Università Politecnica delle Marche

Corso di Laurea Magistrale in Ingegneria Informatica e Dell'Automazione

Ristrutturazione edilizia con impianti 3D

Building renovation with 3D systems

Relatore:
Spalazzi Luca

Correlatore:
Francesco Spegni

Laureando:
Matè Felipe

Sommario

1. Introduzione 3
   1.1. Stato dell'arte 3
   1.2. Ambito 3
2. Descrizione del problema 3
3. Progettazione realizzazione del software 6
   3.1. Trasformatore: 7
   3.2. Applicazione web 12
      3.2.1. Sub applicazione ifc_to_gltf 15
      3.2.2. Sub applicazione nfcReader 17
      3.2.3. Sub applicazione workflow 17
   3.3. Applicazione in Unity per HoloLens 20
      3.3.1. Settings 23
      3.3.2. Collegamento con il server 24
      3.3.3. GameManager 24
      3.3.4. Importo della Solution 24
      3.3.5. Riconoscimento tag 25
      3.3.6. Allineamento 26
      3.3.7. Selezione e commenti 30
4. Sperimentazione 32
      4.1.1. Test sull'esperienza dell'utente utilizzando gli HoloLens 32
      4.1.2. Test sull'import dei modelli dentro Unity 33
      4.1.3. Test differenze con HoloLens2 34
5. Conclusioni 35

1. Introduzione

Anche se poche, esistono già idee e applicazioni sviluppate per coinvolgere la realtà mixata nel mondo dell'edilizia. Il lavoro di questa tesi svolta nel dipartimento del DICEA all'Università Politecnica delle Marche, propone come contributo originale il coinvolgimento del BIM, un processo di sviluppo, progettazione, esecuzione e manutenzione di un edificio, con l'uso di modelli 3D e le visualizzazioni di essi nel reale a scala uno a uno, per il miglioramento della capacità di individuare o scoprire punti positivi, a migliorare o a togliere presenti nei modelli dell'edificio, oltre alla capacità di condividere in forma immediata questi aspetti, con gli altri membri o esperti nel progetto edilizio, tutto in una esperienza semplice, dinamica e innovativa.

1.1. Stato dell'arte

Nel mercato attuale il progetto Trimble XR10 ha la maggior somiglianza. Questo progetto è stato sviluppato per la ditta Trimble ed anche è un prodotto che utilizza gli occhiali sviluppati da Microsoft per la realtà mista. Il loro prodotto permette la visualizzazione di modelli 3D in tempo reale e di allinearli manualmente tramite il riconoscimento dei muri del modello. Il progetto svolto propone come aggiunta un allineamento automatico tramite riconoscimento di elementi chiavi più un sistema di legame tra gli esperti progettisti e gli esperti in cantiere, rendendo la comunicazione di modifiche più veloce ed efficace.

Un po' più diverso si trovano progetti come ARKI, software per la visualizzazione di modelli 3D su planimetrie 2D, per mostrare come si vedrebbe il prodotto finale. Il sistema innovativo proposto sulla tesi mostrerebbe il modello 3D con le dimensioni reali e l'allineamento corretto rendendo più semplice l'incontro con errori o problematiche.

1.2. Ambito

L'edilizia è l'insieme delle azioni, tecniche e conoscenze rivolte alla progettazione, realizzazione, modificazioni, riparazioni di costruzioni o più specificatamente di edifici.

Questo campo di studio può essere diffuso in molti mestieri ognuno più fascinante dall'altro, ma allontanandoci dal proposito di questo documento non verranno menzionati, ma è importante dire che è vincolato all'uomo fin dalle sue origini e che accanto a lui si è evoluto nel passare del tempo e che continua finora a farlo dovuto alla continua ambizione senza fine dell'uomo per tecniche e strumenti sempre migliori.

In un mondo attuale dove le tecnologie del 3D e della realtà aumentata o mista cominciano a crescere sempre è impensabile non venga coinvolta nel mondo dell'edile ed è qui dove entra a far parte il progetto proposto in questo documento, per coinvolgere questi due aspetti.

2. Descrizione del problema

Il campo dell'edilizia essendo molto grande e coinvolgono diverse figure professionali, ciascuna per la propria disciplina o area d'interesse e molte volte queste persone rimangono sconnesse tra di loro o ci sono dei dati che possono essere persi o vengono cominciati in tempistiche non immediate. Per questo motivo è di grande importanza che

i professionisti coinvolti possano scambiarsi informazioni, collaborando efficacemente alla realizzazione del progetto.

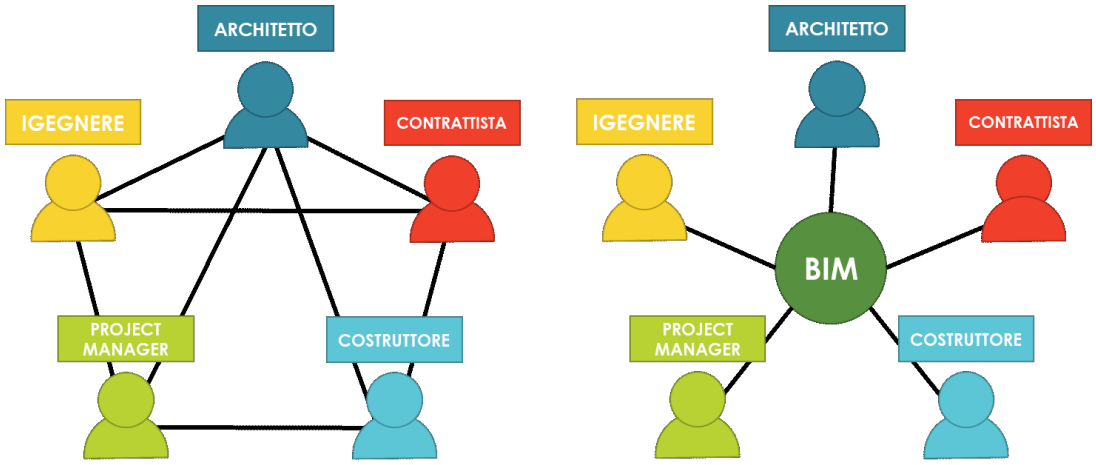

Figura 1: BIM

Per compiere questo proposito è nato il BIM, un processo di sviluppo, progettazione, esecuzione e manutenzione di un edificio che contiene tutte le informazioni che appartengono all'intero ciclo di vita dell'edificio.

È necessario, poi, un formato standard che permetta l'interoperabilità e lo scambio di dati in modo sicuro, senza errori e/o perdite di informazioni. Questo è esattamente lo scopo del formato IFC. Si tratta di un formato aperto, neutro, non controllato dai produttori di software, nato per facilitare l'interoperabilità tra i vari operatori.

Anche con la comparsa del BIM, l'esperto in cantiere che ha il modello dell'edificio deve controllare le possibili incongruenze con il reale e normalmente è l'esperienza chi li permette determinare questi punti a modificare nel modello, ma a volte rimangono in sospese certe aree che dopo diventano modifiche da realizzare.

Per aumentare la velocità con cui analizzare il modello nel reale si propone l'uso di strumenti di realtà mixata dove facilita lo studio del modello sul reale.

Il sistema che viene proposto approfitta le nuove tecniche del BIM e aggiunge la possibilità di visualizzare il modello in scala reale permettendo così la possibilità tramite un unico dispositivo di realtà mixata la visualizzazione e la raccolta delle modifiche, le quali sono in maniera immediata disponibile per le altre persone coinvolte nel progetto.

Gli esperti progettisti si aspettano tramite questa idea innovativa la possibilità di avere tutti i vantaggi proposti dal BIM più il feedback immediato dei cambiamenti da fare tramite file audio lasciati dagli esperti in cantiere.

Gli esperti in cantiere oltre i vantaggi proposti dal BIM guadagnano anche la possibilità di visualizzare nel reale il modello degli esperti progettisti, più la capacità di interagire su di esso rendendo più semplice la capacità di trovare pinti a modificare e tramite lo stesso

strumento poter salvare file audio dei commenti associati ai diversi elementi presenti dentro il modello.

3. Progettazione realizzazione del software

Per lo sviluppo di tutto il sistema si decise di optare per una architettura di tipo orientata ai servizi, dove si centralizza l'informazione in un punto, in questo caso il server installato dentro il dipartimento del DICEA all'Università Politecnica delle Marche e per la visualizzazione dei dati interfaccia web e dei servizi Rest i quali vengono usati dall'applicativo per la visualizzazione in realtà aumentata, sviluppato in Unity 3D per L'HoloLens.

