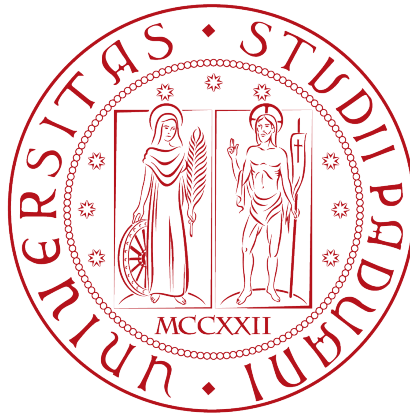


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Integrazione di Amazon Rekognition e Lex  
all'interno di una web app basata su AWS  
Lambda

*Tesi di laurea triennale*

*Relatore*

Prof. Lamberto Ballan

*Laureanda*

Nicla Faccioli

---

ANNO ACCADEMICO 2021-2022



# Sommario

Il presente documento descrive l'attività di stage svolta presso l'azienda Zero12 s.r.l. Lo stage è stato svolto alla conclusione del percorso di studi della laurea triennale in Informatica ed ha avuto la durata di circa trecento ore. L'obiettivo dello stage è stato l'integrazione dei servizi AWS di image recognition (Amazon Rekognition), automatic speech recognition e natural language understanding (Amazon Lex) all'interno di un'applicazione web serverless basata su AWS Lambda con lo scopo di facilitare l'inserimento dei dati.



*“Let us light up the night, we shine in our own ways.  
Shine, dream, smile ”*

— Bangtan

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. NomeDelProfessore, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.*

*Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.*

*Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.*

*Padova, Luglio 2022*

Nicla Faccioli



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	L'azienda . . . . .	1
1.2	L'idea . . . . .	1
1.3	Struttura del documento . . . . .	2
<b>2</b>	<b>Descrizione dello stage</b>	<b>3</b>
2.1	Introduzione al progetto . . . . .	3
2.2	Obiettivi formativi . . . . .	3
2.3	Requisiti . . . . .	3
2.3.1	Requisiti obbligatori . . . . .	4
2.3.2	Requisiti desiderabili . . . . .	4
2.3.3	Requisiti facoltativi . . . . .	4
2.4	Pianificazione . . . . .	4
<b>3</b>	<b>Tecnologie e strumenti</b>	<b>7</b>
3.1	Tecnologie per il back-end . . . . .	7
3.1.1	Serverless Framework . . . . .	7
3.1.2	Node.js . . . . .	7
3.1.3	AWS Lambda . . . . .	8
3.1.4	Amazon API Gateway . . . . .	8
3.1.5	Amazon DynamoDB . . . . .	8
3.1.6	Amazon S3 . . . . .	9
3.1.7	Amazon Rekognition . . . . .	9
3.1.8	Amazon Lex . . . . .	9
3.2	Tecnologie per il front-end . . . . .	10
3.2.1	TypeScript . . . . .	10
3.2.2	Angular . . . . .	10
3.2.3	Nebular . . . . .	10
3.3	Strumenti di supporto a progettazione e codifica . . . . .	11
3.3.1	Git . . . . .	11
3.3.2	AWS CodeCommit . . . . .	11
3.3.3	VisualStudio Code . . . . .	11
3.3.4	Balsamiq Wireframes . . . . .	12
<b>4</b>	<b>Descrizione dell'applicativo esistente</b>	<b>13</b>
4.1	Architettura serverless . . . . .	13
4.1.1	Definizione di una tabella DynamoDB . . . . .	13
4.1.2	Definizione di una funzione Lambda . . . . .	13

4.1.3	Definizione di un bucket S3 . . . . .	13
4.1.4	Deploy del back-end . . . . .	13
4.2	Web-App . . . . .	13
4.2.1	Funzionalità disponibili . . . . .	13
4.2.2	Deploy del front-end . . . . .	13
<b>5</b>	<b>Integrazione di Amazon Rekognition</b>	<b>15</b>
5.1	Presentazione del problema . . . . .	15
5.2	Progettazione . . . . .	15
5.2.1	Architettura . . . . .	15
5.2.2	Funzionamento generale . . . . .	16
5.2.3	Design dell'interfaccia . . . . .	17
<b>6</b>	<b>Integrazione di Amazon Lex</b>	<b>21</b>
6.1	Presentazione del problema . . . . .	21
6.2	Progettazione . . . . .	21
6.2.1	Architettura . . . . .	21
6.2.2	Design dell'interfaccia . . . . .	21
<b>7</b>	<b>Conclusioni</b>	<b>23</b>
7.1	Consuntivo finale . . . . .	23
7.2	Raggiungimento degli obiettivi . . . . .	24
7.3	Conoscenze acquisite . . . . .	24
7.4	Valutazione personale . . . . .	24
	<b>Glossary</b>	<b>25</b>
	<b>Bibliografia</b>	<b>27</b>



