

**PROGETTAZIONE E SVILUPPO DI  
UN SISTEMA SOFTWARE PER IL  
RICONOSCIMENTO DI IMMAGINI  
RELATIVE A MONUMENTI E OPERE  
D'ARTE**

"A mio nonno MATTEO, che mi ha insegnato l'importanza del sacrificio e la passione nell'affrontare ogni attività, per vivere ogni giorno al massimo delle nostre capacità"

"Ai miei genitori, che mi hanno supportato ogni giorno nel raggiungere i miei obiettivi"



## ABSTRACT

---

La presente tesi mostra lo sviluppo e l'implementazione di MonuReg, un sistema software concepito per il riconoscimento di immagini correlate a monumenti e opere d'arte, con l'obiettivo di migliorare l'esperienza turistica.

Il lavoro di tesi, più specificamente, ha comportato la progettazione integrale e l'effettiva realizzazione di un prototipo funzionante. Questo prototipo è stato sviluppato con l'obiettivo di dimostrare la fattibilità del progetto e prepararlo per una possibile integrazione futura nei processi aziendali, al fine di arrivare al prodotto finale.



## INDICE

<b>ABSTRACT .....</b>	<b>3</b>
<b>INDICE .....</b>	<b>4</b>
<b>1. INTRODUZIONE.....</b>	<b>6</b>
1.1 DESCRIZIONE DEL PROBLEMA .....	6
1.2 OBIETTIVO .....	6
<b>2. BACKGROUND .....</b>	<b>8</b>
2.1 CONTESTO DI APPLICAZIONE .....	8
2.2 ANALISI DEI COMPETITORS.....	8
2.2.1 <i>Getcoo Travel</i> .....	9
2.2.2 <i>Ciceros</i> .....	10
2.3 PERCHÉ MONUREG?.....	11
<b>3. IL SISTEMA MONUREG .....</b>	<b>13</b>
3.1 ANALISI DEI REQUISITI.....	13
3.1.1 <i>Requisiti funzionali</i> .....	13
3.2 CASI D'USO .....	14
3.2.1 <i>Diagramma dei casi d'uso</i> .....	14
3.2.2 <i>Scenari</i> .....	15
3.3 ARCHITETTURA DEL SISTEMA .....	20
3.3.1 <i>Diagramma delle componenti</i> .....	20
3.4 USER INTERFACE .....	21
3.4.1 <i>Flutter</i> .....	21
3.5 API REST.....	22
3.5.1 <i>Flask</i> .....	23
3.5.2 <i>Monument Recognition</i> .....	24
3.5.3 <i>Web Scraping</i> .....	29
3.6 DATABASE .....	31
3.6.1 <i>MongoDB</i> .....	31
3.7 WEB APP .....	34
3.7.1 <i>ASP.NET Core</i> .....	34
3.7.2 <i>Pattern Model-View-Controller (MVC)</i> .....	34



3.7.3 <i>ASP.NET Core MVC</i> .....	35
<b>4.     Sperimentazione in campo.....</b>	<b>42</b>
4.1   SCENARIO D'USO: MONUMENTO RICONOSCIUTO CORRETTAMENTE .....	42
4.2   SCENARIO D'USO: MONUMENTO NON RICONOSCIUTO .....	47
4.3   LAVORI FUTURI .....	49
<b>5.     Conclusioni .....</b>	<b>51</b>
<b>RIFERIMENTI .....</b>	<b>52</b>



## 1. INTRODUZIONE

---

### 1.1 Descrizione del problema

Quante volte, visitando nuovi posti e nuove città ci si è imbattuti ad osservare una statua, una chiesa e a chiedersi l'origine e la storia. Un tempo, per scoprire questa informazione, era necessario portare con sé mappe voluminose e guide. Oggi, grazie agli smartphone, bisogna prima determinare la propria posizione attuale, individuare il monumento più vicino sperando che sia quello di nostro interesse e quindi effettuare una ricerca utilizzando i principali motori di ricerca, con la possibilità di ottenere informazioni più o meno affidabili. Inoltre, questa situazione potrebbe ripetersi più volte durante il viaggio per ogni monumento, per ogni museo, per ogni chiesa.

Questo processo richiede molto tempo a un turista che esplora una nuova città, poiché è costretto a utilizzare diversi strumenti all'interno del suo smartphone, senza neanche la certezza di aver individuato il monumento corretto e le informazioni accurate. Uno dei problemi principali è la frustrazione tecnologica, soprattutto per le persone meno esperte, costrette a utilizzare vari strumenti in situazioni scomode come i viaggi, spesso con poco tempo a disposizione, oppure essendo in vacanza preferirebbero non perdere tempo nella ricerca di informazioni.

In generale, per ottenere qualsiasi tipo di informazione come gli orari di apertura di un museo o il costo dei ticket, il turista è costretto ad effettuare ricerche multiple, causando tutti i problemi precedentemente menzionati.

### 1.2 Obiettivo

L'obiettivo principale di questa tesi consiste nella progettazione e nello sviluppo di un sistema mirato ad assistere e orientare il turista in ogni aspetto del suo viaggio.

L'utente, in possesso del sistema, sarà in grado di reperire tutto quello di cui ha bisogno in un solo luogo e tutto a portata di mano.



La priorità è stata data al riconoscimento delle immagini di monumenti e opere d'arte, ma si è cercato di abbracciare il più ampio spettro possibile degli aspetti di un viaggio, al fine di creare qualcosa di autenticamente utile.

Monureg è un compagno ideale per chiunque voglia esplorare una nuova città o località. Fornisce informazioni dettagliate, suggerimenti di luoghi da visitare, informazioni sugli eventi in corso e una modalità per conservare i ricordi del viaggio attraverso un book fotografico personalizzato.



## 2. BACKGROUND

---

### 2.1 Contesto di applicazione

Il turismo culturale è un settore in crescita con sempre più persone interessate a esplorare luoghi storici e culturali in tutto il mondo. Tuttavia, uno dei principali ostacoli a un'esperienza di visita soddisfacente è la mancanza di conoscenza pregressa dei monumenti e delle opere d'arte incontrati durante il viaggio.

Mentre si passeggiava in una città mai visitata, è possibile incappare in un monumento o un'opera d'arte di cui non si ha alcuna conoscenza, suscitando il desiderio di reperire informazioni a riguardo. È in questo contesto che il sistema Monureg diventa fondamentale, poiché consente agli utenti di ottenere tutte le informazioni necessarie semplicemente scattando una foto o caricandola, eliminando ogni incertezza e migliorando notevolmente l'esperienza di visita.

L'adozione di questo sistema può avere un impatto positivo anche sulle comunità locali. L'aumento del turismo dovuto a un'esperienza di visita migliorata può portare a un aumento delle entrate per le attività (vista anche la funzionalità dedicata agli eventi vicini).

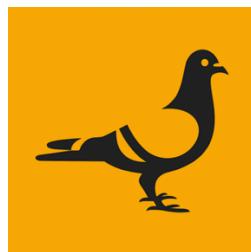
L'applicazione funziona in modo semplice: basterà inquadrare il punto d'interesse con la fotocamera e scattare una foto oppure caricarla dalla galleria. Successivamente, l'app fornirà informazioni correlate se il riconoscimento avrà successo. In caso contrario, darà la possibilità di segnalare il luogo o la città e fornire suggerimenti per migliorare ulteriormente la funzionalità.

### 2.2 Analisi dei competitors

All'interno del mercato italiano sono due i principali competitors che sono riusciti ad individuare e sono GetCoo Travel e Ciceros.



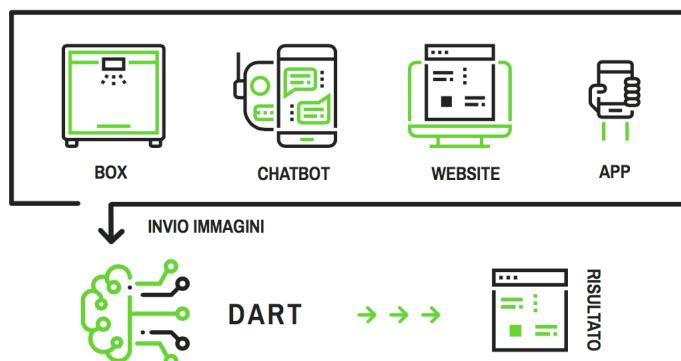
### 2.2.1 Getcoo Travel



**Figura 1: Logo GetCoo**

Getcoo è una startup che sviluppa soluzioni di Computer Vision basate su intelligenza artificiale. Il core delle soluzioni Getcoo è la tecnologia proprietaria DART, leggera e flessibile.

L'architettura di DART è altamente scalabile e personalizzabile, consentendo la sua integrazione con sistemi esistenti e l'adattamento alle esigenze specifiche. Inoltre, la tecnologia è in grado di gestire grandi volumi di dati in tempo reale, consentendo applicazioni in cui è richiesta una risposta istantanea, come il monitoraggio di sicurezza e la sorveglianza.



**Figura 2: DAR**

Indubbiamente, la presenza di questa tecnologia scalabile, accurata e veloce rappresenta un notevole vantaggio per questo sistema. Tuttavia, se devo individuare dei punti negativi, questi includono le funzionalità limitate del sistema in generale e la sua incompatibilità con i dispositivi più recenti, poiché l'ultimo aggiornamento risale al 2019 e non è pertanto compatibile con le versioni più recenti di Android e IOS. Questi sono sicuramente dei limiti per un sistema che ha un grosso potenziale, ma che evidentemente non ha riscontrato molto successo.

**Figura 3**

Esaminando le recensioni dell'applicazione sul Play Store, emerge una frequente presenza di opinioni negative, accompagnate da dichiarazioni di scuse da parte dell'azienda. Queste scuse tendono a sottolineare che il dataset di dati utilizzato nell'applicazione non è ancora sufficientemente ampio o completo per soddisfare appieno le aspettative degli utenti.

Questa carenza nel dataset è chiaramente una sfida per l'azienda, poiché impatta negativamente sull'esperienza dell'utente e sulle prestazioni dell'applicazione stessa. È evidente che, per migliorare l'apprezzamento da parte degli utenti, l'azienda dovrà concentrarsi sull'espansione e l'arricchimento delle informazioni disponibili, al fine di garantire una copertura più completa e accurata delle informazioni necessarie per il corretto funzionamento dell'applicazione.

### **2.2.2 Ciceros**

Ciceros è un progetto creato da due giovani baresi e finanziato dalla regione Puglia e permette anch'essa di riconoscere monumenti puntando la fotocamera dello smartphone. Una funzionalità interessante è quella di poter creare dei percorsi tramite la mappa il che la rende già più utilizzabile di Getcoo.

Ciceros offre anche una console dedicata a guide, storici dell'arte che possono aggiungere audioguide gratis o a pagamento.