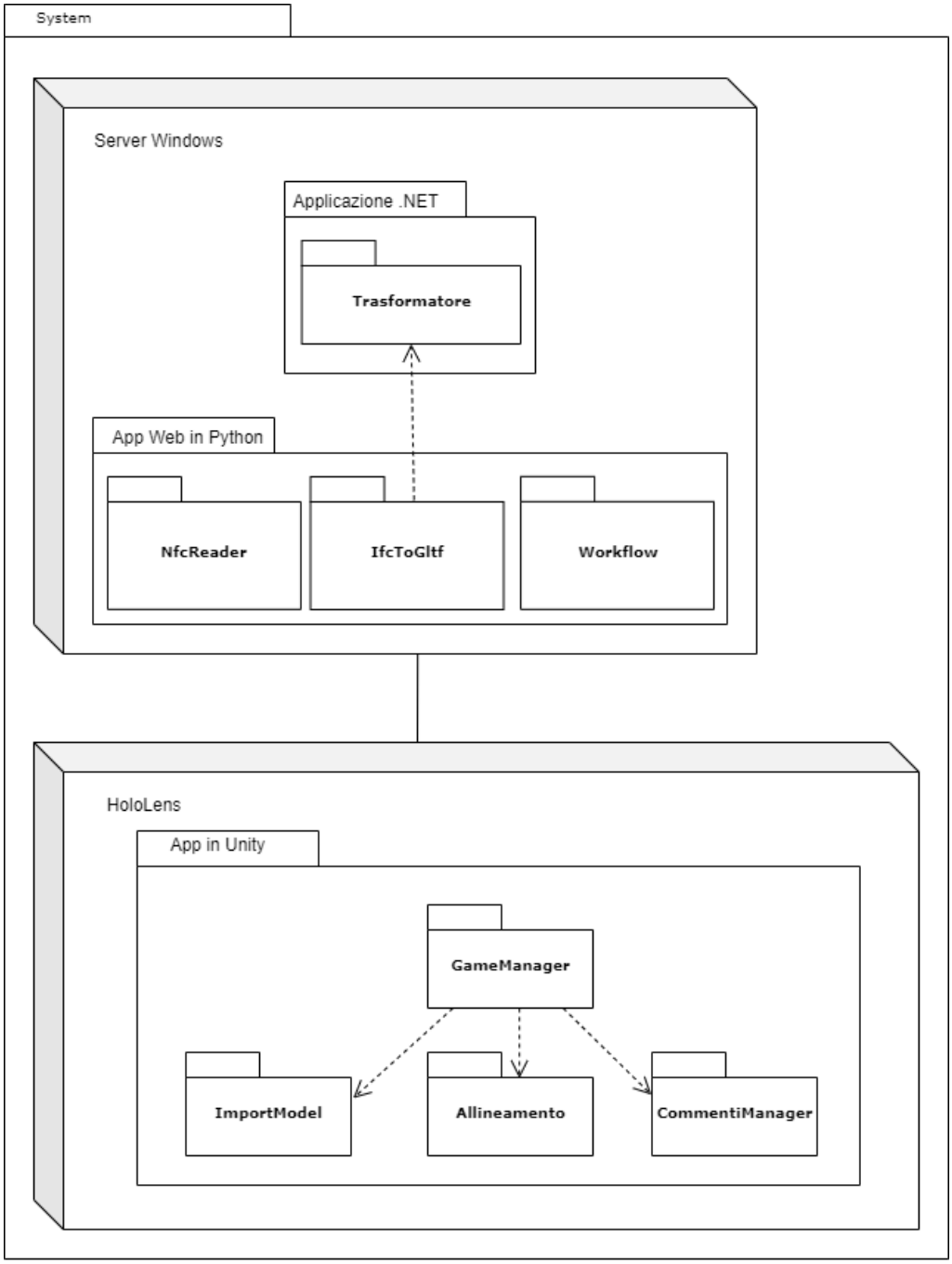

Figura 2: Diagramma di componenti del sistema

Il server compie il ruolo di punto centrale dell'informazione dove tutte le persone coinvolte nel lavoro di edilizia possono rivolgersi per trovare i dati dell'edificio; perciò, si decise di avere una interfaccia web da è possibile estrare e visualizzare questi dati. Anche è il punto centrale dove vengono caricati i file IFC degli edifici.

Ma il formato IFC non è un formato che venga usato come standard nei software o framework per rendering di modelli 3D o sviluppo di applicazioni in ambiente 3D. Allora è stato necessario lo sviluppo di un convertitore dei file IFC a un formato più standard come il glTF.

Per la traduzione di questo file è stato usato il progetto opensource di XbimGltf il cui genera un file glTF con la struttura basica del modello e un file Json con l'informazione che verrebbe persa, visto che non è compresa dentro il glTF.

Come il progetto riportato prima è stato sviluppato in .NET Framework 4.7, ed è una componente essenziale in tutto il sistema è stato deciso che il server sarebbe una macchina virtuale in windows, evitando così problemi di incompatibilità al momento di eseguire la traduzione dei file.

Una volta deciso il server è stato deciso che la piattaforma web sarebbe stata implementata con Python di preciso con DJango per la simplicità e velocità con la quale si può implementare una applicazione web e dei servizi Rest e oltre per la possibilità che a futuro venga aggiornato l'applicazione di traduzione a .NET Core o sostituito e quindi il server sia cambiato un machina virtuale in Linux.

Per ultimo l'applicazione in realtà mixata sarà sviluppata per HoloLens perché Microsoft a mostrato in grande interesse nel continuo sviluppo dello strumento, creando così una grande possibilità di supporto, usando il framework Unity 3D visto che esiste tutto un set di strumenti per lo sviluppo su HolloLens proposto dello stesso fornitore Microsoft.

Di seguito verrà spiegato in dettaglio il lavoro svolto il quale è suddiviso in tre progetti anche se è composto di due grandi componenti, piattaforma web e applicazione HoloLens, perché come detto precedentemente è stato necessario la costruzione di un sistema di traduzione tra file IFC a glTF.

- Applicazione trasformatore dei file ifc a gltf e json
- Applicazione web (Django)
- Applicazione HoloLens (Unity)

3.1. Trasformatore:

Applicazione costruita in **C#**, **.NET Framework 4.7** come un eseguibile, basata sul progetto **XbimGltf** il quali si può trovare nel seguente repository su GitHub https://github.com/xBimTeam/XbimGltf.

Svolge il compito di leggere un file **ifc** e di generare una traduzione **gltf** con la informazione degli oggetti 3D, come dimensione e materiale ed un file **json** con caratteristiche più dettagliate riportate nel file originale.

Lo schema del **json** generato è il seguente:

```
{
    "type": "object",
    "properties": {
        "geometryExporterVersion": {
            "type": "string"
        },
        "uniZiteMetadataExporterVersion": {
            "type": "string"
        },
        "ifcSchemaVersion": {
            "type": "string"
        },
        "elements": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "ifcSystemIndex": {
                        "type": "integer"
                    },
                    "ifcStoreyIndex": {
                        "type": "integer"
                    },
                    "ifcSpaceIndex": {
                        "type": "integer"
                    },
                    "propertySets": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "propertySetName": {
                                    "type": "string"
                                },
                                "properties": {
                                    "type": "array",
                                    "items": {
                                        "type": "object",
                                        "properties": {
                                            "Name": {
                                                "type": "object"
                                            },
                                            "Value": {
                                                "type": "object"
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    },
```

```
                    "ifcExpressTypeName": {
                        "type": "string"
                    },
                    "ifcGuid": {
                        "type": "string"
                    },

                    "ifcEntityLabel": {
                        "type": "integer"
                    },
                   "ifcInfo": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "Name": {
                                    "type": "string"
                                },
                                "Value": {
                                    "type": "string"
                                }
                            }
                        }
                    }
                }
            }
        },
        "systems": {
            "type": "array",
            "items": {
                "ifcExpressTypeName": {
                    "type": "string"
                },
                "ifcGuid": {
                    "type": "string"
                },
                "ifcEntityLabel": {
                    "type": "integer"
                },
                "ifcInfo": {
                    "type": "array",
                    "properties": {
                        "Name": {
                            "type": "string"
                        },
                        "Value": {
                            "type": "string"
                        },
                    }
                }
            }
        },
        "storeys": {
            "type": "string",
            "items": {
                "Elevation": {
                    "type": "integer"
                },
```

```
                "ifcExpressTypeName": {
                    "type": "string"
                },
                "ifcGuid": {
                    "type": "string"
                },
                "ifcEntityLabel": {
                    "type": "integer"
                },
                "ifcInfo": {
                    "type": "array",
                    "properties": {
                        "Name": {
                            "type": "string"
                        },
                        "Value": {
                            "type": "string"
                        },
                    }
                }
            }
        }
    }
}
```

La voce principale che viene utilizzata nell'applicazione in Unity che va installata negli HoloLens è **"elements"**. In questa voce vengono riportati tutti gli elementi presenti dentro il modello. Dentro questa voce principale troviamo la famiglia alla cui appartiene l'elemento, l'identificativo id, il numero della linea dove si trova dentro il file ifc, il nome dell'oggetto e un altro l'identificativo unico.

- **ifcExpressTypeName:** Usato per determinare a quale famiglia appartiene l'oggetto. (fornitura, system, envelope, etc.)
- **ifcGuid:** Il globale identificatore GUID, standard dentro i file ifc.
- **ifcEntityLabel:** Il numero della linea dell'oggetto corrispondente dentro il file ifc.
- **ifcInfo:** Usato per determinare il nome e l'id unico dell'oggetto.

Di seguito un esempio di un file json.

```
{
  "geometryExporterVersion": "1.2",
  "uniZiteMetadataExporterVersion": "1.2",
  "ifcSchemaVersion": "ifc2x3",
  "elements": [
    {
      "ifcSystemIndex": 0,
      "ifcStoreyIndex": -1,
      "ifcSpaceIndex": -1,
      "propertySets": [
        {
          "propertySetName": "Pset_DistributionFlowElementCommon",
          "properties": [
            {
```

```
              "Name": "Reference",
              "Value": "45 Degree"
            }
          ]
        }
      ],
      "ifcExpressTypeName": "IfcFlowFitting",
      "ifcGuid": "1CDrloTIv6OfT6JLKHlQe3",
      "ifcEntityLabel": 10679,
      "ifcInfo": [
        {
          "Name": "name",
          "Value": "M_Rectangular to Round Transition - Angle:45 Degree:954220"
        },
        {
          "Name": "ObjectType",
          "Value": "M_Rectangular to Round Transition - Angle:45 Degree"
        }
      ]
    }
  ],
  "systems": [
    {
      "ifcExpressTypeName": "IfcSystem",
      "ifcGuid": "36rPknJw90wRCl3GfSBIUy",
      "ifcEntityLabel": 94387,
      "ifcInfo": [
        {
          "Name": "name",
          "Value": "Mechanical Supply Air"
        },
        {
          "Name": "ObjectType",
          "Value": "Supply Air"
        }
      ]
    }
  ],
  "storeys": [
    {
      "Elevation": 300.0,
      "ifcExpressTypeName": "IfcBuildingStorey",
      "ifcGuid": "1XxXZlR7L40BDbToLm9VGH",
      "ifcEntityLabel": 151,
      "ifcInfo": [
        {
          "Name": "name",
          "Value": "Ceiling - Existing"
        },
        {
          "Name": "ObjectType",
          "Value": "Livello:8mm Head"
        }
      ]
    }
  ]
}
```

L’applicazione riceve un minimo di 2 parametri e un massimo di 4 parametri i quali sono:

1. Codice indicando se il file a trasformare si trova nello stesso computer o se deve essere scaricato da un indirizzo web. (“url” o “path”)
2. Indirizzo web o indirizzo locale dove si trova il file a tradurre.
3. Indirizzo dove devono essere salvati prodotti dopo la traduzione, se il parametro non esiste i file verranno salvati in una cartella predisposta dal software.
4. Nome della cartella e dei file prodotti dopo la traduzione.

Di seguito viene riportata la struttura gerarchica:

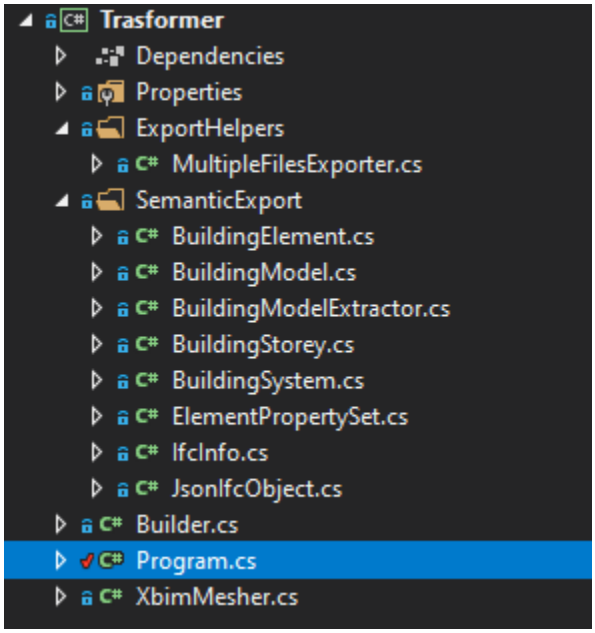

Figura 3: Gerarchia Trasformer

La classe ‘Program’ svolge il ruolo di classe principale, Una volta determinati che i parametri sono corretti scarica il file **ifc** ed inizia con la creazione del file **gltf** e **json**.