# Elenco delle figure

1.1	Logo di Zero12 s.r.l. . . . .	1
3.1	Logo Serverless Framework . . . . .	7
3.2	Logo Node.js . . . . .	7
3.3	Logo AWS Lambda . . . . .	8
3.4	Logo Amazon API Gateway . . . . .	8
3.5	Logo Amazon DynamoDB . . . . .	8
3.6	Logo Amazon S3 . . . . .	9
3.7	Logo Amazon Rekognition . . . . .	9
3.8	Logo Amazon Lex . . . . .	10
3.9	Logo TypeScript . . . . .	10
3.10	Logo Angular . . . . .	10
3.11	Logo Nebular . . . . .	10
3.12	Logo Git . . . . .	11
3.13	Logo AWS CodeCommit . . . . .	11
3.14	Logo Visual Studio Code . . . . .	11
3.15	Logo Balsamiq . . . . .	12
5.1	Schermata iniziale di calcetto . . . . .	17
5.2	Selezione dei giocatori . . . . .	18
5.3	Formazione squadre di calcetto . . . . .	18
5.4	Schermata iniziale di Duck Game . . . . .	19
5.5	Schermata iniziale di Mario Kart . . . . .	19
5.6	Selezione dei giocatori di Mario Kart . . . . .	20

# Elenco delle tabelle

2.1	Pianificazione delle attività . . . . .	5
7.1	Pianificazione delle attività . . . . .	24

# Capitolo 1

## Introduzione

### 1.1 L'azienda

Zero12 s.r.l. è un'azienda informatica nata nel 2012 specializzata nello sviluppo di soluzioni cloud native, sempre in prima linea nel seguire l'evoluzione di questo paradigma tecnologico.

L'azienda è partner [AWS](#) e si occupa di progettazione e sviluppo software Web e Mobile per clienti provenienti da ambiti molto diversificati.

L'obiettivo di Zero12 s.r.l. è aiutare i propri clienti a definire percorsi di innovazione includendo le tecnologie più avanzate tra cui per esempio il cloud ed il machine learning per l'analisi di linguaggio naturale, immagini, video e per fare previsioni.



**Figura 1.1:** Logo di Zero12 s.r.l.

### 1.2 L'idea

Attualmente in azienda è presente una piattaforma denominata MariBa con lo scopo di registrare i risultati di gioco del personale a Mario Kart e calcetto balilla. L'inserimento di tali dati però è completamente manuale: ogni partita va inizializzata con l'inserimento dei nickname di tutti i giocatori e, una volta conclusa, i risultati devono essere inseriti manualmente all'interno della piattaforma. L'idea dello stage è di semplificare l'inserimento di questi dati attraverso l'utilizzo di tecnologie [AWS](#) per il riconoscimento automatico dei giocatori e per la registrazione dei risultati comunicandoli vocalmente alla piattaforma.

### 1.3 Struttura del documento

**Il secondo capitolo** descrive il progetto di stage e la pianificazione delle attività;

**Il terzo capitolo** definisce le tecnologie utilizzate durante lo stage;

**Il quinto capitolo** approfondisce lo sviluppo del sistema di image recognition;

**Il quinto capitolo** approfondisce lo sviluppo del sistema di image recognition;

**Il sesto capitolo** approfondisce lo sviluppo del sistema di voice service;

**Nel settimo capitolo** sono descritte le conclusioni dell'esperienza di stage e gli obiettivi raggiunti.

## Capitolo 2

# Descrizione dello stage

### 2.1 Introduzione al progetto

In Zero12 s.r.l. è stata creata una piattaforma denominata MariBa con lo scopo di registrare i risultati di gioco del personale a Mario Kart e calcetto balilla. Tale piattaforma è dotata di un sistema di intelligenza artificiale che, in base ai giocatori (o alle coppie nel caso del calcetto), è in grado di predire il risultato del match di gioco. Il limite della piattaforma attuale è che tutti i dati, dall'inizializzazione di una partita ai risultati finali, devono essere inseriti manualmente.