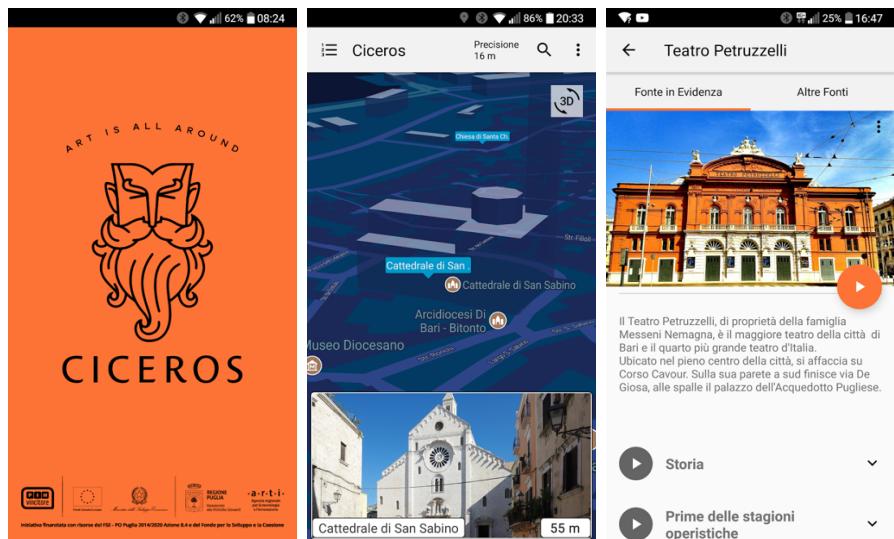


Figura 4

Non sono disponibili informazioni sulle specifiche tecnologie utilizzate.

Senza dubbio, l'inclusione di guide e storici per la condivisione di contenuti la distingue nettamente dalle altre soluzioni, costituendo un evidente vantaggio. Tuttavia, come già evidenziato per il concorrente, anche questo sistema sembra essere stato trascurato, con l'ultimo aggiornamento risalente addirittura al 2018.

### 2.3 Perché Monureg?

Monureg si propone di non solo eccellere nel suo compito principale, ovvero il riconoscimento delle immagini, ma anche di fornire un insieme di funzionalità complementari per rendere l'applicazione il più versatile possibile. L'obiettivo è garantire che l'utente non abbandoni l'app una volta completata la fase di riconoscimento, ma che, al contrario, la utilizzi in tutte le attività turistiche.

Una delle caratteristiche distintive che eleva Monureg sopra le altre soluzioni è la sua capacità di assistere l'utente in una vasta gamma di scenari. Monureg non è semplicemente un'applicazione, ma uno strumento completo che mira a essere una risorsa preziosa e affidabile in molteplici contesti. La sua versatilità si traduce in un supporto efficace e completo per un visitatore, garantendo una risposta accurata ed efficiente.

Sotto un profilo tecnologico, l'approccio adottato si è concentrato sulla scalabilità e sul perfezionamento dei sistemi di riconoscimento, oltre all'ottimizzazione delle singole componenti. Ogni volta che il sistema viene



utilizzato, si perfeziona ulteriormente, integrando nuove informazioni nel suo database e arricchendo la sua libreria di monumenti. Questo processo di miglioramento continuo si traduce in un sistema sempre più competente e informato, in grado di offrire un servizio sempre più completo e di alta qualità.



### 3. IL SISTEMA MONUREG

---

Nella costruzione della soluzione, ci si è inizialmente concentrati sulla fase cruciale della progettazione. In questa fase sono stati individuati, partendo dai requisiti, i casi d'uso e sono state gettate le solide basi su cui costruire il sistema, fornendo una visione chiara e dettagliata del percorso da seguire.

Di seguito vengono descritte tutte le fasi della progettazione e successivamente vengono analizzate le soluzioni adottate e gli strumenti utilizzati.

#### 3.1 Analisi dei requisiti

I requisiti descrivono ciò che l'utente si aspetta dal sistema, e specificarli significa esprimere in modo chiaro, univoco, consistente e completo.

##### 3.1.1 Requisiti funzionali

I requisiti funzionali descrivono cosa deve fare il sistema, generalmente in termini di relazioni fra dati di ingresso e dati di uscita, oppure fra stimoli e risposte del sistema. Un'analisi accurata dei requisiti funzionali è fondamentale per evitare ambiguità o incomprensioni tra il team di sviluppo e gli stakeholder. La chiarezza dei requisiti funzionali aiuta a stabilire le basi per il successo del progetto, garantendo che il software risponda alle esigenze degli utenti e soddisfi i requisiti aziendali.

Nella seguente tabella vengono descritti tutti i requisiti funzionali individuati durante la fase di progettazione.

Requisito	Descrizione
RF - 01 Riconoscimento immagine	Il sistema gestisce lo scatto o il caricamento di un'immagine salvata sul dispositivo ed è in grado di restituire la descrizione del monumento/opera rappresentato se è riconosciuto, altrimenti la possibilità di effettuare una segnalazione.
RF - 02 Monumenti vicini	Il sistema visualizza una mappa che evidenzia tutti i punti d'interesse rispetto alla posizione in tempo reale.

RF - 03 Eventi vicini	Il sistema visualizza un elenco con tutti gli eventi disponibili nella zona in cui si trova l'utente con una descrizione e prezzo se disponibili.
RF - 04 Book Fotografico	Il sistema gestisce il caricamento di immagini al fine di salvarle e raggrupparle all'interno di album.
RF - 05 Segnalazione	Il sistema gestisce il mancato riconoscimento di un monumento e visualizza un form per creare la segnalazione.

## 3.2 Casi d'uso

I casi d'uso sono scenari che rappresentano le interazioni tra il sistema software e i suoi utenti. Ogni caso d'uso spiega in modo chiaro e dettagliato le azioni che un utente compie interagendo con il sistema.

### 3.2.1 Diagramma dei casi d'uso



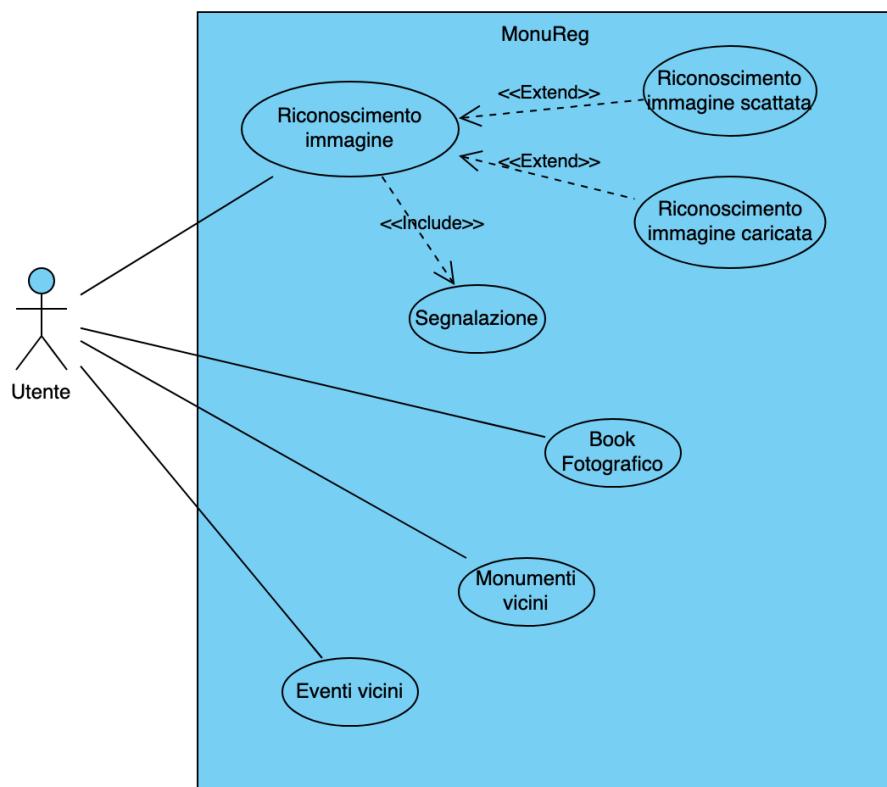


Figura 5

### 3.2.2 Scenari

Nome		Riconoscimento immagine
ID		CU1
Breve Descrizione		L'utente vuole riconoscere un monumento scattando o caricando una foto
Attori Primari		Utente
Attori Secondari		Nessuno
Precondizioni		Nessuna



<b>Sequenza Principale degli eventi</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso inizia quando l'utente accede alla sezione relativa al riconoscimento</li> <li>2. L'utente inquadra al momento il soggetto da riconoscere o ne carica una sua immagine dalla galleria</li> <li>3. Il sistema restituisce il nome del soggetto con una sua descrizione.</li> </ol>
<b>Post-Condizioni</b>	Monumento riconosciuto
<b>Sequenza Alternativa degli Eventi</b>	Segnalazione

<b>Nome</b>	Segnalazione
<b>ID</b>	CU2
<b>Breve Descrizione</b>	Il sistema visualizza un form da compilare per inviare una segnalazione
<b>Attori Primari</b>	Utente
<b>Attori Secondari</b>	Nessuno
<b>Precondizioni</b>	Monumento non riconosciuto
<b>Sequenza Principale degli eventi</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso può iniziare dopo il punto 2 del CU1, nel caso in cui l'immagine non venga riconosciuta</li> <li>2. Il sistema visualizza un form contenente più campi da compilare.</li> </ol>



	3. Le informazioni vengono salvate e inviate al database.
<b>Post-Condizioni</b>	Segnalazione creata
<b>Sequenza Alternativa degli Eventi</b>	Nessuna

Nome	Monumenti vicini
<b>ID</b>	CU3
<b>Breve Descrizione</b>	L'utente vuole visualizzare tutti i monumenti nella sua zona
<b>Attori Primari</b>	Utente
<b>Attori Secondari</b>	Nessuno
<b>Precondizioni</b>	Localizzazione attiva
<b>Sequenza Principale degli eventi</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso inizia quando l'utente accede alla sezione relativa alla visualizzazione dei monumenti vicini</li> <li>2. Viene visualizzata una mappa con i monumenti nella zona in base alla localizzazione</li> <li>3. Cliccando su un apposito bottone l'utente può aggiornare la posizione e quindi i monumenti vicini</li> </ol>
<b>Post-Condizioni</b>	Mappa visualizzata



<b>Sequenza Alternativa degli Eventi</b>	Nessuna
--	---------

Nome	Eventi Vicini
<b>ID</b>	CU4
<b>Breve Descrizione</b>	L'utente vuole visualizzare tutti gli eventi nella sua zona
<b>Attori Primari</b>	Utente
<b>Attori Secondari</b>	Nessuno
<b>Precondizioni</b>	Localizzazione attiva
<b>Sequenza Principale degli eventi</b>	<ol style="list-style-type: none"> <li>Il caso d'uso inizia quando l'utente accede alla sezione relativa alla visualizzazione degli eventi vicini</li> <li>Vengono visualizzati tutti gli eventi nella zona e le possibili attività con tutte le informazioni relative</li> </ol>
<b>Post-Condizioni</b>	Lista eventi vicini
<b>Sequenza Alternativa degli Eventi</b>	Nessuna

Nome	Book fotografico
<b>ID</b>	CU5
<b>Breve Descrizione</b>	L'utente vuole salvare le proprie foto in una sezione dedicata



