Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Da un Set di Foto al Modello 3D: Utilizzare uno Smartphone come 3D Scanner attraverso la Fotogrammetria

Tesi di laurea triennale

Relatore

Prof. Massimiliano De Leoni

Laureando

Edoardo Pavan
matr. 1201117

Anno Accademico 2021-2022

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Edoardo Pavan presso l'azienda Zero12 S.R.L. di Padova.
L'obiettivo dello stage era la realizzazione di un sistema per generare un modello 3D, partendo da una semplice sequenza di foto catturate con uno smartphone. In particolare, durante lo svolgimento dello stage, sono stati richiesti: lo sviluppo di un'applicazione iOS, lo sviluppo di un'applicazione macOS e la creazione di un piatto rotante per la cattura automatica delle foto.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

L'azienda Zero12 S.R.L è una realtà nata nel 2012, specializzata nello sviluppo di applicazioni; basate su una piattaforma in cloud, per dispositivi mobili, tra cui smartphone e tablet. Zero12 è partner di *Amazon Web Services* e $\text{MongoDB}_{\text{G}}$, con i quali collabora per offrire soluzioni scalabili per l'analisi dei $\text{Big Data}_{\text{G}}$ in tempo reale. Zero12, inoltre, nel 2019 è stata acquisita dalla multinazionale VarGroup, partner per l'innovazione delle imprese e azienda leader nel panorama Information and Communication Technologies $(\text{ICT})_{\text{G}}$.

1.2 Lo stage

1.2.1 Il problema

Negli ultimi anni, i clienti dei negozi online sono diventati sempre più esigenti, dal momento che desiderano vedere il prodotto direttamente nelle proprie mani o nella propria casa prima di acquistarlo. Per questo motivo, molti e-commerce si sono innovati per dare la possibilità al cliente di vedere gli oggetti in 3D sullo schermo, oppure nella stanza tramite l'uso della realtà aumentata. Questa novità, da un lato ha contribuito a un incremento delle vendite dei prodotti, ma allo stesso tempo, anche ad un aumento esponenziale dei costi a carico dell'azienda. Infatti, fino a pochi anni fa scattare le foto espositive di un articolo aveva un costo limitato, al giorno d'oggi, elaborare un modello 3D può arrivare a costare centinaia o migliaia di euro per ogni articolo.

1.2.2 La soluzione proposta

Dopo aver analizzato il problema, l'azienda ha proposto uno stage per la creazione di un sistema per generare un modello 3D. Partendo da una semplice sequenza di foto, catturate con uno smartphone o una macchina fotografica, il sistema deve essere in grado di fare il rendering di un oggetto in 3D.
Per raggiungere tale obiettivo, ci si è appoggiati su *Object Capture*, una $\text{Application Program Interface}_{\text{G}}$ presentata da Apple alla conferenza sviluppatori

2021. Per sfruttare questa API, è stato necessario realizzare due applicazioni: una iOS, dedicata alla raccolta delle foto, arricchite dai dati di profondità derivati dal sensore Lidar$_G$, presente negli iPhone 12 Pro e successivi. L'altra applicazione da realizzare, doveva essere per il sistema operativo macOS, con lo scopo di realizzare il modello 3D, sfruttando le foto scattate con l'applicazione iOS.
Infine, insieme all'azienda, si è pensato di progettare un piatto rotante connesso all'applicazione iOS, in modo che la raccolta delle foto potesse avvenire in modalità automatica.

1.3 Organizzazione del testo

Il secondo capitolo descrive le modalità di svolgimento dello stage presso Zero12, spiegandone i relativi vincoli, gli obiettivi pianificati, le tecnologie e discutendo delle possibili problematiche che sarebbero potute emergere nel corso dello svolgimento del progetto.

Il terzo capitolo approfondisce, in modo dettagliato, i requisiti posti per lo sviluppo e i casi d'uso delle applicazioni realizzate, individuati nella fase di analisi del progetto di stage.

Il quarto capitolo tratta la progettazione delle applicazioni e del piatto rotante, descrivendone il funzionamento e i design pattern utilizzati. A seguire, viene esposta la codifica dell'intero progetto.

Il quinto capitolo approfondisce i test effettuati su alcuni oggetti reali con la tecnica della fotogrammetria$_G$, utilizzando il piatto rotante e l'acquisizione automatica.

Il sesto capitolo contiene una valutazione retrospettiva sull'attività di stage.

Nel testo sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati in questo elaborato, vengono definiti nel glossario, collocato alla fine del testo;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*$_G$;
* i termini in lingua straniera e che si riferiscono al gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione dello Stage

Il seguente capitolo descrive le modalità di svolgimento dello stage presso Zero12, spiegandone i relativi vincoli, gli obiettivi pianificati, le tecnologie e discutendo delle possibili problematiche che sarebbero potute emergere nel corso dello svolgimento del progetto.

2.1 Nel dettaglio

Lo stage presso Zero12 aveva come scopo la realizzazione di un sistema per la creazione di modelli 3D a partire da una sequenza di foto, arricchite da metadati posizionali. Per raggiungere l'obiettivo, è stato necessario realizzare due componenti:

1. **Applicazione iOS**: dedicata alla raccolta delle foto, arricchite dai dati di profondità; derivanti dal sensore Lidar presente negli iPhone 12 Pro e nei modelli successivi;

2. **Applicazione macOS**: dedicata alla creazione del modello 3D a partire dalle foto generate grazie all'app iOS. L'applicazione doveva prevedere la ricezione degli asset dall'app iOS e l'esportazione del modello 3D completato.

Lo schema in **Figura 2.1** riassume l'architettura del progetto di stage:

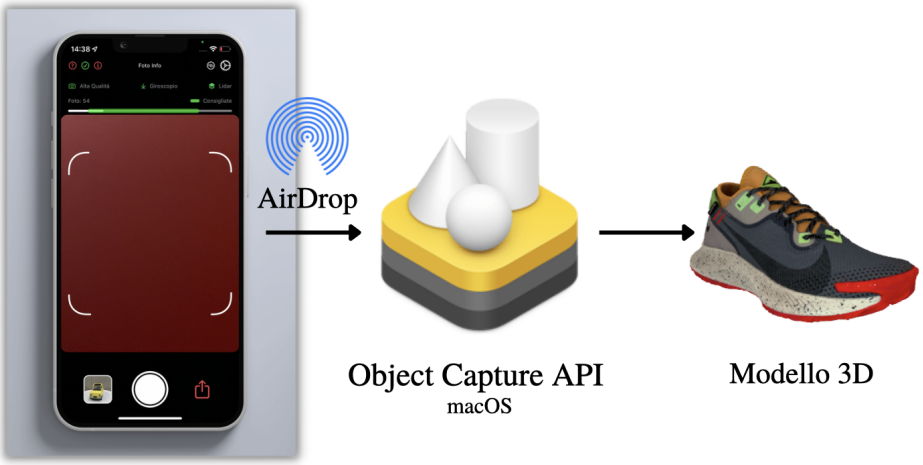

Figura 2.1: Schema riassuntivo dell'architettura del progetto di stage

Le foto vengono acquisite, in un numero variabile tra 20 e 200, tramite l'applicazione iOS; sia in modalità manuale, muovendosi fisicamente intorno all'oggetto, oppure, in automatico, attraverso l'utilizzo del piatto rotante. Successivamente, le immagini e i metadati vengono compressi in una cartella *ZIP*, che l'utente potrà trasferire, con *Airdrop* o via cavo, al Mac. Dall'applicazione macOS l'utente potrà generare il modello 3D attraverso le API *Object Capture* del framework *RealityKit*.

Per entrambe le applicazioni era stato posto dall'azienda il vincolo di dover utilizzare il framework *SwiftUI*, creato da Apple. Il progetto richiedeva, inoltre, la creazione di comunicazione tra le due applicazioni, da attuare tramite la funzione *Airdrop*, nativa in entrambi i sistemi operativi.
Per quanto riguarda il piatto rotante, l'unico vincolo era l'uso del Raspberry$_{\text{G}}$, con libertà di scelta sulla tecnologia per la comunicazione con l'app iOS e sui componenti elettronici da utilizzare per costruirlo.

2.2 Obiettivi del progetto

Il progetto di stage, inizialmente prevedeva lo sviluppo di due applicazioni, quella iOS per la cattura delle foto e quella macOS per l'elaborazione. Ciononostante, dato che l'applicazione di cattura delle foto era già stata sviluppata da Apple, non è stato necessario crearne una nuova, ma solo implementarne le funzionalità mancanti previste dai requisiti del progetto. Questo ha comportato un significativo risparmio di tempo e lo stage sarebbe finito in meno di 200 ore. Di conseguenza, dopo un'analisi con il tutor aziendale, gli obiettivi del progetto sono stati rivisti e si è deciso di creare un piatto rotante, per automatizzare l'acquisizione delle foto con lo smartphone. In base a questa rimodulazione, gli obiettivi del progetto sono diventati i seguenti:

- Studio e valutazione dell'applicazione iOS per la successiva modifica;

* Analisi e progettazione dell'applicazione macOS;
* Analisi dei motori passo-passo$_G$ presenti in commercio;
* Analisi dei driver$_G$ per il collegamento del motore al *Raspberry*;
* Analisi delle librerie per controllare i motori passo-passo;
* Analisi dei vari sistemi di comunicazione, come il Bluetooth Low Energy$_G$ e *AWS IoT*;
* Sviluppo/Codifica del progetto seguendo le norme aziendali;
* Redazione di una guida dettagliata per replicare il piatto rotante;
* Creazione di un'icona per le applicazioni.

2.3 Analisi preventiva dei rischi

In questa sezione vengono analizzati i rischi individuati durante la fase iniziale di analisi, a cui si sarebbe potuto andare incontro durante lo sviluppo del progetto. Per ogni rischio vengono riportate le soluzioni possibili per poterli affrontare.

1. Inesperienza tecnologica
Descrizione: le tecnologie da utilizzare erano nuove, il che poteva portare a ritardi nello sviluppo del progetto.
Soluzione: è stato organizzato un periodo iniziale di formazione individuale sulle tecnologie da utilizzare per lo sviluppo. Il tutor aziendale è sempre stato presente per qualsiasi chiarimento.

2. Problematiche hardware
Descrizione: il computer in cui sviluppare poteva avere malfunzionamenti o non essere compatibile con tutte le API di Apple, questo poteva essere fonte di gravi ritardi.
Soluzione: sono stati creati due repository su *CodeCommit* in cui caricare il codice e l'azienda ha messo a disposizione un *MacBook Pro* sostitutivo in caso di malfunzionamento di quello personale.

3. Sistemi esterni
Descrizione: il progetto richiedeva di interfacciarsi con sistemi esterni, tra cui *AWS IoT* o il *Bluetooth Low Energy*, che potevano avere una documentazione non esaustiva o presentare bug.
Soluzione: con il tutor interno si sono ipotizzate più soluzioni per realizzare la medesima funzionalità con tecnologie differenti.

4. Impegni del tutor aziendale
Descrizione: è possibile che il tutor aziendale sia impegnato nel suo lavoro e

non sia sempre disponibile per chiarimenti.
Soluzione: nei giorni di assenza del tutor aziendale sono state pianificate più attività; in modo che qualora si riscontrassero problemi con una, si potesse continuare ugualmente il progetto con qualcos'altro.

5. Interpretazione dei requisiti
Descrizione: dopo una prima analisi dei requisiti, è possibile che si evidenzi la necessità di modificare o aggiungere nuovi requisiti al progetto iniziale.
Soluzione: almeno una volta a settimana è stato fatto uno *stand up* con il tutor aziendale. Successivamente, si è proceduto alla valutazione delle modifiche e sono state delineate nuove modalità di procedere per evitare un eventuale rallentamento nello sviluppo.

2.4 Pianificazione

Di seguito sono elencate le varie fasi previste per lo stage, in ordine cronologico:

1. **Prima settimana**:
    * presentazione e introduzione al progetto;
    * installazione dell'ambiente di sviluppo e creazione dei certificati necessari per provare l'applicazione nell'iPhone;
    * formazione sul framework *SwiftUI* (studio individuale);
    * analisi delle funzionalità da aggiungere o modificare alll'applicazione iOS;
    * codifica delle funzionalità individuate.
2. **Seconda settimana**:
    * analisi e progettazione dell'applicazione macOS;
    * inizio della codifica dell'applicazione.
3. **Terza settimana**:
    * completamento della codifica dell'applicazione macOS;
    * inizio dei primi test dell'intero processo di cattura ed elaborazione.
4. **Quarta settimana**:
    * studio delle tecnologie e dei componenti necessari per realizzare il piatto rotante;
    * acquisto dei componenti e inizio della codifica del sistema di comunicazione tra *Raspberry* e l'app iOS.
5. **Quinta settimana**:

    * completamento della codifica del sistema di comunicazione tra *Raspberry* e l'app iOS.

6. **Sesta settimana**:

    * progettazione della User Interface $(UI)_G$ per il controllo del piatto rotante;
    * codifica della UI e connessione con la logica.

7. **Settima settimana**:

    * continuazione dei test di scansione con l'uso del piatto rotante;
    * presentazione del prodotto sviluppato all'Amministratore Delegato dell'azienda;
    * implementazione di nuove funzionalità nell'anteprima del modello all'interno dell'app macOS.

8. **Ottava settimana**:

    * continuazione dei test di scansione con l'uso del piatto rotante;
    * realizzazione dell'icona delle applicazioni;
    * presentazione del prodotto sviluppato a tutta l'azienda.

2.5 Strumenti utilizzati

Per la realizzazione del progetto si è adoperato il *MacBook Pro* personale con sistema operativo *macOS Monterey v12.4* e un *Raspberry Pi 3*, fornitomi dall'azienda.

2.5.1 Sviluppo delle applicazioni

Per lo sviluppo delle applicazioni sono stati utilizzati:

2.5.1.1 Xcode

Xcode è un Integrated Development Environment $(IDE)_G$ sviluppato, e mantenuto da Apple. Si tratta di un programma utile allo sviluppo di software per i sistemi macOS, iOS, iPadOS, watchOS e tvOS. È uno strumento che consente la scrittura di applicazioni con il linguaggio *Swift*.

2.5.1.2 Versionamento

Per il versionamento del codice è stato utilizzato *Git*, con cui sono stati creati due repository: uno per l'applicazione mobile e l'altro per quella desktop. Entrambi i repository sono ospitati su *CodeCommit*, il servizio di versionamento offerto da *AWS*.

2.5.1.3 SourceTree

SourceTree permette di lavorare con *Git* attraverso un'interfaccia grafica semplice e funzionale.

2.5.2 Progettazione delle applicazioni

Per la progettazione dell'interfaccia dell'applicazione macOS e per la creazione dell'icona delle app si è utilizato:

2.5.2.1 Sketch

Sketch è uno strumento di grafica vettoriale$_G$ per la progettazione della UI di un'applicazione o di un logo. Ha un'interfaccia semplice e immediata, che consente a chiunque di poterla utilizzare senza difficoltà.

2.5.3 Sviluppo del piatto rotante

Per la scrittura dello script del piatto rotante sono stati utilizzati:

2.5.3.1 Visual Studio Code

Visual Studio Code è un editor per lo sviluppo in qualsiasi linguaggio di programmazione, grazie alla moltitudine di estensioni disponibili. In particolare, ci si è avvalsi dell'estensione *Remote-SSH* per la connessione tra il computer e il *Raspberry* tramite il protocollo SSH.

2.5.3.2 Motore Nema17

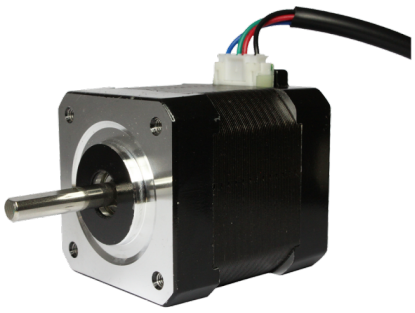

Figura 2.2: Motore Nema17

Per mettere in movimento il piatto, è stato utilizzato un motore passo-passo *Nema17*, raffigurato in **Figura 2.2**. *Nema* è uno standard che identifica la dimensione del quadrato frontale da cui fuoriesce l'albero del motore. In questo caso, il numero 17 indica che i lati del quadrato sono di 1.7", ovvero 42mm. La lunghezza del motore, da cui dipende la massima coppia erogata, non segue uno standard.

2.5.3.3 Driver A4988

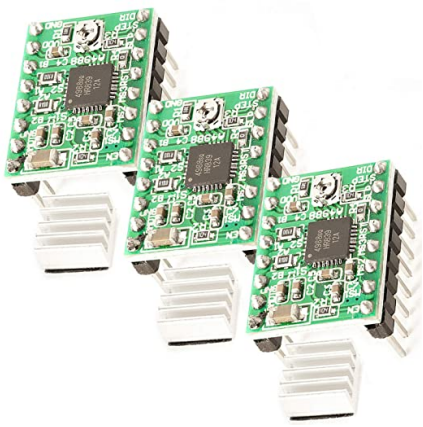

Figura 2.3: Driver A4988

Per pilotare il motore passo-passo con il *Raspberry*, è stato necessario l'utilizzo di un driver, ovvero di una piccola scheda elettronica, come quella in **Figura 2.3**, con il compito di tradurre i messaggi in ingresso, in movimento fisico del motore.

2.6 Tecnologie utilizzate

Per lo sviluppo del progetto sono state utilizzate le seguenti tecnologie:

2.6.1 Sviluppo iOS e macOS

2.6.1.1 Swift

Swift è un linguaggio di programmazione Open Source$_G$ presentato e creato da Apple per facilitare lo sviluppo di app per iOS, iPadOS, watchOS e tvOS.

2.6.1.2 SwiftUI

SwiftUI è un framework, presentato da Apple nel 2019; per semplificare la creazione di interfacce grafiche per dispositivi Apple; utilizzando il linguaggio *Swift* e il pattern Model-View-ViewModel. Si è fatto uso di questo framework per sviluppare le applicazioni richieste dall'azienda.