Come possibili miglioramenti si suggerisce la implementazione di un event log del processo per tenere traccia del processo e la implementazione di un processo asincrono per rendere la traduzione più veloce.

3.2. Applicazione web

Applicazione sviluppata con **Python 3.6.10**, **Django REST framework** e database **sqlite3**.

Dentro il progetto sono state specificate le diverse dipendenze con le versioni usate da installare per il corretto funzionamento dell’applicazione, dentro il file “requirements.txt”, le quali sono le seguenti.

- Django==3.1.5
- djangorestframework==3.12.2
- django-filter==2.4.0
- django-sslserver==0.22
- gevent==21.1.2
- fleep==1.0.1

Tutte queste possono essere facilmente installate con il comando **pip install**.

L'applicazione è stata installata assieme al componente precedente in un server dentro il dipartimento ed è stato deciso di usare Python di preciso Django per la semplicità d'implementare un server e di lavorare la serializzazione e deserializzazione dei dati.

Sono stati implementati in totale otto entità per la gestione dei dati più il sistema di autenticazione base di Django.

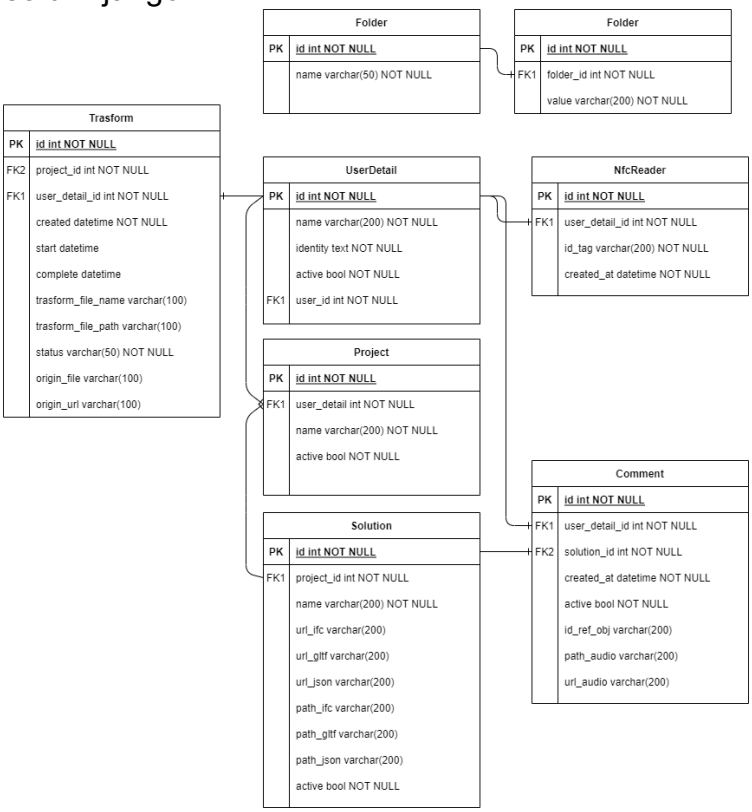

Figura 4: Diagramma ER applicazione web

Anche se è una unica applicazioni esistono tre diverse applicazioni che la compongono:

- **ifc_to_gltf:** Sezione incaricata di tradurre i file ifc a gltf
- **nfc:** Sezione dove vengono riportate le letture fatte dei sensori nfc.
- **workflow:** Sezione gestisce gli utenti e che categorizza i modelli in progetti.

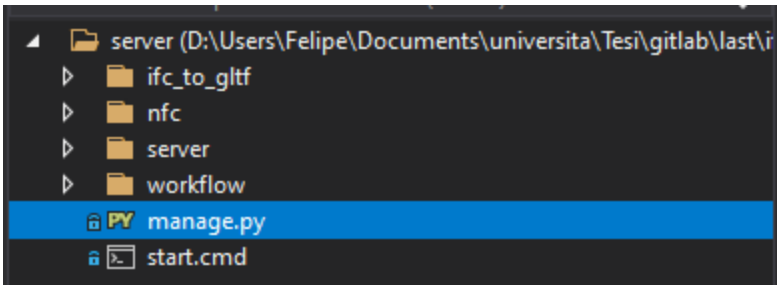

Figura 5: Gerarchia applicazione web

La sezione server gestisce le configurazioni principali del server.

Ogni sub applicazione contiene i seguenti file:

- **models.py:** dove si gestiscono le entità dentro il DB.
- **serializers.py:** dove viene gestita la serializzazione dei modelli.
- **admin.py:** dove gestire la presenza dentro l'area riservata nel portale web.
- **urls.py:** dove gestire i percorsi delle request.
- **views.py:** dove vengono gestite le request e le azioni da eseguire.

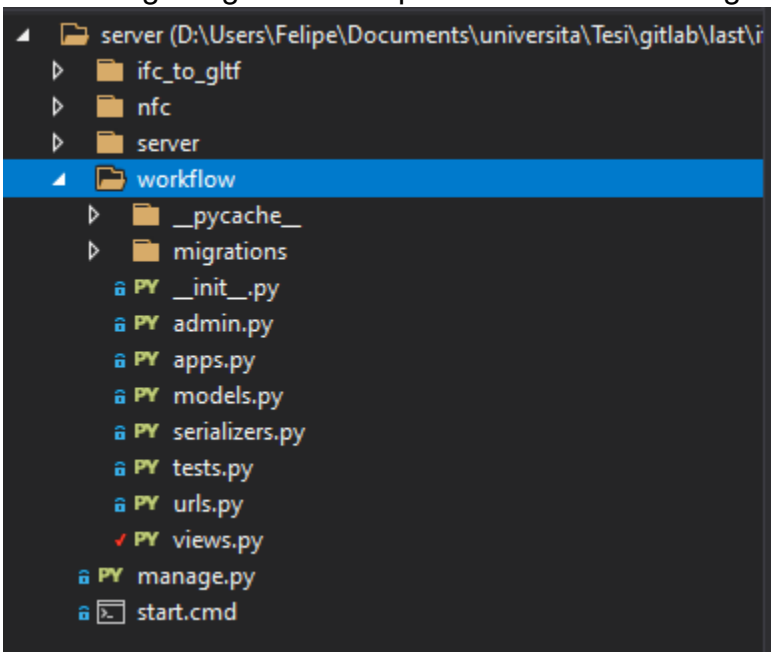

Figura 6: Applicazione web sub applicazioni files

Di seguito verranno spiegati i diversi funzionamenti per ogni singola sub applicazione ma un esempio del funzionamento verrà riportato nella sezione di "Sperimentazione".

3.2.1. Sub applicazione ifc_to_gltf

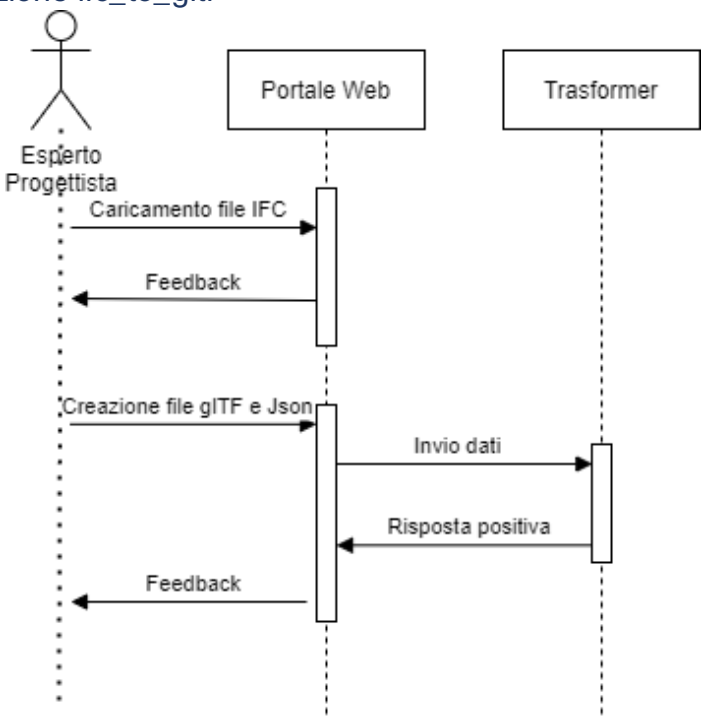

Figura 7: Diagramma di sequenza ifcToGltf

Nella sub applicazione ifc_to_gltf viene gestita la traduzione del file ifc a gltf, viene creato un nuovo registro dentro Trasform con il path del file ifc salvato dentro il server.

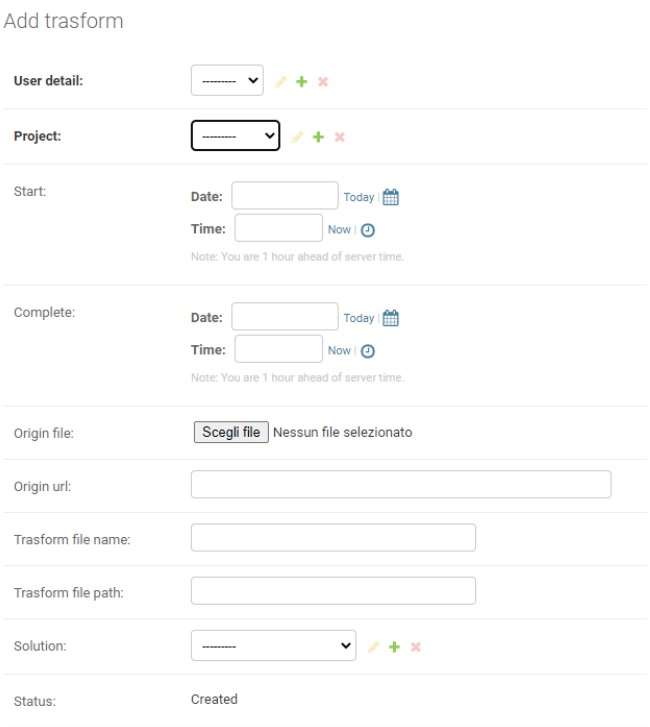

Figura 8: Applicazione web add Trasform

Il file ifc può essere caricato direttamente dal computer o può essere anche condiviso l'indirizzo web dove si trova.