Al fine di rendere più immediato l'inserimento dei dati si vuole evolvere la piattaforma includendo le seguenti funzionalità:

- \* Sistema di [Image Recognition](#) per riconoscere i giocatori e ruoli durante la fase di inizializzazione della partita e formazione delle squadre;
- \* Servizio vocale per l'inserimento dei risultati dei match giocati.

### 2.2 Obiettivi formativi

Gli obiettivi formativi dell'attività di stage sono i seguenti:

- \* Apprendere come sviluppare un applicativo web con controlli vocali;
- \* Apprendere come svolgere attività di integrazione con servizi di Machine Learning in ambito [Image Recognition](#), [Automatic Speech Recognition](#) (ASR) e [Natural Language Understanding](#) (NLU);

### 2.3 Requisiti

Nel primo giorno di stage si è svolto un incontro con il tutor aziendale per definire in modo dettagliato i requisiti. Nel corso dello stage il livello di obbligatorietà di tali requisiti è variato in risposta alle esigenze dell'azienda. Di seguito viene riportata la versione finale dell'analisi effettuata.

### 2.3.1 Requisiti obbligatori

Di seguito vengono elencati i requisiti obbligatori:

- \* Sviluppo di un micro-servizio per le attività di *face detection* e *recognition*;
- \* Sviluppo di un micro-servizio per le attività di controllo vocale via web

### 2.3.2 Requisiti desiderabili

Di seguito vengono elencati i requisiti desiderabili:

- \* Integrazione di un nuovo gioco all'interno della piattaforma;
- \* Implementazione di una versione semplificata per l'inserimento dei dati di Mario Kart per l'utilizzo della piattaforma in occasione del Summit [AWS](#) di Milano.

### 2.3.3 Requisiti facoltativi

Di seguito vengono elencati i requisiti facoltativi:

- \* Sviluppo di una [skill](#) Alexa con le stesse funzionalità del [chatbot](#) vocale richiesto come requisito obbligatorio e integrato sulla piattaforma web.

## 2.4 Pianificazione

La durata complessiva dello stage è stata di 8 settimane di lavoro a tempo pieno per un totale di circa 320 ore.

Secondo il piano di lavoro iniziale definito con l'azienda, le attività sono distribuite come segue:

Durata in ore	Settimana	Descrizione
40	1	* Studio delle tecnologie necessarie.
80	2, 3	* Progettazione e sviluppo di un micro-servizio per attività di <i>face detection</i> per la creazione di squadre di gioco; * Integrazione con la piattaforma esistente.
80	4, 5	* Progettazione e sviluppo di un micro-servizio per il controllo vocale; * Integrazione con la piattaforma esistente.

80	6, 7	<ul style="list-style-type: none"><li>* Sviluppo della skill Alexa per il controllo vocale e l'aggiornamento dei risultati;</li><li>* Integrazione con la piattaforma esistente.</li></ul>
40	8	<ul style="list-style-type: none"><li>* Testing e stesura della documentazione di progetto delle attività di sviluppo condotte nelle settimane precedenti.</li></ul>
<b>Totale ore:</b>		<b>320</b>

**Tabella 2.1:** Pianificazione delle attività

Durante il periodo di stage in azienda il piano di lavoro ha subito modifiche e di conseguenza la versione finale della pianificazione riportata nel [Capitolo 7](#) diverge da quella qui presentata. Tali modifiche sono state effettuate in risposta alle esigenze e richieste dell'azienda. Durante tutta la durata del tirocinio sono stati effettuati stand-up giornalieri con il tutor aziendale in affiancamento per monitorare lo stato di avanzamento ed evidenziare eventuali problemi sorti.





## Capitolo 3

# Tecnologie e strumenti

In questo capitolo vengono presentate le tecnologie utilizzate durante lo stage.