2.6.1.3 RealityKit

RealityKit è un framework creato, da Apple, per lo sviluppo di applicazioni in realtà aumentata. Di questo framework sono state impiegate la API *Object Capture*, presentate alla Worldwide Developers Conference (WWDC2021)$_G$, che dispongono delle seguenti impostazioni:

* **Qualità del modello**: *RealityKit* supporta fino a 5 livelli di dettaglio per un modello, rispettivamente:
  - **Preview**: è il livello base. Serve per vedere un'anteprima del modello, impiegando il minor tempo possibile. Contiene solo la *texture map*$_{\text{G}}$ base e quella normale;
  - **Reduced**: è ideale per il web. Contiene la *texture map* base, la normale e l'occlusione ambientale;
  - **Medium**: è adatto per applicazioni complesse. Contiene le medesime *texture map* del livello *Reduced*;
  - **Full**: è la massima qualità possibile. Contiene tutte le *texture map*, ovvero: base, normale, occlusione ambientale e rugosità.
  - **Raw**: è ideale solo per la post-produzione e non è adatto per la realtà aumentata.
* **Ordinamento delle foto**: per impostazione predefinita, *RealityKit* presuppone che le foto non siano in ordine. Se le immagini adiacenti sono una dopo l'altra, impostando l'ordinamento su "Sequenziale", le prestazioni di elaborazione del modello miglioreranno. Questa impostazione non ha alcun effetto sulla qualità del modello generato.
* **Feature Sensitivity**: impostando la sensibilità su "Alta", *RealityKit* cercherà i punti di riferimento tra le immagini utilizzando un algoritmo che analizza in modo più approfondito un'immagine. Questo algoritmo richiede più tempo per l'elaborazione modello 3D.

2.6.1.4 SceneKit

SceneKit è un framework grafico, sviluppato da Apple, per creare scene ed effetti 3D nelle applicazioni. È stato sfruttato questo framework, per realizzare l'anteprima del modello nell'applicazione macOS.

2.6.1.5 AirDrop

AirDrop è un servizio proprietario per il trasferimento di uno o più file tra due dispositivi Apple. AirDrop è stato impiegato per il trasferimento delle foto dall'applicazione iOS al Mac.

2.6.2 Piatto rotante

2.6.2.1 Python

Python è un linguaggio di programmazione interpretato, orientato agli oggetti, noto per la sua chiarezza e flessibilità. Python è stato utilizzato per sviluppare lo script del piatto rotante.

2.6.2.2 MQTT

MQTT è un protocollo molto leggero, sviluppato per i dispositivi Internet of Things$_G$ (IoT). Si basa sul pattern publish-subscribe$_G$ e richiede pochissima banda ed energia.

2.6.2.3 AWS IoT

AWS IoT Core è un servizio di AWS, che permette di stabilire una comunicazione, attraverso il protocollo MQTT, tra diversi dispositivi fisici o tra un dispositivo e i servizi di AWS.

2.6.2.4 RpiMotorLib

RpiMotorLib è una libreria Python 3 open source, che permette di controllare una vasta gamma di motori. È stata utilizzata per controllare il motore passo-passo *Nema17* del piatto rotante.

Capitolo 3

Analisi dei requisiti

In questo capitolo vengono descritti, in modo dettagliato, i requisiti posti per lo sviluppo e i casi d'uso delle applicazioni, individuati durante la fase di analisi del progetto di stage.

3.1 Individuazione delle funzionalità

Prima della codifica delle applicazioni, ci si è concentrati nell'analisi delle funzionalità richieste dall'azienda. Grazie a questa analisi e ad un confronto con il tutor interno, sono state individuate due tipologie di attori che andranno ad utilizzare l'applicazione iOS e una tipologia per l'applicazione macOS.

3.2 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto, sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono di tipo Unified Modeling Language $(\text{UML})_{\text{G}}$; servono per descrivere delle funzioni o dei servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.
I casi d'uso dell'applicazione iOS descrivono solo le funzionalità che dovranno essere implementate, tralasciando quelle già presenti.
Ogni caso d'uso ha un codice gerarchico ed univoco che lo identifica, nella forma:

Per l'applicazione macOS:
UCM<CodicePadre>.<CodiceFiglio>

Per l'applicazione iOS:
UCI<CodicePadre>.<CodiceFiglio>

I casi d'uso con la lettera "E" rappresentano i messaggi di errore o di avvertimento.

3.2.1 Attori

Gli attori individuati per l'applicazione iOS sono i seguenti:

Figura 3.1: Utenti applicazione iOS

Nell'applicazione iOS non tutte le funzionalità sono disponibili per tutti gli utenti, alcune richiedono che l'utilizzatore disponga del piatto rotante per poter essere utilizzate.
Per l'applicazione macOS, invece, è stato individuato un solo utente.

3.2.2 Applicazione macOS

3.2.2.1 UCM1 - Selezione assets

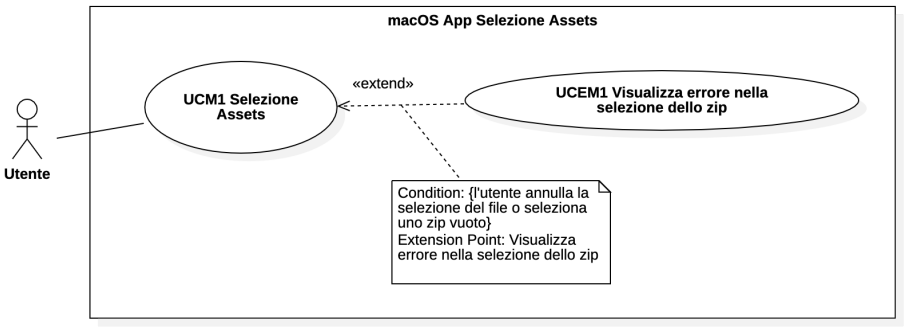

Figura 3.2: Use Case - UCM1: Selezione Assets

* **Descrizione**: L'utente visualizza un bottone per la selezione degli assets.
* **Attore primario**: Utente.
* **Precondizione**: L'utente si trova all'interno dell'applicazione.
* **Postcondizione**: L'utente visualizza a schermo le impostazioni dell'API *Object Capture*.
* **Scenario principale**:
    1. L'utente si trova all'interno dell'applicazione;

2. L'utente clicca sul pulsante per la selezione degli assets e sceglie il file zip esportato dall'applicazione iOS.

* **Estensioni:** Nel caso in cui l'utente annulli la selezione del file zip:

1. L'utente si trova all'interno dell'applicazione;
2. L'utente clicca sul pulsante per la selezione degli assets;
3. L'utente non trova il file e annulla la selezione;
4. Viene mostrato un messaggio d'errore(UCEM1 §3.2.3).

3.2.2.2 UCM2 - Modifica impostazioni

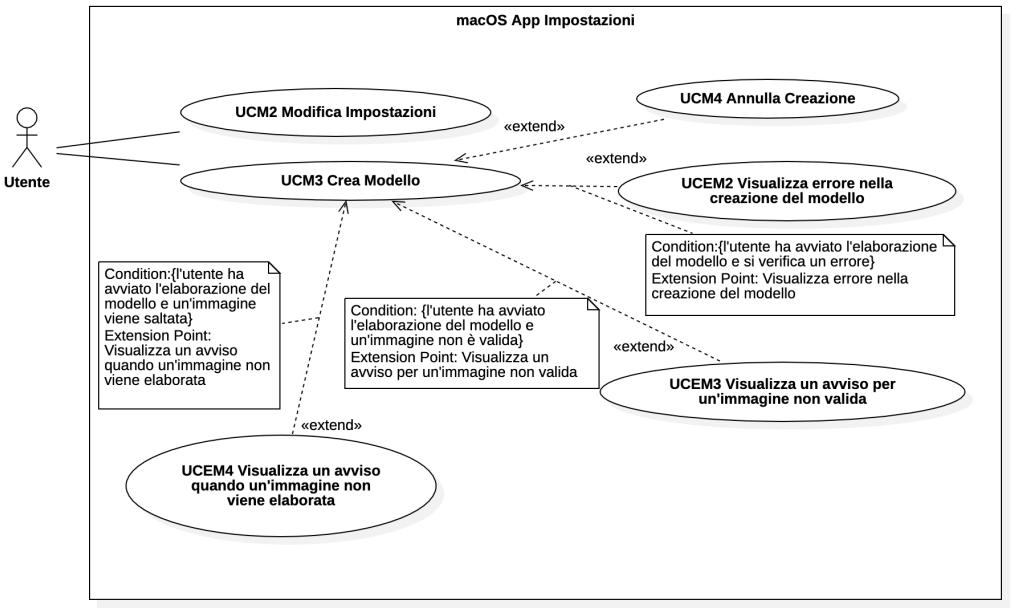

Figura 3.3: Use Case - UCM2 e UCM3: Impostazioni

* **Descrizione**: L'utente visualizza a schermo le impostazioni dell'API *Object Capture*.

* **Attore primario**: Utente.

* **Precondizione**: L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni.

* **Postcondizione**: Le impostazioni vengono salvate.

* **Scenario principale**:

1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
2. L'utente sceglie le impostazioni per la creazione del modello.

* **Sottocasi:**

    1. UCM2.1 Qualità del modello;
    2. UCM2.2 Ordinamento delle foto;
    3. UCM2.3 Feature Sensitivity.

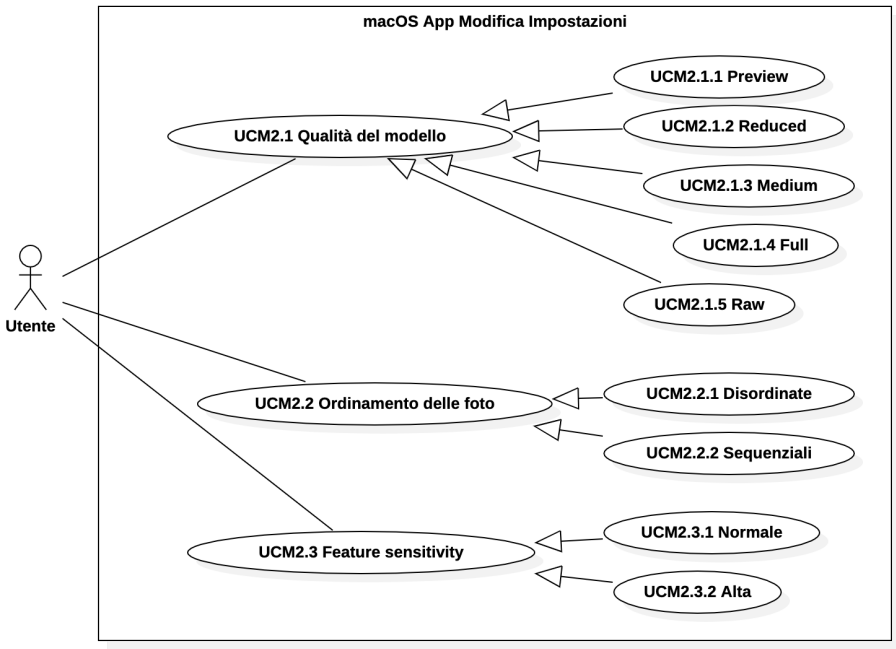

Figura 3.4: Use Case - Sottocasi UCM2

UCM2.1 - Qualità del modello

* **Descrizione**: L'utente sceglie la qualità del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità del modello.

* **Sottocasi:**

    1. UCM2.1.1 Preview;

2. UCM2.1.2 Reduced;
3. UCM2.1.3 Medium;
4. UCM2.1.4 Full;
5. UCM2.1.5 Raw.

UCM2.1.1 - Preview

* **Descrizione**: L'utente sceglie la qualità *preview*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità *preview*.

UCM2.1.2 - Reduced

* **Descrizione**: L'utente sceglie la qualità *reduced*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità *reduced*.

UCM2.1.3 - Medium

* **Descrizione**: L'utente sceglie la qualità *medium*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.

* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità *medium*.

UCM2.1.4 - Full

* **Descrizione**: L'utente sceglie la qualità *full*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità *full*.

UCM2.1.5 - Raw

* **Descrizione**: L'utente sceglie la qualità *raw*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la qualità *raw*.

UCM2.2 - Ordinamento delle foto

* **Descrizione**: L'utente sceglie l'ordinamento delle foto.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.

* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie l'ordinamento delle foto.

* **Sottocasi:**

    1. UCM2.2.1 Disordinate;
    2. UCM2.2.2 Sequenziali.

UCM2.2.1 - Disordinate

* **Descrizione**: L'utente sceglie l'ordinamento delle foto: *disordinate*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie l'ordinamento delle foto: *disordinate*.

UCM2.2.2 - Sequenziali

* **Descrizione**: L'utente sceglie l'ordinamento delle foto: *sequenziali*.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:

    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie l'ordinamento delle foto: *sequenziali*.

UCM2.3 - Feature Sensitivity

* **Descrizione**: L'utente sceglie la precisione dell'algoritmo di riconoscimento dei punti di riferimento nelle foto.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la precisione dell'algoritmo di riconoscimento dei punti di riferimento.
* **Sottocasi:**
    1. UCM2.3.1 Normale;
    2. UCM2.3.2 Alta.

UCM2.3.1 - Normale

* **Descrizione**: L'utente sceglie la precisione *normale* dell'algoritmo di riconoscimento dei punti di riferimento nelle foto.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la precisione *normale* dell'algoritmo di riconoscimento dei punti di riferimento.

UCM2.3.2 - Alta

* **Descrizione**: L'utente sceglie la precisione *alta* dell'algoritmo di riconoscimento dei punti di riferimento nelle foto.
* **Attore primario**: Utente.

* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente sceglie la precisione *alta* dell'algoritmo di riconoscimento dei punti di riferimento.

3.2.2.3 UCM3 - Crea modello

* **Descrizione**: L'utente dopo aver deciso le impostazioni crea il modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha scelto la cartella con gli assets e sta visualizzando le impostazioni.
* **Postcondizione**: L'utente visualizza l'anteprima del modello.
* **Scenario principale**:
    1. L'utente ha selezionato la cartella con gli assets e sta visualizzando le impostazioni;
    2. L'utente clicca il pulsante per creare il modello.
* **Estensioni:**
    1. Nel caso si verifihi un errore nella creazione del modello:
        (a) L'utente ha avviato l'elaborazione del modello;
        (b) L'algoritmo solleva un errore e interrompe l'elaborazione;
        (c) Viene mostrato un messaggio d'errore(UCEM2 §3.2.3)
    2. Nel caso in cui un'immagine non sia valida:
        (a) L'utente ha avviato l'elaborazione del modello;
        (b) L'algoritmo trova un'immagine non valida;
        (c) Viene mostrato un messaggio di errore con il motivo(UCEM3 §3.2.4)
    3. Nel caso in cui un'immagine venga saltata dall'elaborazione:
        (a) L'utente ha avviato l'elaborazione del modello;
        (b) L'algoritmo salta un'immagine;
        (c) Viene mostrato un messaggio di errore con il nome dell'immagine saltata(UCEM4 §3.2.5)

3.2.2.4 UCM4 - Annulla creazione

* **Descrizione**: L'utente annulla la creazione del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha iniziato la creazione del modello.
* **Postcondizione**: La creazione del modello viene annullata.
* **Scenario principale**:
    1. L'utente, dopo aver iniziato creazione del modello, decide di annullarla;
    2. La creazione del modello viene annullata.

3.2.2.5 UCM5 - Visualizza anteprima

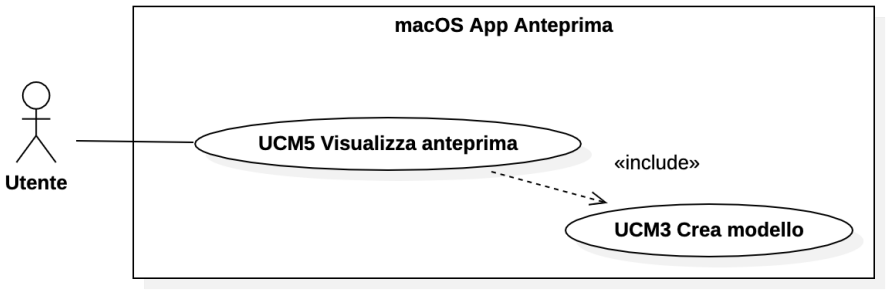

Figura 3.5: Use Case - UCM5: Visualizza anteprima

* **Descrizione**: L'utente ha generato il file usdz con successo e ora può vedere un'anteprima di questo modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: L'utente visualizza l'anteprima.
* **Scenario principale**:
    1. L'utente ha generato il modello usdz con successo;
    2. L'utente visualizza l'anteprima del modello.

3.2.2.6 UCM6 - Cattura Screenshot

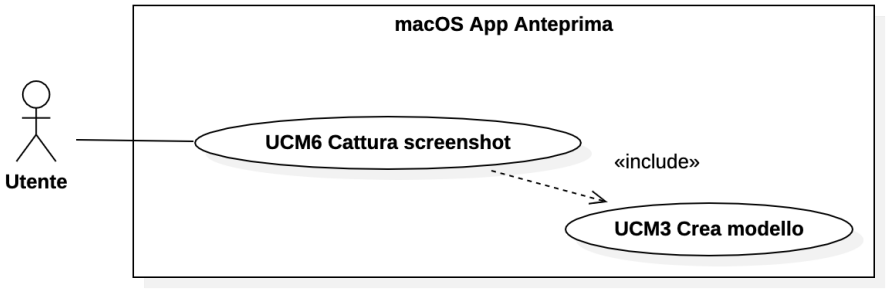

Figura 3.6: Use Case - UCM6: Cattura screenshot

* **Descrizione**: L'utente cattura uno screenshot dell'anteprima.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Lo screenshot viene salvato.
* **Scenario principale**:
    1. L'utente clicca nel pulsante per cattura uno screenshot;
    2. Lo screenshot viene salvato.

