ISSN 2039-7941
anno 2019 numero 405

RAPPORTI TECNICI INGV

Il sistema *TECH4DA* per l'acquisizione e la condivisione di informazioni geografiche

ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

405

REGISTRAZIONE AL TRIBUNALE DI ROMA N.174 | 2014, 23 LUGLIO

ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

ISSN 2039-7941
anno **2019** numero **405**

RAPPORTI TECNICI INGV

Il sistema *TECH4DA* per l'acquisizione e la condivisione di informazioni geografiche

The TECH4DA system for the collection and sharing of geographic information

Sergio Falcone[1], Antonio Costanzo[1], Rosalbino Bisignano[2], Giuseppe Ritacco[2], Michele Straface[2], Carmelo La Piana[1], Massimo Musacchio[1], Francesco Calimeri[2], Corrado Castellano[3], Valerio De Rubeis[3], Paola Sbarra[3], Patrizia Tosi[3]

[1]INGV | Istituto Nazionale di Geofisica e Vulcanologia, Osservatorio Nazionale Terremoti
[2]Università della Calabria | Dipartimento di Matematica e Informatica
[3]INGV | Istituto Nazionale di Geofisica e Vulcanologia, Sezione Sismologia e Tettonofisica

Accettato 9 novembre 2018 | *Accepted 9 November 2018*
Come citare | *How to cite* Falcone S. et al., (2019). Il sistema *TECH4DA* per l'acquisizione e la condivisione di informazioni geografiche. Rapp. Tec. INGV, 405: 1-36.

In copertina Acquisizione di immagini e ricostruzione di un modello 3D attraverso l'app *TE CH 4DA* | *Co ve r Image capture and reconstruction of a 3D model by using TECH4DA app*

405

INDICE

Introduzione

La sequenza sismica che ha recentemente colpito l'Italia Centrale, danneggiando significativamente diversi centri abitati, ha reso palese come in un breve lasso di tempo possano essere cancellati i segni della nostra cultura. Per custodire la memoria fotografica dei luoghi prima dell'occorrenza di un evento calamitoso e per supportare le attività di rilievo del conseguente danneggiamento è stata ideato il sistema *TECH4DA* (acronimo di *Technologies for Damage Assessment*). Si tratta di un sistema semi-automatico basato su un'architettura di tipo *client-server* ottimizzata per dispositivi mobili *Android* e *iOS*, concepito per la raccolta di testi, immagini e ricostruzioni tridimensionali basate sulle stesse immagini. La possibilità di poter archiviare la posizione e l'aspetto degli edifici in particolari istanti di tempo, consente poi di valutare il cambiamento e l'eventuale impatto di un terremoto. L'applicazione *TECH4DA* può essere utilizzata da un esperto di rilevamento del danno, ma essa è stata appositamente ideata per essere fruibile anche dai cittadini che potrebbero così fornire il contributo principale. La necessità di una partecipazione attiva e capillare dei cittadini rende *TECH4DA* una piattaforma basata sul *crowdsourcing*.
La sequenza sismica dell'Italia Centrale ha anche messo in luce la difficoltà di interpretare l'effetto cumulativo di una serie di terremoti che si susseguono a brevi distanze temporali. La possibilità di associare ad ogni evento un'immagine ed una descrizione permette di congelare la progressione del danneggiamento a date specifiche consentendo, allo stesso tempo, un'analisi progressiva e comparabile degli effetti di un sisma.
TECH4DA si inserisce nel filone delle attività, sviluppate nell'ambito del progetto *PON MASSIMO* per il monitoraggio di Beni Culturali in area sismica [Montuori et al., 2016], basate su indagini non distruttive e non invasive che fanno uso di sensori satellitari, da aereo e terrestri, completando la proposta tecnologica maturata nel progetto stesso.
In questo lavoro si presenta prioritariamente lo sviluppo tecnico dell'applicazione, quindi la logica *client-server* e i moduli che compongono l'infrastruttura. Vengono inoltre delineate alcune applicazioni e presentati possibili sviluppi futuri.

Introduction

The 2016-2017 seismic sequence that struck the Central Italy severely damaging different towns, it showed how the marks of our culture can be deleted in an instant. The TECH4DA system (acronym of Technologies for Damage Assessment) was developed both to preserve the photographic memory and support the survey activities after a disaster. TECH4DA is a semi-automatic system based on a client-server architecture optimized for the Android and iOS mobile devices. The capability to store location and aspect of the buildings in different time allow to assess their changes due to an earthquake. TeCH4DA can be used by expert involved in damage assessment, although it has been designed for obtaining relevant contribution also from the citizens. The active and widespread participation of the citizens makes TECH4DA a platform based on the crowdsourcing.
The Central Italy seismic sequence showed the difficulty to understand the cumulative effect due to a series of earthquakes occurred at short temporal distances. The capability to interrelate an image and a description to each event makes possible to freeze the damage progression at a specific date, enabling, a progressive and comparable analysis of the effects of an earthquake.
TECH4DA is part of the products developed in the framework of the PON MASSIMO project aimed to the monitoring of the cultural heritage in seismic area [Montuori et al., 2016], which integrates non-destructive and non-invasive surveys based on satellite, airborne and terrestrial sensors.
In this report, the technical development of the TECH4DA system is described, in addition to client-server logic and the modules which compose the infrastructure. Moreover, some applications and potential future developments are outlined.

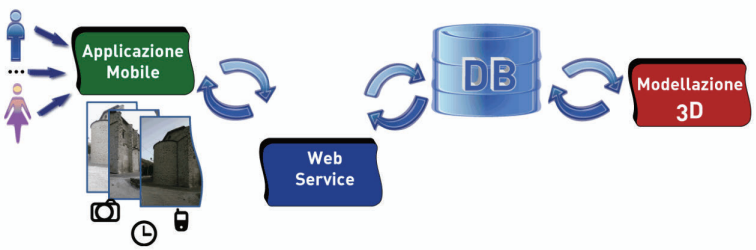
1. Il sistema *TECH4DA*

TECH4DA è un sistema tecnologico che, sulla base di indagini, rilievi e immagini fotografiche acquisibili tramite smartphone/tablet, può supportare le analisi necessarie alla stima del danneggiamento indotto da eventi calamitosi o da azioni antropiche. Il presente lavoro illustra le principali caratteristiche delle varie componenti già implementate nel sistema prototipale, anche al fine di raccogliere suggerimenti e osservazioni per le successive fasi di sviluppo. Una rappresentazione schematica del funzionamento del sistema è mostrata in Figura 1.
Le attività principali che qualificano questo sistema sono:

1. l'implementazione di un *Web Service* per lo *storage* e catalogazione di dati acquisiti e la restituzione delle elaborazioni (§2);
2. lo sviluppo di un'applicazione per *smartphone/tablet*, quale strumento per l'acquisizione dei dati e la visualizzazione delle elaborazioni (§3);
3. la realizzazione di un servizio Windows per interagire con il *software* di ricostruzione 3D (§4).

Figura 1 Schema di funzionamento del sistema TECH4DA.

Figure 1 *Operating scheme of the TECH4DA system.*



Al fine di illustrare le potenzialità di questo sistema tecnologico, alcune possibili applicazioni in forma ancora preliminare sono mostrate nel §5. La struttura del sistema e le funzionalità già implementate sono state sviluppate durante il lavoro di tre tesi di Laurea in Informatica presso il Dipartimento di Matematica e Informatica dell'Università della Calabria, con la supervisione di alcuni degli autori. Alcune conclusioni relative a questa prima parte del progetto ed i principali sviluppi futuri già pianificati sono discussi nel §6.

2. Infrastruttura e Web-Service

I *Web Services* sono applicazioni *software* modulari utilizzabili per supportare l'interoperabilità in un ambiente distribuito [cfr. Alonso et al., 2004]. L'idea è quella di mettere a disposizione un set definito di tecnologie basate su *standard* che siano in grado di facilitare l'interoperabilità tra sistemi diversi, massimizzando il grado di automazione. Per il sistema *TECH4DA*, è stato sviluppato un apposito *Web Service* per la gestione dello scambio di informazioni tra i dispositivi di raccolta dati e il database.
La prima fase dello sviluppo, pertanto, ha riguardato la progettazione della base di dati su cui si basa l'intero sistema; in particolare, sono state individuate le categorie di informazioni che necessitano di essere salvate in modo permanente:

- dati relativi al territorio nazionale (Regioni, Province, Comuni);
- dati relativi ai beni immobili e monumentali oggetto dello studio;
- dati relativi all'evento sismico;
- dati di rilevazione degli effetti prodotti dall'evento.

Per rappresentare al meglio queste informazioni sono state quindi definite le entità:

- *Regiones*, *Provincias*, *Comunes* per i dati relativi al territorio nazionale;
- *Dimostratores*, *Tipo_Dimostratore* per definire i beni censiti;
- *EventoSismicoes* per indicare l'evento che ha prodotto il danno;
- *Misuras*, *Tipo_Misura*, *Files*, *Rilievoes*, *Volumes*, per la rilevazione degli effetti prodotti.

In Figura 2 è mostrato il modello relazionale del DB con il dettaglio delle tabelle. La tabella *Evento_sismicoes* contiene i dati relativi al terremoto selezionato, direttamente ottenuti da apposita richiesta al *web service* dell'INGV.