### 3.1 Tecnologie per il back-end

#### 3.1.1 Serverless Framework

*Serverless Framework* è un [Framework](#) web che permette di costruire applicazioni [Serverless](#) basate sul concetto [FaaS](#). Esso permette di definire funzioni Lambda e infrastrutture [AWS](#) utilizzando sintassi [YAML](#). Per effettuare il [deploy](#) delle Lambda, delle tabelle DynamoDB e dei bucket S3 definiti sarà sufficiente eseguire il comando `serverless deploy`.



**Figura 3.1:** Logo Serverless Framework

#### 3.1.2 Node.js

Node.js è un ambiente runtime open source per l'esecuzione di codice JavaScript all'esterno di browser web. Esso consente infatti di utilizzare JavaScript come linguaggio di programmazione lato server. All'interno del progetto viene utilizzata la versione 12.x per compatibilità con il codice già presente.



**Figura 3.2:** Logo Node.js

### 3.1.3 AWS Lambda

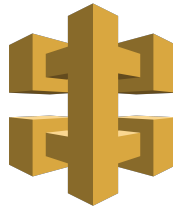
AWS Lambda è un servizio di calcolo serverless che permette l'esecuzione di codice per qualsiasi tipo di applicazione o servizio back-end senza bisogno di gestire un'infrastruttura server. Lambda gestisce le risorse di elaborazione scalando automaticamente in risposta alla potenza di calcolo richiesta. Il linguaggio utilizzato per lo sviluppo di funzioni Lambda è *Node.js v12.x*.



**Figura 3.3:** Logo AWS Lambda

### 3.1.4 Amazon API Gateway

API Gateway è un servizio Amazon che consente di creare API RESTful per permettere una comunicazione bidirezionale in tempo reale tra applicazioni e servizi di back-end. Le [Application Program Interface](#) definite nell'applicazione sviluppata sono state integrate alle rispettive funzioni Lambda.



**Figura 3.4:** Logo Amazon API Gateway

### 3.1.5 Amazon DynamoDB

DynamoDB è un database [NoSQL](#), serverless, completamente gestito che supporta l'inserimento di dati di tipo documento o di tipo chiave-valore. Facendo parte della famiglia di servizi messi a disposizione da Amazon, DynamoDB si integra senza difficoltà con tutti i servizi [AWS](#) e Amazon.

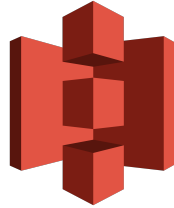


**Figura 3.5:** Logo Amazon DynamoDB

### 3.1.6 Amazon S3

*Amazon Simple Storage Service (S3)* è un servizio di archiviazione oggetti scalabile, sicuro e con ottime prestazioni. Al suo interno i dati sono organizzati in *bucket*. All'interno di ogni *bucket* è possibile definire dei prefissi per poter organizzare al meglio gli oggetti caricati.

All'interno del progetto questo servizio è stato utilizzato per effettuare l'hosting della Web App e per il trasferimento indiretto di immagini e audio tra front-end e back-end.



**Figura 3.6:** Logo Amazon S3

### 3.1.7 Amazon Rekognition

Amazon Rekognition è un software *cloud-based* che mette a disposizione capacità di visione artificiale pre-addestrate e personalizzabili per estrarre informazioni dettagliate da immagini e video. Alcuni esempi di utilizzo sono la moderazione di contenuti e *sentiment analysis*.

All'interno del progetto è stato utilizzato per implementare la ricerca facciale per il riconoscimento dei giocatori in fase di inizializzazione di una partita.



**Figura 3.7:** Logo Amazon Rekognition

### 3.1.8 Amazon Lex

Amazon Lex è un servizio di intelligenza artificiale completamente gestito che mette a disposizione modelli avanzati di linguaggio naturale per permettere lo sviluppo di interfacce di comunicazione all'interno di applicazioni software. Nel progetto è stato utilizzato per implementare un chatbot vocale per interagire con MariBa e registrare i risultati delle partite giocate.



**Figura 3.8:** Logo Amazon Lex

## 3.2 Tecnologie per il front-end

### 3.2.1 TypeScript

TypeScript è un linguaggio di programmazione sviluppato e mantenuto da Microsoft. Esso è un estensione del linguaggio di programmazione JavaScript: utilizza la stessa sintassi ma con l'aggiunta del supporto alla tipizzazione e alle interfacce.