3.2.2.7 UCM7 - Modifica colore di sfondo

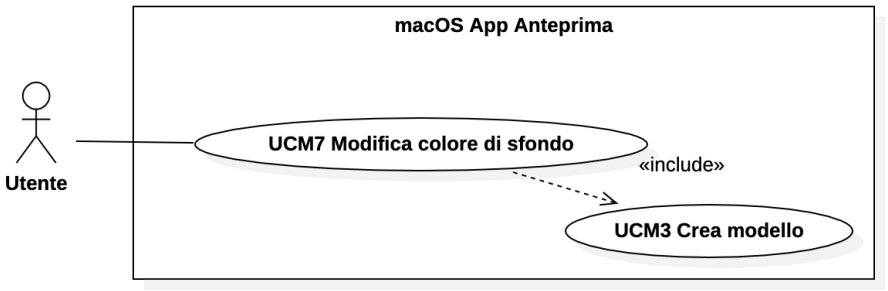

Figura 3.7: Use Case - UCM7: Modifica colore di sfondo

* **Descrizione**: L'utente modifica il colore di sfondo dell'anteprima.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il colore viene applicato.

* **Scenario principale**:

    1. L'utente clicca nel pulsante per cambiare lo sfondo;
    2. Sceglie il colore;
    3. Lo sfondo viene applicato.

3.2.2.8 UCM8 - Modifica luminosità

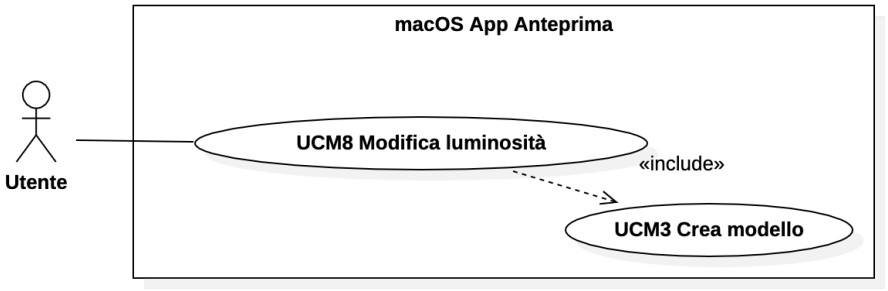

Figura 3.8: Use Case - UCM8: Modifica luminosità

* **Descrizione**: L'utente modifica la luminosità del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: La luminosità viene modificata.
* **Scenario principale**:

    1. L'utente modifica il valore della luminosità;
    2. La luminosità scelta viene applicata.

3.2.2.9 UCM9 - Salva modello

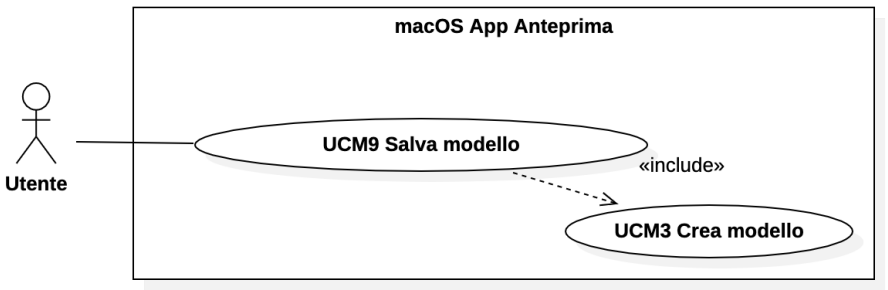

Figura 3.9: Use Case - UCM9: Salva modello

* **Descrizione**: L'utente esporta il modello nel formato scelto con le relative modifiche visualizzate nell'anteprima.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il modello viene salvato nella cartella preselta.
* **Scenario principale**:
    1. L'utente sceglie di esportare il modello generato;
    2. Il modello viene salvato.
* **Sottocasi**:
    1. UCM9.1 Salva modello in USDZ;
    2. UCM9.2 Salva modello in OBJ;
    3. UCM9.3 Salva modello in DAE;
    4. UCM9.4 Salva modello in STL.

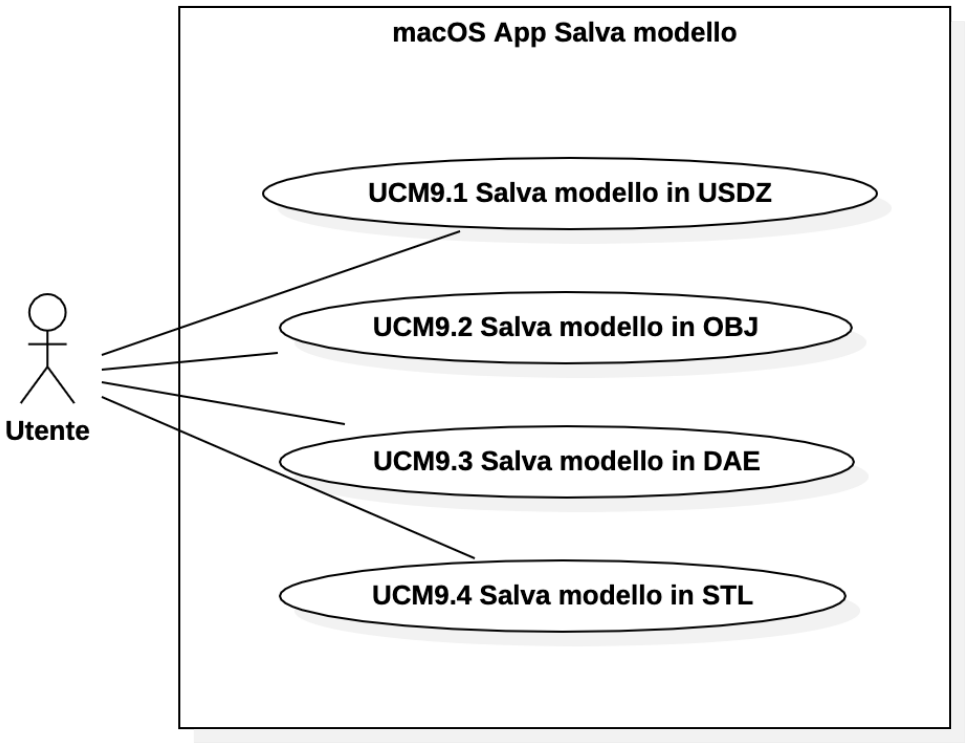

Figura 3.10: Use Case - Sottocasi UCM9

UCM9.1 - Salva modello in USDZ

* **Descrizione**: L'utente esporta il modello nel formato USDZ_{G}.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il modello in formato usdz viene salvato nella cartella preselta.
* **Scenario principale**:
    1. L'utente sceglie di salvare il modello nel formato usdz;
    2. Il modello viene salvato.

UCM9.2 - Salva modello in OBJ

* **Descrizione**: L'utente esporta il modello nel formato Obj_{G}.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il modello in formato obj viene salvato nella cartella preselta.
* **Scenario principale**:
    1. L'utente sceglie di salvare il modello nel formato obj;
    2. Il modello viene salvato.

UCM9.3 - Salva modello in DAE

* **Descrizione**: L'utente esporta il modello nel formato Dae_{G}.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il modello in formato dae viene salvato nella cartella preselta.
* **Scenario principale**:
    1. L'utente sceglie di salvare il modello nel formato dae;
    2. Il modello viene salvato.

UCM9.4 - Salva modello in STL

* **Descrizione**: L'utente esporta il modello nel formato Stl_{G}.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha generato il modello usdz.
* **Postcondizione**: Il modello in formato dae viene salvato nella cartella preselta.
* **Scenario principale**:
    1. L'utente sceglie di salvare il modello nel formato stl;
    2. Il modello viene salvato.

3.2.2.10 UCM10 - Ruota il modello

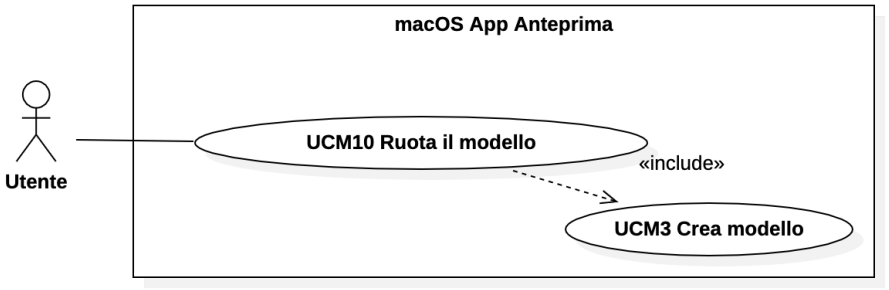

Figura 3.11: Use Case - UCM10: Ruota il modello

* **Descrizione**: L'utente ruota il modello nei 3 assi: X,Y,Z.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima.
* **Postcondizione**: Il modello viene ruotato.
* **Scenario principale**:
    1. L'utente sceglie di quanto ruotare il modello e in quale asse;
    2. Il modello viene ruotato.

3.2.2.11 UCM11 - Scala il modello

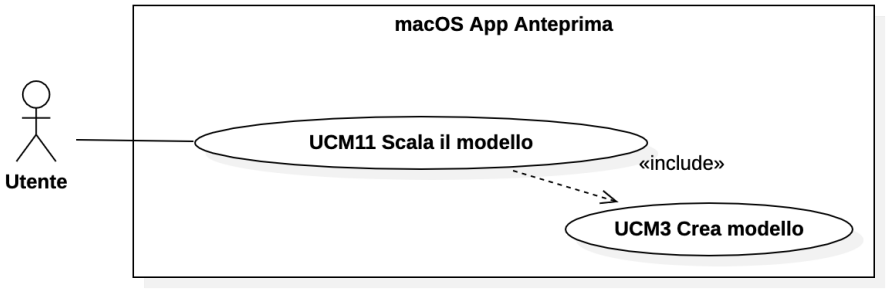

Figura 3.12: Use Case - UCM11: Scala il modello

* **Descrizione**: L'utente scala il modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima.
* **Postcondizione**: Il modello viene scalato.
* **Scenario principale**:
  1. L'utente sceglie di quanto scalare il modello;
  2. Il modello viene scalato.

3.2.2.12 UCM12 - Modifica la trama base

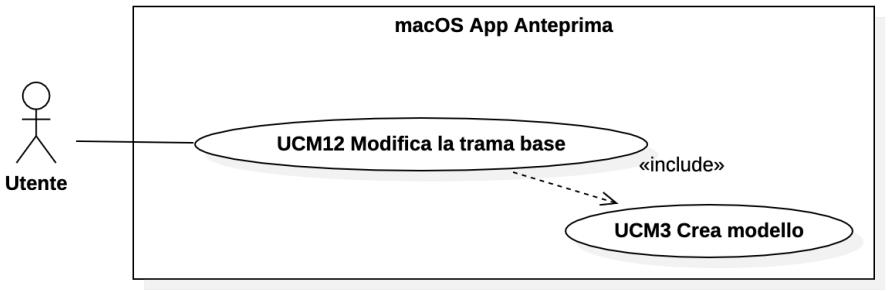

Figura 3.13: Use Case - UCM12 - Modifica la trama base

* **Descrizione**: L'utente modifica la trama base del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima.
* **Postcondizione**: La trama base viene modificata.

* **Scenario principale**:

    1. L'utente sceglie se impostare un colore o un'immagine;
    2. L'utente sceglie il colore o l'immagine;
    3. La trama base viene cambiata.

3.2.2.13 UCM13 - Modifica la trama normale

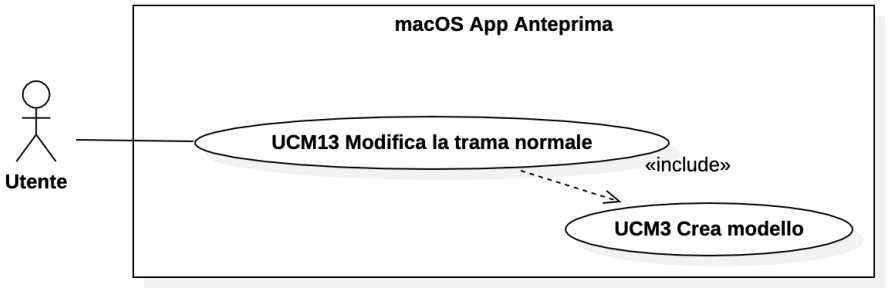

Figura 3.14: Use Case - UCM13 - Modifica la trama normale

* **Descrizione**: L'utente modifica la trama normale.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima e ha generato il modello con la qualità reduced, medium o full.
* **Postcondizione**: La trama normale viene modificata.
* **Scenario principale**:

    1. L'utente sceglie se impostare un colore o un'immagine;
    2. L'utente sceglie il colore o l'immagine;
    3. La trama normale viene cambiata.

3.2.2.14 UCM14 - Modifica la trama ruvidezza

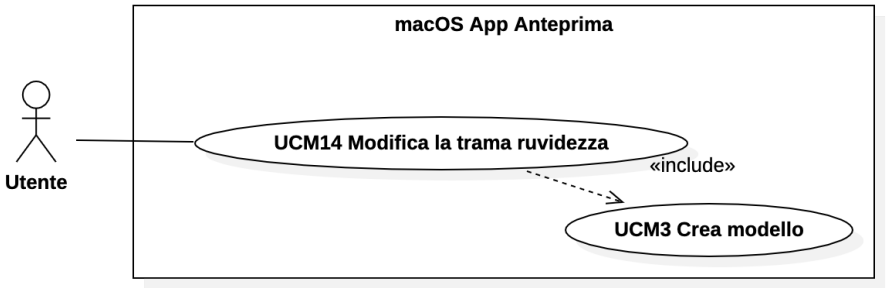

Figura 3.15: Use Case - UCM14 - Modifica la trama ruvidezza

* **Descrizione**: L'utente modifica la trama ruvidezza.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima e ha generato il modello con la qualità full.
* **Postcondizione**: La trama ruvidezza viene modificata.
* **Scenario principale**:
    1. L'utente sceglie se impostare un colore o un'immagine;
    2. L'utente sceglie il colore o l'immagine;
    3. La trama ruvidezza viene cambiata.

3.2.2.15 UCM15 - Modifica la trama occlusione

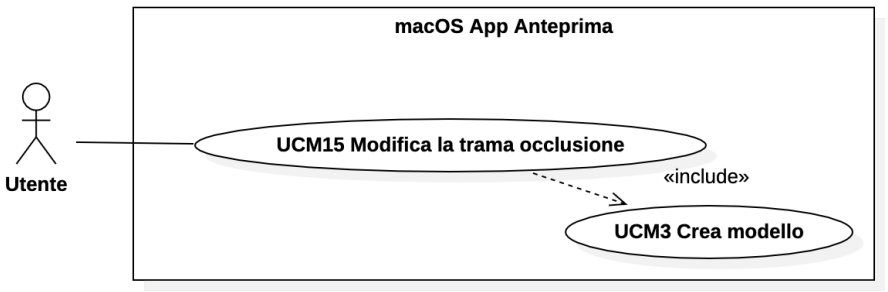

Figura 3.16: Use Case - UCM15 - Modifica la trama occlusione

* **Descrizione**: L'utente modifica la trama occlusione.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima e ha generato il modello con la qualità reduced, medium o full.
* **Postcondizione**: La trama occlusione viene modificata.
* **Scenario principale**:
    1. L'utente sceglie se impostare un colore o un'immagine;
    2. L'utente sceglie il colore o l'immagine;
    3. La trama occlusione viene cambiata.

3.2.2.16 UCM16 - Ripristina la trama base

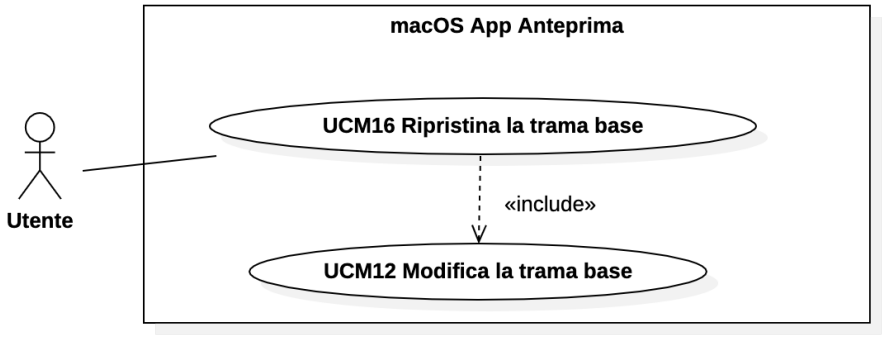

Figura 3.17: Use Case - UCM16 - Ripristina la trama base

* **Descrizione**: L'utente ripristina la trama base originale del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha cambiato la trama base.
* **Postcondizione**: La trama base originale viene ripristinata.
* **Scenario principale**:
    1. L'utente sceglie di ripristinare la trama base originale;
    2. La trama base originale viene ripristinata.

3.2.2.17 UCM17 - Ripristina la trama normale

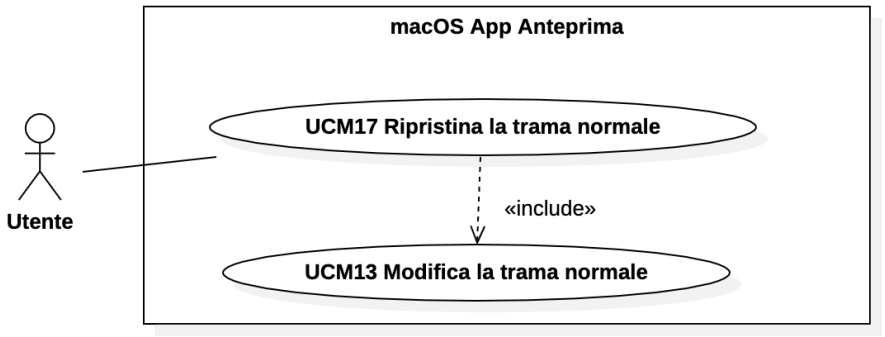

Figura 3.18: Use Case - UCM17 - Ripristina la trama normale

* **Descrizione**: L'utente ripristina la trama normale originale del modello.
* **Attore primario**: Utente.