Quando viene richiesta la funzione di traduzione, l'attuale componente chiamata al componente precedentemente spiegato per costruire così i file gltf e json rispettivi i cui path vengono salvati come una nuova "Solution" ed il riferimento di questa viene salvata dentro "Trasform".

```
def convert(self):
    msg = ""
    # Determina il folder dove salvare i file trasformati
    path_file = self.trasform_file_path
    if not self.trasform_file_path:
        os.makedirs(settings.WORKFLOW_BUILD_SOURCE_PATH + "/" + self.trasform_file_name, exist_ok=True)
        path_file = settings.WORKFLOW_BUILD_SOURCE_PATH + "/" + self.trasform_file_name

    if self.origin_url is not None and len(self.origin_url) > 0:
        origin = self.origin_url
        code = "url"
    elif self.origin_file:
        origin = self.origin_file.path
        code = "path"
    else:
        raise ValueError("You must specify a file path or an URL")


    self.start = datetime.utcnow().replace(tzinfo=pytz.utc)

    cmd_path = None

    if settings.TRASFORMER_EXE_PATH:
        cmd_path = settings.TRASFORMER_EXE_PATH

        # Determina l'esecuzione del Trasfomer e i suoi parametri
        app = "%s %s %s %s %s" % (settings.TRASFORMER_EXE_PATH, code, origin, path_file, self.trasform_file_name)

        print("Launch process: %s (working dir: %s)" % (app, settings.TRASFORMER_ROOT,))
        p = subprocess.Popen(app, cwd=settings.TRASFORMER_ROOT, shell=True)
        print("subprocess.Popen returned %s" % p)
        p.communicate()
        res = p.poll()

    else:
        print("No process specified. Simulate the conversion ...")
        res = 0


    if res != 0:
        # print(":(")
        self.complete = datetime.utcnow().replace(tzinfo=pytz.utc)
        self.status = "Error (%s)" % res
        msg = "Errore trasformando il file"
    else:
        # print(":)")
        self.complete = datetime.utcnow().replace(tzinfo=pytz.utc)
        self.status = "Complete"
        msg = "Il file è stato trasformato"
        self.solution = self.create_solution(path_file, code)

    self.save()
    return msg
```

Figura 9: Applicazione web chiamata Trasformer

3.2.2. Sub applicazione nfcReader

Nella sub applicazione nfc vengono gestite tutte le letture fatte in cantiere dei sensori nfc, salvando così l'id corrispondente di ogni sensore.

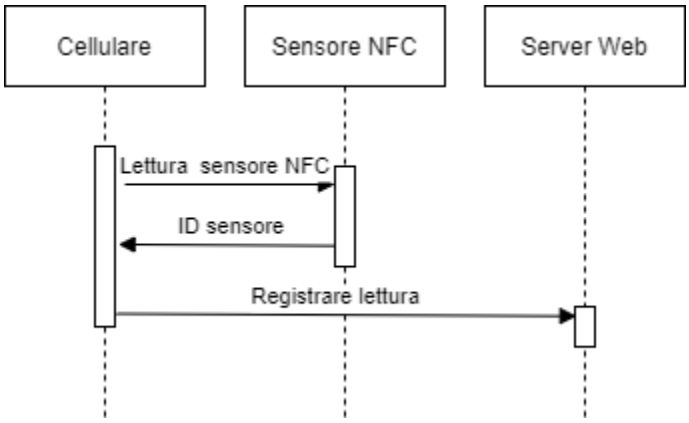

Figura 10: Diagramma di sequenza NFCReader

3.2.3. Sub applicazione workflow

In questa sub applicazione vengono gestiti tutti i dati che sono inviati all'HoloLens come i diversi progetti esistenti, le soluzioni associati ad essi, i commenti e la suddivisione in categoria degli elementi esistenti dentro ogni soluzione.

Per primo è importante dire che il modello costruito dall'esperto progettista e trasformato in file gltf e json viene salvato come una "Solution", la quale è associata a un "Project".

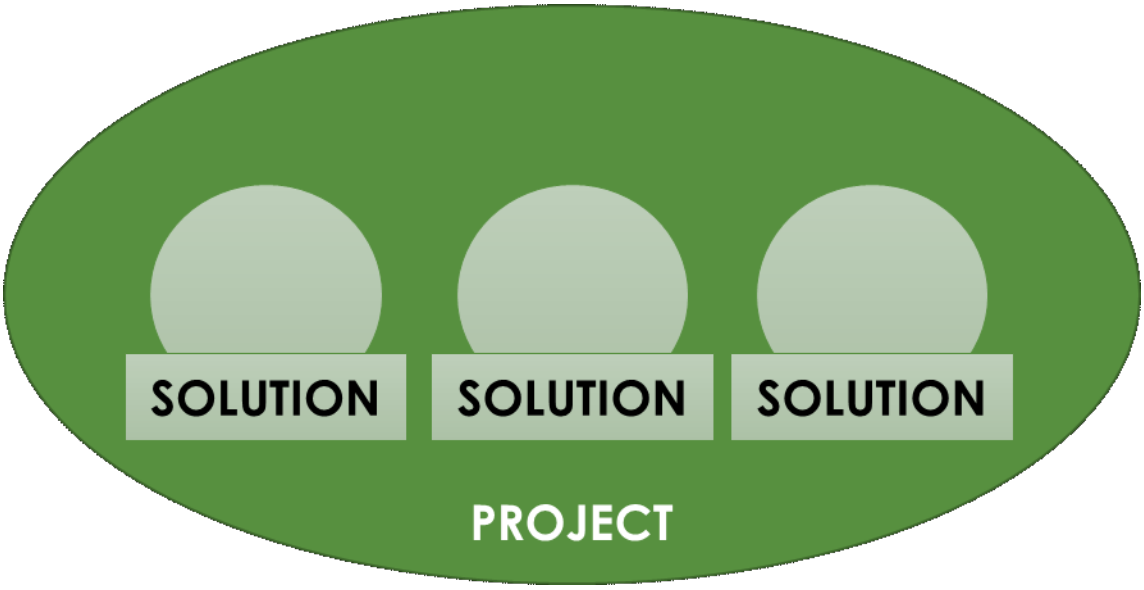

Figura 11: Diagramma generale solution-project

Questa suddivisione è stata fatta con il principio che diversi modelli possono essere associate a un cantiere o ad un progetto in particolare e non è possibile mostrare tutti i

modelli in contemporaneo, per incongruenza tra di loro e per limitazioni fisiche come memoria insufficiente.

Come si vede nel diagramma ER, il “Project” è associato ad un utente, evitando così che gli utenti abbiano accesso ai progetti e le soluzioni che non gli corrispondono.

Nel seguente diagramma di flusso viene mostrata la sequenza come viene fatto l’importo del modello dentro L’HoloLens.

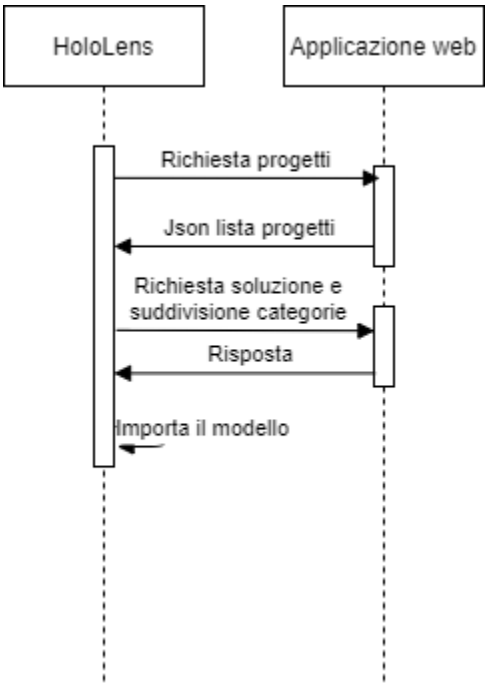

Figura 12: Diagramma di sequenza Import Model

Alla fine di questo processo si ottiene il seguente GameObject dentro l’applicazione di Unity con la seguente gerarchia.

Figura 13: Gerarchia categoria GameObject model

La suddivisione di questa gerarchia viene stabilita dalle entità “Folder” e “Expresion” dentro il DB, dove la prima contiene la categoria e la seconda l’espressione regolare ad usare per determinare quale elemento dentro il modello è compreso dentro la categoria.

L’espressione regolare viene applicata alla voce ‘ifcExpressTypeName’ che esiste dentro il json della soluzione e se coincide viene associato alla categoria corrispondente. Tutti gli oggetti che non appartengano ad una categoria vengono messi direttamente sulla categoria “Others”. Esiste una unica categoria riservata, la quale espressione e nome non possono cambiare la cui è TAG, perché vengono usati posteriormente per l’allineamento.

Questa gerarchia anche permette la customizzazione del menu dentro HoloLens, facendo comparire bottoni per ogni categoria e al premerli nasconde o mostra gli elementi compresi dentro la categoria scelta.

Figura 14: Menu Categorie HoloLens

Per ultimo il server salva i commenti associati ad una soluzione e ad un oggetto dentro di essa. Questi possono essere ascoltati direttamente dall’applicazione web o vengono trasmessi all’applicazione HoloLens come mostrato nel seguente diagramma.

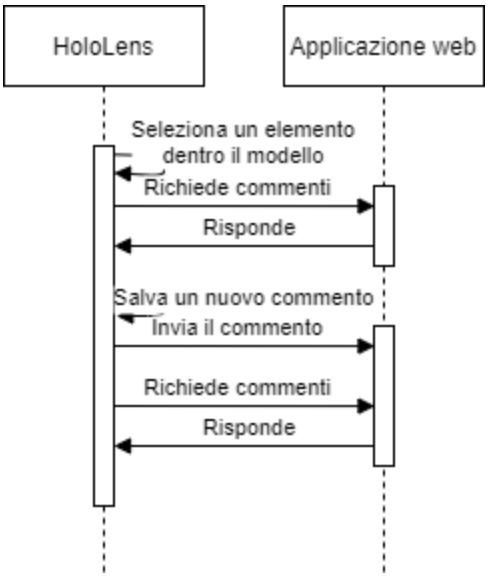

Figura 15: Diagramma di sequenza commenti

3.3. Applicazione in Unity per HoloLens

L'applicazione risultate di questo lavoro è la continuazione di un progetto precedente sviluppato nel dipartimento DICEA all'Università Politecnica delle Marche usando **Unity 2019.4.** Per evitare problemi di compatibilità di versioni e stato deciso di continuare lo sviluppo in questa versione di Unity.

Di seguito una lista con i diversi assets e le sue versioni usati:

- **Piglet: glTF Importer**, per l'importazione dei file glTF dentro Unity, scaricabile dalla Unity asset store.
- **Microsoft Mixed Reality Toolkit v2.5 (MRTK)** essenziale per lo sviluppo di applicazioni con HoloLens. Scaricabile dal repository ufficiale in GitHub.
- **Vuforia Engine 9.6.4** per il riconoscimento d'immagini principali per l'allineamento del modello.

Come IDE per lo sviluppo degli script è stato usato **Visual Studio Community 2019** e il suo complemento per Unity.

L'applicazione contiene una unica Scena principale composta dei seguenti elementi:

- **Directional Light:** GameObject per illuminare l'intera scena.
- **MixedReality Toolkit:** GameObject base per il supporto con HoloLens
- **MixedReality Playspace:** GameObject base per il supporto con HoloLens e contenitore della camera principale.
- **Main Camera:** Camera principale della scena ed incorporata con gli script di Vuforia per il riconoscimento d'immagini.
- **LoadingProcess:** GameObject estetico per gestire la transizione della scarica dei dati.