**Figura 3.9:** Logo TypeScript

### 3.2.2 Angular

Angular è un framework open-source sviluppato da Google. Esso permette lo sviluppo di applicazioni web organizzate in componenti attraverso l'utilizzo di TypeScript, [HTML](#) e [CSS](#).



**Figura 3.10:** Logo Angular

### 3.2.3 Nebular

Nebular è una libreria di Angular gratuita e open-source per la creazione di interfacce utente.



**Figura 3.11:** Logo Nebular

## 3.3 Strumenti di supporto a progettazione e codifica

### 3.3.1 Git

*Git* è un sistema di [controllo di versione](#) distribuito. *Git* permette di tenere traccia di tutte le modifiche avvenute all'interno di un progetto o di un singolo file associando a ciascuna di esse il relativo autore. Permette inoltre di tornare ad una versione precedente del software eliminando le modifiche effettuate successivamente allo stato desiderato. Tutto ciò rende più semplice la collaborazione tra sviluppatori nella stesura del codice durante la fase di sviluppo software.



Figura 3.12: Logo Git

### 3.3.2 AWS CodeCommit

CodeCommit è un servizio gestito altamente scalabile e sicuro che consente l'hosting di [repository Git](#) privati. Esso custodisce i [repository](#) nel cloud [AWS](#) e supporta tutti i comandi *Git*. Si è scelto di utilizzare CodeCommit rispetto ad altri servizi equivalenti per compatibilità con la politica aziendale.



Figura 3.13: Logo AWS CodeCommit

### 3.3.3 VisualStudio Code

Visual Studio Code (VS Code) è un editor per il codice sorgente sviluppato da Microsoft. Esso permette il controllo per *Git* integrato e mette a disposizione numerose estensioni per facilitare la stesura del codice. Un esempio è *Prettier*, estensione che automatizza la formattazione del codice in modo da mantenerlo ordinato e con uno stile consistente.



Figura 3.14: Logo Visual Studio Code

### 3.3.4 Balsamiq Wireframes

*Balsamiq Wireframes* è uno strumento grafico per la creazione di schizzi per interfacce utente e schermate (*wireframes*) di siti web e applicazioni. Durante lo stage è stata utilizzata la versione cloud. I wireframes creati sono stati revisionati dal tutor aziendale, il quale ha potuto inserire commenti sulle modifiche da apportare.



**Figura 3.15:** Logo Balsamiq

## Capitolo 4

# Descrizione dell'applicativo esistente

In questo capitolo viene descritto l'applicativo preesistente le cui funzionalità sono state estese durante lo stage.

### 4.1 Architettura serverless

#### 4.1.1 Definizione di una tabella DynamoDB

#### 4.1.2 Definizione di una funzione Lambda

#### 4.1.3 Definizione di un bucket S3

#### 4.1.4 Deploy del back-end

### 4.2 Web-App

#### 4.2.1 Funzionalità disponibili

#### 4.2.2 Deploy del front-end





## Capitolo 5

# Integrazione di Amazon Rekognition

In questo capitolo viene approfondito lo sviluppo e l'integrazione del sistema di image recognition

### 5.1 Presentazione del problema

MariBa è una piattaforma per il salvataggio dei risultati delle partite giocate dai dipendenti di Zero12 s.r.l. durante le pause pranzo e caffè.

L'inizializzazione di una partita prevede l'inserimento dei nomi di tutti i giocatori, specificando anche la posizione (attacco o difesa) e le squadre nel caso del calcetto balilla. Questo procedimento di inserimento dei dati iniziali risultava laborioso ed è stata quindi espressa la necessità di velocizzare tale procedura.

L'idea è quella di scattare una foto ai giocatori partecipanti in modo tale che l'applicazione li riconosca in modo automatico.

### 5.2 Progettazione

Di seguito viene descritta la progettazione ed il funzionamento della funzionalità di riconoscimento facciale implementata.

#### 5.2.1 Architettura

Di seguito vengono descritte in modo approfondito le componenti sviluppate per il servizio di image recognition, i servizi utilizzati e come essi lavorino insieme per il raggiungimento dell'obiettivo.