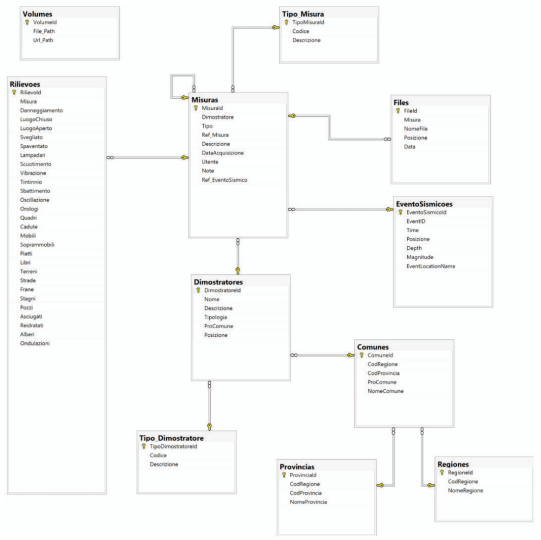


Figura 2 Modello relazionale del database.

Figure 2 *Relational model of the database.*

Per lo sviluppo è stata utilizzata la tecnologia *ASP.NET MVC 4* con *WebAPI*, integrata nell'ambiente di sviluppo *Microsoft Visual Studio*. L'approccio usato è *Code First* di *Entity Framework*, che permette di concentrarsi sulla progettazione del dominio delle classi; invece, il *framework* si occuperà di creare automaticamente il database e di sincronizzarlo con le eventuali modifiche apportate al dominio. Per ogni classe del dominio è generata una tabella nel DB, le cui chiavi e relazioni sono definite attraverso le *Data Annotations*. In Figura 3, la classe *Comune* è mostrata come esempio.

```
public class Comune
{
    [Key]
    public int ComuneId { get; set; }

    public int CodRegione { get; set; }

    [ForeignKey("CodRegione")]
    public Regione Regione { get; set; }

    public int CodProvincia { get; set; }

    [ForeignKey("CodProvincia")]
    public Provincia Provincia { get; set; }

    [Required]
    public int ProComune { get; set; }

    [Required]
    [MaxLength(60)]
    public string NomeComune { get; set; }

}
```

Figura 3 Esempio della classe *Comune*.

Figure 3 *Example of the Comune class.*

Per gestire le richieste HTTP, ed eseguire quindi le operazioni *CRUD* (*Create*, *Read*, *Update*, *Delete*) sui dati, è stato necessario associare ad ogni classe una classe *Controller*. Le classi *Controller* sono state create seguendo il *template "Web Api 2 controller con azioni"* utilizzando *Entity Framework* proposto da Visual Studio.
I metodi principali sono stati implementati in modo analogo per tutte le classi *Controller* [cfr. Larman, 2005]. Per le operazioni di lettura sono disponibili due metodi: il primo restituisce il singolo oggetto a partire dal suo ID; il secondo restituisce un elenco di oggetti opportunamente strutturato per la paginazione. Le operazioni di modifica e cancellazione ricevono come parametro l'ID dell'oggetto da modificare o eliminare. Per la creazione di nuovi oggetti è stato definito il metodo *post*, che riceve come parametro l'oggetto da creare.
In alcuni *Controller* sono stati aggiunti metodi specifici:

- per la classe *DimostratoreController*: un metodo di lettura che restituisce i punti di interesse in una determinata area geografica, una volta ricevuti come parametri latitudine, longitudine e raggio;
- per la classe *FileController*: un metodo di scrittura che riceve come parametro un'immagine codificata in *Base64*, occupandosi della sua decodifica e del salvataggio all'interno del *repository* sia a risoluzione originale, che ridotta (per la visualizzazione in miniatura 150x150 px);
- per la classe *ShapeFileController*: un metodo di lettura che riceve come parametri latitudine e longitudine per restituire il comune entro il quale il punto è collocato.

Per uniformare i valori di ritorno dei metodi delle differenti classi *Controller*, è stata definita una classe comune per la rappresentazione dei dati chiamata *Ws_Result* (Figura 4).

Figura 4 Struttura della classe *Ws_Result*.

Figure 4 *Structure of the Ws_Result class.*

```
public class Ws_Result
{
    public int State { get; set; }

    public string Message { get; set; }

    public object Data { get; set; }

    public int Skip { get; set; }

    public int Take { get; set; }

    public decimal Lat { get; set; }

    public decimal Lon { get; set; }

    public int Radius { get; set; }

}
```

Quest'ultima classe rappresenta il valore di ritorno dei metodi *Get*, *Post e Put*. Nel campo *Data* è riportato l'insieme degli oggetti richiesti e/o inviati. Il campo *State* indica la corretta esecuzione o la presenza di eventuali errori, mentre i campi *Skip*, *Take*, *Lat*, *Lon e Radius* tengono traccia dei parametri aggiuntivi utilizzati. L'eventuale messaggio di errore è riportato nel campo *Message*.
Al fine di evitare accessi non autorizzati ai dati, è stato adottato il sistema di autenticazione *Asp.NET Identity* basato su framework *OWIN* e il protocollo *OAuth* per la gestione di autorizzazioni basate su *token*. Con la scelta di questa opzione sono state create le classi di supporto *AccountViewModel* e *AccountBindingModel* e la classe controller *AccountController*. Inoltre, nel database sono state aggiunte le tabelle *AspNetUsers* e *AspNetUserLogins* per la gestione degli utenti. Per la creazione di un nuovo utente si effettua una richiesta *Post* all'indirizzo *../Tech4DA_WS/api/Account/Register* contenente i parametri *email*, *password* e *confirmPassword*

relativi al nuovo utente.
Il *token* per l'accesso ai metodi *CRUD* è richiesto tramite *Post* all'indirizzo *../Tech4DA_WS/token* inserendo nel corpo della richiesta la stringa: *grant_type=password&username=XXX&password=YYY*.
All'esito positivo della richiesta è restituito il *token* generato (Figura 5), che, preceduto dalla parola *bearer*, viene utilizzato per l'accesso ai metodi protetti inserendolo come valore del parametro *Authorization*.


```
"access_token": "BLQY0HrpAqAFsR4sepz6NLDASnP43…",
"token_type": "bearer",
"expires_in": 1209599,
"userName": "test",
".issued": "Fri, 16 Jul 2017 09:28:15 GMT",
".expires": "Fri, 29 Jul 2017 09:28:15 GMT"
```


Figura 5 Esempio di risposta contenente il *token*.

Figure 5 *Example of the response with the token.*

I metodi *CRUD* dei *Controller* sono stati protetti specificando il *tag [Authorize]*.
Il *web service* è stato installato su *server* dislocati presso la sede INGV di Rende. L'*host* che pubblica all'esterno il *web service* è una macchina virtuale con *Windows Server 2016 Standard Edition*, architettura a 64 bit, 4 core da 2,50 GHz e 8,00 GB di RAM. Per far fronte a tutte le necessità sia *hardware* che *software*, la macchina virtuale risulta facilmente scalabile grazie all'ambiente di virtualizzazione VMware ESXi 5.5. La pubblicazione all'esterno del servizio è gestita attraverso Internet Information Service (IIS) di Windows Server.

3. Sviluppo dell'Applicazione

L'applicazione mobile per *tablet* e/o *smartphone* rappresenta lo strumento per l'acquisizione dei dati e garantisce la possibilità di una partecipazione diffusa sul territorio. L'obiettivo principale è di supportare ed ampliare le attività di rilievo del danneggiamento in seguito ad un evento calamitoso, consentendone la condivisione tra personale specializzato e verso i cittadini. L'applicazione consente di:

- acquisire informazioni e gallerie di immagini fotografiche georiferite relative ad un punto di interesse (*PDI*);
- visualizzare su una mappa ibrida (satellitare e stradale) un insieme di *PDI* nelle vicinanze, rappresentati da segnaposti;
- visualizzare le informazioni relative ai *PDI*;
- compilare questionari post-sisma.

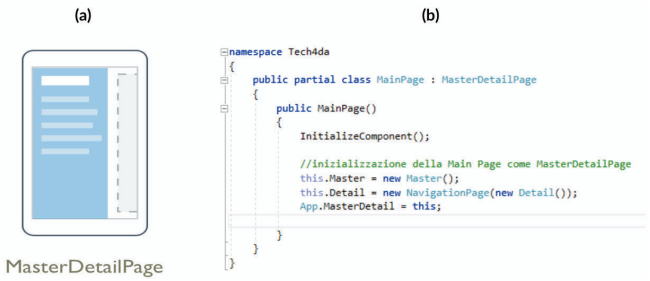
Per tutte le operazioni di acquisizione dei dati è necessaria l'autenticazione dell'utente; pertanto, l'applicazione fornisce un sistema di login per adempiere a tale scopo. Vista la necessità di coprire una vasta fetta di pubblico, si è scelto di rendere l'applicazione disponibile per le principali piattaforme presenti sul mercato, ovvero *Android* e *iOS*. Per lo sviluppo si è scelto quindi di utilizzare il *framework* Xamarin, integrato nell'ambiente di sviluppo Microsoft Visual Studio, che consente di realizzare contemporaneamente applicazioni *mobile* per diverse piattaforme. In dettaglio, lo sviluppo prevede sia la scrittura di codice condiviso che specifico per le due piattaforme, attraverso l'uso del linguaggio di programmazione orientato ad oggetti *C#*. La realizzazione dell'applicazione ha visto come piattaforma principale di distribuzione *Android*, in modo da avere fin da subito un prototipo funzionante per la maggior parte dei dispositivi mobile in commercio. Le versioni di *Android* supportate sono *Android 6.0 API level 23 – Marshmallow* come destinazione, *Android 4.4 API level 19 – Kit Kat* come versione minima e per la compilazione del progetto *Android 7.1 Nougat*.