- **LoadingProcess_2:** GameObject estetico per gestire la transizione della scarica dei dati.
- **LogCanvas:** GameObject per gestire gli errori in ambiente di Test
- **Bounding:** GameObject per gestire l'allineamento in forma manuale.
- **ImageTarget:** GameObject per la gestione della visualizzazione dei tag.
- **QCode:** Tag dentro della scena
- **Player:** GameObject contenitore dei manager.
- **GameManager:** GameObject con lo script principale che gestisce tutta la scena.
- **SelectionManager:** GameObject con lo script per la selezione di un oggetto.
- **Menu:** GameObject contenitore dei vari menu per l'utente

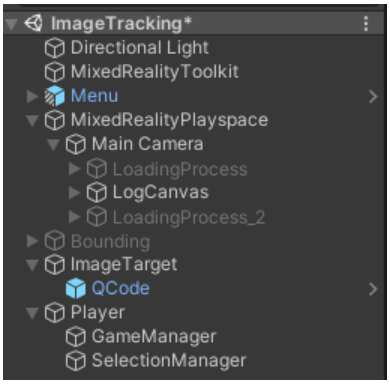

Figura 16: Gerarchia scena

Oltre ad usare i complementi aggiuntivi di Vuforia e MRTK, gli script implementati vengono riportati nei seguenti due diagrammi di classi. Sono stati divisi in due categorie tra Modelli e Componenti Logici, dove i primi sono usati per il ricevimento e l'invio d'informazione al server e il secondo l'implementazione logica usata.

Figura 17: Diagramma delle classi – Componenti logici

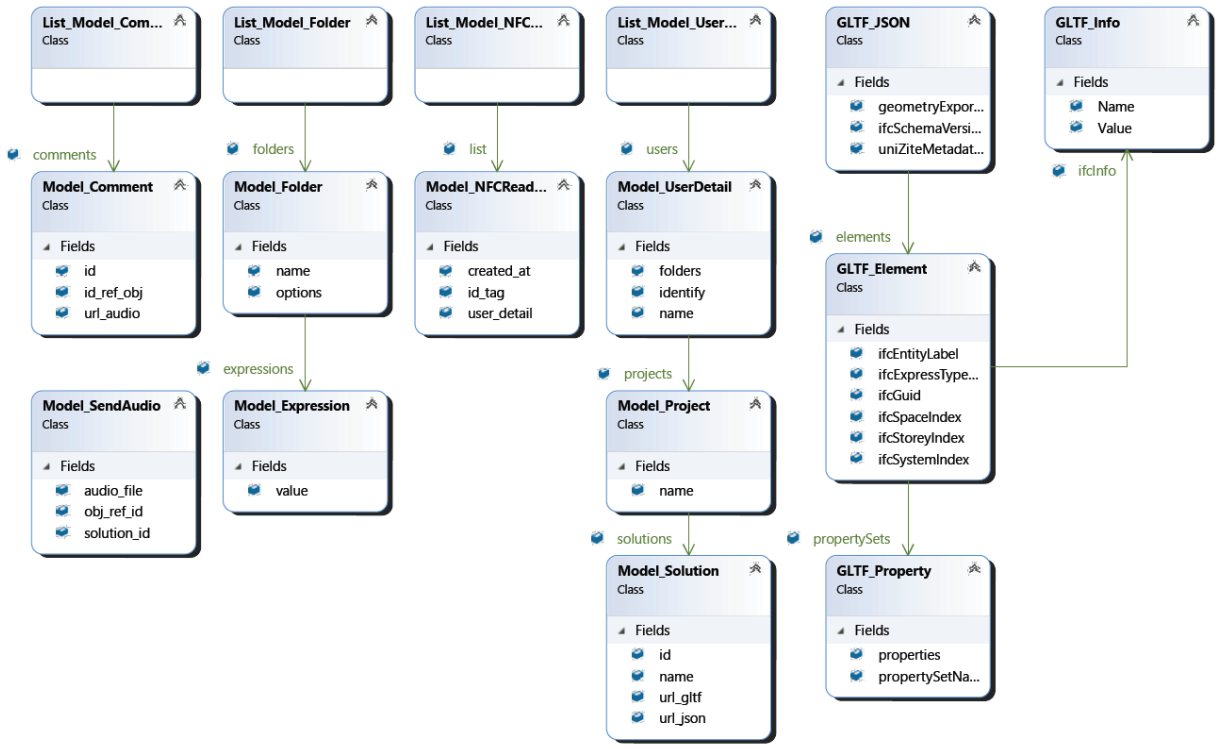

Figura 18: Diagramma delle classi - Modelli

Le classi di Unity e dei complementi MRTK e di Vuforia non vengono riportate nei diagrammi.

Per una più semplice comprensione del funzionamento dell'applicazione è stato diviso la spiegazione in sotto temi.

3.3.1. Settings

Sono i dati configurazione precaricati obbligatori che permette l'avvio dell'applicazione i quali suddividono in caratteristiche funzionali ed estetiche.

In quanto alle funzionali sono tutte quelle che cambiano il comportamento attuale dell'applicazione:

- l'indirizzo del server web
- l'utente
- la password
- l'estensione con cui verrà salvato l'audio
- il tempo di registrazione audio massima per commento.

Le funzioni estetiche sono tutte quelle che alterano il modo come vengono visualizzate le cose.

- numero di colonne per il menu dei commenti

- valore in decimale della trasparenza di un oggetto quando viene nascosto

La suddivisione in queste due categorie è stata fatta per una futura configurazione dove l'utente poteva gestire a proprio gusto i colori dei menu, la trasparenza degli oggetti nascosti, il colore dell'elemento selezionato, etc. mentre le altre impostazioni venivano gestite direttamente dal sistema.

Attualmente i settings sono impostati al momento della compilazione dell'applicazione, non rendendo così possibile il cambiamento di essi.

3.3.2. Collegamento con il server

L'applicativo in Unity deve poter collegarsi al server sia per richiedere o inviare i dati necessari allo stesso tempo che non deve bloccarsi mentre aspetta la risposta del server. Per questo motivo che ogni richiesta è stata implementata in modo asincrona e in contemporaneo mostra una icona di caricamento per migliorare l'esperienza all'utente ed informarlo del processo.

3.3.3. GameManager

Il GameManager è uno degli script principali di tutta l'applicazione i quale svolge le due funzioni descritte prima; quindi, è lui l'incaricato di caricare i settings e ogni chiamata al server viene fatta da lui.

Oltre a queste due funzioni s'incarica anche di gestire i tag riconosciuti e funzioni extra dentro i componenti di Vuforia e del MRTK, ad esempio stoppare ed avviare il riconoscimento d'immagini di Vuforia, nascondere la mesh costruita dal MRTK o di aggiungere o togliere ancore al modello. Un'ancora spaziale rappresenta un punto importante nel mondo che il sistema segue nel tempo. Ogni ancoraggio ha un sistema di coordinate regolabile, basato su altri ancoraggi o sistemi di riferimento, per garantire che gli ologrammi ancorati rimangano esattamente al loro posto.

Questo schema c'centralizzato è stato implementato per permettere ad altre funzione chiamare queste funzioni senza dover implementarle ogni volta e per rendere più semplice la gestione di cambi a futuro.

3.3.4. Importo della Solution

Una volta l'utente abbia scelto dal menu la Solution che vuole visualizzare lo script ImportModel tramite il GameManager scarica dal server i file glTF e Json più la suddivisione in categorie.

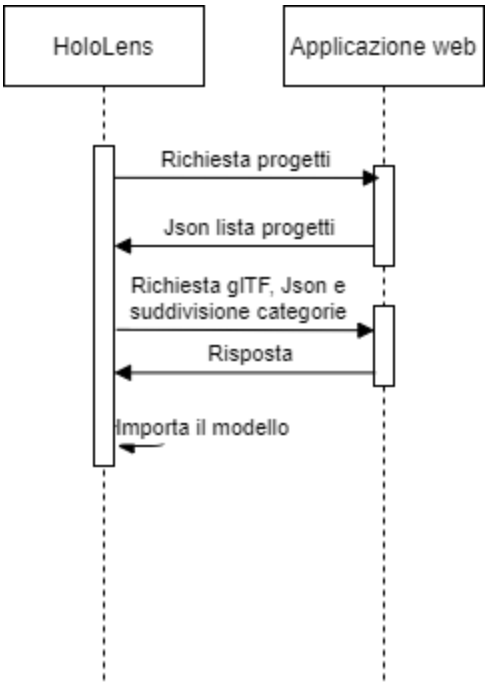

Figura 19: Diagramma di sequenza importo soluzione

Tramite Piglet il glTF viene convertito in una lista di componenti riconoscibile per Unity mentre il Json viene salvato dentro la classe GLTF_JSON. Per prima cosa dopo lo scaricamento viene fatta la creazione di un dizionario con chiave una stringa con il nome della categoria e come oggetto il GameObject che contenderà gli elementi del modello associati alla categoria, per poi cercare dentro la lista di componenti se il suo nome compie con alcuna espressione e determinare a quale categoria appartiene ed associarlo.

Alla fine di tutto questo processo si ottiene un unico GameObject, il modello, con dentro i GameObject di ogni categoria e i corrispondenti elementi della Solution, e importante dire che viene anche creata la categoria Other per ogni elemento non appartenente ad una categoria.

Mentre avviene tutto questo processo, l'utente vede una icona indicando il caricamento,

3.3.5. Riconoscimento tag

Per prima cosa è importante dire che i tag sono punti di riferimento essenziali per poter ubicare il modello e muoverlo e ruotarlo nella posizione corretta. Questi tag sono dei sensori NFC con informazione del ID, che verranno inseriti dentro le pareti e che per la loro lettura verrà utilizzata una funzione aggiuntiva al componente web, ma che attualmente è ancora in sviluppo. Questa funzione ha il compito di rilevare l'ID associato al sensore ed inviarlo al server chi tiene traccia di ogni lettura fatta e di mostrare l'immagine associata al tag.