* **Precondizione**: L'utente ha cambiato la trama normale.
* **Postcondizione**: La trama normale originale viene ripristinata.
* **Scenario principale**:
    1. L'utente sceglie di ripristinare la trama normale originale;
    2. La trama normale originale viene ripristinata.

3.2.2.18 UCM18 - Ripristina la trama ruvidezza

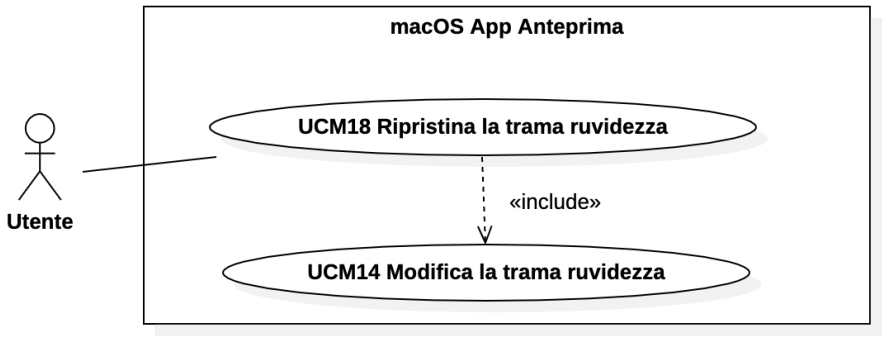

Figura 3.19: Use Case - UCM18 - Ripristina la trama ruvidezza

* **Descrizione**: L'utente ripristina la trama ruvidezza originale del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha cambiato la trama ruvidezza.
* **Postcondizione**: La trama ruvidezza originale viene ripristinata.
* **Scenario principale**:
    1. L'utente sceglie di ripristinare la trama ruvidezza originale;
    2. La trama ruvidezza originale viene ripristinata.

3.2.2.19 UCM19 - Ripristina la trama occlusione

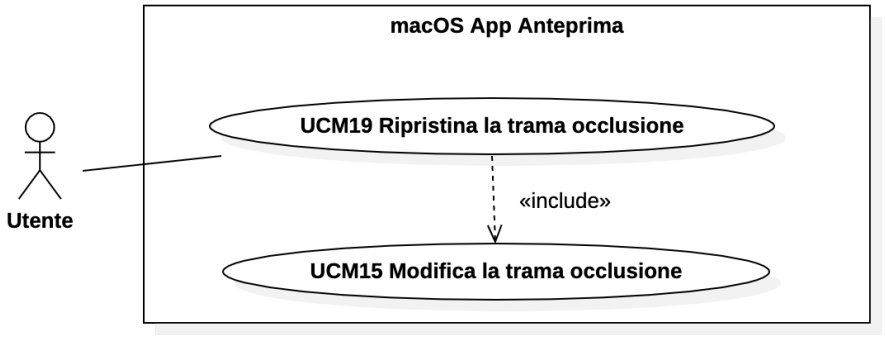

Figura 3.20: Use Case - UCM19 - Ripristina la trama occlusione

* **Descrizione**: L'utente ripristina la trama occlusione originale del modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha cambiato la trama occlusione.
* **Postcondizione**: La trama occlusione originale viene ripristinata.
* **Scenario principale**:
    1. L'utente sceglie di ripristinare la trama occlusione originale;
    2. La trama occlusione originale viene ripristinata.

UCM20 - Nuovo modello

* **Descrizione**: L'utente inizia la creazione di un nuovo modello.
* **Attore primario**: Utente.
* **Precondizione**: L'utente sta visualizzando l'anteprima.
* **Postcondizione**: L'utente viene reindirizzato alla selezione assets.
* **Scenario principale**:
    1. L'utente preme il bottone per iniziare a creare un nuovo modello;
    2. L'utente viene reindirizzato alla selezione assets.

3.2.2.20 UCEM1 - Visualizza errore nella selezione dello zip

* **Descrizione**: L'utente annulla la selezione dello zip o seleziona uno zip vuoto.
* **Attore primario**: Utente.
* **Precondizione**: L'utente annulla la selezione dello zip o seleziona uno zip vuoto.
* **Postcondizione**: Viene mostrato un messaggio d'errore.
* **Scenario principale**:
    1. L'utente annulla la selezione dello zip o seleziona uno zip vuoto.
    2. Viene mostrato il corrispondente messaggio d'errore;
    3. All'utente viene data la possibilità di tentare nuovamente la selezione dello zip.

3.2.2.21 UCEM2 - Visualizza errore nella creazione del modello

* **Descrizione**: Durante l'elaborazione del modello si è verificato un errore.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha iniziato l'elaborazione del modello.
* **Postcondizione**: Viene mostrato il relativo errore.
* **Scenario principale**:
    1. L'utente inizia la crezione del modello;
    2. Durante l'elaborazione del modello si verifica un errore;
    3. Viene mostrato il relativo errore.

3.2.2.22 UCEM3 - Visualizza un avviso per un'immagine non valida

* **Descrizione**: Durante l'elaborazione del modello l'algoritmo ha trovato un'immagine non valida.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha iniziato l'elaborazione del modello.
* **Postcondizione**: Viene mostrato un errore con il relativo motivo.
* **Scenario principale**:
    1. L'utente inizia la crezione del modello;
    2. Durante l'elaborazione del modello l'algoritmo trova un'immagine non valida;
    3. Viene mostrato un errore con il relativo motivo.

3.2.2.23 UCEM4 - Visualizza un avviso quando un'immagine non viene elaborata

* **Descrizione**: Durante l'elaborazione del modello l'algoritmo ha saltato un'immagine.
* **Attore primario**: Utente.
* **Precondizione**: L'utente ha iniziato l'elaborazione del modello.
* **Postcondizione**: Viene mostrato un errore con il nome dell'immagine saltata.
* **Scenario principale**:
    1. L'utente inizia la crezione del modello;
    2. Durante l'elaborazione del modello l'algoritmo non elabora un'immagine;
    3. Viene mostrato un errore con il nome dell'immagine saltata.

3.2.3 Applicazione iOS

3.2.3.1 UCI1 - Inserimento Credenziali

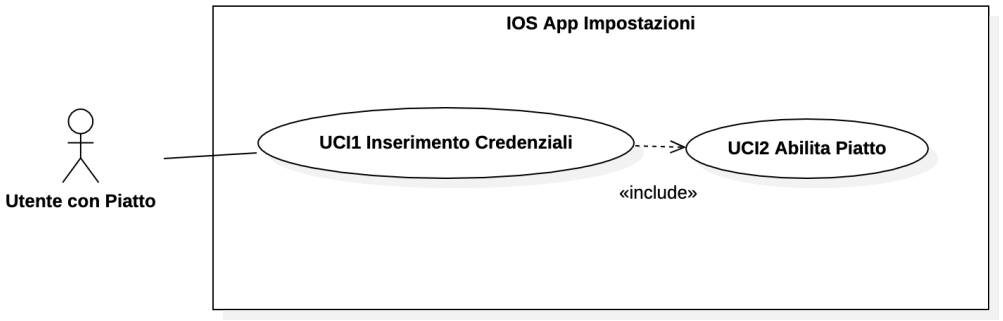

Figura 3.21: Use Case - UCI1 e UCI2: Inserimento Credenziali

* **Descrizione**: L'utente può scegliere di connettere un piatto rotante.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'utente può scattare utilizzando il piatto rotante.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione;
    2. L'utente inserisce le credenziali richieste.
* **Sottocasi**:
    1. UCI1.1 AWS Endpoint;
    2. UCI1.2 AWS Regione;
    3. UCI1.3 AWS Certificato;
    4. UCI1.4 Password del certificato.

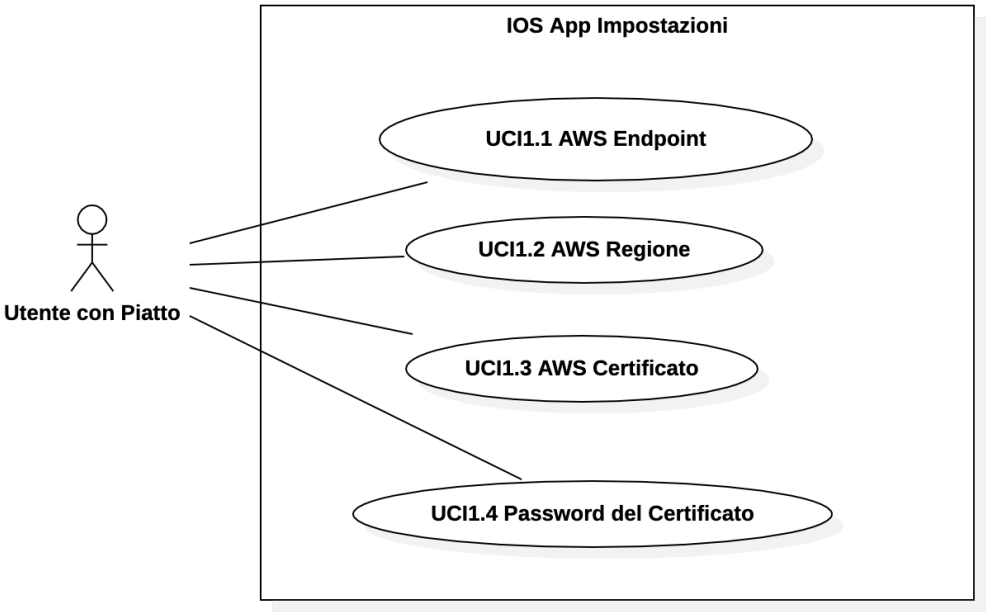

Figura 3.22: Use Case - Sottocasi UCI1

3.2.3.2 UCI2 - Abilita Piatto Rotante

* **Descrizione**: L'utente abilita le impostazioni del piatto rotante.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione
* **Postcondizione**: L'impostazione viene salvata e ora può inserire le credenziali.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione;
    2. L'utente abilita il piatto rotante;
    3. L'impostazione viene salvata e ora può inserire le credenziali (UCI1).

3.2.3.3 UCI3 - Modalità Automatica

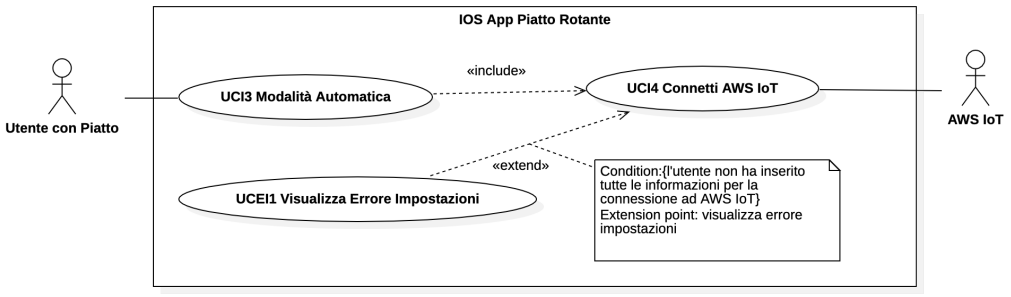

Figura 3.23: Use Case - UCI3: Modalità Automatica

* **Descrizione**: L'utente sceglie le impostazioni per la modalità automatica
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'utente ora può iniziare una sessione di cattura automatica.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente si connette ad AWS IoT;
    3. L'utente sceglie le impostazioni per la modalità automatica;
    4. L'utente ora può iniziare una sessione di cattura automatica.
* **Sottocasi**:
    1. UCI3.1 Senso di rotazione;
    2. UCI3.2 Numero di foto.

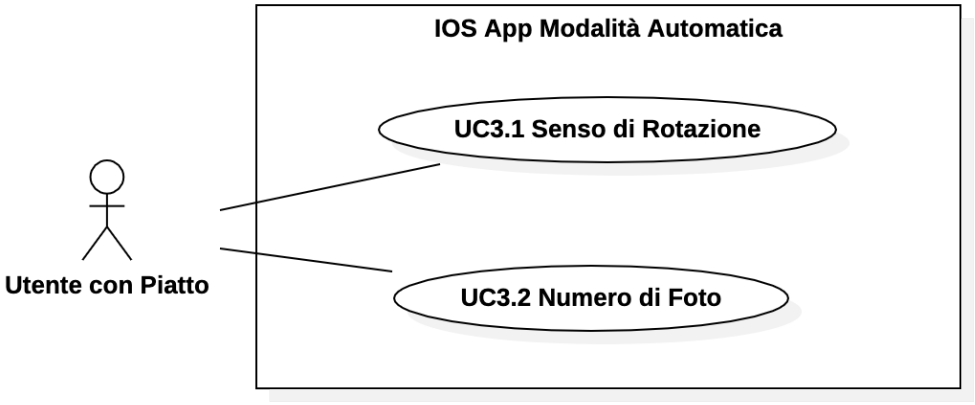

Figura 3.24: Use Case - Sottocasi UCI3

UCI3.1 - Senso di Rotazione

* **Descrizione**: L'utente decide se ruotare il piatto in senso orario o in senso antiorario.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente sceglie il senso di rotazione;
    3. L'impostazione viene salvata.

UCI3.2 - Numero di Foto

* **Descrizione**: L'utente decide il numero di foto da scattare in modalità automatica.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente sceglie il numero di foto;
    3. L'impostazione viene salvata.

3.2.3.4 UCI5 - Modalità Manuale

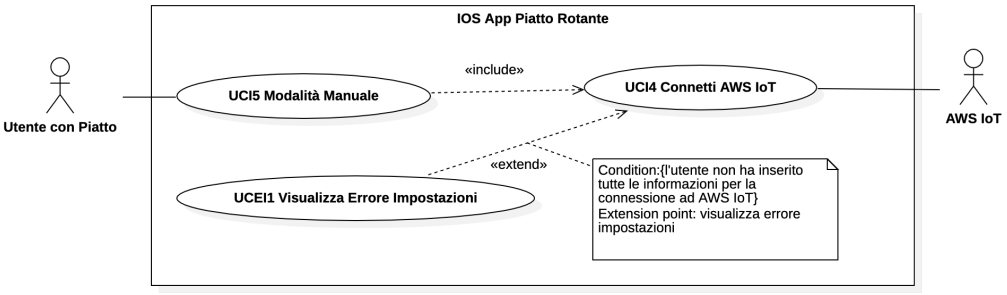

Figura 3.25: Use Case - UCI5: Modalità Manuale

* **Descrizione**: L'utente può scegliere come muovere il piatto.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: Il piatto ruota.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente si connette ad AWS IoT;
    3. L'utente sceglie come muove il piatto;
    4. Il piatto ruota.
* **Sottocasi**:
    1. UCI5.1 Senso di rotazione;
    2. UCI5.2 Steps.

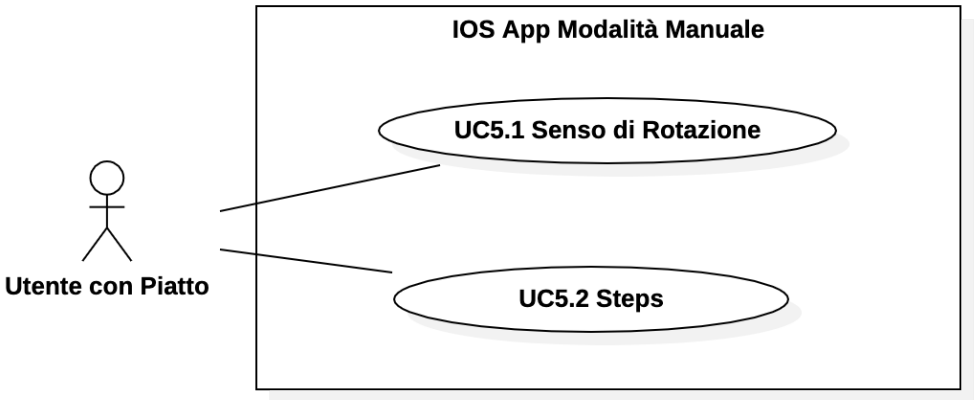

Figura 3.26: Use Case - Sottocasi UCI5

UCI5.1 - Senso di Rotazione

* **Descrizione**: L'utente decide se ruotare il piatto in senso orario o in senso antiorario.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente sceglie il senso di rotazione;
    3. L'impostazione viene salvata.

UCI5.2 - Steps

* **Descrizione**: L'utente decide il numero di quanti steps muovere il piatto rotante. .
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante.
* **Postcondizione**: L'impostazione viene salvata.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha abilitato il piatto rotante;
    2. L'utente sceglie il numero di steps;
    3. L'impostazione viene salvata.