##### 5.2.1.1 Amazon Rekognition

*Amazon Rekognition* è un software *cloud-based* che mette a disposizione capacità di visione artificiale pre-addestrate e personalizzabili per estrarre informazioni dettagliate da immagini e video. Nel caso del progetto è stato utilizzato per indicizzare le facce e

permetterne quindi il riconoscimento. In particolare, tutte le facce sono state salvate all'interno di una raccolta (*collection*) e a ciascuna di esse è stato assegnato un ID univoco (*faceId*). Per effettuare un riconoscimento si procede ad una ricerca di eventuali corrispondenze all'interno di tale *collection*.

Di seguito sono elencate le funzioni utilizzate:

- \* **DetectFaces**: individua le cento facce di dimensione maggiore presenti nell'immagine. Per ogni viso individuato ne restituisce i dettagli, in particolare la *bounding box*;
- \* **IndexFaces**: Individua i visi all'interno di un'immagine e li aggiunge ad una *collection* specificata. Per questioni di sicurezza *Rekognition* non salva direttamente l'immagine contenente la faccia ma ne salva solamente le caratteristiche che ne permettano il riconoscimento.
- \* **SearchFacesByImage**: data un'immagine, vengono identificate le facce presenti e successivamente ne vengono cercate delle corrispondenze all'interno di una *collection* specificata.

### 5.2.1.2 S3

descrizione del bucket con la lifecycle rule

*Amazon Simple Storage Service* (S3) è un servizio di archiviazione oggetti. Al suo interno i dati sono organizzati in *bucket*. All'interno di ogni *bucket* è possibile definire dei prefissi per poter organizzare al meglio gli oggetti caricati.

Per evitare un passaggio diretto delle immagini tra front-end e back-end si è utilizzato questo servizio. S3 infatti fornisce la possibilità di generare un URL per effettuare operazione in una specifica posizione all'interno del *bucket*.

### 5.2.1.3 AWS Lambda

descrizione delle lambda e di come vengono utilizzate

## 5.2.2 Funzionamento generale

- \* front-end richiede url per caricamento
- \* caricamento su s3
- \* chiamata callRekognition
- \* scarica l'immagine e chiama elaborateImage
- \* chiama rekognition.detectFaces per l'individuazione dei visi
- \* per ciascuno dei visi chiama rekognition.searchFaceByImage per il riconoscimento
- \* per i visi riconosciuti restituisce il faceid della faccia il cui match ha maggiore confidenza
- \* per i visi non riconosciuti restituisce il crop della foto per permettere un'eventuale indicizzazione in caso di richiesta
- \* ritorna i dati a callRekognition

- \* callRekognition torna al frontend i dati
- \* il front-end li visualizza
- \* Se viene richiesto di registrare un nuovo utente:
  - il crop viene indicizzato nella collection di rekognition
  - viene restituito il faceId del viso indicizzato
  - il nuovo utente viene registrato con il faceId ricevuto in modo che la volta successiva possa essere riconosciuto direttamente
- \* Se viene richiesto di associare un nickname già esistente ad un viso: il crop della foto viene indicizzato
  - viene restituito il faceId
  - viene aggiornato il record dell'utente selezionato aggiungendo o sostituendo il faceId

### 5.2.3 Design dell'interfaccia

Per il design dell'interfaccia, prima della sua effettiva codifica, sono stati realizzati dei *wireframes* utilizzando **Balsamiq** che definissero il flusso di funzionamento a livello di front-end.

Successivamente, la realizzazione dell'interfaccia è avvenuta utilizzando **Angular 12.x** in combinazione con la libreria **Nebular**, specifica per lo sviluppo di interfacce utente. Il procedimento di inserimento dei giocatori è leggermente differente a seconda del gioco utilizzato.

#### 5.2.3.1 Inserimento di una nuova partita di calcetto

Nella pagina per l'inizializzazione di una nuova partita di calcetto è stato aggiunto un pulsante per attivare la webcam e scattare la foto contenente i volti dei giocatori (Figura 5.1). Lo scatto viene quindi inviato a *Amazon Rekognition* per il riconoscimento.

The screenshot shows a web interface for setting up a soccer game. At the top, a green header bar contains the word "Partita". Below this, the interface is divided into two main sections: "Squadra Rossa" (Red Team) on the left and "Squadra Blu" (Blue Team) on the right. Each team section contains two dropdown menus: "Difensore" (Defender) and "Attaccante" (Attacker). For the Red Team, the dropdowns currently show "Difensore Rosso" and "Attaccante Rosso". For the Blue Team, they show "Difensore Blu" and "Attaccante Blu". Below these sections, centered, is a blue button with a camera icon and the text "USA FOTOCAMERA". Above this button is the word "oppure" (or). At the bottom of the interface is a grey button labeled "PREDIZIONE RISULTATO".

