse formule matematiche per mostrare a schermo il contenuto di un vettore (che contiene la mappa), questo grazie a diversi "raggi che quando colpiscono un punto dove dovrebbe stare un muro o un nemico ne permetteno di ottenere la distanza e successivamente disegnarlo; questo viene fatto per ogni linea verticale dello schermo."
  - **SDL2:**: per mostrare a schermo il render e gestire i vari eventi è stato necessario usare la libreria SDL2.
  - **Event Handle:**: Per gestire diverse casistiche come l'eliminazione dei nemici, il movimento o l'apertura della porta è stato necessario un handle degli eventi nel loop principale del gioco, sempre tramite SDL2.