. Matteo Calvanico

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

Skills

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

Education

- **"Alma Mater Studiorum" - University of Bologna** — Cesena (FC), Italy
  Bachelor's Degree in Computer System Technologies — *2022 – Ongoing*
  - **Skills**: Object-Oriented Programming (Java, C#), Mobile Systems Programming (Kotlin, Android Studio, Room, Gradle, Retrofit), Databases (SQL, MySQL), Web Systems Engineering (JavaScript, TypeScript, CSS, SCSS/SASS, NodeJS, Vue.js, Bootstrap), Basics of Neural Networks, Machine Learning, Image Processing (OpenCV, CUDA, Python, PyTorch, TensorFlow, Scikit Learn, Pandas), Embedded Systems and Internet of Things (Arduino, ESP32, Raspberry Pi, MQTT), Development Platforms for Automation and Virtualization (Docker, Vagrant, Kubernetes, Active Directory), Network Systems, Development in Asp.NET
- **I.T.T. "Blaise Pascal"** — Cesena (FC), Italy
  Diploma in Computer Science — *September 2017 – June 2022*

Work Experience

- **Maggioli S.P.A.** — Santarcangelo di Romagna (FC), Italy
  Intern Software Engineer — *November 2024 – Current*
  - **Technologies used**: .
  - **Description**: Italian company specialized in developing applications for public administration.

Projects

- **Game Vault**: Android app for tracking your game collection, with rating and search capabilities.
  - **Play Store:**: the application was tested by 20 people and approved by Google, now available for download directly from the Google Play Store
  - **API:**: the search is done through public APIs, which provide all the necessary information about the game to be displayed to the user; the returned JSON files were managed using Moshi and Retrofit.
  - **DB:**: to save information persistently, Room was used.
- **GetEat**: Web app for restaurant management, with the ability for users to purchase products and for the admin to manage them.
  - **Multi-technology**: the web app was created using the Vue.js framework, utilizing TypeScript and Scss with the use of Express.js to handle calls to the relational DB created using SQL.
  - **Modularity**: : the application was divided first into Front-end/Back-end and then further subdivided into modular files within each part.
  - **Authentication**: the application includes a registration and login part with permission control, thus hiding the admin part from basic users
  - **Accessibility**: the goal was not only to have a functional application but also a responsive and accessible web app.
- **Virtual Casino**: Desktop application in Java simulating some casino games, such as Blackjack, Roulette, and dice.
  - **MVC**: to make the project less complex and facilitate easy modification and testability, the MVC Pattern was used, separating the presentation part from the data part, ensuring communication between the two through the controller

- **Collaboration**: the project was developed together with two other colleagues, giving us a first taste of teamwork, leading to an equitable division of work, regular virtual calls to discuss the tasks, and helping each other in case of difficulties.
- **Test-Driven Development**: a crucial part of the project was to perform periodic and automated tests to ensure that each part works correctly, made possible by JUnit.
- **Gradle**: to automate the compilation and execution of tests, we used Gradle, which was also very useful for dependency management.

- **MultiClass Classification**: Neural network for creating and testing a machine learning model for multi-class classification of different types of fruit. The network is a classic CNN with a non-linear activation function (ReLU) and an SGD optimizer.
  - **CONDA:**: Anaconda was essential for creating an environment with everything needed for the development and testing of the network.
  - **Multi-library**: To create, train, and test the network, several libraries such as Pytorch and OpenCV were necessary, which allowed modifying the dataset images to adjust certain aspects.
  - **Kaggle**: to find a dataset that was perfect for my needs, I conducted several searches and ended up on Kaggle, which also allowed me to use its API for downloading images.

- **DoomClone**: Video game developed in C++ similar to old classics like Doom or Wolfenstein 3D.
  - **Basics of computer graphics:**: to create a 3D effect through a 2D map, I ventured into the technique of Raycasting, where various mathematical formulas are used to display the content of a vector (which contains the map) on the screen; this is done for each vertical line of the screen.
  - **SDL2:**: to display the render on the screen and handle various events, the SDL2 library was necessary.
  - **Event Handle:**: To handle various cases such as enemy elimination, movement, or door opening, an event handler was necessary in the main game loop, also using SDL2.