3.2.3.5 UCI4 - Connetti AWS IoT

* **Descrizione**: L'utente connette l'applicazione ad AWS IoT.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha inserito tutte le credenziali di AWS IoT.
* **Postcondizione**: La connessione è stata stabilita.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione e ha inserito tutte le credenziali di AWS IoT;

2. L'utente clicca un bottone per connettersi;
3. La connessione viene stabilita.

* **Estensioni:**

1. Nel caso in cui l'utente non abbia inserito tutte le informazioni per la connessione ad AWS IoT:
   (a) L'utente con piatto vuole connettersi ad AWS IoT;
   (b) L'utente con piatto non ha inserito tutte le informazioni per la connessione ad AWS IoT;
   (c) Viene mostrato un messaggio d'errore(UCEI1 §3.2.7)

3.2.3.6 UCI6 - Cattura Automatica

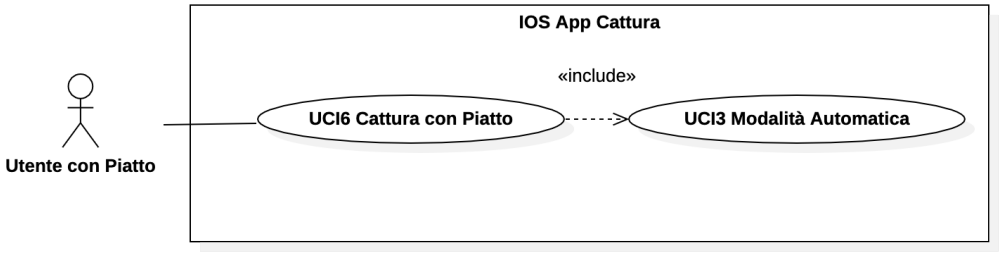

Figura 3.27: Use Case - UCI6: Cattura Automatica

* **Descrizione**: L'utente cattura le foto automaticamente con il piatto rotante.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente è all'interno dell'applicazione e ha impostato la modalità automatica.
* **Postcondizione**: La cattura automatica inizia.
* **Scenario principale**:

1. L'utente è all'interno dell'applicazione e ha impostato la modalità automatica;
2. L'utente clicca un bottone per iniziare a scattare;
3. La cattura automatica inizia.

3.2.3.7 UCI7 - Esporta Sessione

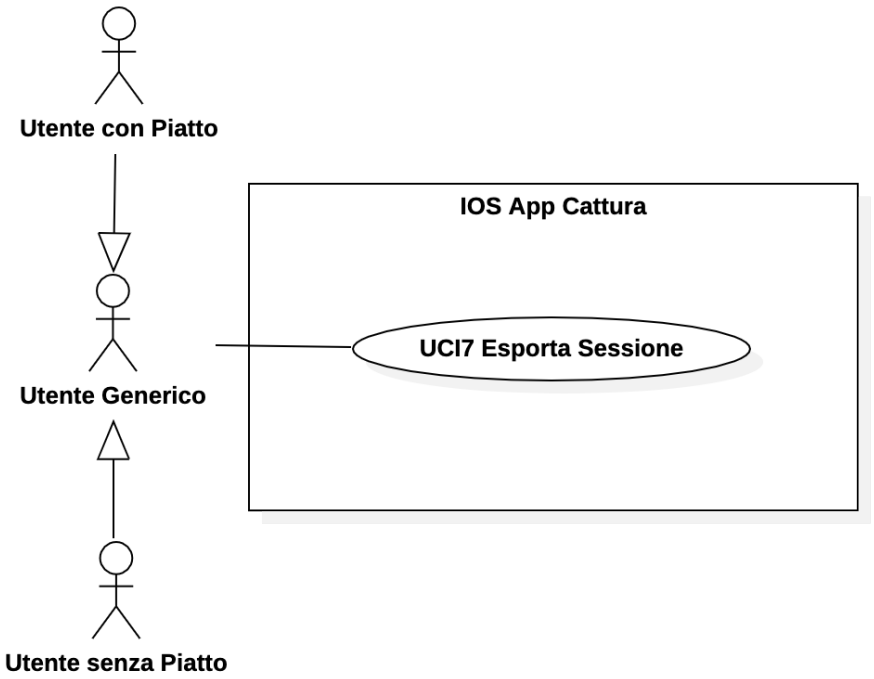

Figura 3.28: Use Case - UCI7: Esporta Sessione

* **Descrizione**: L'utente esporta in un file zip la sessione.
* **Attore primario**: Utente generico.
* **Precondizione**: L'utente è all'interno dell'applicazione.
* **Postcondizione**: L'esportazione viene eseguita.
* **Scenario principale**:
    1. L'utente è all'interno dell'applicazione;
    2. L'utente clicca un bottone per esportare la sessione;
    3. L'esportazione viene eseguita.

3.2.3.8 UCEI1 - Visualizza errore nel caso in cui non vengano inserite tutte le credenziali

* **Descrizione**: L'utente con piatto non inserisce tutte le credenziali per la connessione ad AWS IoT.
* **Attore primario**: Utente con piatto.
* **Precondizione**: L'utente non inserisce tutte le credenziali.

* **Postcondizione**: Viene mostrato un messaggio d'errore e la connessione non viene stabilita.

* **Scenario principale**:

    1. L'operazione di inserimento del nome fallisce perché viene inserita una stringa vuota;
    2. Viene mostrato il corrispondente messaggio d'errore;
    3. All'utente viene data la possibilità di tentare nuovamente la registrazione al sistema.

3.3 Tracciamento dei requisiti

In seguito all'analisi dei requisiti e degli use case del progetto, è stata stilata una tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti, distinti da un codice identificativo. Ogni requisito è strutturato nel seguente modo:

* **Codice Identificativo**: ogni codice identificativo è univoco e conforme alla seguente codifica:

  R[Applicazione][ID][Tipologia]

  Il significato è il seguente:

  - **Applicazione**: ogni requisito può assumere i seguenti valori:
    * I: relativo all'applicazione iOS;
    * M: relativo all'applicazione macOS;
    * IM: relativo alle applicazioni iOS e macOS.
  - **ID**: identificatore del requisito in forma gerarchica, la presenza del secondo numero indica che si tratta di un requisito figlio.
  - **Tipologia**: ogni requisito può assumere i seguenti valori:
    * F: funzionale;
    * Q: qualitativo;
    * V: vincolo;
    * P: prestazionale.
  - **Importanza**: riporta l'importanza del requisito:
    * obbligatorio: requisito irrinunciabile;
    * desiderabile: requisito non strettamente necessario ma con valore aggiunto riconoscibile;
    * facoltativo: requisito relativamente utile.

Tabella 3.2: Tabella del tracciamento dei requisiti funzionali dell'applicazione macOS

Requisito	Descrizione	Classificazione	Use Case
RM1F	L'utente deve poter scegliere il file zip esportato dall'applicazione iOS	Obbligatorio	UCM1
RM2F	L'utente deve poter modificare le impostazioni di creazione del modello	Obbligatorio	UCM2
RM2.1F	L'utente deve poter modificare la qualità del modello	Obbligatorio	UCM2.1
RM2.1.1F	L'utente deve poter sceglier la qualità *Preview*	Obbligatorio	UCM2.1.1
RM2.1.2F	L'utente deve poter sceglier la qualità *Reduced*	Obbligatorio	UCM2.1.2
RM2.1.3F	L'utente deve poter sceglier la qualità *Medium*	Obbligatorio	UCM2.1.3
RM2.1.4F	L'utente deve poter sceglier la qualità *Full*	Obbligatorio	UCM2.1.4
RM2.1.5F	L'utente deve poter sceglier la qualità *Raw*	Obbligatorio	UCM2.1.5
RM2.2F	L'utente deve poter modificare l'impostazione di ordinamento delle foto	Desiderabile	UCM2.2
RM2.2.1F	L'utente deve poter modificare scegliere l'ordinamento disordinato	Desiderabile	UCM2.2.1
RM2.2.2F	L'utente deve poter modificare scegliere l'ordinamento sequenziale	Desiderabile	UCM2.2.2
RM2.3F	L'utente deve poter modificare l'impostazione dell'algoritmo Feature Sensitivity	Desiderabile	UCM2.3
RM3F	L'utente deve poter creare il modello	Obbligatorio	UCM3
RM4F	L'utente deve poter annullare la creazione del modello	Desiderabile	UCM4
RM5F	L'utente deve poter visualizzare un'anteprima del modello creato	Obbligatorio	UCM5
RM6F	L'utente deve poter catturare uno screenshot dell'anteprima	Desiderabile	UCM6
RM7F	L'utente deve poter modificare il colore di sfondo dell'anteprima	Desiderabile	UCM7
RM8F	L'utente deve poter modificare la luminosità del modello	Desiderabile	UCM8

RM9.1F	L'utente deve poter salvare il modello nel formato usdz	Obbligatorio	UCM9.1
RM9.2F	L'utente deve poter salvare il modello nel formato obj	Desiderabile	UCM9.2
RM9.3F	L'utente deve poter salvare il modello nel formato dae	Desiderabile	UCM9.3
RM9.4F	L'utente deve poter salvare il modello nel formato stl	Desiderabile	UCM9.4
RM10F	L'utente deve poter ruotare il modello	Desiderabile	UCM10
RM11F	L'utente deve poter scalare il modello	Desiderabile	UCM11
RM12F	L'utente deve poter modificare la trama base	Desiderabile	UCM12
RM13F	L'utente deve poter modificare la trama normale	Desiderabile	UCM13
RM14F	L'utente deve poter modificare la trama ruvidezza	Desiderabile	UCM14
RM15F	L'utente deve poter modificare la trama occlusione	Desiderabile	UCM15
RM16F	L'utente deve poter ripristinare la trama base originale	Desiderabile	UCM16
RM17F	L'utente deve poter ripristinare la trama normale originale	Desiderabile	UCM17
RM18F	L'utente deve poter ripristinare la trama ruvidezza originale	Desiderabile	UCM18
RM19F	L'utente deve poter ripristinare la trama occlusione originale	Desiderabile	UCM19
RM20F	L'utente deve poter iniziare la creazione di un nuovo modello	Obbligatorio	UCM20

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali dell'applicazione iOS

Requisito	Descrizione	Classificazione	Use Case
RI1F	L'utente con piatto deve poter inserire le credenziali per il piatto rotante	Obbligatorio	UCI1
RI1.1F	L'utente con piatto deve poter inserire il proprio AWS Endpoint	Obbligatorio	UCI1.1
RI1.2F	L'utente con piatto deve poter inserire la propria regione AWS	Obbligatorio	UCI1.2
RI1.3F	L'utente con piatto deve poter inserire il proprio AWS Certificato	Obbligatorio	UCI1.3
RI1.4F	L'utente con piatto deve poter inserire la propria password del certificato	Obbligatorio	UCI1.4
RI2F	L'utente con piatto deve poter abilitare e disabilitare il piatto rotante	Obbligatorio	UCI2
RI3F	L'utente con piatto deve poter impostare la modalità automatica di rotazione del piatto	Obbligatorio	UCI3
RI3.1F	L'utente con piatto deve poter impostare il senso di rotazione del piatto	Obbligatorio	UCI3.1
RI3.2F	L'utente con piatto deve poter impostare il numero di foto da scattare	Obbligatorio	UCI3.2
RI4F	L'utente con piatto deve potersi connettere ad AWS IoT	Obbligatorio	UCI4
RI5F	L'utente con piatto deve poter muovere il piatto in modalità manuale	Obbligatorio	UCI5
RI5.1F	L'utente con piatto deve poter impostare il senso di rotazione del piatto	Obbligatorio	UCI5.1
RI5.2F	L'utente con piatto deve poter impostare il numero di steps da compiere	Obbligatorio	UCI5.2
RI6F	L'utente con piatto deve poter iniziare una sessione di cattura automatica	Obbligatorio	UCI6
RI7F	L'utente deve poter esportare un file zip con le foto	Obbligatorio	UCI7

Tabella 3.3: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Classificazione
RIM1Q	Tutto il codice delle due applicazioni essere ospitato all'interno di Obbligatorio repository su CodeCommit	Obbligatorio

Tabella 3.4: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Classificazione
RM1V	Deve essere sviluppata un'applicazione macOS	Obbligatorio
RI2V	Deve essere utilizzato un Raspberry per il piatto	Obbligatorio
RM3V	L'applicazione macOS deve essere scritta utilizzando il framework SwiftUI	Obbligatorio
RM4V	L'applicazione macOS deve utilizzare le API *Object Capture*	Obbligatorio
RI5V	La comunicazione tra applicazione iOS e piatto rotante deve essere implementata tramite il Bluetooth Low Energy o AWS IoT	Desiderabile
RI6V	La comunicazione deve essere implementata con l'SDK di AWS IoT	Facoltativo

Capitolo 4

Progettazione e codifica

In questo capitolo viene trattata la progettazione delle applicazioni e del piatto rotante, descrivendone il funzionamento e i design pattern utilizzati. A seguire, viene esposta la codifica dell'intero progetto.

4.1 Applicazione iOS

L'applicazione iOS non ha richiesto una progettazione totale, dal momento che Apple l'aveva già realizzata in occasione della presentazione delle nuove API *Object Capture* alla *WWDC2021*. Di conseguenza, il tutor interno, dopo aver valutato la qualità del codice già presente, ha deciso di procedere all'estensione dell'applicazione già sviluppata.

Originariamente aveva le seguenti viste:

* **Cattura**: la vista principale dell'applicazione, da cui è possibile: scattare una foto, avviare una cattura automatica, visualizzare i sensori disponibili e il numero di foto già salvate;
* **Galleria**: la vista che mostra le foto scattate in una sessione, da cui è possibile eliminarla o ingrandirla con un tap, oltre a visualizzare se è stata scattata con il lidar e il giroscopio. È possibile, inoltre, iniziare una nuova sessione;
* **Sessioni**: in questa vista è possibile vedere le sessioni precedenti, aprirne una o cancellarle una alla volta;
* **Suggerimenti**: in questa vista, organizzata in 3 schede, sono riportati dei suggerimenti per realizzare un'ottima sessione di cattura.

Basandosi su quanto già realizzato da Apple e dai requisiti, sono state progettate, attraverso l'utilizzo di *Sketch*, le seguenti nuove viste:

* **Splash**: la vista di benvenuto, che viene mostrata all'avvio dell'applicazione, necessaria per la pubblicazione sull'App Store;

* **Impostazioni**: da questa vista è possibile abilitare il piatto rotante e inserire le credenziali necessarie per il collegamento ad *AWS IoT*, quali l'endpoint, la regione, il nome del certificato e la password per aprirlo;

* **Piatto rotante**: in questa vista l'utente può connettersi ad *AWS IoT*, impostare una cattura automatica, o muovere il piatto manualmente;

* **Onboarding**: la vista di introduzione, viene visualizzata al primo avvio, dopo l'installazione dell'applicazione. La schermata è organizzata in 3 schede, tra cui quella dei *Suggerimenti*; nella terza scheda, inoltre, è presente un bottone per non visualizzare più questa vista all'avvio dell'applicazione.

È stato pianificata la modifica delle viste esistenti con l'aggiunta delle seguenti funzionalità:

* **Condividi**: un pulsante per inviare il file zip della sessione di cattura al Mac. È stato aggiunto nelle viste *Cattura* e *Galleria*;

* **Elimina tutto**: un pulsante, nella vista *Sessioni*, per eliminarle tutte in un click;

* **Cattura automatica**: quando l'utente imposta la cattura automatica con il piatto rotante, il pulsante di scatto, nella vista *Cattura*, indicherà che questa è attiva, e quando l'utente cliccherà su di esso, la cattura automatica avrà inizio;

Tutte le viste sono state progettate aderendo allo stile grafico dell'applicazione originale e del sistema operativo iOS.

4.2 Applicazione macOS

Per la progettazione dell'applicazione macOS, si è deciso di partire dallo studio delle API *Object Capture*; in modo particolare, dalla documentazione fornita da Apple nel portale sviluppatori. Successivamente, con l'utilizzo di *Sketch*, sono stati realizzati alcuni mockup$_G$ delle viste:

1. **Benvenuto**: la prima schermata dell'applicazione, dove è possibile selezionare il file zip esportato dall'applicazione iOS;

2. **Impostazioni**: in questa vista l'utente può vedere una foto contenuta nello zip scelto, per assicurarsi di aver selezionato il file corretto. Inoltre, può scegliere le impostazioni per l'elaborazione del modello e iniziare la sua creazione;

3. **Anteprima**: una volta generato il modello, da questa vista l'utente può vederne un'anteprima, esportarlo, ruotarlo, scalarlo, fare uno *screenshot*, cambiare il colore di sfondo, cambiare le texture e modificare la luminosità del modello.

Si è scelto di partire dallo studio della logica, dal momento che queste tre viste hanno un ordine preciso e non hanno senso prese singolarmente, ad esempio la vista *Anteprima*, senza aver prima generato il modello, non potrà mostrare nulla.

4.3 Piatto rotante

4.3.1 Componenti

Per realizzare il piatto rotante è stato richiesto l'utilizzo del *Raspberry Pi 3*, che l'azienda aveva già in casa, pertanto è stato necessario trovare componenti compatibili con questa scheda elettronica. Dopo numerose ricerche, grazie all'esperienza pregressa nella stampa 3D e nei progetti con *Arduino* e *Raspberry*, sono stati individuati i seguenti componenti, che l'azienda ha acquistato:

* **Motore Nema17**: per ruotare il piatto è stato necessario un motore passo-passo. È stato scelto un Nema17 con un'altezza di 38mm e una coppia di 40Ncm/57,1Nm, dal momento che serviva abbastanza potenza per ruotare anche oggetti di qualche chilo;
* **Driver A4988**: per pilotare un motore passo-passo da una scheda elettronica è fondamentale un driver. Si è scelto un *Pololu A4988* per la sua economicità e semplicità di utilizzo;
* **Alimentatore**: il driver viene alimentato dal *Raspberry*, ma il motore passo-passo ha bisogno di un'alimentazione esterna, per cui è stato selezionato un trasformatore da 12V in corrente continua e 1.67A;
* **Cuscinetti a sfera**: per migliorare la stabilità del piatto e distribuire il peso, sono stati progettati 4 supporti, posizionati agli angoli 90°, 180°, 270° e 360°, con dei cuscinetti a sfera 9x24x7mm che quello in *Figura 4.2*;
* **Staffa**: per fissare il piatto all'albero del motore è stato indispensabile inserire una staffa. È stata scelta quella della figura *Figura 4.1*.

Figura 4.1: Staffa per il fissaggio del piatto

Figura 4.2: Cuscinetto a sfera

Oltre ai componenti elettronici, per costruire il piatto sono stati necessari due dischi di legno da 40cm di diamentro, delle viti e del colore bianco.