Figura 5.1: Schermata iniziale di calcetto

Una volta ricevuto il risultato dell'elaborazione, i giocatori vengono visualizzati in card selezionabili per la formazione delle squadre (Figura 5.2). I giocatori possono appartenere a tre categorie differenti:

- \* **Giocatore riconosciuto:** viene mostrato il nickname corrispondente al volto riconosciuto;

- \* **Giocatore non riconosciuto ma registrato:** quando selezionato richiede l'associazione del nickname al viso scegliendo tra i nickname già presenti nel database;
- \* **Giocatore non riconosciuto e non registrato:** quando selezionato richiede la registrazione di un nuovo giocatore;

Nel caso in cui uno o più giocatori non fossero presenti all'interno della fotografia scattata, è possibile inserirli manualmente. Una volta inserito il nickname desiderato, viene mostrata la card corrispondente.

Il numero di giocatori selezionabili per ciascuna squadra è esattamente due.

**Figura 5.2:** Selezione dei giocatori

Dopo la selezione dei giocatori sarà possibile scambiare i due nickname all'interno di ciascuna squadra per associare loro il ruolo desiderato e formare le squadre definitive (Figura 5.3) .

**Figura 5.3:** Formazione squadre di calcetto

### 5.2.3.2 Inserimento di una nuova partita di Mario Kart e Duck Game

Il procedimento da seguire per l'inserimento di una partita di Mario Kart o di Duck Game tramite l'utilizzo del riconoscimento facciale dei giocatori è pressoché il medesimo. La sola differenza tra i due giochi è individuabile nel numero di giocatori selezionabili:

- \* per Mario Kart devono essere inseriti esattamente quattro giocatori (Figura 5.5);
- \* per Duck Game si possono inserire da un minimo di due ad un massimo di otto giocatori (Figura 5.4).

Ad entrambe le schermate, come per calcetto, è stato quindi aggiunto un pulsante per attivare la webcam e scattare la fotografia da inviare ad *Amazon Rekognition*.