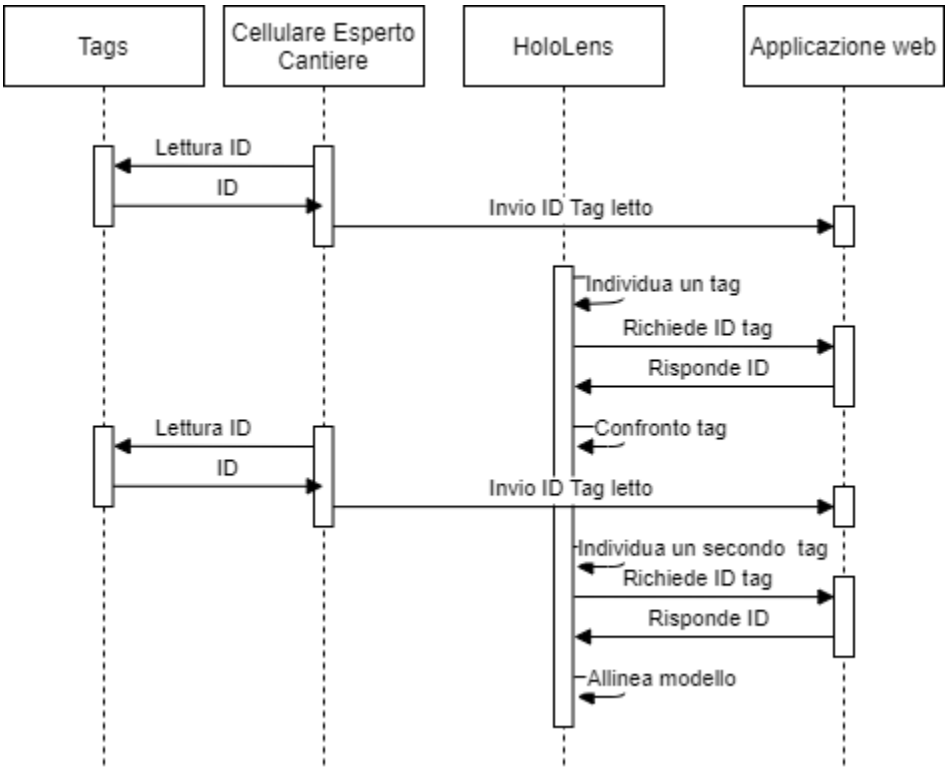

Figura 20: Diagramma di sequenza riconoscimento tag e allineamento

Il riconoscimento dei tag viene dopo che una Solution è stata importata dentro la scena. Lo script DefaultTracklaEventHandler è l'incaricato di gestire questa parte e appena la telecamera dell'HoloLens inquadra il tag e la riconosce chiede al server tramite il GameManager l'ultimo ID trovato per confrontarlo con i tag dentro la Solution importata e determinare quale sia.

Se il tag è stato trovato dentro il modello, allora l'utente può salvare la posizione premendo il rettangolo del tag che compare difronte a lui. Questo rettangolo cambia di colore a verde se la posizione è stata salvata o a rosso se è stata rimossa la posizione. Una volta salvati due tag diversi è possibile avviare l'allineamento del modello.

3.3.6. Allineamento

Trovati due tag e salvate le sue posizioni avviene l'allineamento del modello, il quale viene fatto dallo script AlingModelToIMG. Per capire come avviene questo è prima necessario capire che in un ambiente 3D lo spazio viene identificato da un piano cartesiano a tre assi ed ogni elemento dentro la scena ha delle coordinate che lo ubicano dentro di essa e normalmente queste coordinate sono associate al punto di origine dell'elemento.

Normalmente il punto d'origine di un elemento è associato al centro di esso ma questo può cambiare a seconda di come è stato costruito.

Un GameObject composto da altri GameObject, ad esempio un modello, il quale è composto da diversi elementi comprende anche esso delle coordinate e se vengono cambiate, tutti gli elementi compresi dentro vengono spostati insieme.

Con le due premesse precedenti possiamo spiegare l'allineamento. Per prima cosa viene creato un GameObject, al cui chiameremo dummy, con coordinate uguali ha quelle del modello importato e vengono anche copiati tutti i tag esistenti di esso sul dummy, rispettando le coordinate di ognuno.

Quando viene salvato il secondo tag, il dummy viene spostato al centro della scena, coordinate (0,0,0) per questione di precisione.

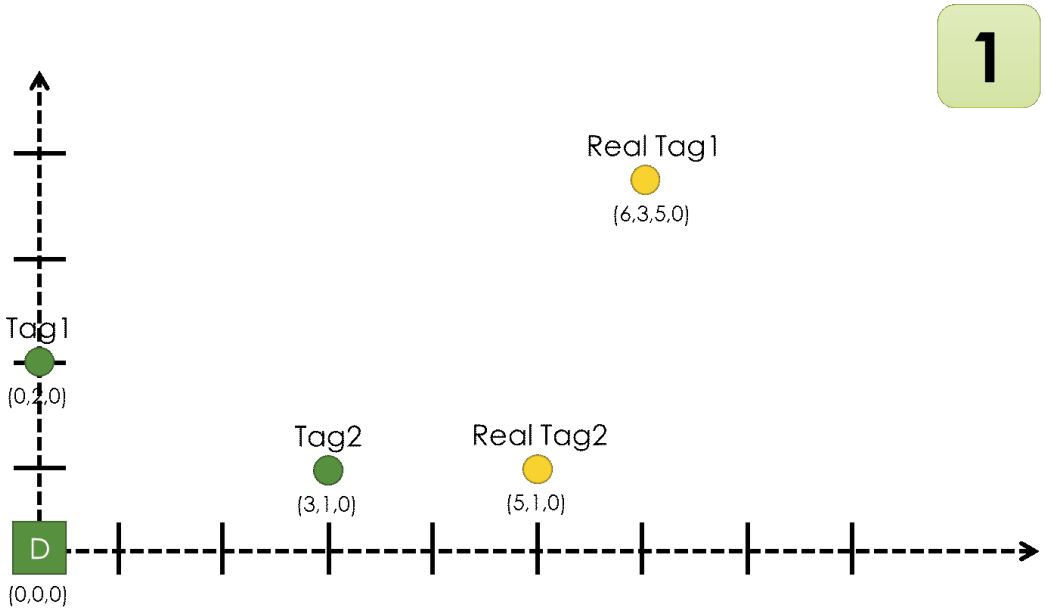

Figura 21: Calcolo allineamento 1

Vengono calcolato il vettore tra il primo tag trovato e il secondo, sia per i tag dentro il dummy come per quelli trovati e salvati.

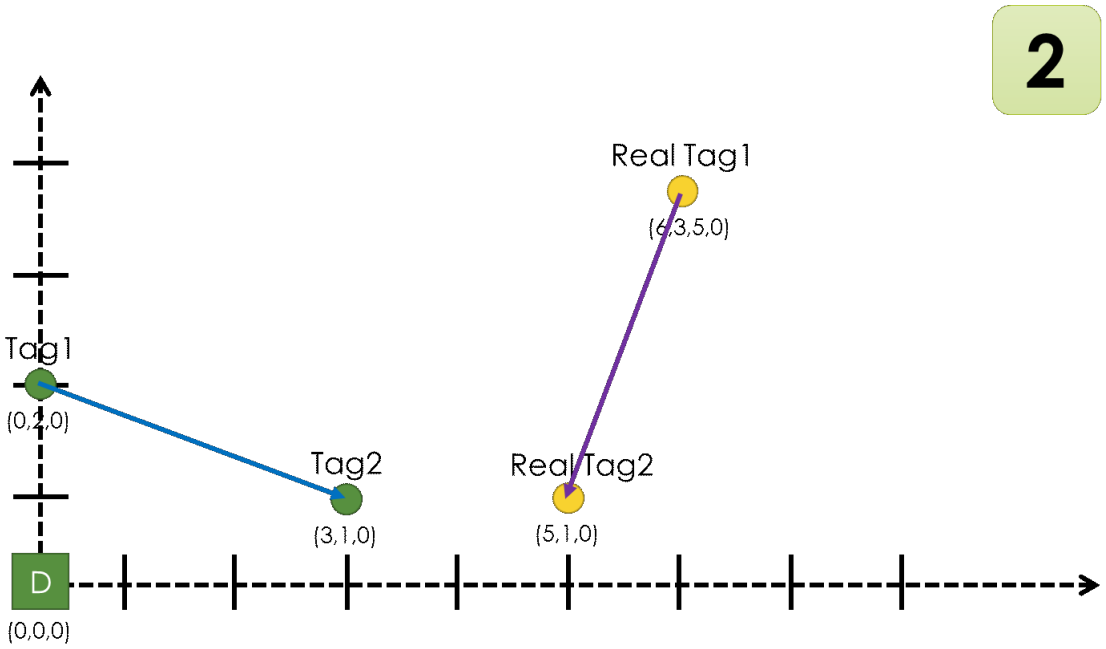

Figura 22: Calcolo allineamento 2

Appena calcolati i due vettori, viene calcolata la rotazione dal vettore dummy, quello dai tag del dummy (blue) a quello nel reale (viola).

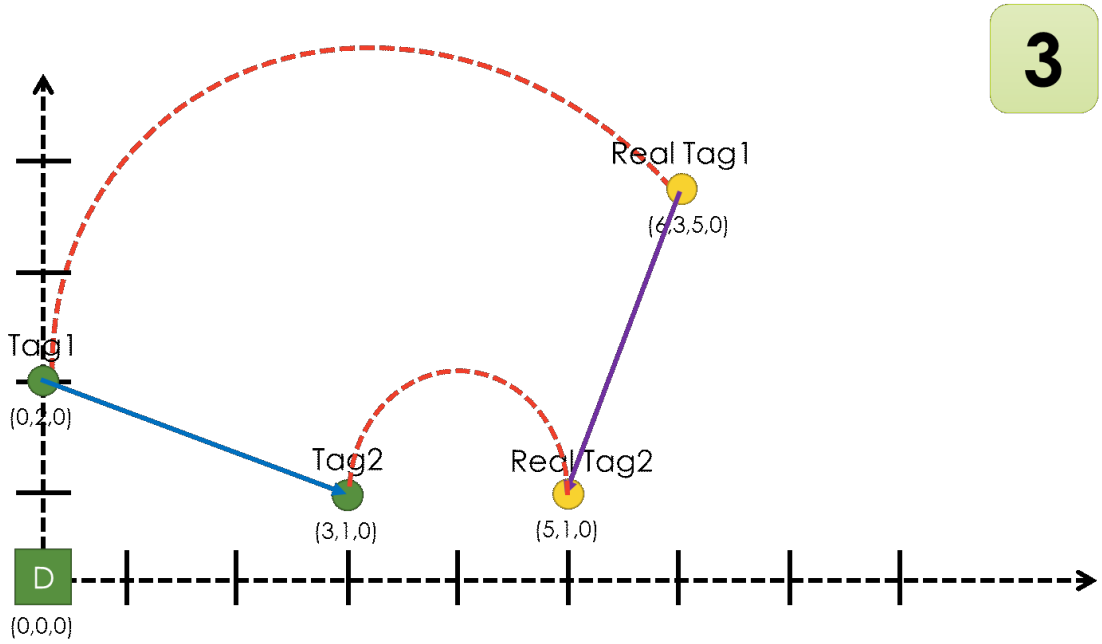

Figura 23: Calcolo allineamento 3

La rotazione viene applicata al dummy, ruotando così anche i tag associati.