4.3.2 Comunicazione con l'applicazione iOS

Per impartire i comandi al piatto rotante dall'applicazione iOS, è stato necessario implementare un protocollo di comunicazione; per questo motivo, insieme al tutor interno sono state individuate alcune possibili tecnologie da poter utilizzare, in ordine di preferenza:

1. **Bluetooth Low Energy**: crea una connessione diretta tra i due dispositivi, senza la necessità di fare il pairing$_{\text{G}}$;

2. **AWS IoT**: stabilisce una connessione con il *Raspberry*, attraverso i server di *AWS* e il protocollo *MQTT*. Richiede la creazione di un account *AWS*;

3. **API REST**: crea una connessione con il *Raspberry*, attraverso una API REST_{G} pubblica. Richiede la creazione di un account *AWS*.

4.4 Design Pattern utilizzati

I design pattern utilizzati nello sviluppo delle applicazioni sono due: Model-View-ViewModel e Observed Pattern.

4.4.1 Model-View-ViewModel

Model-View-ViewModel, abbreviato MVVM, è una generalizzazione del pattern Model-View-Controller(MVC). Il pattern MVVM è stato pensato per semplificare le interfacce caratterizzate da una programmazione a eventi ed è costituito da tre parti:

* **Model**: in cui risiede il modello dei dati e la logica di gestione e validazione degli stessi;
* **View**: rappresenta le viste dell'applicazione, ossia ciò che vede l'utente;
* **View-Model**: è un ponte di collegamento tra la vista e il modello ed è specifico per ogni **View**. Qui sono contenuti e vengono esposti, tutti i metodi che sono necessari per il funzionamento della vista.

Questo pattern è stato utilizzato per lo sviluppo di tutte le viste di entrambe le applicazioni.

4.4.2 Observer Pattern

L'observer pattern è un design pattern pensato per la gestione di eventi in cui c'è un observer che resta in ascolto di eventuali cambiamenti che gli vengono notificati da un subject. Questo pattern è incluso in *SwiftUI*, quindi è stato inserito ogni qualvolta fosse necessario registrare un cambiamento di stato, ad esempio quando l'utente clicca su un pulsante o completa un campo di testo.

4.5 Codifica

4.5.1 Applicazione iOS

Prima della codifica delle funzionalità mancanti nell'applicazione iOS ci si è soffermati qualche giorno allo studio del framework *SwiftUI* leggendo il libro *SwiftUI by Tutorials 4.0* fornitomi dal tutor interno e visionando i video tutorial di Apple. Successivamente, è stata studiata l'applicazione di esempio, che aveva già creato Apple in occasione della *WWDC2021*, per capire come fosse strutturata e per individuare dove inserire la vista di benvenuto, la vista di introduzione, il pulsante "Elimina tutto" e il tasto "Condividi". Per quest'ultimo è stato scelto di utilizzare l'interfaccia standard di iOS per la condivisione di un contenuto tramite *Airdrop*, e la libreria *ZipArchive* per realizzare un file zip compresso in modo da diminuire il tempo di trasferimento. Infine, sono state implementate le traduzioni in Italiano e Inglese, che cambiano automaticamente in base alla lingua scelta nelle impostazioni del dispositivo. La parte relativa al piatto rotante è stata sviluppata dopo aver terminato la comunicazione tra l'applicazione iOS e il *Raspberry* perchè inizialmente non era prevista, come già accennato nella *Sezione 2.2*.
Il logo dell'applicazione per iOS è in *Figura 4.3*.

Figura 4.3: Logo dell'applicazione

Le Figure 4.4 - 4.9 illustrano l'applicazione iOS, realizzata come sopraccitato nella progettazione alla *Sezione 4.1*.

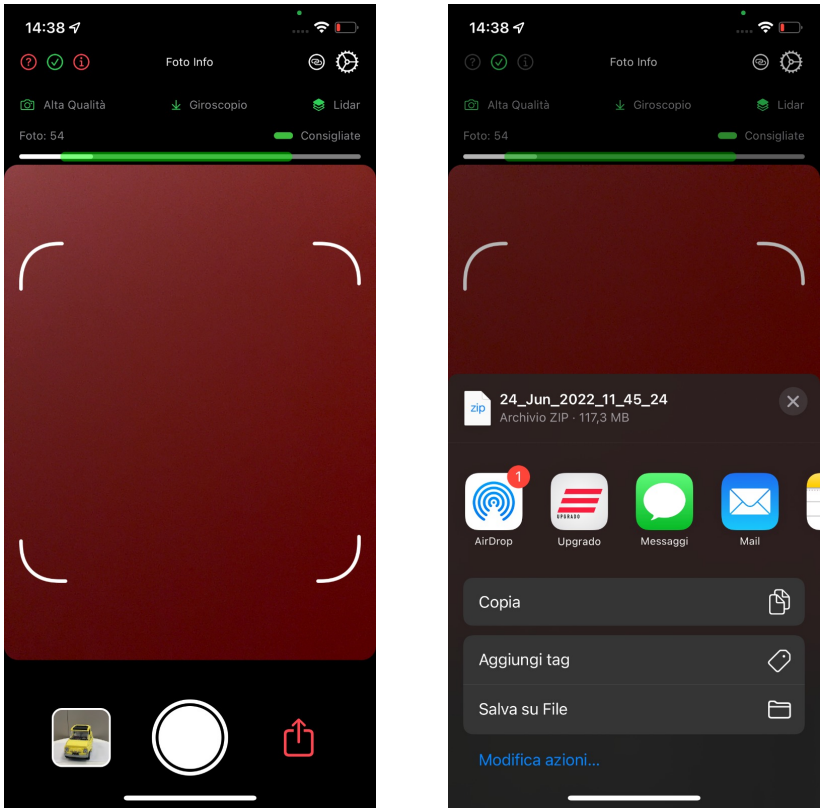

Figura 4.4: Schermata principale con esportazione delle foto scattate

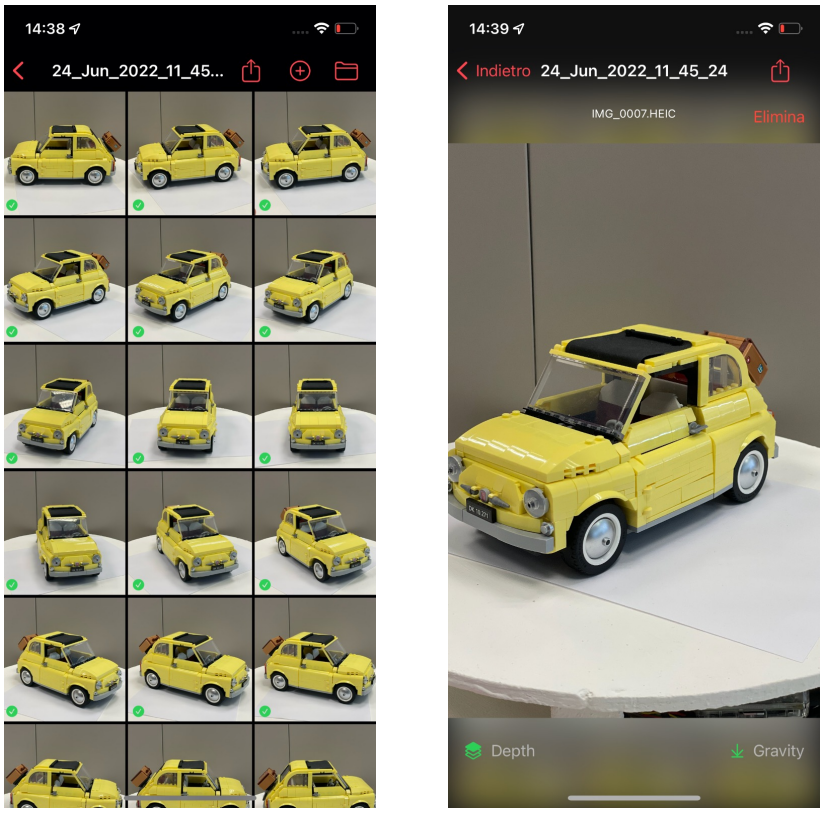

Figura 4.5: Galleria: vista tutte e singola foto

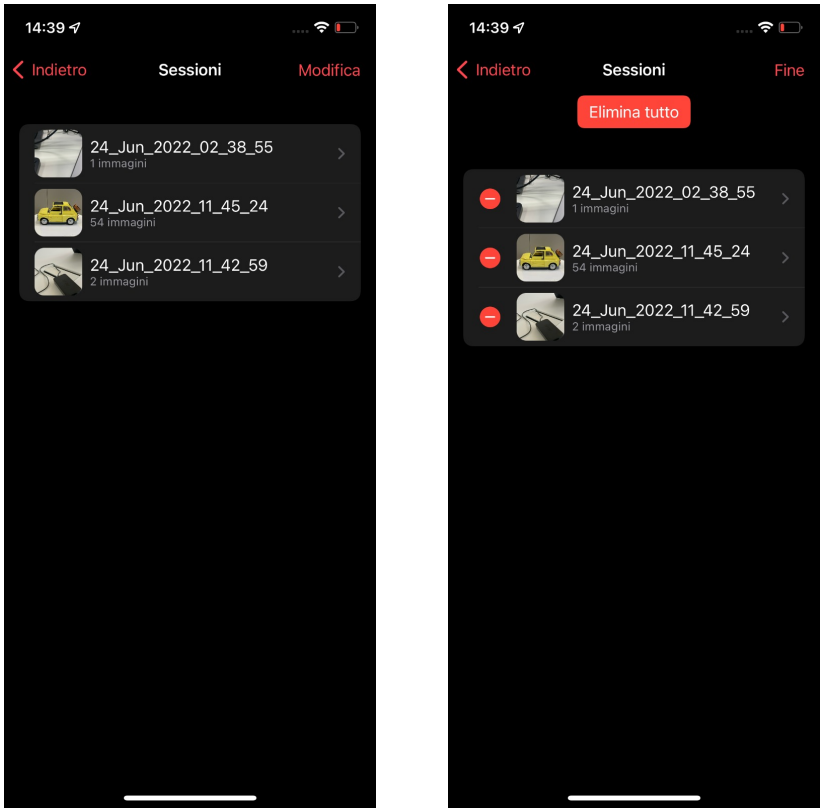

Figura 4.6: Sessioni precedenti con possibilità di cancellazione

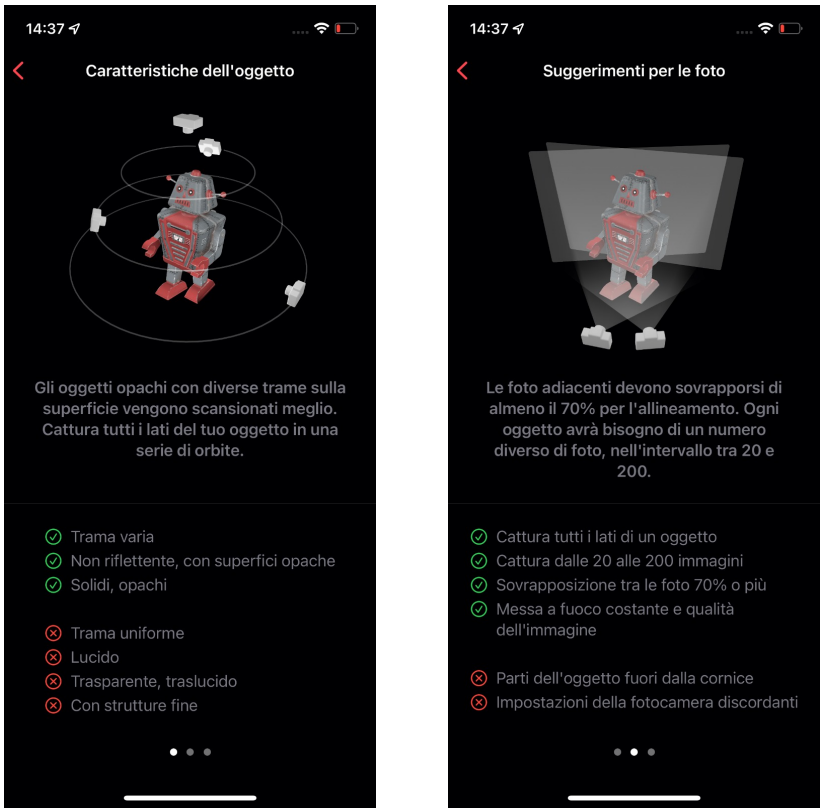

Figura 4.7: Suggerimenti sulla fotogrammetria

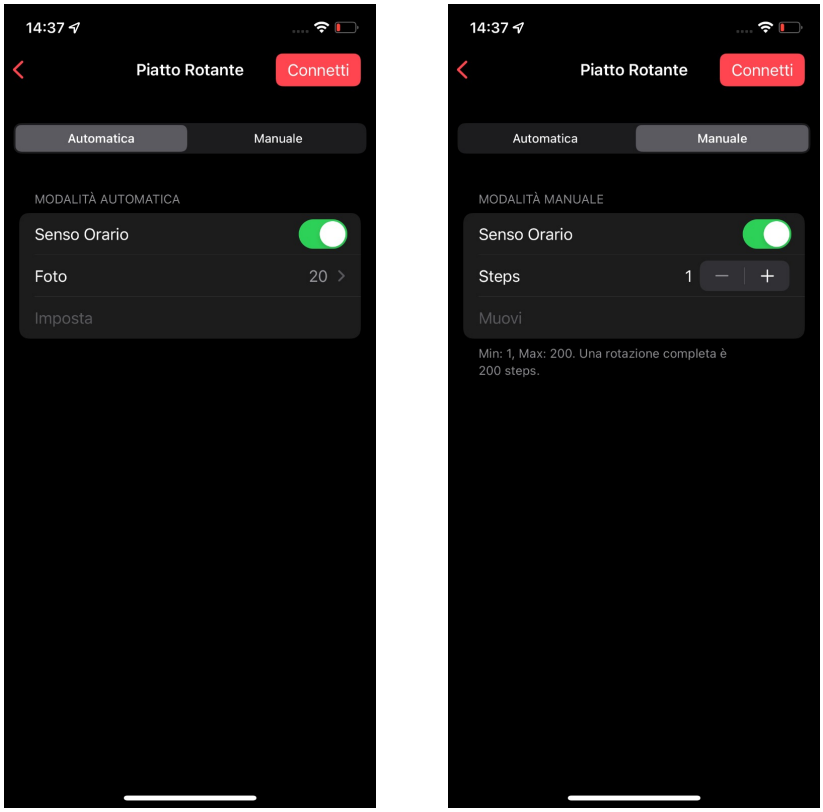

Figura 4.8: Piatto rotante: modalità automatica e manuale

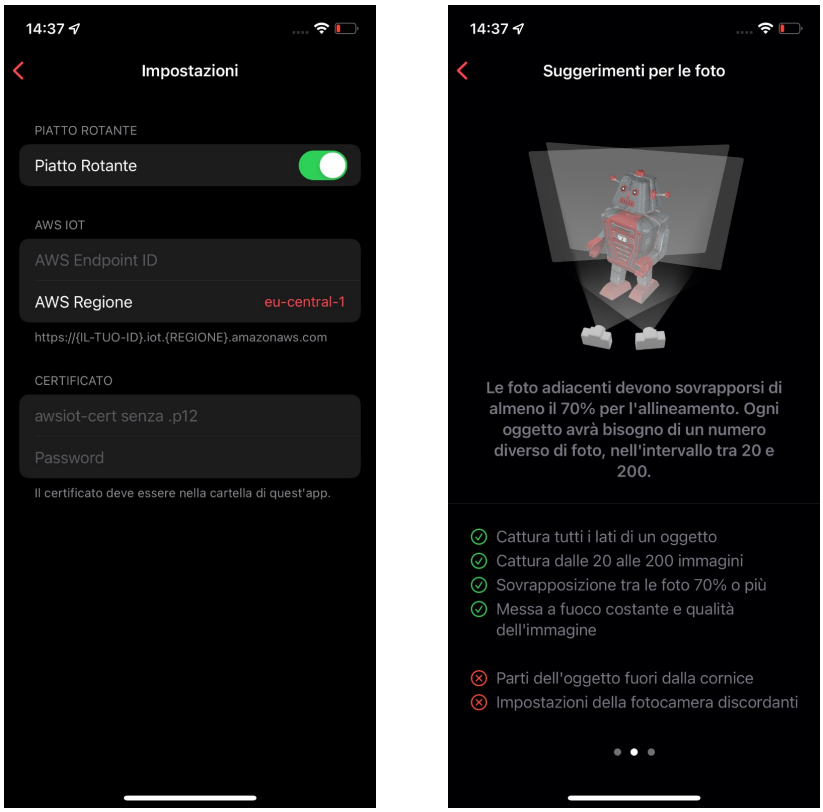

Figura 4.9: Impostazioni dell'applicazione

4.5.2 Applicazione macOS

La codifica dell'interfaccia dell'applicazione macOS è stata più veloce perchè si aveva già familiarizzato con il linguaggio *Swift* e il framework *SwiftUI*. Tuttavia, per sviluppare l'anteprima del modello si doveva ricorrere al framework *SceneKit*, che ha sollevato numerosi problemi a causa della mancanza di una documentazione esaustiva per la nuova classe *SceneView*, creata da Apple per includere un contenuto *SceneKit* in *SwiftUI*. Così, dopo che si è scoperto che tutte le funzionalità della vecchia classe *SCNView* non erano state incluse nella nuova *SceneView*, in accordo con il tutor interno, si è deciso di utilizzare quella più vecchia.
Le Figure 4.10 - 4.16 mostrano alcuni *screenshot* dell'applicazione macOS, realizzata come indicato nella *Sezione 4.3*.