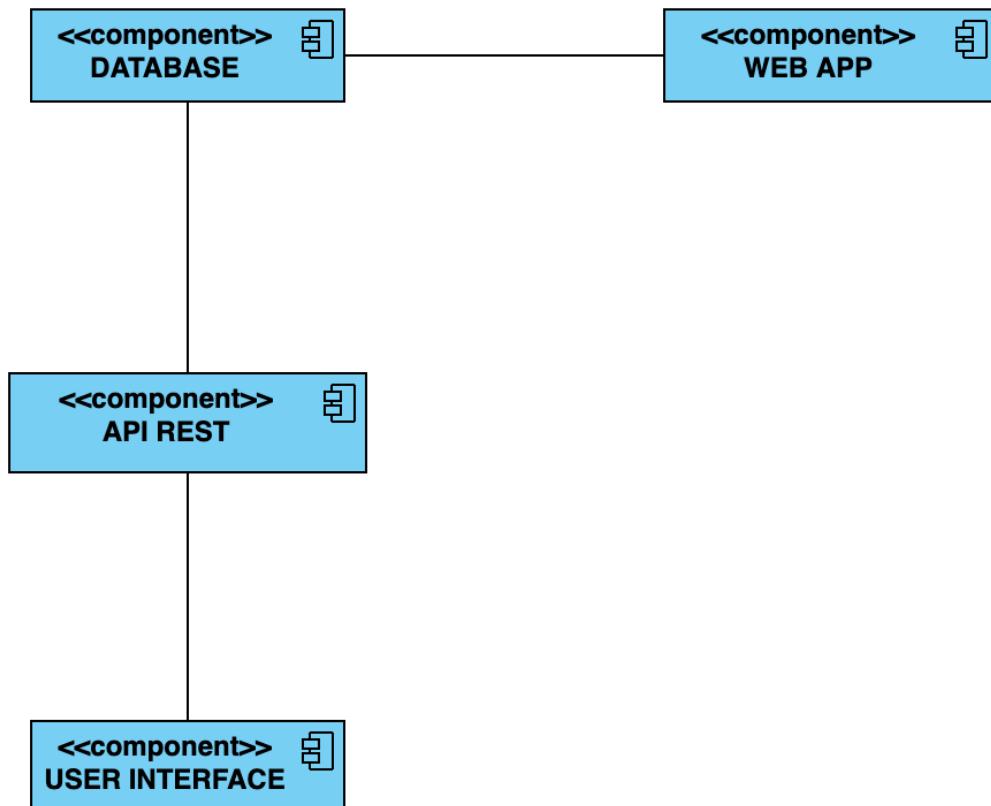
<b>Attori Primari</b>	Utente
<b>Attori Secondari</b>	Nessuno
<b>Precondizioni</b>	Nessuna
<b>Sequenza Principale degli eventi</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso inizia quando l'utente accede alla sezione relativa al salvataggio di immagini organizzate per viaggio.</li> <li>2. L'utente carica un'immagine e una sua descrizione</li> <li>3. L'immagine viene salvata nel sistema e può essere visualizzata in qualsiasi momento</li> </ol>
<b>Post-Condizioni</b>	Visualizzazione foto
<b>Sequenza Alternativa degli Eventi</b>	Nessuna



### 3.3 Architettura del sistema

#### 3.3.1 Diagramma delle componenti

Le principali componenti individuate nel processo di progettazione sono:



**Figura 6**

- **User Interface:** Interfaccia grafica che permette all'utente di interagire con il sistema ed eseguire tutti i casi d'uso individuati.
- **API Rest:** Componente principale del sistema che tramite chiamate da parte dell'interfaccia, si occuperà del riconoscimento delle immagini e della ricerca delle informazioni dal database o dal web.
- **Database:** Salvataggio e gestione di tutti i dati utili al funzionamento del sistema.
- **Web App:** Componente per la gestione interna del database.



### 3.4 User Interface

L'interfaccia utente è il mezzo attraverso cui l'utente interagisce con il computer e dalla quale partono la maggior parte delle funzionalità.

Per lo sviluppo di questa soluzione, l'interfaccia, è stata implementata utilizzando Flutter.

#### 3.4.1 Flutter

Flutter è un framework open source sviluppato e supportato da Google. Gli sviluppatori utilizzano Flutter per sviluppare l'interfaccia utente di un'applicazione per più piattaforme con un'unica base di codice.

Questo è il motivo principale della scelta di questo framework, la possibilità di sviluppare un'applicazione multiplataforma utilizzando un unico codice. Questo porta un enorme vantaggio in termini di costi e tempo avendo comunque delle prestazioni quasi native.

Flutter utilizza il linguaggio di programmazione Dart, sviluppato da Google. Dart è ottimizzato per la creazione di UI e molti dei suoi punti di forza vengono utilizzati in Flutter.[1]

Flutter è basato principalmente sui **Widget** ossia elementi visuali che possono modificarsi dinamicamente in base a logiche implementative. Inoltre, utilizzando Dart come linguaggio, il framework possiede tutti i suoi meccanismi asincroni per i compiti long-running, come ad esempio una chiamata di rete. Per rendere possibile la scrittura di applicazioni mobile cross-platform, Flutter mette a disposizione delle astrazioni che si comportano in modo analogo o quasi a quelle native di Android e iOS.

In Flutter ogni elemento visuale è un Widget. Essi sono immutabili, ossia una volta istanziate non possono cambiare. Esistono due tipi di Widget:

- Stateless, non mantengono informazioni sullo stato;
- Stateful, mantengono informazioni sullo stato.

Gli Stateful Widget, mantenendo informazioni sullo stato possono essere modificate pure essendo immutabili. Quando viene modificato il loro stato Flutter crea un nuovo albero di istanze dai Widget sostituendo quelle che



hanno subito modifiche. Questa tipologia viene usata quando sono necessarie delle modifiche in seguito ad un evento. Gli Stateless Widget invece, non mantenendo le informazioni sul loro stato, sono a tutti gli effetti immodificabili e vengono quindi utilizzate per elementi visuali come immagini.[2]

Nel capitolo 4 verrà mostrata l'interfaccia andando ad analizzare uno specifico caso d'uso.

### **3.5 API Rest**

Questa componente è il cuore dell'applicazione in quanto si occupa di tutte le operazioni principali. All'interno dell'architettura comunica sia con l'interfaccia utente per ricevere gli input e mostrare gli output, sia con il database per effettuare il salvataggio dei nuovi monumenti e delle nuove segnalazioni.

API è l'acronimo di "Application Programming Interface", un'API è un'interfaccia che consente un dialogo tra parti diverse di uno stesso software o tra parti di programmi diversi. Esse nascono con lo scopo di permettere ad uno sviluppatore di usare lo stesso codice in contesti diversi; con le API il programmatore può evitare di riscrivere ogni volta tutte le funzioni necessarie al programma dal nulla, si può quindi dire che rientrano nel più vasto concetto del riutilizzo del codice. Grazie alle API un servizio o un prodotto possono comunicare con altri servizi o prodotti senza sapere come questi vengono implementati, semplificandone lo sviluppo con un notevole risparmio di tempo e denaro. Esse si possono considerare delle vere e proprie "scatole nere" di cui il programmatore non deve conoscere il funzionamento ad un livello più basso ma solamente il compito generale, ciò permette di riprogettare o migliorare le funzioni all'interno dell'API senza dover cambiare il codice che si affida ad essa. Tutte le API che utilizzano un protocollo HTTP per le richieste e per le risposte ottenute sono considerate dei "Web Service". Un Web Service non è altro che un'interazione richiesta-risposta tra un Client e un Server. In particolare, il Client richiede una risorsa e l'API Server invia una risposta utilizzando qualsiasi linguaggio di programmazione; il linguaggio utilizzato infatti non è importante in quanto



la richiesta e la risposta sono effettuate tramite dei protocolli HTTP. Tutto ciò rende i Web Service indipendenti dal linguaggio di programmazione e interoperabili tra diverse piattaforme e sistemi; ecco spiegata la grande utilità di questi servizi, non è importante con quale linguaggio di programmazione si stia lavorando, un Web Service sarà utilizzabile in qualsiasi caso. I Web Service includono più tipi di API, tra cui le REST API.

L'API Rest, non è altro che un'API che segue i principi REST e che per funzionare ricorre ai comandi dell'HTTP, ovvero GET, POST, PUT e DELETE.

I vantaggi nell'utilizzo sono numerosi, come l'indipendenza e la separazione Client-Server, la scalabilità.

Nel contesto specifico, il client, attraverso l'interfaccia utente, invia richieste di tipo GET al nostro server, che rappresenta la nostra API. Il server elabora queste richieste e fornisce una risposta utilizzando il metodo POST. Questo processo di comunicazione tra il client e il server tramite richieste GET e risposte POST costituisce il fondamento del nostro sistema di interazione.

Questa componente all'interno del sistema Monureg si occupa di tutti i compiti principali come il riconoscimento dei monumenti, l'estrazione dei dati dal web, il salvataggio su database. Di seguito, si analizzano in dettaglio le soluzioni adottate.

### **3.5.1 Flask**

Flask è un micro-framework web leggero e altamente flessibile, scritto in Python, il quale si distingue per la sua capacità di semplificare la creazione di applicazioni web. Questo è particolarmente prezioso per progetti di piccole e medie dimensioni, in cui l'adozione di un framework "full-stack" sarebbe eccessiva e inutile. La sua popolarità nelle implementazioni di REST API in Python si deve alla sua agilità e adattabilità.

Per iniziare a utilizzare Flask, è necessario prima installarlo attraverso il comando "pip install Flask" e quindi importarlo nel proprio progetto mediante "from flask import Flask". Una volta importato, è possibile definire le route per gestire le operazioni e i metodi HTTP desiderati. Flask supporta una vasta gamma di metodi, tra cui GET, POST, PUT e DELETE, offrendo così una grande versatilità nella gestione delle richieste.



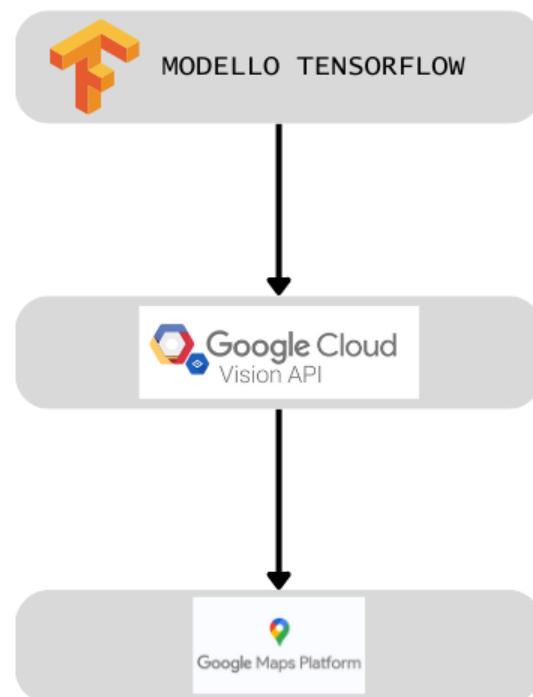
Un vantaggio significativo di Flask è la sua funzione "jsonify", che semplifica la conversione degli oggetti Python in risposte JSON, rendendo la manipolazione dei dati ancora più agevole.

In ambito aziendale, durante la fase di deployment, il codice sviluppato con Flask viene solitamente ospitato su un server web. In particolare, nel contesto descritto, il codice è stato eseguito localmente, specificando una porta per l'ascolto delle richieste in arrivo.

Questo framework si distingue per la sua capacità di adattarsi alle esigenze specifiche del progetto, garantendo al contempo una leggerezza che lo rende una scelta ideale per sviluppatori che desiderano maggiore controllo e flessibilità nella creazione di applicazioni web.[7]

### 3.5.2 Monument Recognition

Il riconoscimento dei monumenti è stato sviluppato in più livelli, utilizzando diverse tecnologie al fine di avere una soluzione che sia il più veloce e accurata possibile.