3.1 Struttura dell'Applicazione

La navigazione tra le pagine dell'applicazione segue il modello a *stack*, in cui una *Root Page* sta alla base della navigazione, in questo caso costituita dalla pagina *MainPage.xaml.cs*. Per la *Root Page* è stato utilizzato un tipo di pagina chiamata *MasterDetaisPage* (Figura 6), visto che offre la possibilità di gestire due pagine di informazioni correlate: una *Master* che presenta degli elementi e l'altra *Detail* che presenta i dettagli dell'elemento selezionato nella pagina *Master*. Di seguito viene illustrata la struttura della *MasterDetailPage* e la sua implementazione e la navigazione a *stack*.
Navigare verso una nuova pagina corrisponde a inserirla in cima allo *stack* (*Push*), mentre navigare indietro corrisponde alla sua rimozione (*Pop*). Per gestire la navigazione all'interno dell'applicazione sono stati implementati dei metodi statici appositi nella classe *App.xaml.cs*, utilizzati anche per nascondere il menu laterale (*Master*) della *MasterDetailPage* quando si naviga tra le pagine di dettaglio.

Figura 6 Struttura della pagina *MasterDetailPage* (a) e sua implementazione nel file *MainPage.xaml.cs* (b).

***Figura 6** Structure of the MasterDetailPage page (a) and its implementation code on the file MainPage.xaml.cs (b).*

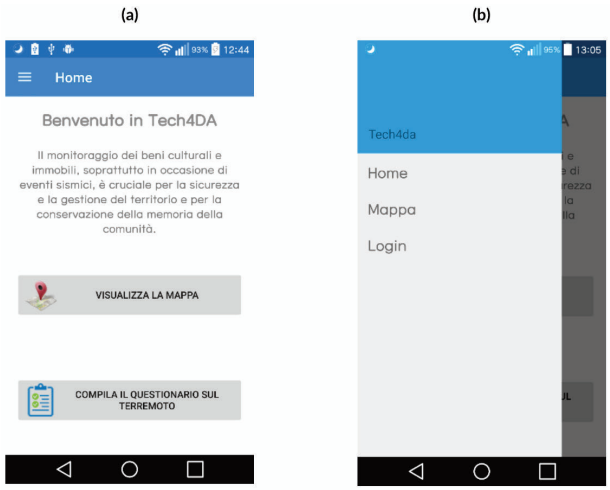


3.2 Schermate e funzionalità

Una volta avviata l'applicazione, compare la schermata *home* (Figura 7). Nella Figura 7.a è visibile la struttura della *MasterDetailPage* composta da una barra di navigazione superiore, con un pulsante per visualizzare il menu laterale (cfr. Figura 7.b) contenente le sezioni principali dell'applicazione (*Master*), e la pagina di dettaglio relativa selezionata. Dalla pagina di *home* è possibile decidere: di visualizzare la mappa con l'insieme dei punti d'interesse nelle vicinanze (§3.2.1); di acquisire direttamente dati e/o gallerie fotografiche (§3.2.2); di effettuare il login (§3.2.3).

Figura 7 Pagina di *home* (pannello sinistro) e con il menu laterale (pagina *Master*) attivo (pannello destro).

***Figura 7** Homepage (left frame) and the same page with sliding lateral menu (Master page) (right frame).*



3.2.1 Visualizzazione su mappa dei Punti di Interesse

Questa schermata consente di visualizzare la posizione attuale dell'utente e l'insieme dei punti di interesse nelle vicinanze su una mappa ibrida (satellitare e stradale) (Figura 8).

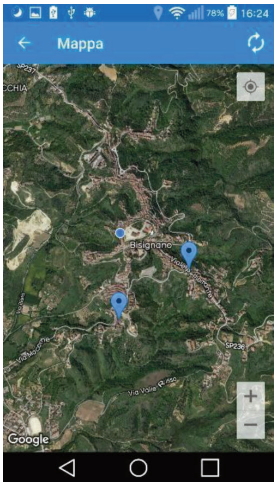


Figura 8 Visualizzazione su mappa dei punti d'interesse nelle vicinanze.

Figure 8 *View of the map with the points of interest.*

Per realizzare questa visualizzazione è stato necessario inserire il *plugin Xamarin.Forms.Maps* in tutti i progetti della soluzione. Successivamente, al fine di utilizzare le *API v2* di *Google* per le mappe su *Android*, è stato necessario utilizzare i seguenti pacchetti aggiuntivi dei servizi Google: *Xamarin.GooglePlayServices.Base*, *Xamarin.GooglePlayServices.Basement*, *Xamarin.GooglePlayServices.Maps*, *Xamarin.GooglePlayServices.Location*. La visualizzazione su mappa, allo stato attuale, è disponibile solo per *Android* attraverso la *Google Maps API key*, ovvero la chiave per poter utilizzare le mappe integrandole all'interno del file *AndroidManifest.xml*. All'interno dello stesso file è stato necessario abilitare i seguenti permessi: *AccessCoarseLocation*, *AccessFineLocation*, *AccessLocationExtraCommands*, *AccessMockLocation*, *AccessNetworkState*, *AccessWifiState*, *Internet*.
All'interno del progetto contenente il codice condiviso dalle due piattaforme (*Android* e *iOS*) vengono inizializzate le proprietà della pagina e della mappa, viene restituita la posizione attuale dell'utente, attraverso il GPS del dispositivo, e vengono inoltrate le richieste al *web service* per ottenere i punti d'interesse, che sono individuati sulla mappa attraverso dei segnaposti. Per quanto riguarda l'interfaccia utente di questa pagina, si è usato un *custom renderer*, visto che *Xamarin.Forms.Maps* non permette la gestione dell'evento di *click* di un segnaposto all'interno della mappa. Un *custom renderer* rende possibile la definizione di un'interfaccia che può essere successivamente implementata in modo nativo per ogni piattaforma. Nella *MapPage* per poter ottenere la posizione dell'utente sono presenti i metodi *initGeolocator* (Figura 9), per inizializzare il geolocalizzatore, e *PositionChanged* (Figura 10), per ascoltare i cambiamenti di posizione, a intervalli regolari di tempo e distanza. Entrambi i metodi fanno uso del metodo *RetrievePosition* della classe di utilità *GeolocationHelper.cs*.

```
private async void initGeolocator() //inizializzazione geolocalizzatore per ottenere la posizione corrente a intervalli regolari
{
    Plugin.Geolocator.Abstractions.Position position = await GeolocationHelper.Instance.RetrievePosition(); //inizializzazione geolocalizzatore e
                                                                                                            //restituzione posizione attuale
    CurrentPosition = new Xamarin.Forms.Maps.Position(position.Latitude, position.Longitude);

    GeolocationHelper.Instance.getGeolocator().PositionChanged += PositionChanged; //inizializzazione listener per cambio posizione
    if (GeolocationHelper.Instance.getGeolocator().IsListening)
    {
        await GeolocationHelper.Instance.getGeolocator().StopListeningAsync();
    }
    await GeolocationHelper.Instance.getGeolocator().StartListeningAsync(minTime: 2000, minDistance: 0.5); //listener asincrono per poter restituire
                                                                                                            //la posizione al variare del tempo o
                                                                                                            //della posizione
}
```

Figura 9 Metodo *initGeolocator*.

Figure 9 *initGeolocator method.*

Nella classe di utilità *GeolocationHelper*, implementata utilizzando il *design pattern Singleton*, il metodo *RetrievePosition* (Figura 11) ha il compito effettivo di restituire le coordinate geografiche della posizione attuale, pertanto è stato installato il *plugin Xam.Plugin.Geolocator*.

Figura 10 Metodo *PositionChanged*.

Figure 10 *PositionChanged method.*

```
private async void PositionChanged(object sender, PositionEventArgs e)
{
    Plugin.Geolocator.Abstractions.Position position = await GeolocationHelper.Instance.RetrievePosition(); //restituisce posizione
    CurrentPosition = new Xamarin.Forms.Maps.Position(position.Latitude, position.Longitude);
    getPins();
    if (firstAccess)
    {
        MoveToCurrentLocation(); //spostamento sulla posizione attuale se è il primo accesso
        firstAccess = false;
    }

}
```

Figura 11 Parte del metodo *RetrievePosition* responsabile della restituzione della posizione attuale.

Figure 11 *Part of the RetrievePosition method, which returns the actual position.*

```
Position currentPosition = new Position();
TaskScheduler scheduler = TaskScheduler.FromCurrentSynchronizationContext();
CancellationTokenSource cancelSource = new CancellationTokenSource();
_geolocator = CrossGeolocator.Current;
_geolocator.DesiredAccuracy = 20;

//restituzione posizione attuale se possibile
await _geolocator.GetPositionAsync(5000, cancelSource.Token, false).ContinueWith(t =>
{
    if (t.IsFaulted)
    {
        Console.WriteLine(((GeolocationException)t.Exception.InnerException).Error.ToString());
    }
    else if (t.IsCanceled)
    {
        Console.WriteLine("Cancelled");
    }
    else
    {
        Console.WriteLine(t.Result.Timestamp.ToString("G"));
        currentPosition.Latitude = t.Result.Latitude;
        currentPosition.Longitude = t.Result.Longitude;
        Debug.WriteLine("PRIMA " + currentPosition.Latitude + "       " + currentPosition.Longitude);
    }
}, scheduler);

return currentPosition;
```

Per visualizzare sulla mappa i punti di interesse, nel metodo *getPins* viene preparata una richiesta da inviare al *web service* contenente i parametri richiesti dalle *API* (Figura 12). La richiesta è inviata al *server* tramite il metodo *GetDataAsync* (Figura 13) della classe di utilità *RestService.cs*, se questa va a buon fine il risultato viene salvato nel campo *Data* della classe *MyObjectResult.cs*. Ottenuti i punti di interesse, e deserializzando poi la stringa JSON che li rappresenta nella risposta dal *server*, sono assegnati a dei *Custom Pin* mostrati come segnaposti personalizzati su mappa, tramite il *custom rende*rer.