Figura 4.10: Selezione zip di immagini

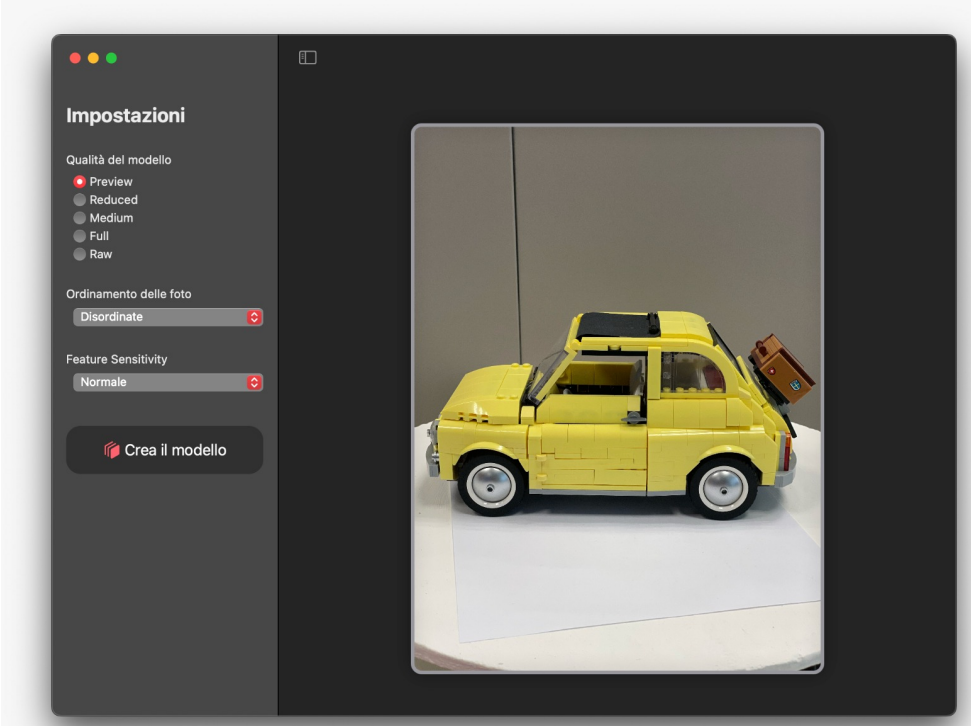

Figura 4.11: Schermata impostazioni

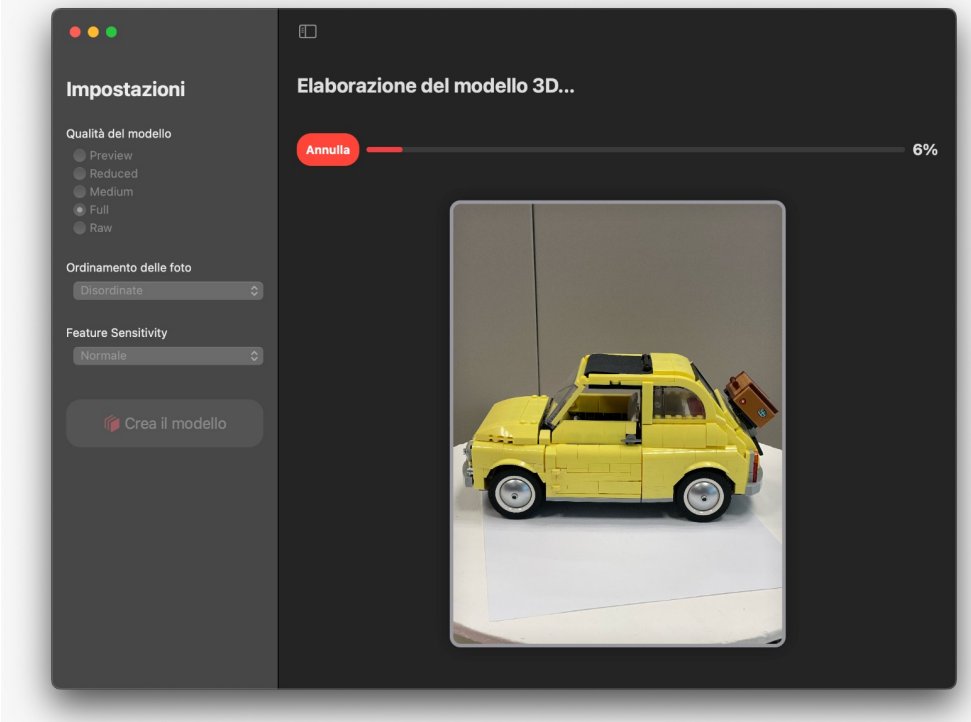

Figura 4.12: Schermata impostazioni: elaborazione del modello

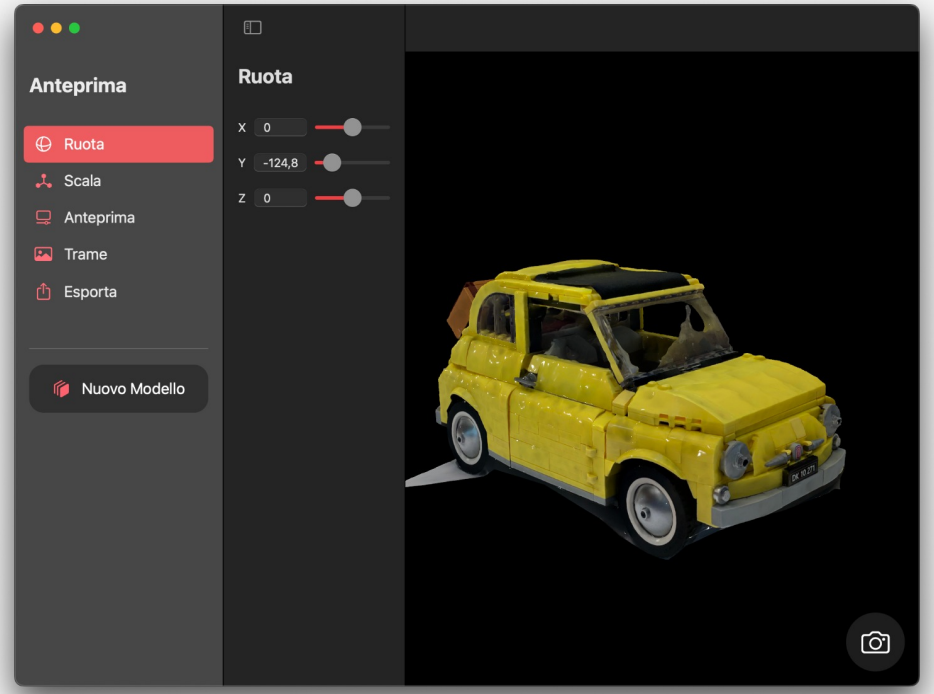

Figura 4.13: Schermata anteprima: ruota il modello

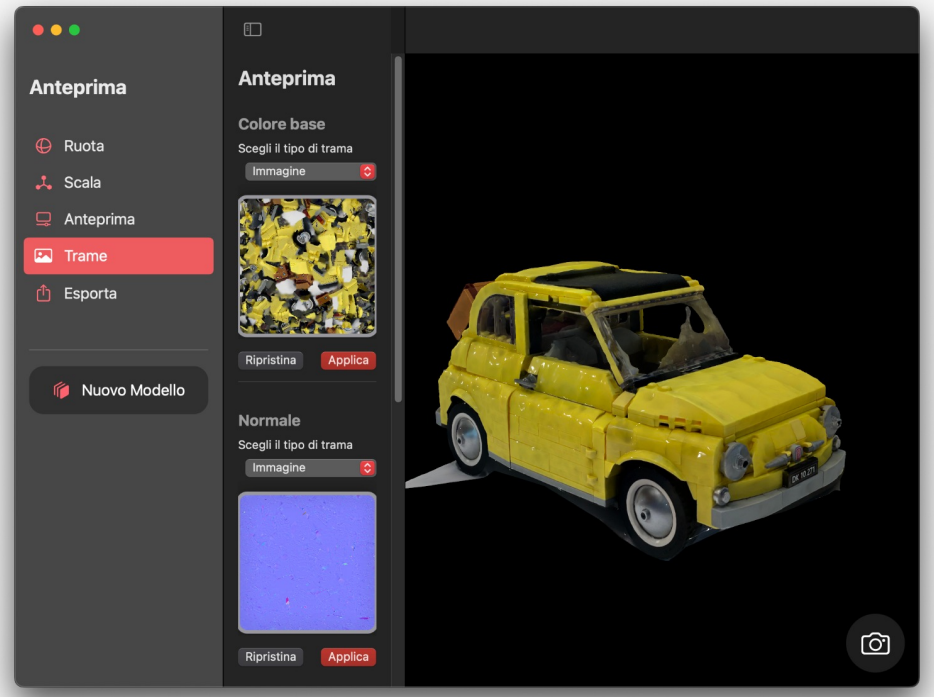

Figura 4.14: Schermata anteprima: modifica delle textures

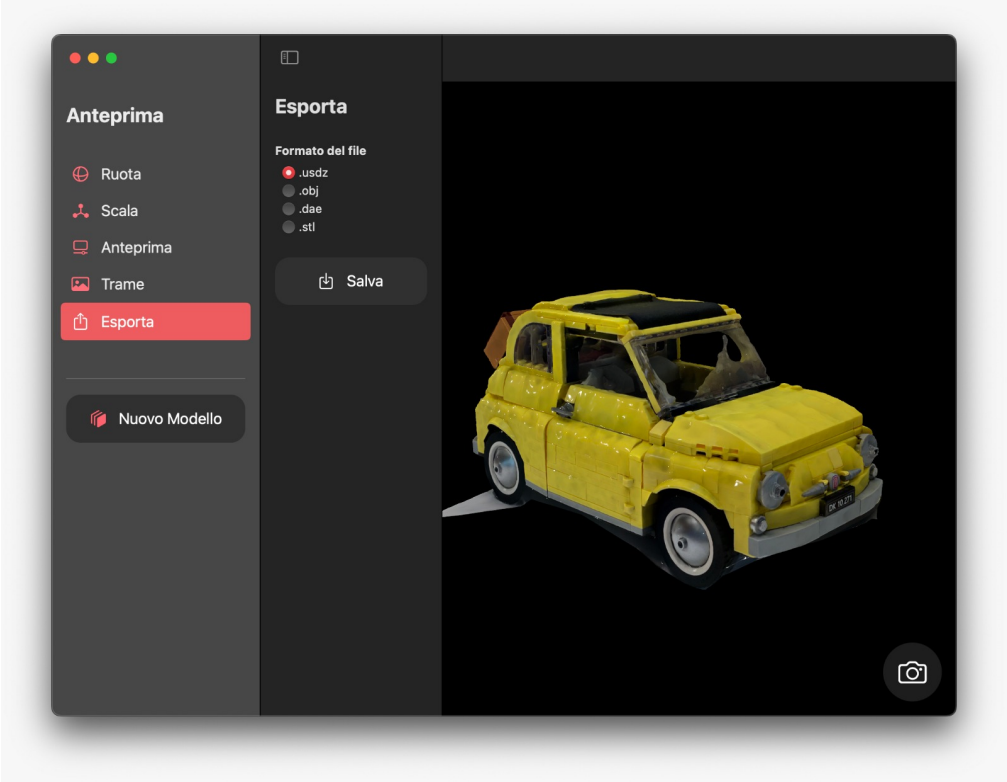

Figura 4.15: Schermata anteprima: esportazione del modello

Figura 4.16: Screenshot dell'anteprima

4.5.3 Piatto rotante

Dal momento che, il progetto del piatto rotante è open-source e successivamente verrà pubblicato su *Github*, era preferibile utilizzare il *Bluetooth Low Energy* per una questione di facilità di replicazione del progetto. Tuttavia, dopo numerose prove con questa tecnologia, si è constatato che non era sufficentemente stabile e più di qualche volta richiedeva comunque il *pairing*, che non doveva esserci. Di conseguenza, in accordo con il tutor interno, si è deciso di adottare la seconda tecnologia precedentemente individuata: *AWS IoT* con il protocollo *MQTT*. La *Figura 4.17* riassume la comunicazione tra l'applicazione iOS e il *Raspberry*. L'applicazione iOS instaura una connessione con *AWS IoT* e quando l'utente inizia la cattura automatica invia un messaggio, che in *MQTT* viene chiamato *payload*, con le informazioni per ruotare il piatto. *AWS IoT* inoltrerà il messaggio, ricevuto dall'applicazione, al *Raspberry*, che sarà continuamente in attesa di nuovi *payload*. Quando il *Raspberry* riceve il messaggio ruota il piatto e invia all'applicazione iOS un *payload* di conferma. Dopo aver ricevuto quest'ultimo messaggio l'applicazione scatterà la foto. Questo processo si ripete per ogni foto.

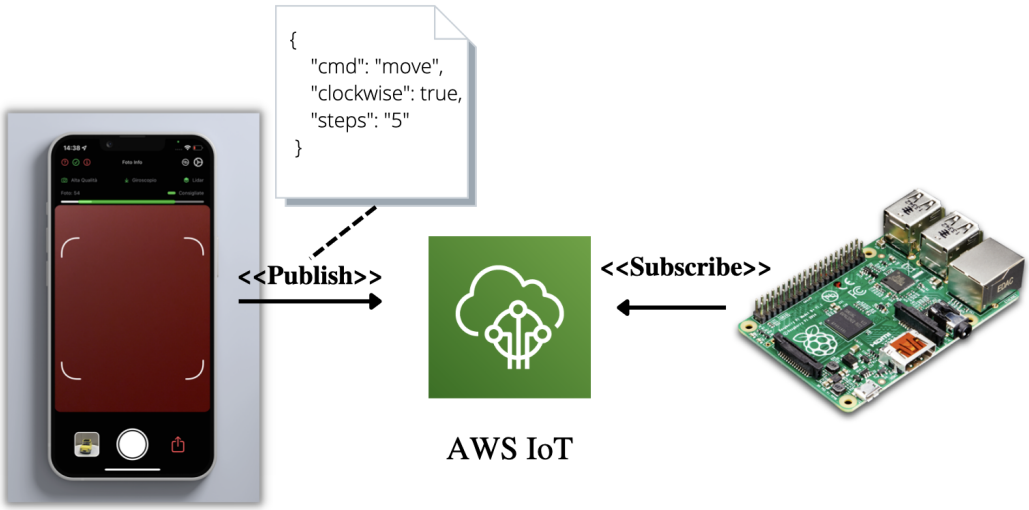

Figura 4.17: Schema del protocollo *MQTT* con *AWS IoT*

4.5.3.1 Comunicazione tra il Raspberry e AWS IoT

Per la comunicazione tra il *Raspberry* e *AWS IoT* inizialmente si era pensato di utilizzare il linguaggio *Swift*, ma *AWS* non fornisce una libreria per questo linguaggio e quelle trovate online sono tutte obsolete o piene di bug. Di conseguenza, si è deciso di ricorrere all'SDK *AWS* con il linguaggio *Python*, che non ha sollevato nessun problema.

4.5.3.2 Comunicazione tra AWS IoT e iOS

Dopo aver completato la codifica della comunicazione tra il *Raspberry* e *AWS IoT* sono stati individuati due modi per realizzare quella tra *AWS IoT* e

l'applicazione iOS:

1. **Senza certificato**: *AWS Cognito User Pool* autentica l'utente e genera un identificativo univoco per la connessione con *AWS IoT*;

2. **Con certificato**: l'utente con piatto deve generare un certificato *X.509* dal servizio *AWS IoT*, che dovrà inserire all'interno della cartella dell'applicazione iOS.

Come già accennato alla *Sezione 4.5.3*, per la facilità di replicazione del progetto, il tutor interno ha richiesto di provare con il primo modo, cioè senza certificato. Tuttavia, data la scarsa qualità dell'*SDK AWS*, non è stato possibile portare a termine questa richiesta. Di conseguenza, è stata completata la codifica della comunicazione tra l'app iOS e *AWS IoT* nel secondo modo, con il compromesso legato al certificato, che deve essere generato e trasferito al dispositivo iOS, nella cartella dell'applicazione.
Le figure *4.18* e *4.19* mostrano il piatto rotante, autocostruito con due dischi di legno dipinti da 40 cm di diametro, il motore centrale e i 4 supporti dotati di cuscinetti a sfera.

Figura 4.18: Piatto rotante: vista dall'alto

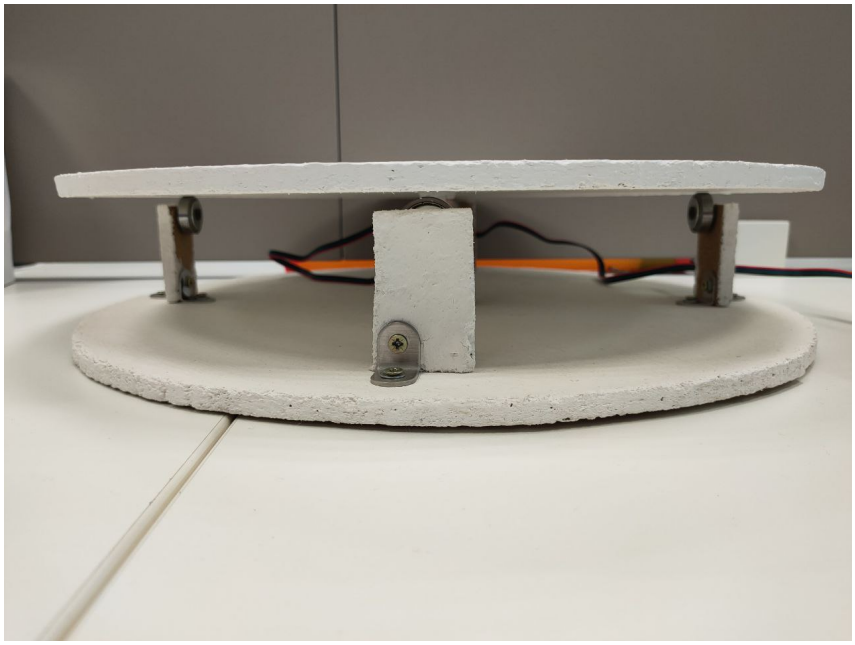

Figura 4.19: Piatto rotante: motore e cuscinetti a sfera

Capitolo 5

Test delle API Object Capture

In questo capitolo si descrivono i test effettuati con la tecnica della fotogrammetria e con l'acquisizione delle foto in modalità automatica, con l'uso del piatto rotante.