**Figura 7**

Come si può notare in figura, il processo di riconoscimento segue una sequenza ben definita. Inizialmente, avviene un tentativo di riconoscimento utilizzando un modello pre-addestrato disponibile su TensorFlow. Questo



modello pre-addestrato è la prima risorsa impiegata per identificare l'oggetto o il monumento in questione.

Se il risultato del riconoscimento tramite TensorFlow è negativo o insoddisfacente, allora si attiva una seconda fase. In questa fase, si fa ricorso all'API di Google chiamata Vision AI. Questo servizio esterno di Google offre funzionalità di riconoscimento avanzato, che può essere utilizzato come risorsa supplementare per migliorare la precisione del riconoscimento.

Tuttavia, se anche l'API di Google Vision AI non riesce a dare un risultato positivo, si passa a una terza opzione. In questo caso, si sfrutta la posizione geografica per determinare il monumento più vicino. Questa opzione è utile quando le risorse di riconoscimento visivo non sono sufficienti, ma la posizione geografica dell'utente può fornire indizi preziosi per identificare l'oggetto o il monumento nelle vicinanze.

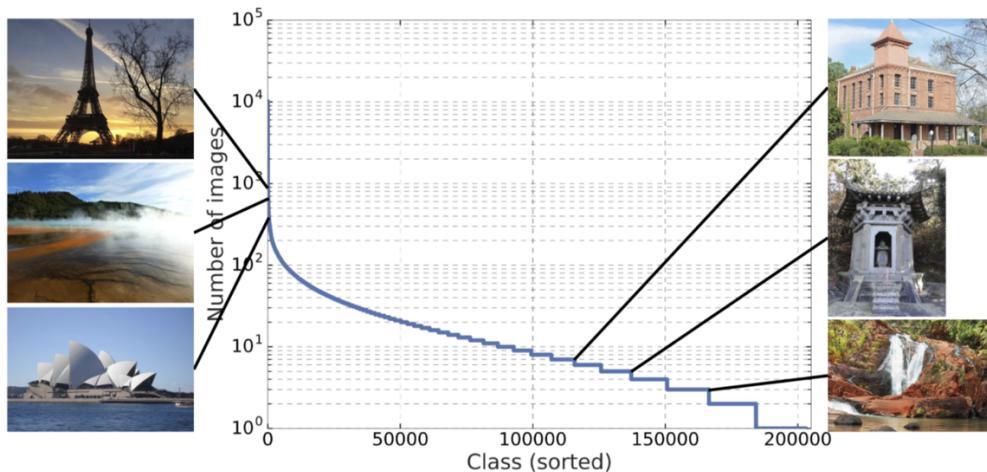
Questo approccio a tre fasi garantisce un livello crescente di precisione nel riconoscimento e assicura che, anche in situazioni complesse, si possa fornire all'utente una risposta adeguata basata su diverse fonti di dati e informazioni.

### **3.5.2.1 Modello TensorFlow**

Il primo strumento utilizzato è un modello disponibile su TensorFlow Hub che è stato addestrato utilizzando il più grande dataset sui punti di riferimento disponibile in modo gratuito: il Google Landmarks Dataset v2(GLDv2).

GLDv2 è di gran lunga uno dei dataset più grandi ad oggi, con oltre 5 milioni di immagini e 200.000 etichette di istanze distinte. Il suo set di test è composto da 118.000 immagini con annotazioni di verità sul campo per i compiti di recupero e riconoscimento. Il dataset proviene da Wikimedia Commons, la più grande raccolta di fotografie di punti di riferimento ottenute attraverso il crowdsourcing nel mondo.





**Figura 8**

Nella seguente figura, effettuiamo un confronto tra i dataset attualmente disponibili per il riconoscimento dei punti di riferimento e il nostro ampio GLDv2. È interessante notare che alcuni di questi dataset si limitano a raccogliere immagini da singole città, mentre altri presentano un numero di punti di riferimento notevolmente inferiore rispetto al dataset che abbiamo selezionato. La vasta quantità di immagini e le numerose etichette di istanze distinte presenti nel GLDv2 offrono una prospettiva eccezionalmente ricca per il riconoscimento dei punti di riferimento e il recupero delle immagini, rendendolo uno strumento prezioso per la ricerca e lo sviluppo di applicazioni pratiche in questo ambito.

Dataset name	Year	# Landmarks	# Test images	# Train images	# Index images	Annotation collection	Coverage	Stable
Oxford [42]	2007	11	55	-	5k	Manual	City	Y
Paris [43]	2008	11	55	-	6k	Manual	City	Y
Holidays [28]	2008	500	500	-	1.5k	Manual	Worldwide	Y
European Cities 50k [5]	2010	20	100	-	50k	Manual	Continent	Y
Geotagged StreetView [32]	2010	-	200	-	17k	StreetView	City	Y
Rome 16k [1]	2010	69	1k	-	15k	GeoTag + SfM	City	Y
San Francisco [14]	2011	-	80	-	<b>1.7M</b>	StreetView	City	Y
Landmarks-PointCloud [36]	2012	1k	10k	-	205k	Flickr label + SfM	Worldwide	Y
24/7 Tokyo [56]	2015	125	315	-	1k	Smartphone + Manual	City	Y
Paris500k [61]	2015	13k	3k	-	501k	Manual	City	N
Landmark URLs [7, 22]	2016	586	-	140k	-	Text query + Feature matching	Worldwide	N
Flickr-SfM [45]	2016	713	-	120k	-	Text query + SfM	Worldwide	Y
Google Landmarks [40]	2017	30k	<b>118k</b>	1.2M	1.1M	GPS + semi-automatic	Worldwide	N
Revisited Oxford [44]	2018	11	70	-	5k + 1M	Manual + semi-automatic	Worldwide	Y
Revisited Paris [44]	2018	11	70	-	6k + 1M	Manual + semi-automatic	Worldwide	Y
Google Landmarks Dataset v2	2019	<b>200k</b>	<b>118k</b>	<b>4.1M</b>	762k	Crowsourced + semi-automatic	Worldwide	Y

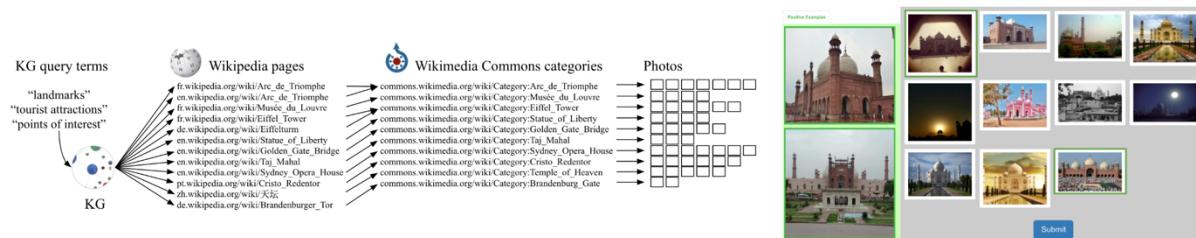
**Figura 9**

Tutte le immagini in GLDv2 sono sotto licenza gratuita, garantendo così che il set di dati sia conservabile indefinitamente e non si riduca nel tempo, consentendo il confronto di approcci di riconoscimento e recupero nel corso di un lungo periodo di tempo. Tutte le immagini possono essere liberamente



riprodotte nelle pubblicazioni, a condizione che venga fornita l'adeguata attribuzione.

Il processo di estrazione delle immagini è avvenuto tramite Wikipedia Commons e in figura viene riassunto. Wikipedia Commons è organizzato in una gerarchia di categorie che costituiscono la sua tassonomia. Ogni categoria ha un URL unico in cui sono elencate tutte le immagini associate ad essa. È stato utilizzato il Google Knowledge Graph per ottenere un elenco esaustivo dei punti di riferimento nel mondo, effettuando una query su "landmarks", "tourist attractions", "point of interest". Per ciascuna entità si sono ottenuti i relativi articoli su Wikipedia. Successivamente, è stato seguito il collegamento delle Categorie di Wikipedia Commons nell'articolo di Wikipedia e sono state scaricate tutte le immagini contenute nelle pagine di Wikipedia Commons.[6]



**Figura 10**

### 3.5.2.2 Google Vision AI

L'API Vision di Google Cloud è uno dei due strumenti (oltre ad AutoML Vision) offerti da Google per la comprensione delle immagini e offre avanzati modelli di machine learning pre-addestrati per diversi compiti, tutti pay-per-use. Tra le funzionalità di questo strumento vi sono il rilevamento di oggetti appartenenti alle categorie più comuni, il confronto delle foto con il catalogo prodotti del cliente, il rilevamento del testo, il rilevamento dei volti, il rilevamento e l'identificazione di luoghi popolari e loghi di marchi famosi, la moderazione dei contenuti espliciti.



**Figura 11**

In questo contesto, la scelta di questo strumento si è basata principalmente sulla sua affidabilità ed efficienza, anche se va notato che è associato a un costo considerevole. La decisione di impiegarlo avviene solo quando il modello precedentemente descritto non riesce a garantire il riconoscimento necessario. Questa strategia si adatta perfettamente alle esigenze del nostro



attuale livello di studio e di progettazione, dove stiamo lavorando su un prototipo con l'obiettivo di valutare la fattibilità del prodotto.

Tuttavia, è importante sottolineare che in fasi più avanzate del processo, potremmo prendere in considerazione l'opportunità di sostituire questa componente con un'altra soluzione, in base agli obiettivi specifici e alle disponibilità economiche. Il nostro sistema è stato progettato con un'attenzione particolare alla modularità e alla scalabilità, il che significa che siamo in grado di adattarci alle esigenze future e di evolvere il sistema in modo coerente con i nostri obiettivi e le risorse disponibili.

A livello implementativo, viene innanzitutto importata la libreria dedicata (from google.cloud import vision). In seguito, viene preparata l'immagine e insieme alle credenziali di accesso all'API vengono passate al metodo che si occupa di effettuare il riconoscimento dell'immagine, restituendo una risposta contenente il nome del monumento e la posizione espressa in latitudine e longitudine.

Dal punto di vista dell'implementazione, il processo inizia con l'importazione della libreria dedicata, che viene eseguita utilizzando l'istruzione "from google.cloud import vision". Successivamente, è necessario preparare l'immagine che si desidera analizzare. Insieme all'immagine, vengono fornite le credenziali di accesso all'API.

Una volta che l'immagine e le credenziali sono pronte, queste informazioni vengono passate al metodo responsabile dell'effettuazione del riconoscimento dell'immagine. Questo metodo eseguirà l'analisi dell'immagine utilizzando le API di Google Cloud Vision e restituirà una risposta. La risposta contiene due informazioni principali: il nome del monumento o dell'oggetto riconosciuto nell'immagine e la posizione del monumento espressa in termini di latitudine e longitudine.



```

image_data = prepare_image_local(image)

client = vision.ImageAnnotatorClient()
va = VisionAI(client, image_data)
landmarks = va.landmark_detection()
serialized = proto.Message.to_json(landmarks)
json_data = json.loads(serialized)
if not 'error' in json_data:
    nome = json_data['landmarkAnnotations'][0]['description']
    latitudine = json_data['landmarkAnnotations'][0]['locations'][0]['latLng']['latitude']
    longitudine = json_data['landmarkAnnotations'][0]['locations'][0]['latLng']['longitude']
elif 'error' in landmarks:
    return Exception

```

**Figura 12**

### 3.5.3 Web Scraping

Dopo aver identificato il monumento e ottenuto il suo nome, il sistema procede con la fase di estrazione delle informazioni dal web. In questo contesto, Wikipedia è stata selezionata come la principale fonte di dati per questa operazione.