Figura 12 Preparazione, invio richiesta e ricevimento risposta di Punti dei Interesse nel metodo *GetPins*.

Figure 12 *Preparation, sending request and receiving response of the Points of interest in the GetPins method.*

```
//inizializzazione parametri della richiesta
parametersList.Clear();
parametersList.Add(new KeyValuePair<string, string>("skip", "0"));
parametersList.Add(new KeyValuePair<string, string>("take", "10"));
parametersList.Add(new KeyValuePair<string, string>("lat", CurrentPosition.Latitude.ToString().Replace(",", ".")));
parametersList.Add(new KeyValuePair<string, string>("lon", CurrentPosition.Longitude.ToString().Replace(",", ".")));
parametersList.Add(new KeyValuePair<string, string>("r", "20"));

try
{

    result = await service.GetDataAsync(ServiceUrl.DIMOSTRATORI, parametersList, TipoRichiesta.Dimostratori); //risultato richiesta
    UserDialogs.Instance.HideLoading();
    if (result == null)
    {
        var answer = await DisplayAlert("Servizio non disponibile", "Passare a modalità offline?", "SI", "NO");
        if (!answer)
        {
            if (Navigation.NavigationStack.Count > 0)
                await Navigation.PopToRootAsync();
        }
        return;
    }

    customMap.CustomPins.Clear();
    customMap.Pins.Clear();
```

Il metodo *getPins*, richiamato automaticamente all'apertura della mappa, può essere manualmente invocato dall'utente per aggiornare i *PDI*, attraverso il bottone in alto a destra nella barra di navigazione della schermata. Essendo necessaria la connessione ad Internet, per inoltrare le richieste al *web service*, e la geolocalizzazione GPS attiva per ottenere la posizione corrente, sono stati aggiunti dei controlli di verifica utilizzando delle finestre di *popup* che segnalino l'eventuale assenza di tali servizi.

```
public async Task<MyObjectResult> GetDataAsync(String uri, List<KeyValuePair<string, string>> list, TipoRichiesta tipo)
{
    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", "");
    MyObjectResult result = null;
    string s = "?" +  string.Join("&", list.Select((x) => x.Key + "=" + x.Value));
    try
    {
        TaskScheduler scheduler = TaskScheduler.FromCurrentSynchronizationContext();
        CancellationToken cancelToken = new CancellationToken();
        HttpResponseMessage response = null;

        await client.GetAsync(uri + s, cancelToken).ContinueWith(t => {
            if (t.IsFaulted)
            {
                Console.WriteLine("Faulted");
            }
            else if (t.IsCanceled)
            {
                Console.WriteLine("Cancelled");
            }
            else
            {
                Console.WriteLine("RESULT: " + t.Result);
                response = t.Result;
            }
        }, scheduler);
        if (response != null)
        {
            if (response.IsSuccessStatusCode)
            {
                var content = await response.Content.ReadAsStringAsync();
                result = JsonConvert.DeserializeObject<MyObjectResult>(content, new MyObjectResultConverter() { Tipo = tipo });
            }
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine(@"ERROR {0}", ex.Message);
    }
    return result;
}
```

Figura 13 Metodo *GetDataAsync*.

Figure 13 *GetDataAsync method.*

Cliccando su un segnaposto riportato sulla mappa è possibile visualizzare un *popup* con indicazione del nome del punto di interesse e i primi 40 caratteri della sua descrizione (Figura 14.a). Al tocco del *popup* si viene rimandati alla finestra con le informazioni di dettaglio del *PDI*: nome, descrizione completa e gallerie fotografiche correlate (Figura 14.b).
Per mostrare l'insieme delle gallerie fotografiche acquisite si fa uso di una lista a scorrimento orizzontale, nella quale ogni elemento rappresenta una galleria ed è caratterizzato dal nome e da un'immagine di anteprima. Per realizzare la lista a scorrimento si è fatto uso della libreria *DLToolKit.Forms.Controls.FlowListView*, che estende la classe *ListView* di *Xamarin.Forms* offrendo ulteriori funzionalità; invece, le anteprime fotografiche sono implementate attraverso la libreria *Xamarin.FFImageLoading*, che risulta utile al caricamento di immagini in modo veloce ed efficiente. Per ottenere le informazioni, come nel caso della schermata precedente, anche per i dettagli relativi al PDI viene inviata al *web service* una richiesta contenente come parametro il suo ID tramite il metodo *GetDataAsync* all'atto del caricamento della pagina. Pertanto anche in questo caso la presenza di una connessione ad Internet è necessaria per la corretta visualizzazione di tutte le informazioni di un punto d'interesse. Nel caso questo servizio non fosse disponibile, nella schermata viene mostrato solo il bottone *"Riprova"*, cliccando il quale è possibile inoltrare nuovamente la richiesta al server.

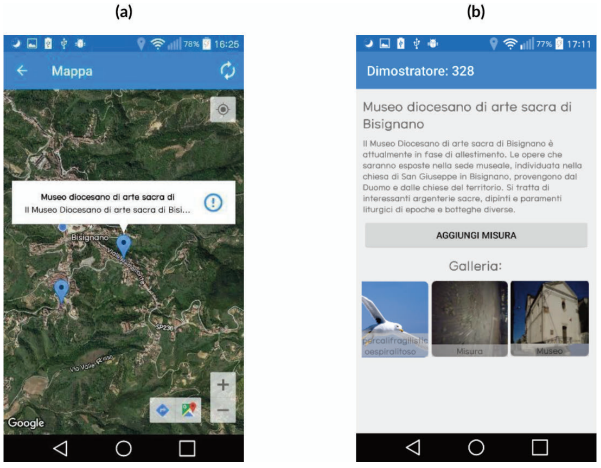


Figura 14 Finestra con le indicazioni essenziali (nome e primi 40 caratteri della descrizione) relative al PDI riportate al tocco del segnaposto sulla mappa (a). Finestra con informazioni di dettaglio e gallerie fotografiche relative al PDI (b).

Figure 14 *Basic information window (name and first 40 characters for the description) related to the points of interest, which are shown at the touch of the placeholder on the map (a). Window with detailed information and photo galleries related to the points of interest (b).*

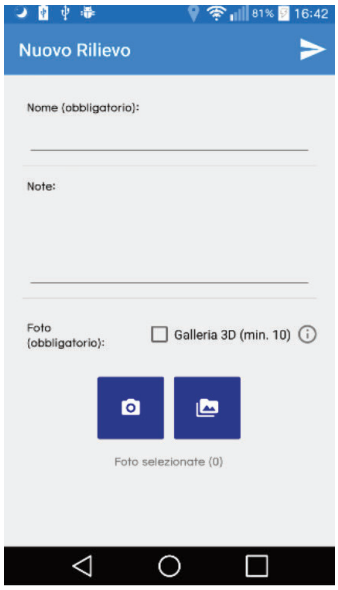
3.2.2 Acquisizione di una nuova galleria ed aggiunta foto ad una esistente

Al tocco del pulsante “Nuovo Rilievo” nella schermata di dettaglio del punto d’interesse, si visualizza la schermata in cui è possibile acquisire una galleria fotografica con i relativi dati.
In Figura 15 è mostrata la pagina di acquisizione, che è composta da:

- due campi per l’inserimento del nome e della descrizione (opzionale);
- due bottoni per l’acquisizione delle immagini fotografiche da fotocamera e dalla galleria del *device*;
- una *checkbox* per indicare se la galleria che si sta acquisendo sia semplice oppure utile ad una ricostruzione di un modello 3D;
- un bottone in alto nella barra di navigazione per l’invio dei dati al *web service*.

Figura 15 Schermata per l’acquisizione di una galleria fotografica.

Figure 15 *Page for the acquisition of a photo gallery.*



Per l’acquisizione delle immagini tramite fotocamera si è utilizzato un *plugin* chiamato *Xam.Plugin.Media* che offre un modo semplice e immediato per svolgere tale funzione (Figura 16).

Figura 16 Codice per l’acquisizione di un’immagine tramite fotocamera del dispositivo.

Figure 16 *Code for the acquisition of a image through the camera of the device.*

```
var file = await CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions
{
    DefaultCamera = Plugin.Media.Abstractions.CameraDevice.Rear,
    Directory = "Tech4Da"
});
```

Per quanto riguarda, invece, l’acquisizione delle foto tramite la galleria del *device*, visto che il *plugin* utilizzato non fornisce la selezione multipla delle immagini, si sono dovute utilizzare le funzionalità native della piattaforma *Android* attraverso il *DependencyService* di *Xamarin.Forms*. Il *DependencyService* consente di richiamare codice nativo della piattaforma dal codice condiviso, permettendo all’applicazione di svolgere qualsiasi cosa possa fare un’app nativa. Essendo un risolutore di dipendenze, in *Tech4da.Services* è stata creata l’interfaccia *IMediaService.cs*, che è utilizzata dal servizio per trovare l’implementazione specifica della piattaforma, in questo caso appunto la classe *MediaService.cs* presente nel progetto *Android*. In Figura 17 è mostrato l’uso

del *DependencyService* per aprire la galleria e quindi scegliere le foto quando viene premuto il bottone con l'icona di una cartella.

```
async void PickPhoto_Clicked(object sender, System.EventArgs e) //listener bottone scelta foto da galleria
{
    await DependencyService.Get<IMediaService>().OpenGallery(); //servizio di dipendenza per chiamare il metodo specifico
                                                                //in base alla piattaforma
}
```

Figura 17 Metodo *PickPhoto_Clicked* contenente il *DependecyService*.