In seguito al termine della codifica dell'applicazione macOS, si è iniziato a testare il processo di creazione di un modello 3D. Prima di cominciare è stato necessario un momento di formazione con il video di Apple dedicato alle API Object Capture, dove vengono dati i seguenti suggerimenti per scattare le foto ad un oggetto:

* **L'oggetto**: l'oggetto deve essere solido, ricco di colori e trame, non riflettente e con superfici opache;
* **Le foto**: per scattare le foto è consigliato utilizzare un fondo a tinta unita chiaro, come un piatto bianco. Per scattare le foto si può girare attorno all'oggetto o ruotare il piatto e per catturare anche il sotto bisogna ruotare l'oggetto, le foto adiacenti devono sovrapporsi di almeno il 70%. Ogni oggetto avrà bisono dalle 20 alle 200 foto;
* **L'ambiente**: l'illuminazione dell'ambiente deve essere uniforme, evitando troppe ombre o troppa luce e ci deve essere abbastanza spazio attorno all'oggetto. Inoltre, ci deve essere la possibilità di ruotare attorno all'oggetto.

Per tutti i test si è utilizzato l'iPhone 12 Pro fornito dall'azienda, alcuni oggetti presenti in ufficio e il *MacBook Pro 16" M1 Pro* personale.

5.1 Test con light box

Nei primi test è stato utilizzato il piatto rotante costruito e un Lightbox$_G$ personale, per avere un'illuminazione uniforme e maggiore rispetto a quella presente in ufficio.

5.1.1 Lego

Figura 5.1: Test con light box - Immagine dell'oggetto: Lego

* **Numero di foto**: 39
* **Lidar**: Si
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 2 minuti
* **Risultato**: Il risultato purtroppo è stato piuttosto deludende, come si può vedere dall'immagine sottostante, probabilmente perchè l'oggetto è complicato, presenta molte parti piccole.

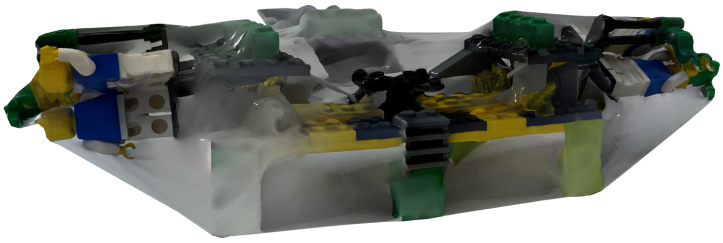

Figura 5.2: Test con light box - Modello 3D: Lego

5.1.2 Alimentatore USB

Figura 5.3: Test con light box - Immagine dell'oggetto: Alimentatore

* **Numero di foto**: 27
* **Lidar**: No
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 46 secondi

* **Risultato**: Il risultato è stato migliore del precedente, considerando che l'oggetto è più lucido, in tinta unita, non è stato utilizzato il lidar e sono state fatte meno foto. L'algoritmo, come si può vedere ha preso anche il piatto rotante perchè è in legno dipinto e presenta nervature.

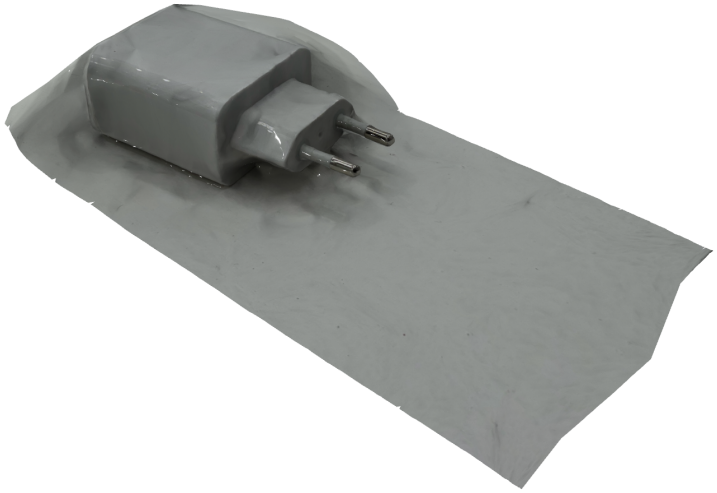

Figura 5.4: Test con light box - Modello 3D: Alimentatore

Ho provato anche con altri oggetti più semplici e non lucidi, tuttavia, utilizzando il light box, con tutti ho riscontrato queste 2 problematiche:

* **Lidar**: per scattare con il lidar serve molto più spazio tra l'iPhone e l'oggetto, con conseguente ripresa anche del piatto;

* **Qualità**: la qualità dei modelli 3D creati dalle foto scattate con il light box è inferiore.

5.2 Test con il piatto rotante

5.2.1 Papa Francesco

Figura 5.5: Test con il piatto rotante- Immagine dell'oggetto: Papa Francesco

* **Numero di foto**: 32
* **Lidar**: Si
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 1 minuto e 40 secondi
* **Risultato**: Il risultato è buono, tuttavia l'oggetto è troppo riflettente quindi si creano queste "bolle" sulla superficie del modello.

Figura 5.6: Test con il piatto rotante - Modello 3D: Papa Francesco

5.2.2 Lego Fiat 500

Figura 5.7: Test con il piatto rotante - Immagine dell'oggetto: Lego Fiat 500

* **Numero di foto**: 54
* **Lidar**: Si
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 3 minuti e 4 secondi
* **Risultato**: Il modello è venuto molto bene considerando la presenza dei finestrini. Per risolvere il problema delle nervature nel piatto rotante ho provato con un foglio di carta, ma se l'oggetto è grande il foglio viene considerato dall'algoritmo come parte integrante del modello.

Figura 5.8: Test con il piatto rotante - Modello 3D: Lego Fiat 500

5.3 Test senza piatto rotante

Le foto sono state catturate posizionando l'oggetto sopra un tavolo bianco e girandoci intorno in una serie di orbite.

5.3.1 Astuccio per occhiali

Figura 5.9: Test senza il piatto rotante - Immagine dell'oggetto: Astuccio per occhiali

* **Numero di foto**: 166
* **Lidar**: Si
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 5 minuti e 58 secondi
* **Risultato**: Qui il piano di appoggio non presenta imperfezioni, infatti non è comparso nel modello. Il modello elaborato è perfetto nella parte frontale e laterale, che sono identiche all'originale. Invece, completamente differente nella parte posteriore, sicuramente a causa di una luce non ottimale.

Figura 5.10: Test senza il piatto rotante - Modello 3D: Astuccio per occhiali

5.3.2 Lego Fiat 500

Figura 5.11: Test senza il piatto rotante - Immagine dell'oggetto: Lego Fiat 500

* **Numero di foto**: 50
* **Lidar**: Si
* **Qualità**: Full
* **Sample Ordering**: Alta
* **Tempo di elaborazione**: 2 minuti e 32 secondi
* **Risultato**: Il modello è molto simile a quello del test alla *Sezione 5.2.2*, ma in questo caso erano state fatte le foto anche alla parte inferiore. Tuttavia, l'algoritmo non ha considerato questo lato perchè non è stato in grado di individuare delle parti in comune con le immagini degli altri lati.

Figura 5.12: Test senza il piatto rotante - Modello 3D: Lego Fiat 500

5.4 Considerazioni finali

Dopo aver testato il processo di creazione di un modello 3D con differenti oggetti e luci, è stato possibile trarre le seguenti conclusioni:

* **Luci**: l'algoritmo produce risultati migliori quando le foto vengono scattate con una luce diffusa e naturale;
* **Piano rotante**: il piano rotante semplifica l'acquisizione delle foto, dal momento che è possibile attaccare il telefono ad un supporto mantenendo la stessa distanza dall'oggetto. Evita anche possibili errori, come foto mosse o troppo spostate l'una dall'altra. Tuttavia, il piano di appoggio deve essere liscio, altrimenti verrà considerato come parte integrante dell'oggetto;
* **Lato inferiore**: il lato inferiore è complicato da scannerizzare, dal momento che l'oggetto deve essere ruotato, dalla foto corrente alla successiva, di pochi gradi e questo non è sempre possibile con oggetti che non rimangono in piedi. Infatti, tra le immagini adiacenti è necessaria una

parte in comune di almeno il 70%, altrimenti l'algoritmo non riuscirà ad individuare i punti in comune;

* **Lidar**: il sensore *Lidar* necessita di almeno 20/30 cm di distanza dall'oggetto. Nel *light box*, a causa della luce artificiale diretta, ha bisogno di una distanza maggiore dall'oggetto.

Riassumendo, i risultati migliori si ottengono quando le foto vengono scattate con un piatto rotante liscio e la luce diffusa.

Capitolo 6

Conclusioni

Il seguente capitolo contiene una valutazione retrospettiva sull'attività di stage.

6.1 Consuntivo finale

Lo stage ha avuto una durata di esattamente 320 ore, come preventivato dal piano di lavoro. È iniziato il 27/04/2022 e si è concluso il 24/06/2022 con una presentazione e demo a tutta l'azienda.

6.2 Raggiungimento degli obiettivi

Come descritto nei capitoli precedenti, il prodotto soddisfa tutti i requisiti individuati durante la fase di analisi dei requisiti, sia obbligatori che desiderabili. Inoltre, sono andato oltre con la realizzazione del piatto rotante, che non era in programma nel piano di lavoro.

6.3 Conoscenze acquisite

Per lo svolgimento di questo progetto sono state fondamentali le nozioni apprese durante il percorso accademico, soprattutto quelle ottenute nel corso di Ingegneria del Software, in cui ho imparato a dare importanza a tutte le attività che vanno a comporre un progetto software e non solo alla loro codifica. In merito alla codifica, nel corso e nel progetto di Ingegneria del Software, ho appreso alcuni design pattern che mi sono stati utili anche per questo progetto di stage.
Oltre alle conoscenze apprese durante il percorso di studi, sono state fondamentali anche quelle pregresse, assimilate dagli innumerevoli hobby, quali: bricolage, stampa 3D e piccoli esercizi con il *Raspberry*.
In aggiunta, per portare a termine il progetto, sono state importanti anche le conoscenze apprese durante lo stage, come:

* **Swift**: necessario per la programmazione in ambito Apple;

* **SwiftUI**: framework, in via di sviluppo, per la creazione di interfacce grafiche con *Swift*. È stato fondamentale per la creazione di entrambe le applicazioni;
* **AWS IoT**: per la gestione delle connessioni attraverso il protocollo MQTT con dispositivi IoT;
* **Nozioni sul 3D**: fondamentali per capire come realizzare l'applicazione macOS.

6.4 Valutazione personale

Il progetto è stato svolto con grande successo e mi ritengo piuttosto soddisfatto dall'esito finale, soprattutto perchè non avevo mai sviluppato un'applicazione mobile e non conoscevo nè il linguaggio *Swift* nè l'ambiente di sviluppo. Sono stato estereffatto dalla velocità con cui ho appreso i concetti per lo sviluppo di applicazioni Apple.
Purtroppo, per ragioni legate alle tempistiche non sono riuscito a realizzare uno strato di comunicazione in cloud tra le due applicazioni, con: autenticazione, area personale e salvataggio in cloud delle sessioni di cattura. Analizzando il progetto con retrospettiva forse avrei preferito portare avanti queste funzionalità, piuttosto che il piatto rotante.
All'interno dell'azienda Zero12 mi sono trovato benissimo e penso che rimarrò qui per approfondire lo sviluppo di applicazioni in ambito Apple.

Acronimi e abbreviazioni

API Application Program Interface. 1

ICT Information and Communication Technologies. 1

IDE Integrated Development Environment. 7

UI User Interface. 7

UML Unified Modeling Language. 13

WWDC2021 Worldwide Developers Conference. 9

Glossario

API REST API REST è una tecnologia che consente di accedere ai dati di un server tramite una richiesta HTTP o HTTPS. 55

Big Data Il termine Big Data si riferisce a dati che contengono una maggiore varietà, che arrivano in volumi crescenti e con più velocità. 1

Bluetooth Low Energy Il Bluetooth Low Energy è una tecnologia wireless progettata per consumare pochissima energia. A differenza del Bluetooth tradizionale il pairing non è obbligatorio e ha un minor numero di canali. 5

Dae Acronimo di "Digital Asset Exchange". DAE è un formato grafico 3D XML aperto adottato come standard internazionale per memorizzare asset 3D nel formato di interscambio COLLADA. Gli asset salvati in formato DAE possono includere fisica, shader, animazione e dati geometrici. 26

Driver Un driver per motori passo-passo è una piccola scheda elettronica che ha il compito di tradurre i messaggi in ingresso in movimento fisico del motore. 5

Editor Programma orientato alla modifica di testo o immagini. 8

fotogrammetria La fotogrammetria è una tecnica che consente di determinare metricamente forma e posizione di oggetti, partendo da almeno due fotogrammi distinti che riprendono lo stesso oggetto. 2

Grafica vettoriale Le immagini vettoriali sono costruite utilizzando punti, linee e curve, il che le rende ideali per essere ingrandite o rimpicciolite per qualsiasi caso d'uso. 8

Internet of Things Internet of Things (IoT) è un neologismo utilizzato per riferirsi al processo di connessione a Internet di oggetti fisici di utilizzo quotidiano. 11

Lidar Lidar è una tecnica di telerilevamento che permette di determinare la distanza di un oggetto o di una superficie utilizzando un impulso laser. 2

Lightbox Il light box è una scatola vuota che permette di scattare fotografie dell'oggetto posizionato al suo interno con uno sfondo bianco o colorato e un'ottima illuminazione. Solitamente viene utilizzato per scattare le fotografie di un prodotto per un e-commerce. 67

Mockup Un mockup è una realizzazione a scopo illustrativo di un oggetto o un sistema, priva delle funzioni dell'originale. 52

MongoDB MongoDB è un database non relazionale basato sui documenti. È classificato come un database di tipo NoSQL. 1

Motori passo-passo Il motore passo-passo o stepper motor è una particolare tipologia di motore sincrono capace di suddividere la propria rotazione in un grande numero di passi. 5

Obj Acronimo di "Object File Format". Obj è un formato di file per la memorizzazione di oggetti 3D. Il formato è stato creato per essere utilizzato con il software 3D Max, ma è anche utilizzabile con altri software 3D. 26

Open Source Con il termine open source ci si riferisce al software rilasciato con una licenza in cui il detentore del copyright concede agli utenti i diritti di utilizzare, studiare, modificare e distribuire il software e il suo codice sorgente a chiunque e per qualsiasi scopo. 9

Pairing Il termine indica il processo di riconoscimento che si attua tra due dispositivi provvisti di tecnologia Bluetooth. Consiste nello scambio e verifica di un codice identificativo al fine di instaurare una connessione. . 54

Pattern publish-subscribe In informatica, l'espressione publish-subscribe si riferisce a un design pattern utilizzato per la comunicazione asincrona fra diversi sistemi. I mittenti e i destinatari dialogano attraverso un broker, che possiamo immaginarlo come una bacheca dove i mittenti pubblicano i messaggi e i destinatari abbonandosi stanno in ascolto per nuovi messaggi. 11

Raspberry Il Raspberry Pi è un piccolo computer a scheda singola sviluppato nel Regno Unito. Ha riscosso un notevole successo per il suo costo contenuto e per i pin GPIO programmabili, con i quali è possibile connettere una vasta gamma di attuatori. 4

SSH In informatica l'SSH è un protocollo che permette di stabilire una sessione remota cifrata, tramite riga di comando, con un altro host della medesima rete informatica. 8

Stl Acronimo di "Stereolithography". Stl è un formato per la memorizzazione di oggetti 3D. È un file che descrive solo la geometria della superficie di un oggetto 3D senza alcuna rappresentazione di colore e texture. Per

questo motivo viene utilizzato nella stampa 3D, dove è importante solo la geometria.. 27

Texture Map Texture map è il processo di definizione delle informazioni sul colore, dei dettagli della superficie e delle proprietà visive di un modello 3D. Le mappe delle texture più utilizzate sono: colore base, normale, rugosità e occlusione ambientale. 10

USDZ USDZ è un formato di file multimediale utilizzato per la realizzazione di oggetti 3D. Il formato è stato creato da Apple e Pixar per consentire la realizzazione di oggetti 3D in formato immagine, ma è anche possibile utilizzarlo per la realizzazione di oggetti 3D in formato video. 26

Bibliografia

Riferimenti alle tecnologie

Di seguito vengono riportati i riferimenti alle tecnologie per lo sviluppo del progetto:

* Swift: `https://www.apple.com/it/swift/`
* SwiftUI: `https://developer.apple.com/xcode/swiftui/`
* API Object Capture: `https://developer.apple.com/augmented-reality/object-capture/`
* RealityKit: `https://developer.apple.com/documentation/realitykit`
* SceneKit: `https://developer.apple.com/documentation/scenekit/`
* RpiMotorLib: `https://github.com/gavinlyonsrepo/RpiMotorLib`
* Python: `https://www.python.org`
* MQTT: `https://mqtt.org`
* ZipArchive: `https://github.com/ZipArchive/ZipArchive`

Riferimenti bibliografici

SwiftUI by Tutorials, By Antonio Bello, Audrey Tam, Bill Morefield and Sarah Reichelt, Fourth Edition, 2021 Razeware LLC

Siti web consultati

* Collegamenti del piatto rotante: `https://iotdesignpro.com/projects/raspberry-pi-stepper-motor-control-through-a-webpage-using-flask`
* SwiftUI: `https://www.hackingwithswift.com`