Il Web Scraping è una tecnologia che mira all'estrazione di informazioni dai siti web. Esso simula l'esplorazione umana del World Wide Web e si concentra sulla trasformazione di informazioni non strutturate (solitamente in formato HTML) presenti sul web in informazioni strutturate che possono essere salvate ed elaborate in un database centralizzato. [3]

Il processo di scraping dei dati da Internet può essere suddiviso in due fasi sequenziali: l'acquisizione delle risorse web e l'estrazione delle informazioni desiderate.

In particolare, un programma di web scraping inizia componendo una richiesta HTTP per acquisire risorse da un sito web specifico. Questa richiesta può essere formattata come un URL contenente una query GET o un messaggio HTTP contenente una richiesta POST.

Una volta che la richiesta è stata ricevuta ed elaborata con successo dal sito web, essa verrà recuperata dal sito web e quindi rinviata al programma di web scraping. La risorsa può essere restituita in più formati, come pagine web costruite a partire da HTML, feed di dati in formato XML o JSON, o dati multimediali come immagini, file audio o video.



Esistono due moduli essenziali di un programma di web scraping: un modulo per la composizione di una richiesta HTTP, come Urllib2 o Selenium, e un altro per il parsing e l'estrazione di informazioni dal codice HTML grezzo, come BeautifulSoup o Pyquery.[4]

## WEB SCRAPING IN MONUREG

Dopo aver identificato il monumento e ottenuto il suo nome, il sistema procede con la fase di estrazione delle informazioni dal web. In questo contesto, Wikipedia è stata selezionata come la principale fonte di dati per questa operazione.

Per effettuare questa operazione è stata impiegata la libreria BeautifulSoup che si occupa di analizzare ed estrarre dati da documenti HTML e XML. Con questa libreria, è possibile analizzare il codice sorgente di una pagina web e isolare i tag HTML, i testi e gli attributi che contengono le informazioni di interesse. Nello specifico, nel contesto di questo progetto, ci si è concentrati sull'estrazione dei primi due paragrafi della sezione descrittiva di Wikipedia.

```
def web_scraping_wikipedia(parola, lingua):
    if lingua == 0:
        url = f"https://en.wikipedia.org/wiki/{parola}"
    else:
        url = f"https://it.wikipedia.org/wiki/{parola}"

    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        paragrafi = soup.findAll('p')

        testi = [p.get_text() for p in paragrafi[:2]]
        return testi
    else:
        print(f"La richiesta HTTP non è stata eseguita correttamente. Codice di stato: {response.status_code}")
```

**Figura 13**

In figura, viene mostrato il codice della funzione che si occupa di effettuare l'estrazione dei dati dal sito.

Per comprendere il processo, innanzitutto, viene effettuata una richiesta GET al sito web di interesse. Se questa richiesta ha successo, viene richiamata la funzione “BeautifulSoup” della libreria “BeautifulSoup” che si occupa di



effettuare il parsing della pagina. Una volta che abbiamo ottenuto il contenuto della pagina, il nostro obiettivo è estrarre informazioni specifiche. Pertanto, selezioniamo i tag HTML di tipo `<p>` all'interno del documento. Per mantenere il focus solo su ciò che ci interessa, cioè i dati dei primi due paragrafi, applichiamo un filtro per isolare solamente questi due paragrafi.

### 3.6 Database

Per la componente legata al salvataggio delle informazioni è stato scelto un database di tipo NoSql. Invece della tipica struttura tabulare di un database relazionale, i database NoSQL ospitano i dati all'interno di una struttura dati, come ad esempio un documento JSON. Tra i vantaggi si notano sicuramente le performance più alte per i tempi di risposta, infatti nei database non relazionali un elemento contiene tutte le informazioni necessarie e dunque non serve usare i dispendiosi "join" come avviene per i database relazionali. Un altro vantaggio deriva dalla semplicità di questi database e che infatti permette di scalare in orizzontale in maniera efficiente. La semplicità di questi database porta però anche degli svantaggi, infatti, non essendoci dei controlli fondamentali sull'integrità dei dati, il compito ricade quindi totalmente sull'applicativo che dialoga con il database. Altro svantaggio è la mancanza di uno standard universale come per l'SQL, ogni database ha infatti le proprie API e il suo metodo di storing e di accesso ai dati.[10]

Nonostante questi svantaggi, i database non relazionali risultano la miglior soluzione in contesti in cui ogni giorno bisogna salvare un numero elevatissimo di dati. Queste sono le caratteristiche perfette per il nostro contesto e in particolare è stato utilizzato MongoDB.

#### 3.6.1 MongoDB

MongoDB memorizza i dati in documenti flessibili JSON-like, il che significa che i campi possono variare da un documento all'altro ed è possibile modificare nel tempo la struttura dei dati. Questo modello di database eredita il meccanismo di storage dal paradigma doc-oriented che consiste nel memorizzare ogni record come documento che possiede caratteristiche predeterminate. Si può aggiungere un numero qualsiasi di campi con una qualsiasi lunghezza.

Nei doc-oriented si segue una metodologia differente rispetto al modello relazionale: si accorpano quanto più possibile gli oggetti, creando delle



macro entità dal massimo contenuto informativo. Questi oggetti incorporano tutte le notizie di cui necessitano per una determinata semantica.

Pertanto MongoDB non possiede uno schema e ogni documento non è strutturato, ha solo una chiave obbligatoria: `_id`, la quale serve per identificare univocamente il documento; essa è comparabile, semanticamente, alla chiave primaria dei database relazionali.[10]

Per lo sviluppo delle funzionalità implementate sono state create tre collections: monuments, reports e mon\_images.

In monuments, vengono accuratamente catalogati e conservati tutti i luoghi di interesse che emergono dall'analisi delle immagini. Questi punti di interesse, acquisiti attraverso il riconoscimento delle immagini, vengono registrati con una serie di informazioni chiave. Ogni monumento è identificato da un unico identificatore (ID), che lo rende univoco all'interno della collection. Oltre all'ID, vengono registrati il nome completo del monumento, una dettagliata descrizione che ne cattura la storia e le caratteristiche uniche, e le coordinate geografiche che definiscono la sua posizione precisa, espressa in latitudine e longitudine.

```

_id: ObjectId('64be896dcbdd6ddb4a285582')
nome: "Stadio San Nicola"
descrizione: "Lo stadio San Nicola è il maggiore impianto sportivo della città di Ba..."
latitudine: 41.0847298
longitudine: 16.840135199999995

_id: ObjectId('64e326a467ad8d1728ce638e')
nome: "Teatro Petruzzelli"
descrizione: "Il teatro Petruzzelli è il maggiore teatro di Bari e il quarto più grande d...
latitudine: 41.123536
longitudine: 16.872726

_id: ObjectId('64e32b74d5b66c38a4d47a')
nome: "Palazzo Mincuzzi"
descrizione: "Il Palazzo Mincuzzi è uno storico edificio commerciale di Bari situato nel ...
latitudine: 41.1232419999995
longitudine: 16.869445799999995

```

**Figura 14**

In reports, vengono invece registrate tutte le segnalazioni raccolte in situazioni in cui un'immagine non può essere classificata correttamente. Questi dati vengono raccolti attraverso un form appositamente progettato all'interno dell'interfaccia del sistema e vengono successivamente archiviati nel nostro database per ulteriori analisi e azioni appropriate. Ogni segnalazione è identificata da un ID univoco, l'email del mittente, la città in cui è stata scattata la foto, il monumento (se lo si conosce) e eventuali note per descrivere al meglio la situazione.



```

_id: ObjectId('64a5ae921569dc609d0832c8')
email: "matteo.ciccone23@gmail.com"
city: "Bari"
monument: ""
note: "vicino il lungomare"

_id: ObjectId('64d338889b69eb1c16f08a3b')
email: "mc@gmail.com"
city: "Bari"
monument: ""
note: "stadio di bari"

```

**Figura 15**

La collection mon\_images è stata creata con l'obiettivo specifico di archiviare tutte le fotografie e assegnare loro un'etichetta. Questo processo è fondamentale per il continuo miglioramento del sistema in futuro. Le immagini etichettate costituiscono una risorsa preziosa che ci consente di sviluppare un modello incrementale basato su dati in costante espansione, parallelamente all'uso crescente del sistema.

Le immagini etichettate svolgono un ruolo cruciale nel nostro approccio di apprendimento automatico. Consentono di addestrare e migliorare costantemente il nostro algoritmo di riconoscimento delle immagini. Man mano che nuove immagini vengono acquisite e etichettate, il nostro modello può essere affinato per riconoscere una gamma più ampia di monumenti e dettagli correlati.

Questo ciclo di apprendimento continuo e progressivo contribuisce notevolmente all'evoluzione del sistema nel tempo, consentendo una classificazione più precisa delle immagini e una migliore comprensione dei punti di interesse.

La collection è formata da tre elementi chiave: il percorso verso l'immagine memorizzata nel file system, l'etichetta corrispondente al nome del monumento e un identificatore univoco (ID).

```

{
  "_id": {"$oid": "64d338889b69eb1c16f08a3b"},
  "image": "/Users/matteocicccone/Desktop/casodistudio/files/Palazzo Mincuzzi/19-50-18..jpg",
  "monument_name": "Palazzo Mincuzzi"
}

{
  "_id": {"$oid": "64d338889b69eb1c16f08a3b"},
  "image": "/Users/matteocicccone/Desktop/casodistudio/files/Palazzo Mincuzzi/19-50-38..jpg",
  "monument_name": "Palazzo Mincuzzi"
}

```

**Figura 16**

### 3.7 Web App

Nel contesto di questo progetto, la back-office web app costituisce l'ultima componente concepita e sviluppata. La sua fondamentale funzione è la gestione dei dati archiviati nel database del sistema. Questa applicazione è progettata per essere utilizzata dai responsabili del sistema, come gli amministratori e il personale interno, allo scopo di eseguire una serie di operazioni cruciali. Tra queste operazioni, si includono la possibilità di modificare o aggiornare le informazioni relative ai monumenti presenti nel database, garantendo che queste descrizioni siano sempre precise e aggiornate per gli utenti finali. Inoltre, la back-office web app consente anche di gestire in modo efficiente le segnalazioni inviate dagli utenti, fornendo uno strumento completo per l'analisi, la risoluzione e il monitoraggio delle questioni sollevate dalla community.

Per realizzare questa applicazione, è stato adottato il framework ASP.NET con l'architettura MVC (Model-View-Controller). Questa scelta architettonica offre un solido fondamento per lo sviluppo di applicazioni web avanzate.

#### 3.7.1 ASP.NET Core

ASP.NET è un framework di sviluppo web robusto e potente sviluppato da Microsoft. È ampiamente utilizzato per creare applicazioni web dinamiche e scalabili che soddisfano le esigenze complesse del mondo moderno dell'informatica. Uno dei punti di forza più notevoli di ASP.NET è la sua capacità di adottare diverse architetture di applicazioni web, con l'architettura Model-View-Controller (MVC) che si distingue per la sua organizzazione e modularità.