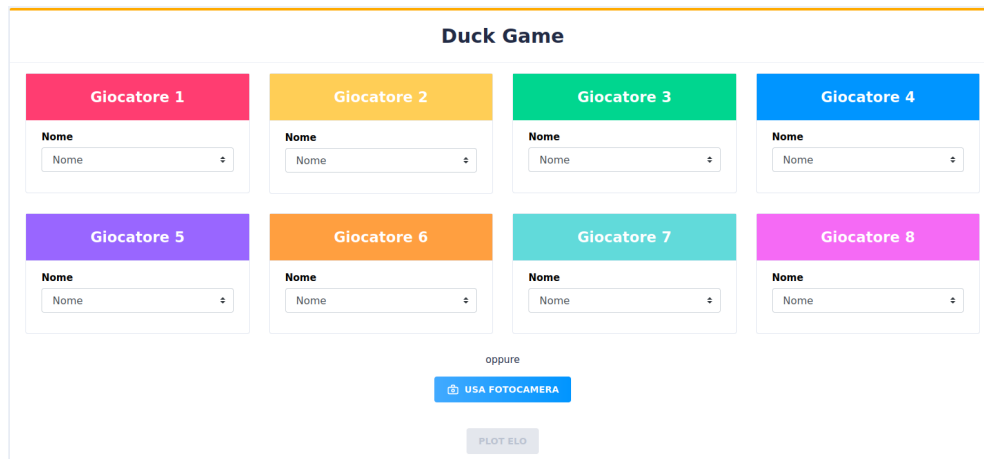


Figura 5.4: Schermata iniziale di Duck Game

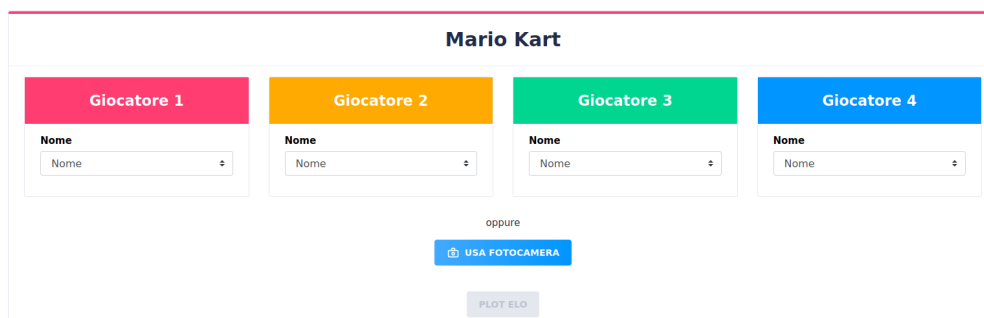


Figura 5.5: Schermata iniziale di Mario Kart

Una volta ricevuto il risultato dell'elaborazione, i giocatori vengono visualizzati in card selezionabili per la formazione delle squadre (Figura 5.6). I giocatori possono appartenere a tre categorie differenti:

- \* **Giocatore riconosciuto:** viene mostrato il nickname corrispondente al volto riconosciuto;
- \* **Giocatore non riconosciuto ma registrato:** il giocatore è già registrato ma non ha ancora volto associato. Quando selezionato richiede l'associazione del nickname al viso scegliendo tra i nickname già presenti nel database;
- \* **Giocatore non riconosciuto e non registrato:** quando selezionato richiede la registrazione di un nuovo giocatore;

Nel caso in cui uno o più giocatori non fossero presenti all'interno della fotografia scattata, è possibile inserirli manualmente. Una volta inserito il nickname desiderato, viene mostrata la card corrispondente.

The screenshot shows a web interface titled "Mario Kart" for player selection. It features three main cards, each with a green checkmark icon in its top-left corner. The first card displays a photo of a person and the name "Nicla". The second card shows a blurred photo and a dropdown menu labeled "Sono:" with an "OK" button below it; an "oppure" (or) link is present, followed by a blue button labeled "CREA NUOVO GIOCATORE". The third card shows a grey circle icon and the name "Ospite", with a blue button labeled "+ AGGIUNGI MANUALMENTE" below it. At the bottom center is a grey button labeled "PLOT ELO". A small modal dialog is open over the third card, containing a "Nickname" dropdown menu and an "OK" button.

**Figura 5.6:** Selezione dei giocatori di Mario Kart

## Capitolo 6

# Integrazione di Amazon Lex

In questo capitolo viene approfondito lo sviluppo e l'integrazione del sistema di image recognition

### 6.1 Presentazione del problema

### 6.2 Progettazione

#### 6.2.1 Architettura

#### 6.2.2 Design dell'interfaccia





## Capitolo 7

# Conclusioni

### 7.1 Consuntivo finale

Durata in ore	Settimana	Descrizione
40	1	* Studio delle tecnologie necessarie.
100	2, 3, 4	* Progettazione e sviluppo di un micro-servizio per attività di <i>face detection</i> per la creazione di squadre di gioco; * Integrazione con la piattaforma esistente.
40	4, 5	* Progettazione e sviluppo di una sezione del sito per l'inserimento dei risultati di un nuovo gioco; * Integrazione con la piattaforma esistente.
100	5, 6, 7	* Progettazione e sviluppo di un micro-servizio per il controllo vocale; * Integrazione con la piattaforma esistente.

40	8	* Stesura della documentazione di progetto delle attività di sviluppo condotte nelle settimane precedenti.
Totale ore:		320

Tabella 7.1: Pianificazione delle attività

## 7.2 Raggiungimento degli obiettivi

## 7.3 Conoscenze acquisite

## 7.4 Valutazione personale

# Glossario

**API** in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [6](#)

**Automatic Speech Recognition** prova. [3](#)

**AWS** prova. [1](#), [4-6](#), [9](#)

**chatbot** prova. [4](#)

**controllo di versione** prova. [9](#)

**CSS** prova. [8](#)

**deploy** prova. [5](#)

**FaaS** prova. [5](#)

**Framework** prova. [5](#)

**HTML** prova. [8](#)

**Image Recognition** prova. [3](#)

**Natural Language Understanding** prova. [3](#)

**NoSQL** prova. [6](#)

**repository** prova. [9](#)

**Serverless** prova. [5](#)

**skill** prova. [4](#)

**YAML** prova. [5](#)



# Bibliografia