Figure 17 *PickPhoto_Clicked method containing the DependecyService.*

Le informazioni relative alla data ed all'orario di acquisizione dell'immagine fotografica sono memorizzate direttamente dal web-service se questa è ottenuta tramite l'applicazione; sono invece estrapolate dai metadati che accompagnano l'immagine stessa se questa è inviata dalla galleria fotografica del dispositivo.
Per poter inviare le foto scelte dal progetto nativo al progetto condiviso, si è reso indispensabile l'utilizzo del *MessagingCenter* di *Xamarin.Forms*. Il *MessagingCenter* è un servizio per ricevere e inviare messaggi di qualsiasi tipo tra due componenti fortemente disaccoppiati. Le parti coinvolte sono due: la parte *Subscribe* in ascolto per l'arrivo dei messaggi e capace di eseguire le azioni alla loro ricezione, e la parte *Send* che invia i messaggi alla parte in ascolto. Di seguito sono mostrati i frammenti di codice presenti in *MainActivity.cs* e in *AddMisuraPage.xaml.cs* che rappresentano, rispettivamente, la parte *Send* (Figura 19) e la parte *Subscribe* (Figura 18) del *MessagingCenter*:

```
MessagingCenter.Subscribe<App, ObservableCollection<ImageItem>>((App)Xamarin.Forms.Application.Current, "ImagesSelected", (s, images) =>
{
    foreach (var p in images)
    {
        if(!Images.Contains(p))
        {
            Images.Add(p);
            if (!Thumbnails.Contains(p.ThumbnailPath))
            {
                Thumbnails.Insert(0, p.ThumbnailPath);
            }
        }
    }
```

Figura 18 Codice *Subscribe* del *MessagingCenter, che è* presente nel metodo *OnAppearing* della classe *AddMisuraPage.xaml.cs*.

Figure 18 *Subscribe code of the MessagingCenter, which is into the OnAppearing method of the AddMisuraPage.xaml.cs class.*

```
MessagingCenter.Send<App, ObservableCollection<ImageItem>>((App)Xamarin.Forms.Application.Current, "ImagesSelected", images);
```

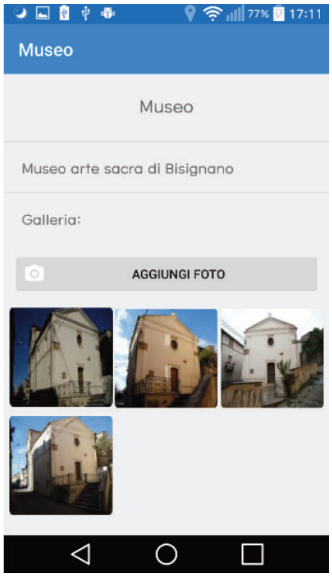
Figura 19 Codice *Send* del *MessagingCenter*, che è presente nel metodo *OnActivityResult* della classe *MainActivity.cs*.

Figure 19 *Send code of the MessagingCenter, which is into the OnActivityResult method of the MainActivity.cs class.*

Le foto ricevute sotto forma di stringhe rappresentanti il *path* della *directory* cui appartengono, dopo essere state ridimensionate per creare delle *thumbnail* salvate temporaneamente sul dispositivo, vengono visualizzate in elenco (cfr. Figura 14.b), come avviene anche per le immagini acquisite da fotocamera. All'aggiunta delle foto nell'elenco, apparirà un bottone *"Deseleziona tutto"*, utilizzabile nel caso in cui si volessero rimuovere le foto presenti. Se l'utente volesse acquisire una galleria fotografica per la realizzazione di un modello 3D, oltre a spuntare la *checkbox* relativa, può visualizzare i consigli sull'acquisizione delle immagini cliccando sul bottone di informazioni (cfr. Figura 15). Quando le immagini fotografiche vengono acquisite tramite fotocamera, viene anche rilevata la posizione geografica attuale dell'utente, tramite la classe *GeolocationManager.cs*, per poter georiferire ogni singola fotografia. Inoltre, per essere inviate al *web service* le fotografie sono convertite in stringhe in base 64, grazie ai metodi presenti nella classe statica di utilità *ConvertHelper.cs*.

Figura 20 Schermata relativa alle informazioni di una galleria fotografica.

Figure 20 *Page related to the information of a photo gallery.*



Dalla schermata di dettaglio del punto di interesse selezionando una galleria dall'elenco a scorrimento orizzontale, è possibile raggiungere la finestra con le informazioni e le immagini presenti nella stessa galleria (Figura 20).
Nella schermata in figura sono mostrati gli elementi presenti caratterizzanti la galleria selezionata: il nome, la descrizione, il bottone per aggiungere ulteriori foto (solo se si tratta di una galleria semplice, cioè per la quale non è richiesta la ricostruzione 3D) e l'insieme delle fotografie che la costituiscono. Le fotografie sono disposte come elenco tramite una lista *FlowListView*, selezionandole sarà possibile visualizzarle a schermo intero o fare uno zoom con un doppio *tap*. Le fotografie ad alta risoluzione visualizzate a schermo intero sono ottenute tramite richiesta al *server*, in quanto le immagini presenti nell'elenco sono solo di anteprima e quindi a bassa risoluzione. È presente un'immagine *placeholder* durante il caricamento di una foto e una di errore in caso di foto non ricevuta o corrotta. Per la visualizzazione delle immagini a pieno schermo si è utilizzato il *plugin ffimageloading.forms*, in modo da caricare soltanto una volta le foto nel formato più adatto allo schermo del *device* e non saturarne la memoria, rendendo quindi più efficiente l'applicazione eliminando possibili *crash*.
Premendo il bottone "Aggiungi Foto" nella schermata in Figura 20, è possibile aggiungere nuove foto ad una galleria semplice già esistente attraverso la finestra in Figura 21.
La schermata in figura è composta da:

- bottoni per l'acquisizione delle immagini fotografiche;
- elenco delle anteprime;

- bottone, nella barra di navigazione, per aggiungere le immagini alla galleria selezionata.

La struttura della pagina e l'implementazione della funzionalità, sono una parte della schermata per l'aggiunta di una nuova galleria fotografica, quindi i metodi all'interno della classe sono molto simili.

Figura 21 Schermata di aggiunta foto ad una galleria già esistente.

Figure 21 *Page for adding photos to an existing gallery.*

3.2.3 Schermata di login e sessione utente

Le sezioni di acquisizione dell'applicazione sono raggiungibili soltanto se un utente ha effettuato il login ed è stato autenticato, altrimenti lo stesso può soltanto visualizzare le informazioni. In sostanza, se non viene scelta dal menu laterale, la schermata di *login* (Figura 22) viene presentata ogni qualvolta un utente non autenticato cerchi di raggiungere le pagine di inserimento di una nuova galleria fotografica, di aggiunta di nuove foto ad una galleria già esistente o di raccolta dati.

Figura 22 Schermata di login.

Figure 22 *Login page.*

Inseriti *username* e *password*, queste credenziali sono incluse come parametri nel corpo del messaggio per una richiesta *Post* inviata al *web service*, tramite il metodo *PostCredentialAsync* della classe *RestService.cs*, ed è effettuata la validazione. Se l'utente viene autenticato, il messaggio di risposta ricevuto da *server* conterrà un *token* di accesso che verrà utilizzato

dall'applicazione per mantenere attiva la sessione dell'utente. Per la gestione della sessione di un utente è stata creata la classe di utilità *AccountManager.cs* e, dopo aver installato la libreria *Xamarin.Auth*, sono stati implementati i metodi:

- *StoreAccount* per memorizzare il *token* di accesso di un utente e quindi la sua sessione;
- *GetAccount* per la restituzione di un utente in sessione, da richiamare nelle sezioni di acquisizione dell'applicazione;
- *RefreshToken* per effettuare una nuova richiesta per un *token* al *Web Service*, in modo trasparente all'utente, nel caso in cui quest'ultimo sia scaduto;
- *RemoveAccount* per eliminare la sessione di un utente nel caso in cui quest'ultimo faccia il *logout*.

4. Ricostruzione tridimensionale dalle immagini fotografiche

Lo strumento automatizzato di modellazione 3D ha il compito di recuperare le gallerie fotografiche caricate sul server, mediante l'applicazione mobile, ed elaborarle per la ricostruzione nello spazio 3D dell'oggetto. Questo blocco di *TECH4DA* è composto da:

- un Servizio Windows installato sul *server* che interagisce con un *database* Microsoft SQL Server [1] per recuperare periodicamente dal *database* la locazione delle gallerie fotografiche caricate;
- software open source *VisualSFM* per la produzione del modello tridimensionale;
- funzionalità utili a garantire la conformità del modello tridimensionale prodotto alle specifiche stabilite, attraverso controlli all'interno del codice;
- un sistema di tracciamento delle operazioni eseguite giornalmente dall'applicazione, attraverso la piattaforma di registrazione *NLog*.