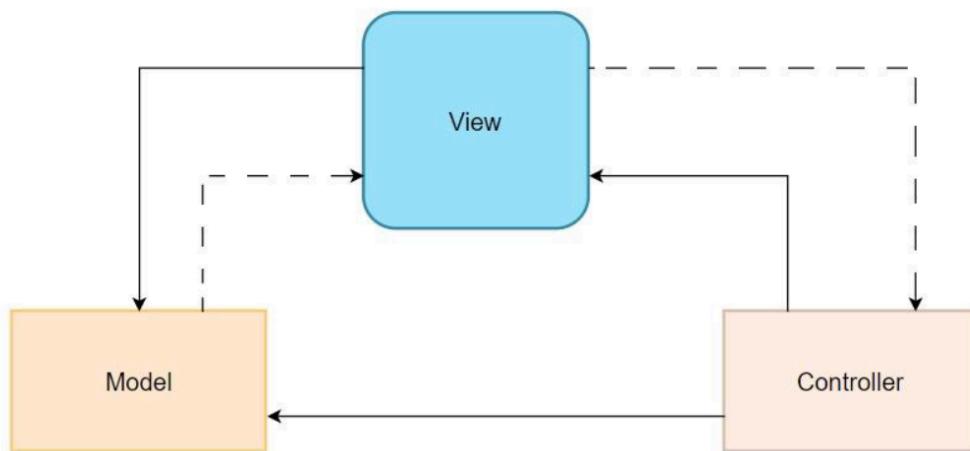
ASP.NET Core è indicato sia per lo sviluppo di pagine web, sia per lo sviluppo di API REST.<sup>[8]</sup>

#### 3.7.2 Pattern Model-View-Controller (MVC)

Model-View-Controller è un pattern utilizzato in programmazione per dividere il codice in blocchi dalle funzionalità ben distinte. Questa architettura separa il sistema in tre componenti principali, ognuna con un ruolo specifico:



- Il **Model** che rappresenta la logica aziendale e i dati dell'applicazione. È responsabile della manipolazione e dell'elaborazione dei dati. È indipendente dalla View e dal Controller.
- La **View** che è responsabile della presentazione dei dati all'utente.
- Il **Controller** che funge da intermediario tra il Model e la View. Riceve le richieste dell'utente dalla View, interagisce con il Model per ottenere o aggiornare i dati e quindi determina quale View deve essere utilizzata per mostrare i risultati.



**Figura 17**

In questo specifico contesto applicativo, sono state implementate diverse View per soddisfare le esigenze di interazione dell'utente. Tra queste View, figurano l'index, che rappresenta la pagina iniziale dell'applicazione, e, in particolare per le sezioni dedicate ai Monumenti e ai Report, sono state create View specifiche per le operazioni di Creazione, Modifica (Edit) e Visualizzazione generale (Index).

### 3.7.3 ASP.NET Core MVC

Il framework ASP.NET Core MVC segue la divisione descritta in precedenza.

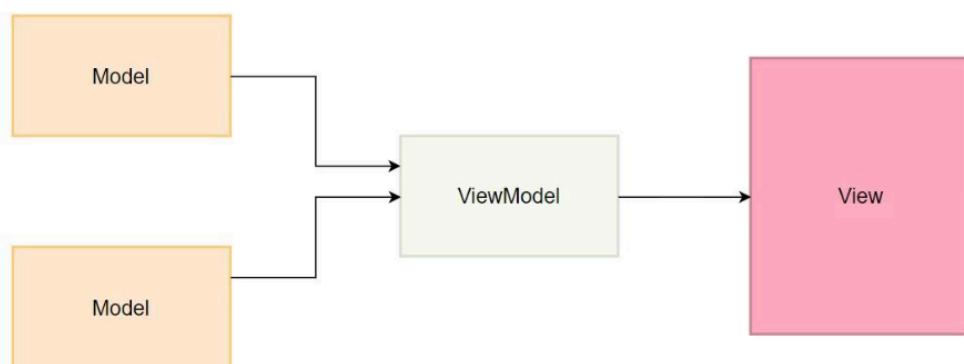
Le View presentano in HTML i dati che vengono forniti dal controller. In ASP.NET Core MVC le View a supporto della presentazione dei dati, possono contenere anche codice C#. Il codice viene inserito tramite il simbolo @.

Un controller è una semplice classe C# che viene creata nella directory Controllers. I metodi pubblici creati al suo interno si chiamano action. I Controller utilizzano il middleware di routing e dei servizi di ASP.NET Core MVC per richiamare un metodo quando viene chiamata una vista che di solito ha il suo stesso nome.



Il Model è la componente MVC dedicata alla logica per l'accesso ai dati, che si occupa di fare da tramite tra l'applicazione e il database sottostante. Il model si preoccupa di creare il necessario livello di astrazione tra il formato in cui i dati sono memorizzati alla fonte e il formato in cui i livelli di Controller e View si aspettano di riceverli.

Per il passaggio dei dati da Model a View può essere usato un ulteriore elemento chiamato ViewModel: classi che hanno la funzione di contenere al proprio interno delle proprietà. Le classi ViewModel contengono solo i dati necessari alla creazione della View a cui vengono passati.[9]



**Figura 18**

### 3.7.3.1 Model

Per quanto riguarda il model sono state implementate due classi che contengono i dati riguardanti Monumenti e Segnalazioni e una classe che contiene solamente due contatori che tengono traccia del numero di Monumenti e Segnalazioni che verranno visualizzati nella schermata principale (HomeIndexViewModel)

```

using System;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

namespace MONUREGDB.Models
{
    public class Monument
    {
        [BsonId]
        public ObjectId Id { get; set; }

        [BsonElement("nome")]
        public string Nome { get; set; } = String.Empty;

        [BsonElement("descrizione")]
        public string Descrizione { get; set; } = String.Empty;

        [BsonElement("latitudine")]
        public double Latitudine { get; set; }

        [BsonElement("longitudine")]
        public double Longitudine { get; set; }
    }
}

```

Figura 19

```

using System;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
using System.ComponentModel.DataAnnotations;

namespace MONUREGDB.Models
{
    public class Report
    {
        [BsonId]
        public ObjectId Id { get; set; }

        [BsonElement("email")]
        [EmailAddress]
        public string Email { get; set; } = String.Empty;

        [BsonElement("city")]
        public string City { get; set; } = String.Empty;

        [BsonElement("monument")]
        public string Monument { get; set; } = String.Empty;

        [BsonElement("note")]
        public string Note { get; set; } = String.Empty;
    }
}

```

Figura 20

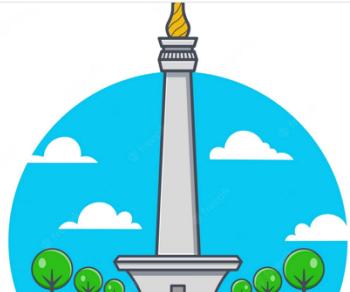
### 3.7.3.2 View

Nella view sono state implementate la schermata principale (Index) e per i Monumenti e le Segnalazioni, le rispettive pagine di visualizzazione e di creazione e modifica.



MONUREG Monuments Reports

### Database manager



**Monuments Collection**  
Nr. Monument: 3

[Modifica](#)



**Reports Collection**  
Numero di report: 10

[Modifica](#)

© 2023 - MONUREG - [Privacy](#)

**Figura 21**

MONUREG Monuments Reports

### Monument List

[Create New Monument](#)

Name	Description	Latitude	Longitude		
Stadio San Nicola	Lo stadio San Nicola è il maggiore impianto sportivo della città di Bari, della regione Puglia e del sud Italia. Progettato da Renzo Piano e soprannominato da lui stesso Astronave per via della sua caratteristica conformazione architettonica, è stato realizzato in occasione della campionato mondiale di calcio 1990. Sorge nella zona sud-ovest della città, nel territorio del IV municipio (nella zona dell'ex circoscrizione Carbonara-Santa Rita) in un'area di 533 000 m <sup>2</sup> . Di proprietà del Comune di Bari, è stato intitolato il 28 aprile 1990 al santo patrono della città a seguito di un referendum popolare svolto in abbinamento alla Gazzetta del Mezzogiorno. Ospita le partite casalinghe della società calcistica del Bari dalla stagione 1990-1991.	41,0847298	16,840135199999995	<a href="#">Edit</a>	<a href="#">Delete</a>
Teatro Petruzzelli	Il teatro Petruzzelli è il maggiore teatro di Bari e il quarto più grande d'Italia, ubicato nel pieno centro del capoluogo pugliese. Il teatro è di proprietà della famiglia Messeni Nemagna.[1]	41,123536	16,872726	<a href="#">Edit</a>	<a href="#">Delete</a>
Palazzo Mincuzzi	Il Palazzo Mincuzzi è uno storico edificio commerciale di Bari situato in via Sparano da Bari, 98. Il palazzo venne eretto tra il 1926 e il 1928 secondo il progetto dell'architetto Aldo Forcignanò e dell'ingegnere Gaetano Palmiotto su commissione della famiglia Mincuzzi, proprietaria degli omonimi grandi magazzini, per dare una nuova sede al commercio di famiglia. I lavori vennero realizzati dall'impresa costruttrice di Giuseppe Garibaldi e fratelli. Il palazzo venne fastosamente inaugurato il 28 ottobre del 1928 con la partecipazione delle principali autorità cittadine e di buona parte della cittadinanza.[2] Il palazzo è divenuto in breve tempo l'icona della Bari commerciale.[3]	41,123244199999995	16,869445799999998	<a href="#">Edit</a>	<a href="#">Delete</a>

© 2023 - MONUREG - [Privacy](#)

**Figura 22**



SERLAB **Progettazione e sviluppo di un sistema software per il riconoscimento** di immagini relative a monumenti e opere d'arte

38 /

The screenshot shows a 'Create Monument' form. At the top left is the navigation bar: MONUREG, Monuments, Reports. Below it is the title 'Create Monument'. The form fields include:

- 'Nome' (Name) with an input field.
- 'Descrizione' (Description) with an input field.
- 'Latitudine' (Latitude) with two input fields: 'Parte Intera' (Whole part) and 'Parte Decimale' (Decimal part).
- 'Longitudine' (Longitude) with two input fields: 'Parte Intera' (Whole part) and 'Parte Decimale' (Decimal part).

At the bottom are two buttons: 'Create' (blue) and 'Cancel' (grey).

At the very bottom of the page, there is a copyright notice: © 2023 - MONUREG - [Privacy](#).

**Figura 23**

### 3.7.3.3 Controller

I controller implementati gestiscono le operazioni all'interno della web app.