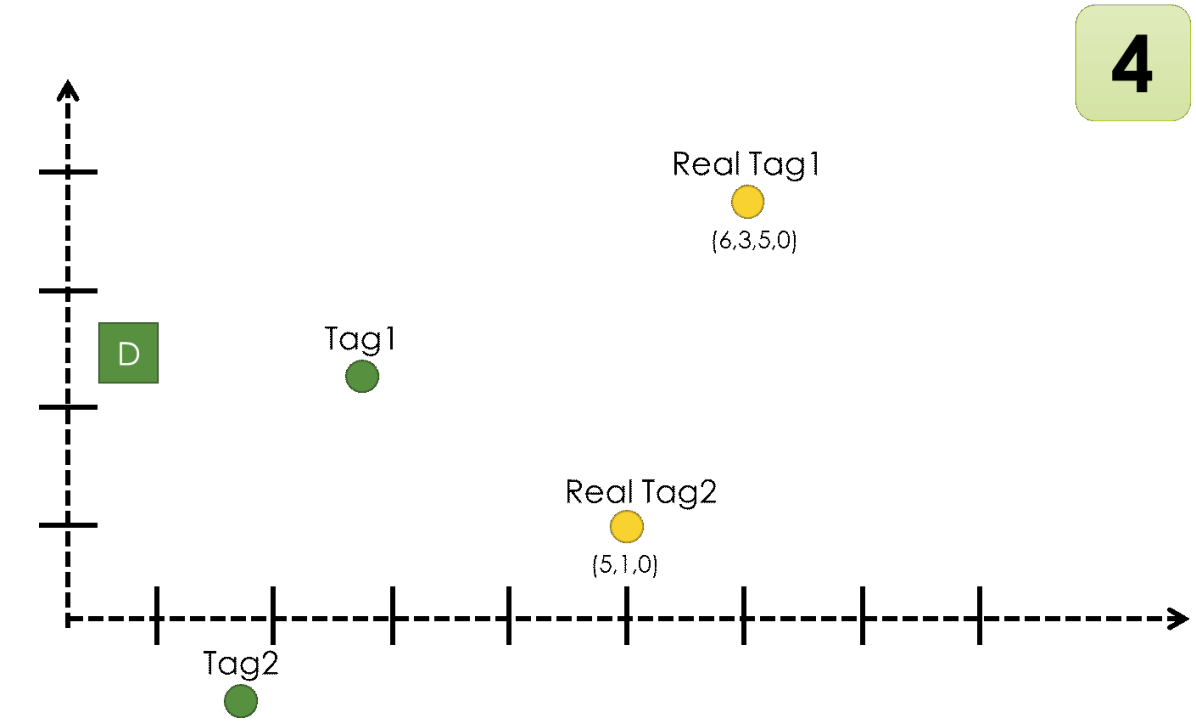

Figura 24: Calcolo allineamento 4

Si calcola il vettore di spostamento dal secondo tag dentro il dummy al secondo tag salvato.

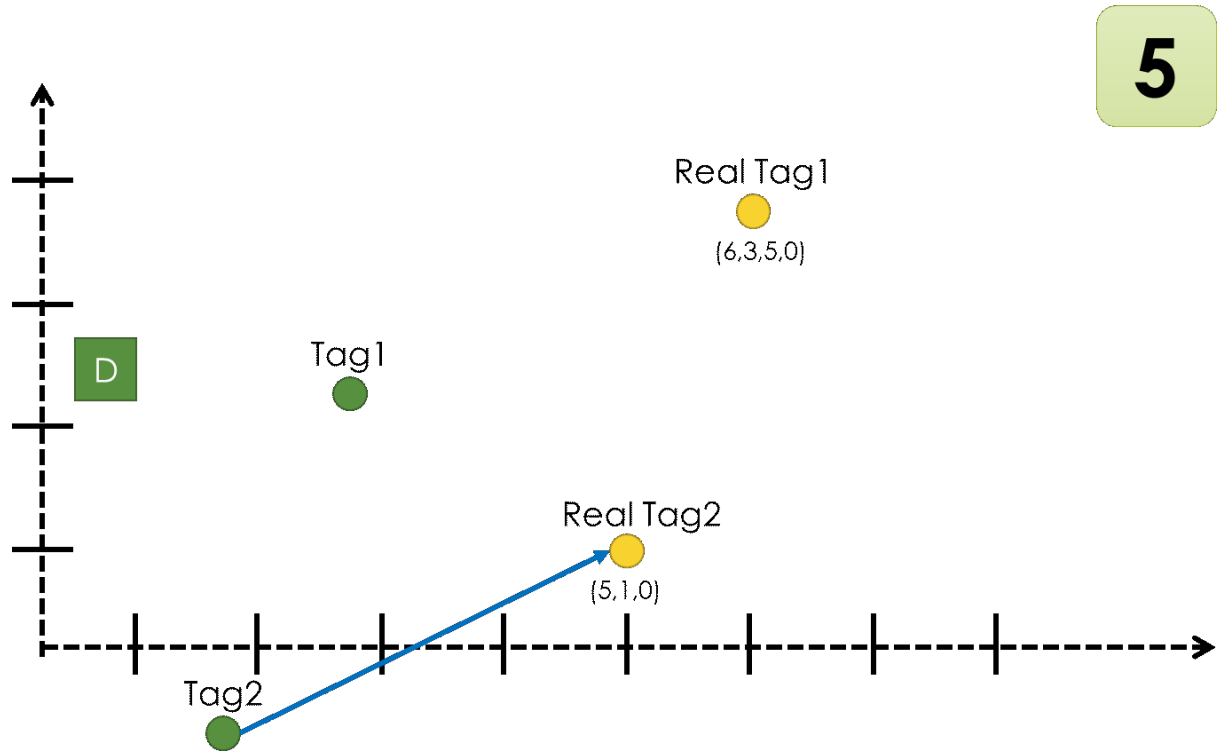

Figura 25: Calcolo allineamento 5

Per ultimo si sposta il dummy del vettore calcolato nel passo precedente.

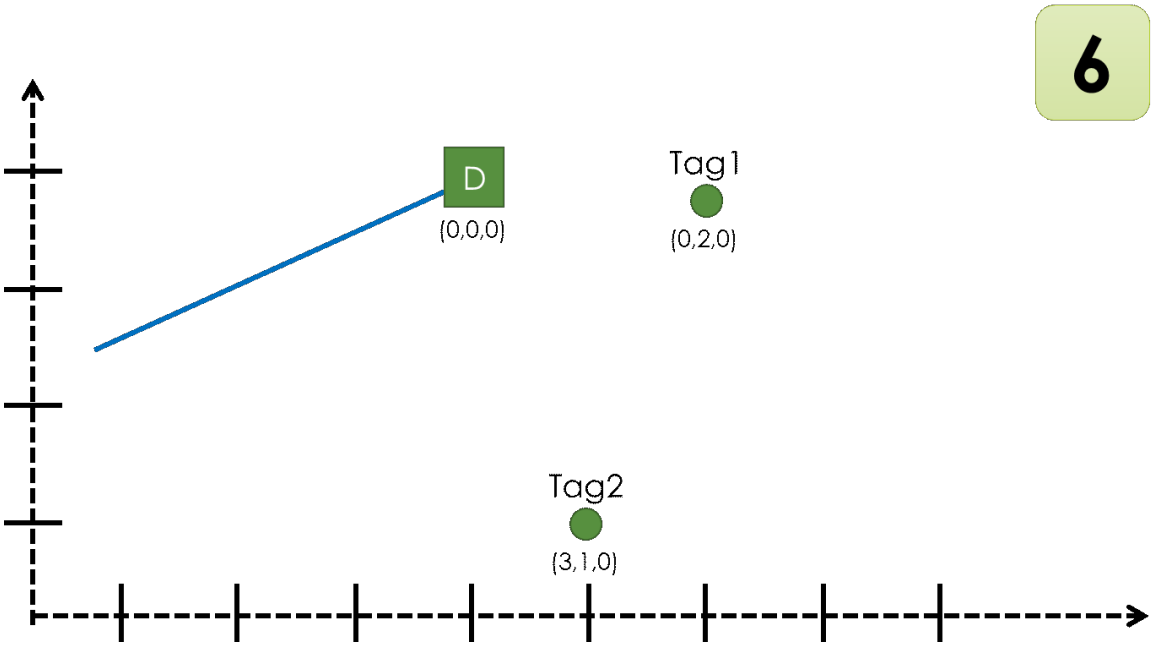

Figura 26: Calcolo allineamento 6

Le immagini riportate mostrano un esempio dell'algoritmo applicato in uno spazio di due dimensioni, ma è comunque applicabile ad uno spazio di tre dimensioni.

3.3.7. Selezione e commenti

Un commento è una registrazione audio assegnata ad un elemento dentro una Solution. Per assegnare un commento è stato implementato la funzione di selezionamento di un oggetto tramite il air touch, gesto in HoloLens per simulare il click del mouse, e viene gestito dallo script SelectionManager.

Questo script appena riconosce il gesto traccia un Raycast dal punto di osservazione fino 30m in avanti e identifica il primo oggetto appartenete al layer di selezione associato agli elementi del modello nel momento dell'import. Le layer in Unity definiscono quali GameObject possono interagire con le diverse funzionalità tra loro. Sono più comunemente usati dalle camere per eseguire il rendering solo di una parte della scena e dalle luci per illuminare solo parti della scena, ma possono anche essere usati dal raycasting per ignorare selettivamente i collider o per creare collisioni.

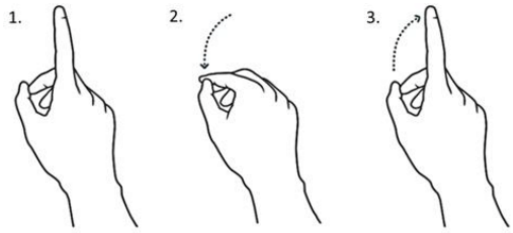

Figura 27: Gesto air tap

Appena viene trovato un primo oggetto a questo viene cambiato il colore facendolo risaltare come un elemento selezionato e compare davanti all'utente anche il menu dei commenti. Questo nuovo menu contiene il pulsante di registrazione audio come una lista di tutti i commenti associati ad esso. Per mostrare i commenti esistenti prima è stata fatta una richiesta al server e se viene premuto un commento viene riprodotto l'audio del commento.

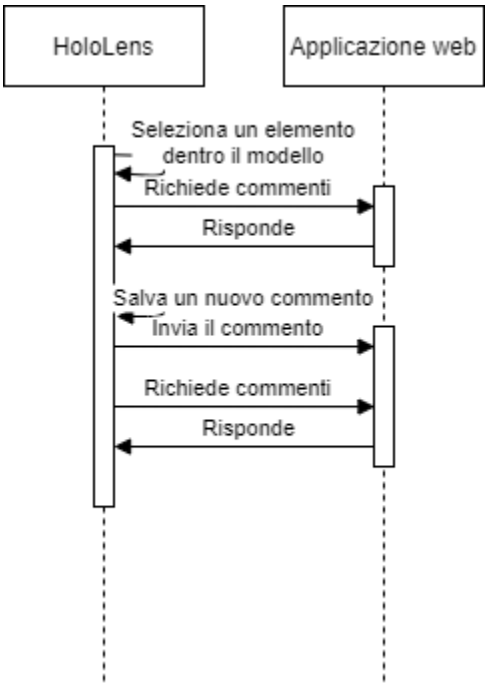

Figura 28: Diagramma di sequenza commenti

Per salvare un nuovo commento l'utente deve premere soltanto il tasto di nuovo commento ed iniziare a parlare. Se preme il pulsante nuovamente prima che finisca il tempo massimo, il commento fa salvato fino quell'instante li. Il nuovo commento viene riprodotto appena finito ed è anche inviato al server.