4.1 Software e metodi di acquisizione delle immagini

Per la scelta del software di modellazione 3D si è partiti da quanto presente in letteratura riguardo la comparazione di differenti *software* disponibili. In letteratura è mostrato il confronto tra quattro *SfM* (*Structure from Motion*) *software* in termini di accuratezza della ricostruzione di elementi geometrici e statue, anche usando differenti dispositivi di acquisizione delle immagini fotografiche [Mousavi et al., 2015]. Dal confronto, *VisualSFM* mostra accuratezza in molti casi confrontabile con gli altri, anche se non sempre la migliore. Rappresentando una delle poche valide soluzioni *open source* è stata scelta per questa fase di implementazione del sistema.
VisualSFM è una interfaccia grafica *open source* strutturata in algoritmi dedicati alla tecnica della *Computer Vision* [Wu, 2013]. Per la *sparse reconstruction* il *software* utilizza gli algoritmi *Sift GPU* [Wu et al., 2012] e *Multicore Bundle Adjustment* [Wu et al., 2011]. Inoltre, integra gli algoritmi *PMVS2* e *CMVS* per il calcolo delle *dense reconstruction* attraverso la scomposizione del problema in *cluster* ragionati.
VisualSFM rielabora un *dataset* di immagini, le suddivide in *cluster* e ne ricava l'orientamento e i punti omologhi, infine, estrae una nuvola di punti che ricostruisce fedelmente l'oggetto indagato.
Una serie di variabili, come ad esempio la corretta acquisizione delle immagini, possono influire significativamente sull'intero processo di elaborazione dei dati, determinando il corretto *processing* della nuvola di punti ed il risultato finale. Seppur non esistano regole precise e veloci su come effettuare le fotografie per la ricostruzione 3D, alcuni suggerimenti generali possono aumentare la probabilità di creare correttamente una nuvola di punti dalle immagini digitali.

Ad esempio, eseguire riprese convergenti (Figura 23) produce risultati migliori rispetto a riprese divergenti o parallele, anche se, in genere i *dataset* fotografici sono prodotti da una combinazione di questi tipi di ripresa.

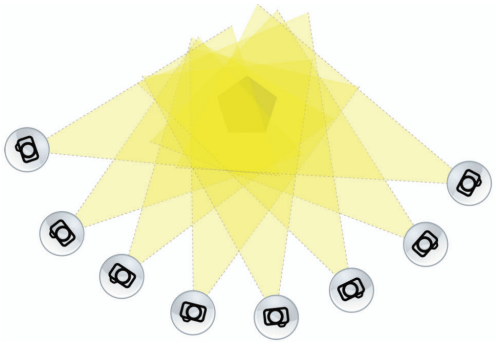


Figura 23 Schema di ripresa convergente.

Figure 23 *Convergent shooting scheme.*

A *dataset* fotografici con poche immagini corrispondono ricostruzioni di nuvole poco dense e, quindi, poco definite e metricamente meno attendibili. Tuttavia, la possibilità di strutturare il *cluster* iniziale ottimizza il rapporto tra numero di foto e qualità delle stesse: migliorando il criterio di presa e la risoluzione dell'immagine, si migliora il grado di definizione e di dettaglio della nuvola. Il suggerimento è di operare la stessa macchina fotografica per le riprese, utilizzando un'ottica fissa ed effettuando prese convergenti con un criterio di copertura e sovrapposizione delle immagini simile a quello fotogrammetrico.
Queste regole di ripresa, che possono garantire un maggior successo nella ricostruzione del modello 3D, sono consultabili direttamente tramite l'applicazione accedendo tramite il pulsante di informazioni della schermata di acquisizione (vedi Figura 15).

4.2 Implementazione del servizio locale

In questo paragrafo vengono descritte le classi relative al servizio Windows, che automatizza il processo di interazione tra l'acquisizione delle gallerie fotografiche e la generazione del modello tridimensionale.

4.2.1 Classe V sualSFMManager

In una tipica esecuzione del servizio, la classe *VisualSFMManager* recupera i *path* delle gallerie da elaborare dal database, lancia l'elaborazione di tali galleria tramite l'oggetto *Launcher* e salva il modello prodotto. In particolare, il costruttore della classe si occupa di recuperare le gallerie fotografiche da elaborare interrogando il database (Figura 24).

```
string filePath = ent.Volumes.First().File_Path;

//Aggiungo a toProcess i path delle gallerie immagini di tutte le misure
//che non hanno un modello 3d che verranno successivamente processati
foreach (var item in ent.Misuras.Where(op => op.Ref_Misura == null &&
    op.Tipo_Misura.Codice == "GLC"))
{
    int dimostratore = item.Dimostratore;
    int misuraId = item.MisuraId;
    MyPath m = new MyPath(filePath, dimostratore, misuraId);
    toProcess.Add(m);
}
```

Figura 24 Costruttore della classe per il recupero delle gallerie fotografiche.

Figure 24 *Builder of the class for the recovery of the photo galleries.*

Nelle istruzioni in Figura 24, la variabile *filePath* rappresenta il *path* generale della cartella all'interno del *server* che contiene tutti i *PDI*. Viene recuperato dalla tabella *Volumes* del DB (cfr. Figura 2). Il *foreach* ricerca nel DB tutte le *tuple* di *Misuras* che hanno la colonna <Ref_misura> a [null] e come <Tipo_misura> [GLC] (Galleria 3D), quindi i record non ancora elaborati. A questo punto, viene creato l'oggetto '*MyPath m*' che serve per comporre il *path* di localizzazione di ogni singola galleria.
Il metodo *start* crea ed avvia l'esecuzione di un *Launcher* (uno per ogni galleria) in maniera sequenziale non concorrente. Nello specifico viene rilevato il primo *path* nella lista *toProcess* e passato ad un nuovo *Launcher*, questo crea il modello e lo salva temporaneamente aggiornando il DB. Se il processo è andato a buon fine, il risultato viene salvato nella cartella finale ed il *path* appena eseguito viene rimosso dalla lista *toProcess*. In particolare, il salvataggio viene fatto richiamando il metodo *saveModel* dell'oggetto *Launcher* (Figura 25).

Figura 25 Metodo *saveModel* dell'oggetto *Launcher*.

Figure 25 *saveModel method of the Launcher object.*

```
launcher.saveModel(m.filePath + @"\" + m.dimostratore.ToString() + @"\" +
    newModelId.ToString(), nameTmp + ".nvm" + ".cmvs");
```

Figura 26 Codice per l'inserimento di un nuovo record nella tabella *Misuras* del DB per la generazione del modello 3D.

Figure 26 *Code for the input of a new record in the Misuras table of the DB for the generation of the 3D model.*

```
Misuras m = new Misuras();
m.Dimostratore = myPath.dimostratore;
m.Tipo = ent.Tipo_Misura.First(a => a.Codice == "3DM").TipoMisuraId;
m.Ref_Misura = null;
m.Descrizione = null;
m.DataAcquisizione = DateTime.Now;
m.Utente = 112;
m.Note = null;
ent.Misuras.Add(m);
ent.SaveChanges();
tmpNewMisuras = m.MisuraId;
```

Figura 27 Codice per l'inserimento di un nuovo record nella tabella *Files* del DB per la generazione del modello 3D.

Figure 27 *Code for the input of a new record in the Files table of the DB for the generation of the 3D model.*

```
Files files = new Files();
files.NomeFile = "result3d";
files.Lat = 0;
files.Lon = 0;
files.Data = DateTime.Now;
files.Misura = m.MisuraId;
ent.Files.Add(files);
ent.SaveChanges();
tmpNewFiles = files.FileId;
ApplicationContext.Instance.LogWriter.Info("Update Files");
```

I parametri che vengono forniti al metodo *saveModel* sono due:

- il primo per identificare la destinazione, che si compone dell'indirizzo del *repository*, del nome della cartella del *PDI* e del nome della cartella del nuovo modello prodotto (tutti questi campi vengono recuperati dal database);
- il secondo per ricercare il modello temporaneo, che si compone del nome precedentemente creato con l'aggiunta dell'estensione *.nvm.cmvs* assegnato dal *software VisualSFM*.

Successivamente, viene chiamato il metodo *updateDB*, che si occupa di aggiornare il *database* inserendo una nuova *tupla* in *Misuras* (Figura 26) per il modello 3D appena creato e, quindi, anche la relativa *tupla* in *Files* (Figura 27). Alla fine di questo processo, viene settato il riferimento

tra la galleria fotografica ed il modello 3D, inserendo nel *<ref_misura>* l'ID della galleria elaborata (Figura 28).

```
ent.Misuras.First(a => a.MisuraId == myPath.misuraId).Ref_Misura = m.MisuraId;
ent.SaveChanges();
```

Figura 28 Codice per l'aggiornamento del legame tra galleria fotografica e modello 3D.

Figure 28 *Code for the update of the link between photo gallery and 3D model.*

4.2.2 Classe Launcher

La classe *Launcher* ha il compito di interagire col programma *VisualSFM* e con le cartelle di origine e di destinazione delle gallerie fotografiche e dei modelli 3D.
Il metodo *createTempModel* si occupa di creare il modello della galleria fotografica ricevuta come input e memorizzare in modo temporaneo il risultato. Nello specifico questo metodo esegue il programma *VisualSFM* da linea di comando:

- *.\VisualSFM.exe sfm+pmvs pathPhoto pathTemp\nameResultTemp*

La linea di comando è definita da:

- *.\VisualSFM.exe* che esegue *VisualSFM*;
- *sfm+pmvs* sono le opzioni per la generazione della nuvola di punti sparsa (*sfm*) e densa (*pmvs*);
- *pathPhoto* definisce la cartella che contiene la galleria fotografica da elaborare;
- *pathTemp* definisce il percorso temporaneo dove salvare il risultato;
- *\nameResultTemp* è il nome del risultato, assegnato sia alla cartella che al file .nvm.