L'HomeController si occupa della connessione a MongoDB, del recupero di tutti i dati salvati all'interno di esso e del conteggio dei Monumenti e delle Segnalazioni totali.



```

public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly IMongoDatabase _database;

    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
        var client = new MongoClient("mongodb://localhost:27017");
        _database = client.GetDatabase("monureg");
    }

    public IActionResult Index()
    {
        var monumentCollection = _database.GetCollection<Monument>("monuments");
        var monumentCount = monumentCollection.CountDocuments(new BsonDocument());

        // Ottenere il numero di documenti nella collezione Report
        var reportCollection = _database.GetCollection<Report>("reports");
        var reportCount = reportCollection.CountDocuments(new BsonDocument());

        var viewModel = new HomeIndexViewModel
        {
            MonumentCount = monumentCount,
            ReportCount = reportCount
        };

        return View(viewModel);
    }
}

```

Figura 24

Il MonumentController e il ReportController si occupano innanzitutto di prendere i dati dal database e di salvarli in modo strutturato all'interno delle classi create nel Model. Inoltre, permettono di effettuare il create e l'edit, impostando l'apertura delle rispettive pagine, e del delete. Un'altra funzionalità che gestiscono è quella dell'invio di una mail ogni volta che viene aggiunto un nuovo elemento nel database.

```

[HttpPost]
public ActionResult Create(Monument monument, string latitudineParteIntera, string latitudineParteDecimale, string longitudineParteIntera, string longitudineParteDecimale)
{
    string latitudineCompleta = $"{latitudineParteIntera}.{latitudineParteDecimale}";
    string longitudineCompleta = $"{longitudineParteIntera}.{longitudineParteDecimale}";

    // Converte il valore completo in double
    if (double.TryParse(latitudineCompleta, NumberStyles.Float, CultureInfo.InvariantCulture, out double latitudine))
    {
        monument.Latitudine = latitudine;
    }

    if (double.TryParse(longitudineCompleta, NumberStyles.Float, CultureInfo.InvariantCulture, out double longitudine))
    {
        monument.Longitudine = longitudine;
    }

    if (_monumentCollection != null)
    {
        // Inserisce un nuovo monumento nel database
        _monumentCollection.InsertOne(monument);
        SendEmailToAdmin(monument);
    }
}

return RedirectToAction("Index");
}

```

Figura 25



```
[HttpPost]
public ActionResult Edit(Monument monument, string latitudineParteIntera, string latitudineParteDecimale, string longitudineParteIntera, string longitudineParteDecimale)
{
    string latitudineCompleta = $"{latitudineParteIntera}.{latitudineParteDecimale}";
    string longitudineCompleta = $"{longitudineParteIntera}.{longitudineParteDecimale}";

    // Converte il valore completo in double
    if (double.TryParse(latitudineCompleta, NumberStyles.Float, CultureInfo.InvariantCulture, out double latitudine))
    {
        monument.Latitudine = latitudine;
    }

    if (double.TryParse(longitudineCompleta, NumberStyles.Float, CultureInfo.InvariantCulture, out double longitudine))
    {
        monument.Longitudine = longitudine;
    }

    // Aggiorna l'utente nel database
    _monumentCollection.ReplaceOne(u => u.Id == monument.Id, monument);
    SendEmailToAdmin(monument);

    return RedirectToAction("Index");
}
```

Figura 26



## 4. Sperimentazione in campo

In questo capitolo vengono sperimentati alcuni casi d'uso al fine di testare e di dimostrare il corretto funzionamento del sistema. Questa parte di codice, ovvero l'interfaccia e l'api rest sono state caricate su GitHub al seguente link: <https://github.com/mCiccone23/monureg->.

### 4.1 Scenario d'uso: Monumento riconosciuto correttamente

Di seguito viene presentata un'illustrazione dettagliata dello scenario d'uso relativo al caso d'uso di riconoscimento dei monumenti, partendo dal caso d'uso relativo ai monumenti situati nelle immediate vicinanze.

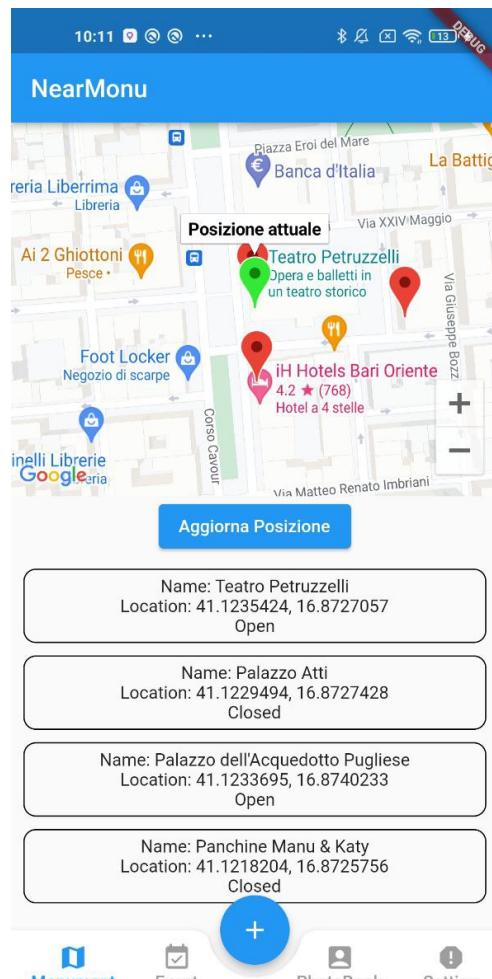


Figura 27



All'apertura dell'applicazione, la prima schermata visualizzata è dedicata alla visualizzazione dei monumenti nelle vicinanze. Questa pagina è composta da una mappa interattiva creata utilizzando le potenti API di Google Maps. Sotto la mappa, si trova un pulsante che consente di aggiornare la tua posizione attuale in tempo reale.

In questa schermata, si possono facilmente individuare e scoprire tutti i monumenti, palazzi e chiese che si trovano entro un determinato raggio dalla posizione attuale. Oltre alla loro posizione precisa sulla mappa, l'app fornisce anche informazioni cruciali, come se un monumento è aperto o chiuso al pubblico.

Si è scelto di utilizzare la tecnologia di Google Maps per questa funzionalità in quanto è notoriamente affidabile, incredibilmente veloce e precisa. Inoltre, offre già un vasto assortimento di dati integrati, il che rende l'esperienza dell'utente più ricca e informativa.

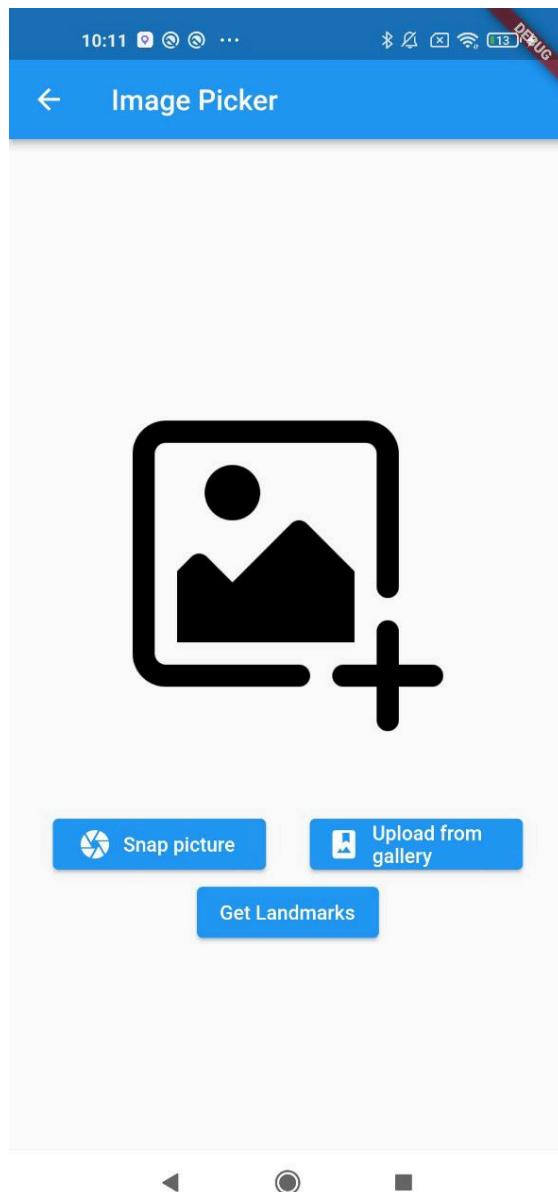
Tuttavia, è importante notare che questa tecnologia è soggetta a costi basati sull'utilizzo. Pertanto, in futuro, durante la fase di produzione o in base ai vincoli di budget e agli obiettivi specifici del progetto, si potrebbero considerare alternative tecnologiche che meglio si adattino alle esigenze di business.

Nella parte inferiore dell'applicazione, è presente una barra di navigazione che consente l'accesso a tutte le sezioni dell'applicazione in modo intuitivo e veloce. Attraverso questa barra, è possibile esplorare le diverse funzionalità dell'applicazione.

Per accedere alla sezione dedicata al riconoscimento dei monumenti, è necessario premere il pulsante (floating action button), rappresentato da un segno più (+). Questo pulsante si trova in una posizione strategica per garantire l'accesso rapido e immediato alla funzione di riconoscimento dei monumenti.

Tutti i widget sono stati scelti e implementati seguendo i principi del material design.





**Figura 28**

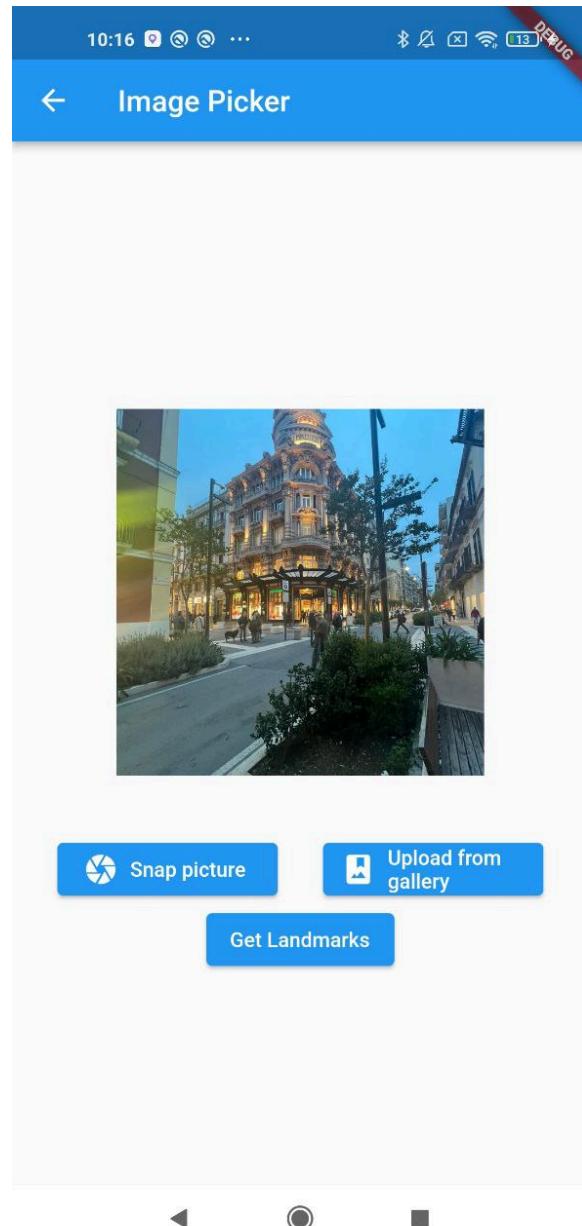
Premendo il pulsante a forma di più (+), come precedentemente descritto, si aprirà la schermata illustrata nella figura. All'interno di questa schermata, sono presenti vari pulsanti che consentono di svolgere diverse azioni. Tra questi, è possibile scattare una foto del monumento utilizzando l'opzione 'Scatta foto' (Snap picture), oppure selezionare un'immagine dalla galleria mediante 'Carica dalla galleria' (Upload from gallery). Successivamente, è possibile ottenere informazioni sui punti di interesse relativi all'immagine selezionata cliccando su 'Ottieni punti d'interesse' (Get Landmarks).

In questo contesto, stiamo scegliendo di caricare un'immagine dalla galleria, e nello specifico, la foto selezionata è la seguente.