Chi gestisce i commenti è lo script RecordingAudioManager

4. Sperimentazione

Non sono state possibili eseguire le esperimentazioni direttamente in un cantiere; quindi, tutti i test furono svolti dentro il dipartimento DICEA all'Università Politecnica delle Marche.

In totale sono stati svolti tre tipi di test. Uno con soggetti di prova, uno per misurare la capacità di caricamento dell'applicazione in Unity e uno sulle prestazioni nuove della versione due dell'HoloLens.

4.1.1. Test sull'esperienza dell'utente utilizzando gli HoloLens

Questo test consiste nel vedere il comportamento delle persone indossando gli HoloLens e utilizzando l'applicazione. Venivano prese in considerazione la capacità dell'utente di usare l'applicazione di forma autonoma senza spiegarli il funzionamento delle cose, la qualità del menu e per ultimo i commenti e opinioni dell'utente sulla sua esperienza usando l'applicazione.

Le prove sono state eseguite per un numero totale di 5 persone, ripetute all'incirca di 10 volte per ognuna in giorni diversi aumentando le funzioni da testare. I risultati ottenuti vengono riportati qui sotto:

- In quanto alla capacità autonoma dell'utente per tutti la prima volta non è stato molto semplice di capire il funzionamento dell'applicazione ma ripetendo le prove nei giorni seguenti, già dalla terza prova, erano in grado di muoversi per conto proprio. Soltanto una persona continuava ad avere difficoltà nell'uso dell'applicazione. Quindi possiamo dire che i resultati ottenuti sono accettabili ma deve essere implementato un menu più semplice e in grado di spiegare per conto proprio le funzioni disponibili, ad esempio associando immagini uniche per funzione e più coerenti alla stessa funzionalità.
- In quanto alla qualità del menu, i risultati sono stati negativi visto che non ci sono stati commenti del tutto positivi ma indifferenti o negativi. Una quinta parte non ha voluto esprimere un pensiero sul menu, mentre il resto preferivano un menu con bottoni più significativi, come descritto dal test precedente. Altro risultato ottenuto era le diverse opinioni sul movimento del menu, mentre segue alla persona, due lo preferivano più rapido ma per altre due era meglio non tanto veloce. In quanto al display delle funzioni sono stati positivi i risultati indicando che era abbastanza semplice abituarsi al suo uso.
- Riguardo all'esperienza usando l'applicazione dei soggetti di prova, ognuno ha proposto le proprie idee di come poter migliorare le funzioni attuali o delle funzioni aggiuntive da eseguire. Vengono riportati i commenti in comune tra i soggetti e non verranno rispiegati le proposte migliorative già scritte nei due test precedenti. Tra le proposte migliorative è l'uso della capacita del microfono per riconoscimento di comandi vocali per evitare dover premere sempre i pulsanti e dare all'utente la libertà di poter usare le mani in altre cose, rendere possibile la customizzazione

del menu all'utente visto che alcune preferenze di una persona erano contradittorie ad un'altra e rendere possibile lo streaming direttamente dagli HoloLens, cosi se per qualche motivo l'esperto progettista non è in cantiere, esempio motivi COVID, potrebbe vedere e partecipare della prova sul luogo tramite internet.

4.1.2. Test sull'import dei modelli dentro Unity

Con questo test si vuole provare a capire i limiti presenti nell'applicazione di Unity più di preciso a capire le dimensioni massime con cui un file può essere importato dentro l'applicazione.

Per eseguire le seguenti prove sono stati costruiti file di diverse dimensioni e sono state importate dentro l'applicazione controllando se viene importato il modello e il tempo impiegato dal momento iniziale della scarica fino alla sua costruzione.

	Elementi	Dimensioni (KB)	Tempo (s)
1	450	1095	5,61
2	1330	17766	14,41
3	2210	31008	24,74
4	5730	83977	60,25
5	6610	97221	-

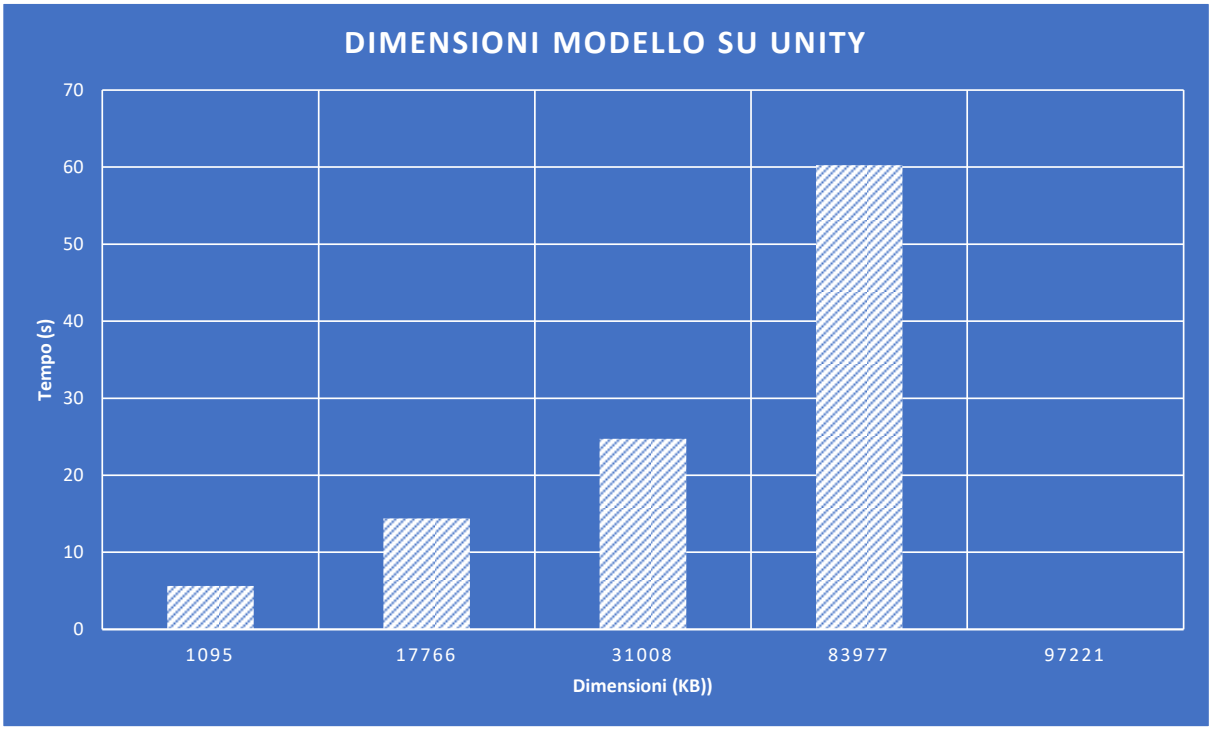

Figura 29: Diagramma dimensioni modelli su Unity

Nella tabella superiore vengono riportate cinque prove eseguite mostrando i casi più interessanti e possiamo vedere come esiste una relazione lineare tra la dimensione del file e il tempo impiegato, se aumenta la prima la seconda aumenta anche.

Nel quinto caso il file ha dimensioni di 97MB il quale non è importati dentro l'applicazione. Osservando il quarto caso, anche se viene caricato impiega all'incirca un minuto, il quale è un tempo molto lungo per lasciare l'utente in attesa. Continuando con l'analisi dal terso caso al primo, i tempi sono accettabili.

Si suggerisce usare modelli di dimensioni massimo a 30MB perché superati i 30s in attesa l'utente inizia ad annoiarsi e quindi peggiora l'esperienza e l'opinione sull'applicazione.

4.1.3. Test differenze con HoloLens2

Per vedere se la componente Unity per HoloLens era compatibile con la versione 2 del visore di realtà mixata di Microsoft, è stata installata l'applicazione anche sulla nuova versione.

L'installazione è avvenuta senza problemi e l'avvio dell'applicativo anche. Tutte le funzioni sviluppate per la prima versione continuarono a funzionare. I cambi riscontrati nelle due versioni sono stati nella performance della nuova versione. La visualizzazione del modello 3D era più chiaro e comodo visto che lo schermo dove viene riportata l'immagine è più grande e di miglior qualità. Gli oggetti ancorati avevano un minor spostamento dal punto di ancoraggio, anche se si allontanava dal luogo dove era stato fissato il modello e si tornava al luogo il modello rimaneva nella posizione indicata nella fase di allineamento o lo spostamento era di un valore molto basso (1cm max), mentre nella versione precedente, la uno, e si faceva la stessa prova, nel 90% dei casi l'allineamento fatto veniva perso con valori di oltre 1cm di distanza dal luogo stabilito.

Esiste anche un miglioramento ergonomico nell'indossare gli HoloLens V2 che quindi si suggerisce l'utilizzo di questa per un funzionamento migliore e più accurato.

5. Conclusioni

Il lavoro proposto non è la prima idea ad implementare la realtà mixata nel campo dell'edilizia, ma propone una nuova forma di allineamento del modello sul reale ed è la prima a coinvolgere i diversi agenti presenti nel progetto edile.

se i test sono molto pochi e sono mancati test su un cantiere reale i risultati sono stati sodisfacenti, dai quali è possibile identificare i limiti attuali esistenti e gli aspetti a modificare, determinando che il sistema si trova attualmente in uno stato di alpha testing.

Dai test svolti si consiglia se possibile l'uso della versione 2 di HoloLens e sono stati decisi come lavori futuri:

- Lo sviluppo del lettore dei sensori NFC, per il riconoscimento dei tag nel processo d'allineamento tra l'ologramma 3D e il reale.
- L'implementazione di un'interfaccia grafica dove l'utente possa gestire le sue preferenze di colori, distanza del menu, trasparenza degli oggetti e future caratteristiche estetiche, migliorando l'esperienza e la comodità con cui si usa l'applicazione in Unity
- Il miglioramento nel processo d'importazione dei modelli 3D sull'applicazione in Unity, rendendolo più veloce e capace di caricare file di dimensioni più grandi, questo cambiando l'asset di Piglet per uno migliore o implementando il proprio processo.

Dal lavoro svolto possiamo dire in conclusione che si è riusciti ad ottenere un primo approccio all'obbiettivo di un sistema innovativo per la gestione di lavori di ristrutturazione edilizia, il quale permette mantenere in connessione i diversi agenti presenti nel progetto e mantiene traccia storica dei cambiamenti dell'edificio rendendolo compatibile al BIM e allo stesso tempo incorpora le tecnologie nuove di realtà mixata creando così un sistema robusto alla avanguardia tecnologica.