Alla fine del processamento, nella cartella temporanea vengono memorizzati i risultati della ricostruzione 3D: la sottocartella denominata *nameResultTemp.nvm.cmvs* ed il file *nameResultTemp.nvm*. Inoltre, nella cartella contenente la galleria fotografica vengono creati per ogni foto i file *.mat* e *.sift*. Il file *.mat* (*Matchesper*) contiene i risultati di calcolo dei punti omologhi tra le immagini; il file *.sift* (*Scale-Invariant-Feature-Transform*), invece, rappresenta le proprietà di una quantità da non mutare qualora sia effettuata una trasformazione di scala.
Successivamente alla creazione del modello, il metodo esegue i controlli di integrità. In dettaglio, viene verificata l'unicità (non frammentazione) del modello prodotto (in formato .ply) controllando che esista soltanto la cartella *<00>* all'interno della cartella *<.nvm.cmvs>* (Figura 29).

```
if (Directory.Exists(this.pathTemp + @"\" + nameResultTemp + ".nvm.cmvs" + @"\00")
    && !Directory.Exists(this.pathTemp + @"\" + nameResultTemp + ".nvm.cmvs" + @"\01"))
{
    executed = true;
    ApplicationContext.Instance.LogWriter.
        Info("Class:Launcher.cs - Method:createTempModel - Temp Model created");
}
```

Figura 29 Codice di verifica della non frammentazione del modello 3D generato.

Figure 29 *Code for the non-fragmentation check of the 3D model.*

Al termine della verifica, il metodo *saveModel* (invocato attraverso il codice in Figura 25) sposta il file .ply del modello 3D dalla cartella temporanea alla cartella di destinazione finale. Nello

specifico, come mostrato in Figura 30, il comando *File.Move()* sposta il file *<option-0000.ply>* dalla sottocartella *<\00\models>* alla cartella di destinazione *pathResult*.

Figura 30 Codice per lo spostamento del file 3D *.ply* nella cartella di destinazione finale.

Figure 30 *Code for moving the 3D .ply file into final destination folder.*

```
public void saveModel(string pathResult, string folderToMove)
{
    bool executed = false;

    try
    {
        Directory.CreateDirectory(pathResult);
        File.Move(this.pathTemp + @"\" + folderToMove +
            @"\00\models\option-0000.ply ", pathResult + @"\option-0000.ply");
        executed = true;
        ApplicationContext.Instance.LogWriter.
            Info("Class:Launcher.cs - Method:saveModel - Model saved");
    }
```

5. Possibili applicazioni di *TECH4DA*

A seguire lo sviluppo tecnologico descritto precedentemente, presentiamo un possibile panorama delle applicazioni per cui *TECH4DA* può essere utilizzata a valle degli eventuali adattamenti, che saranno utili per i diversi scopi che si prefigge di raggiungere. Questa applicazione affianca quelle generaliste già presenti sugli "store" delle diverse piattaforme ma, a differenza di quelle, presenta delle peculiarità che la rendono particolarmente efficace nell'ambito della collezione delle informazioni non solo visive ma anche informative e documentali. Rispetto alle soluzioni comuni pre-installate sui diversi smartphone, *TECH4DA* è dedicata alla acquisizione contemporanea di immagini e testi (da parte di utenti generici) che verranno poi analizzati da sistemi di processamento da cui estrarre l'informazione a valore aggiunto utile per i fini di ricerca e documentazione degli effetti prodotti da eventi naturali quali ad esempio terremoti. La possibilità di ricostruire cronologicamente l'evoluzione di un fenomeno, corredando l'informazione testuale ad un'immagine fotografica permette di ripercorrere la progressione del danneggiamento di un'area soggetta ad eventi ripetuti come, ad esempio, una sequenza sismica. Nel caso di successioni di eventi sismici che interessino una zona geograficamente definita, questa applicazione permette il confronto delle immagini acquisite dopo un primo evento con quelle relative a scosse successive. Inoltre favorisce la comparazione tra le immagini acquisite dopo il primo evento con quelle fornite da librerie di realtà aumentata che fornirebbero lo stato delle cose prima di un danneggiamento. Tali informazioni possono consentire una valutazione sugli effetti che incidono il tessuto economico e sociale di un'area, quindi contribuendo ad una maggiore consapevolezza nella fase di pianificazione della ricostruzione, con particolare attenzione al mantenimento dell'identità dei luoghi.
In Figura 31 vengono riprodotte le principali funzionalità già implementate, da adattare per specifiche esigenze, che interessano i vari blocchi del sistema. Le attività di consultazione dei dati sono aperte a tutti gli utenti; invece, l'inserimento dei dati necessita che l'utente sia loggato affinché siano riconosciuti i permessi ad esso associati.
Nel seguito di questo paragrafo sono illustrate alcune possibili applicazioni, che hanno carattere esemplificativo e non esaustivo. In particolare, i tre esempi servono a mostrare al lettore le potenzialità dello strumento *TECH4DA*:

1. per la raccolta di dati ed immagini da parte di utenti accreditati o non registrati (§5.1). La prima possibilità che questa applicazione mette a disposizione è quella della raccolta di immagini descrittive di un evento disastroso naturale (terremoto, eruzione vulcanica, ecc.) da parte di utenti, accreditati o meno che possono contribuire alla collezione di informazioni utili a delineare singoli effetti sull'ambiente costruito e/o naturale. Questo modulo, che può essere usato per differenti finalità (es. segnalare danni/guasti dovuti ad

eventi di differente natura), nasconde, lato server, un motore per la ricostruzione 3D dell'oggetto fotografato (come descritto nel par. 4.2.2);

2. per la raccolta, attraverso i cittadini, di dati sugli effetti permanenti e transitori dei terremoti attraverso la compilazione del questionario del sistema HSIT (www.haisentitoilterremoto.it) (§5.2). La partecipazione diretta degli utenti è uno degli strumenti su cui questa applicazione intende basare il proprio successo;
3. sempre nel caso di un terremoto, per la raccolta delle informazioni che permettano di (1) definire un quadro dell'area maggiormente danneggiata nella fase di emergenza post-sismica e (2) consentire una valutazione tra diffusione del danno e vulnerabilità dell'edificato per località, attraverso rilievi di dettaglio nell'area più colpita (§5.3), supportando quindi le attività relative ai rilievi macrosismici.

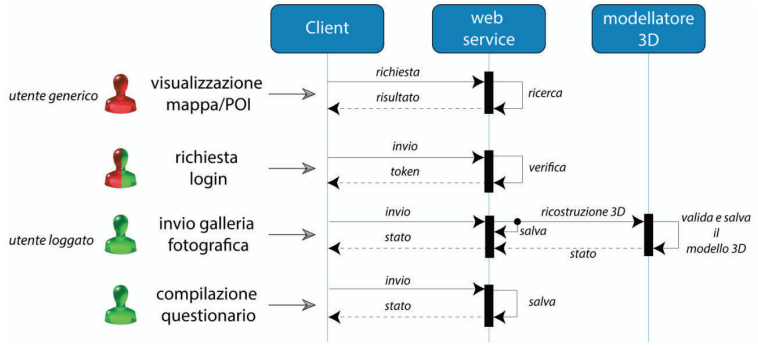


Figura 31 Schema delle funzionalità implementate e interazione con i blocchi del sistema. La visualizzazione dei dati è aperta a qualunque utente (utente rosso); invece, l'inserimento dei dati è disponibile solo una volta che l'utente sia stato loggato (utente verde).

Figure 31 *Diagram of the implemented functionalities and interaction with the system blocks. The data visualization is open to any user (red user); however, data entry is only available once the user has been logged (green user).*

5.1 Raccolta dati e immagini

Un evento catastrofico naturale può implicare la perdita del più importante dei valori cioè quello della vita umana ma anche intaccare violentemente il tessuto sociale ed urbano. Rispetto a quest'ultimo e coerentemente a quanto evidenziato dal protocollo "Make cities and human settlements inclusive, safe, resilient and sustainable" dell'obiettivo 11 "Sustainable Cities and Communities" appartenente ai 17 impegni delle Nazioni Unite sullo sviluppo sostenibile (https://www.unglobalcompact.org/what-is-gc) l'acquisizione digitale della memoria diventa un passo preliminare al raggiungimento degli obiettivi. La sequenza sismica che ha recentemente colpito l'Italia Centrale ha reso evidente con quanta rapidità possano cambiare i luoghi a cui siamo abituati. Per questa ragione *TECH4DA* può operare come strumento per ricostruire la memoria fotografica dei luoghi prima di un evento sismico e per supportare le successive attività di rilievo del danneggiamento. Questa applicazione è stata ideata con lo scopo di facilitare la raccolta di immagini utilizzabili anche per una ricostruzione tridimensionale dell'oggetto fotografato, ma è importante sottolineare che la possibilità di valutare il danneggiamento di beni e artefatti attraverso il confronto tra immagini non è vincolato alla presenza di immagini pre-evento acquisite attraverso la applicazione; *TECH4DA* infatti permetterà il confronto tra le immagini acquisite dopo un accadimento e quelle rese disponibili, ad esempio, attraverso librerie commerciali di realtà aumentata permettendo di fatto un'analisi comparata sullo stato dei beni.

A titolo di esempio, in Figura 32, è mostrata l'acquisizione di fotografie relative al distacco di intonaco su una parete di un edificio dell'Università della Calabria.

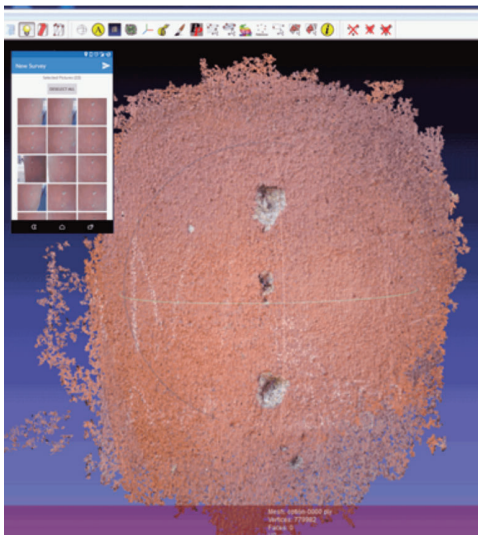
Figura 32 Acquisizione di immagini relative al distacco di intonaco su una parete di un edificio dell'Università della Calabria.