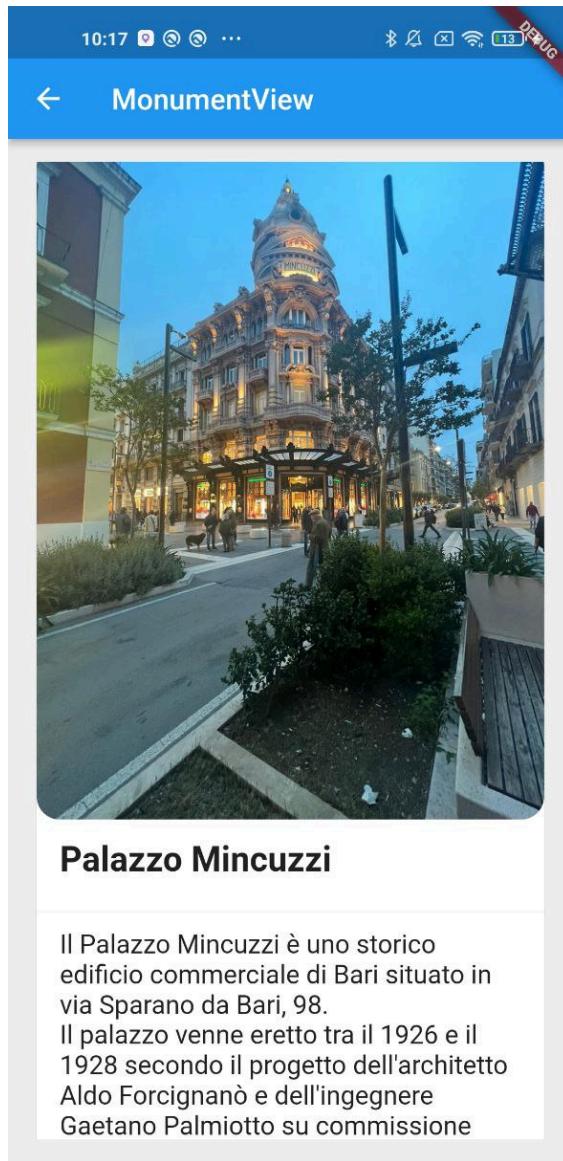
**Figura 29**

L'immagine selezionata ritrae il Palazzo Mincuzzi, un edificio storico situato a Bari. È evidente che la foto è stata catturata durante un momento della giornata prossimo al tramonto, con alcune sfocature ai lati. Questa scelta è stata fatta per mettere alla prova il sistema in diverse condizioni di luce e per valutarne la sua capacità di riconoscimento in situazioni diverse.



**Figura 30**

Questa è la schermata aggiornata dopo il caricamento dell'immagine. A questo punto è possibile premere su Get Landmarks e aspettare il risultato ottenuto.



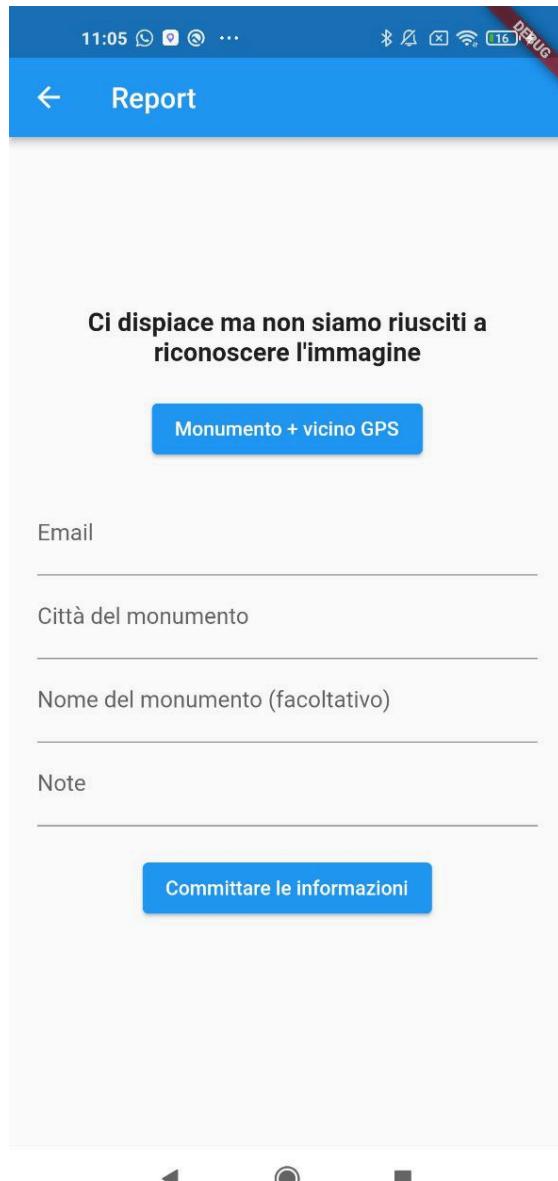
**Figura 31**

Come è possibile notare nella figura, in questo caso l'esito del riconoscimento è andato a buon fine. Nella schermata viene riproposta la foto scattata con il nome del monumento accompagnato da una breve descrizione.

## 4.2 Scenario d'uso: Monumento non riconosciuto

Uno scenario alternativo rispetto a quello precedente è il caso in cui non venga riconosciuto correttamente il punto d'interesse.

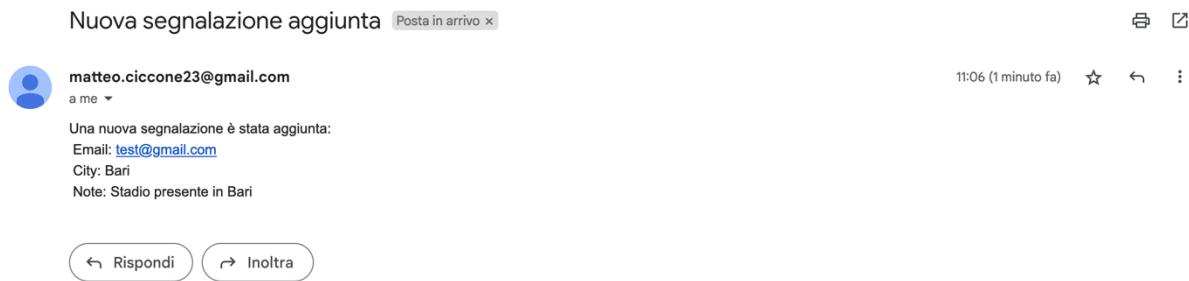


**Figura 32**

Questa schermata presenta due opzioni principali: la possibilità di ottenere informazioni sul monumento più vicino in base alla posizione attuale o di creare una segnalazione. Questo strumento può aiutare a chiarire l'oggetto di interesse che si stava cercando di riconoscere e contribuire a migliorare la precisione del sistema.

Dopo aver inserito le informazioni nei campi e confermato l'azione, i dati vengono memorizzati nel database e contemporaneamente viene inviata una notifica via email all'amministratore. Questo stesso processo di notifica via email si attiva anche quando un nuovo monumento viene aggiunto.



**Figura 33**

### 4.3 Lavori futuri

Gli obiettivi stabiliti in fase di progettazione, considerando i tempi e le risorse disponibili, sono stati completati con successo. Tuttavia, è importante sottolineare che il processo di miglioramento non avrà mai una conclusione definitiva all'interno di questa applicazione. Grazie alla sua struttura modulare, sarà sempre possibile scalare e perfezionare ciascun componente. Questo ci permetterà di rimanere all'avanguardia con l'evolversi delle tecnologie e garantire che il sistema sia sempre in continua evoluzione e miglioramento.

La prossima fase del lavoro prevede il completamento dell'implementazione delle funzionalità relative agli 'eventi vicini'. Questo richiederà l'utilizzo di altre tecniche di web scraping per acquisire informazioni da fonti come TripAdvisor, dove sono disponibili dettagli su tour, guide, itinerari e altre attività, o tramite l'utilizzo di API dedicate. Inoltre, sarà necessario implementare le funzionalità legate ai 'book fotografici', consentendo agli utenti di creare album di foto personalizzati. Oltre a queste nuove caratteristiche, verrà sviluppato un sistema di registrazione e accesso degli utenti (registrazione e log in) che sarà integrato nel database per garantire una gestione sicura dei dati degli utenti.

Un altro obiettivo di sviluppo consiste nella sostituzione del modello di riconoscimento attualmente utilizzato con uno personalizzato. A causa dell'uso continuo dell'app, il database cresce costantemente, diventando una risorsa di grande valore. La proposta è di implementare un modello incrementale (Incremental Online Learning) che possa migliorare con l'utilizzo dell'app e l'aggiunta di nuove immagini. Per questo scopo, all'interno del sistema, vengono memorizzate tutte le foto scattate con le



relative etichette, contribuendo così al miglioramento continuo del modello di riconoscimento.



## 5. CONCLUSIONI

---

Questa tesi illustra il processo di sviluppo di un prototipo di sistema software destinato a migliorare l'esperienza del turismo culturale. Il sistema offre agli utenti un'ampia gamma di funzionalità, che includono il riconoscimento istantaneo dei monumenti con accesso diretto alle loro descrizioni semplicemente attraverso l'inquadratura, la visualizzazione di tutti gli eventi e i luoghi di interesse nella zona e la possibilità di creare album fotografici per conservare i ricordi.

In conclusione, possiamo dire che gli obiettivi prefissati all'inizio di questo lavoro sono stati rispettati. Nei vari capitoli sono state dettagliate tutte le attività svolte e gli strumenti impiegati.

Nel capitolo introduttivo, sono stati definiti gli obiettivi e sono state esposte le problematiche e le necessità che hanno reso utile questo sistema.

Nel secondo capitolo, è stata condotta un'analisi dei principali concorrenti e dei punti di forza di Monureg.

Nel terzo capitolo, è stata presentata l'intera fase di progettazione, comprensiva dell'identificazione dei requisiti, dei casi d'uso e dell'architettura del sistema. Successivamente, ogni componente del sistema è stato analizzato e sono stati illustrati gli strumenti utilizzati a tale scopo.

Nel quarto capitolo, è stato effettuato un dettagliato esame del funzionamento dell'applicazione attraverso un caso d'uso completo, e sono state delineate le prospettive future e le possibili migliorie da apportare al sistema.



## RIFERIMENTI

---

- [1] <https://aws.amazon.com/it/what-is/flutter/>
- [2] <https://docs.flutter.dev/resources/architectural-overview>
- [3] <https://ieeexplore.ieee.org/abstract/document/8821809>
- [4] A study on Web Scraping – Niranjan Krishna
- [5] Web Scraping- Bo Zhao College of Earth, Ocean, and Atmospheric Sciences, Oregon State University, Corvallis, OR,USA.
- [6] Weyand T, Araujo A, Cao B, Sim J. Google Landmarks Dataset v2-A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. Link: <https://arxiv.org/abs/2004.01804>
- [7] FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets. Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeyonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, Minjoon Seo. Link: <https://arxiv.org/abs/2307.10928>
- [8] <https://learn.microsoft.com/it-it/aspnet/core/?view=aspnetcore-7.0>
- [9] E-Commerce Website Using ASP.NET MVC Framework – NOMULA MOHANSATISH, KALPANA
- [10] Exploring the Merits of NoSQL: A Study Based on MongoDB – Benymol Jose, Sajimon Abraham. Link: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8076778>