Figure 32 *Acquisition of images related to the detachment of plaster on a building of the University of Calabria.*

L'applicazione consente sia di ottenere direttamente le fotografie dalla fotocamera, che di caricare fotografie già disponibili nella memoria del dispositivo mobile.*TECH4DA* è pensata per utenti di diversa preparazione: un utente "non preparato" al quale si chiede di documentare un evento collezionando le immagini che, anche attraverso un percorso guidato, siano utilizzabili per coloro che dai rilievi dei danneggiamenti possono trarre informazioni utili per la generazione di un rilievo 3D (utente istituzionale preparato).
Nella Figura 33 è mostrata la ricostruzione 3D della porzione di una parete con alcuni segni di degrado. Il modello è stato generato automaticamente sulla base di un set fotografico di 22 foto acquisite tramite l'applicazione (cfr. Figura 32). Il modello generato secondo quanto descritto in §4.1, in formato .ply può essere visualizzato attraverso software open-source, quali ad esempio *Meshlab*. *TECH4DA* consente attraverso questo servizio di navigare intorno al danneggiamento, anche di piccola scala come il distacco di intonaco mostrato nell'esempio, dotando l'utilizzatore di uno strumento aggiuntivo per valutarne la gravità.

Figura 33 Modello 3D ricostruito a partire dal dataset fotografico e visualizzato con il software open source Meshlab [modificato da Costanzo et al., 2018].

Figure 33 *3D model built from photographic dataset and visualization of the open source software Meshlab [modified by Costanzo et al., 2018].*



5.2 Raccolta dati macrosismici tramite questionario nell'ambito del progetto HSIT (Hai Sentito Il Terremoto)

Il progetto HSIT (www.haisentitoilterremoto.it) dell'INGV coinvolge da anni i cittadini nello scambio di informazioni con i ricercatori dell'INGV. Grazie alla diffusione e alla rapidità di questa comunicazione dal 1997 (e dal 2007 nella forma attuale) il sito raccoglie tramite un questionario le informazioni sugli effetti dei terremoti, le elabora automaticamente e pubblica in tempo quasi reale dati, grafici e mappe dedicati a ciascun terremoto. Ad oggi sono rappresentati più di 11700 eventi sismici, frutto dell'elaborazione di oltre 1000000 questionari compilati sia da volontari che da utenti registrati al sito. A questi ultimi viene inviata automaticamente un'email, con la richiesta di compilazione del questionario, dopo il verificarsi di un terremoto, anche di piccola entità, se situati nell'area di probabile risentimento. La possibilità di poter descrivere gli effetti di qualunque intensità, inclusi quelli transitori dovuti a terremoti di magnitudo molto bassa, fa mantenere alta l'attenzione sul fenomeno terremoto anche in assenza di eventi distruttivi. Per questo motivo gli utenti registrati al sito HSIT (oltre 26000) potrebbero rappresentare un'ottima opportunità per la diffusione di *TECH4DA*, in quanto persone sicuramente interessate al crowdsourcing e abituate a riportare gli effetti di un sisma.
Il sistema HSIT ad oggi opera prevalentemente attraverso il proprio sito web e negli ultimi anni il progetto si è dotato anche di una versione app per Android. La versione per smartphone presenta dei vantaggi: la diffusione del mezzo è vastissima, la localizzazione è molto precisa (per l'uso del GPS o della cella di riferimento) e la portabilità dello smartphone la rende sempre disponibile (riducendo così l'intervallo di tempo tra il verificarsi del sisma e la compilazione del questionario), a tutto vantaggio della precisione e qualità delle risposte. L'attuale app HSIT non è più attiva in quanto, a causa del continuo aggiornamento dei sistemi operativi e dell'avanzamento della tecnologia HW, avrebbe necessità di costanti e rapide modifiche per essere funzionante. Quindi la possibilità di inserire il sistema HSIT all'interno di *TECH4DA* presenterebbe il reciproco vantaggio dell'ampia diffusione e del ripristino della versione *mobile*. Un notevole valore aggiunto sarebbe comunque rappresentato dal poter associare l'eventuale foto del danno al questionario sugli effetti di HSIT. Infatti, la caratterizzazione della struttura e la stima del danneggiamento di un edificio possono essere problematiche se fatte da parte di persone non esperte. Viceversa le immagini dirette e la ricostruzione 3D di *TECH4DA* permettono la verifica del danneggiamento da parte di personale dell'INGV e possono indirizzare nell'assegnazione del grado di intensità ottenuto confrontando anche gli effetti, avvenuti nello stesso luogo, descritti nel questionario.

5.3 Supporto ai rilievi macrosismici

L'Istituto Nazionale di Geofisica e Vulcanologia (INGV) ha, tra i suoi compiti, quello di effettuare i rilievi dei danneggiamenti del patrimonio immobiliare pubblico e privato nel periodo successivo ad un evento sismico, per meglio definirne gli effetti locali in termini di intensità macrosismica. Questi rilievi sono eseguiti da personale esperto appartenente al gruppo QUEST (Quick Earthquake Survey Team). Il rilievo si svolge nell'area di maggior danneggiamento, secondo le modalità di indagine macrosismica condotta da tempo dall'INGV, osservando il danno visibile all'esterno degli edifici e focalizzando l'indagine sull'edilizia civile e residenziale in quanto maggiormente rappresentativa dal punto di vista macrosismico. L'attribuzione dell'intensità in EMS98 [Grünthal, 1998], quindi, tiene conto principalmente degli effetti sull'edilizia convenzionale e del risentimento sulla popolazione. L'analisi dati dei rilievi ha una duplice funzione: (1) definire un quadro dell'area maggiormente danneggiata nella fase di emergenza post-sismica e (2) consentire una valutazione tra diffusione del danno e vulnerabilità dell'edificato per località, attraverso rilievi di dettaglio nell'area più colpita.

Uno dei limiti dell'indagine macrosismica diretta è, da sempre, quello della copertura dell'area danneggiata. Fin quando l'area del danneggiamento è circoscritta a poche decine di località il rilievo viene svolto con estrema rapidità ed efficienza e si riesce a coprire un'area sufficientemente ampia per pubblicare un piano quotato esaustivo (l'insieme delle località rilevate associate ad un grado di intensità macrosismica). Quando invece l'area di danneggiamento è particolarmente ampia subentrano delle inevitabili difficoltà operative. Prima fra tutte quella di delineare l'area di maggiore risentimento che risulterà comunque molto estesa. Per fare questa operazione è spesso necessario raccogliere le informazioni che arrivano dai vari media e cominciare a stilare una lista di comuni presso i quali effettuare il rilievo. Maggiori sono le informazioni relative al danneggiamento di una località più tempestivo possibile sarà il rilievo diretto. A volte si cominciano a rilevare delle località e man mano che si effettua il lavoro si viene a conoscenza di altri comuni ugualmente danneggiati.
In questo quadro di necessità di razionalizzare energie e risorse verso obiettivi chiari e identificabili è possibile individuare una possibile applicazione di *TECH4DA*.
Ci viene incontro, nel nostro caso, la capacità di ognuno di noi, attraverso uno smartphone e una tecnologia dedicata (per l'appunto *TECH4DA*) di essere degli "inviati" nei luoghi colpiti da un sisma. Un pò come avviene con altre diffuse App (es: Waze per il traffico automobilistico in tempo reale) ogni utente registrato su *TECH4DA* potrà segnalare, attraverso una fotografia, eventuali danneggiamenti di edifici nel luogo in cui si trova. *TECH4DA* sarà in grado di mappare le immagini inviate creando un database di contenuti. Infatti, oltre alla semplice immagine, l'utente registrato potrà inserire alcuni metadati, qualora noti, che potranno sostanziare meglio le informazioni inviate seguendo uno schema predisposto (Figura 34).

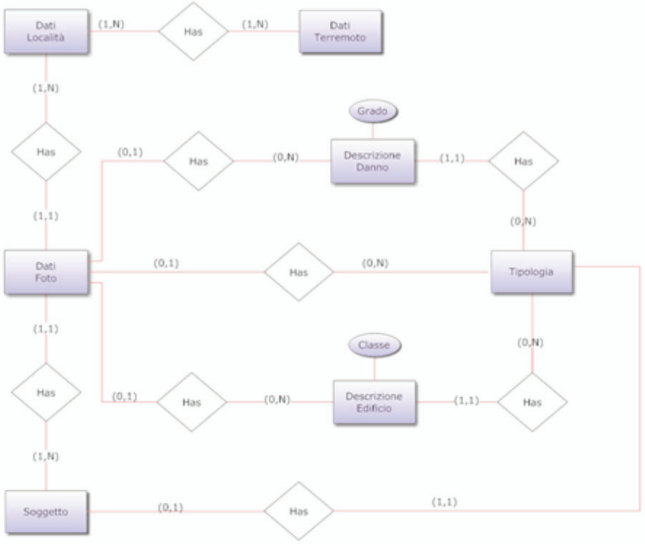


Figura 34 Schema delle informazioni per il database finalizzato alle attività del gruppo QUEST e relativo ai metadati delle immagini (immagine per concessione di Manuela Sbarra).

Figure 34 *Scheme of the information for the database related to the activities of the QUEST group and referred to the metadata of the images (image by courtesy of Manuela Sbarra).*

Pertanto si potrà documentare il danneggiamento anche inserendo delle informazioni relative alla tipologia dell'edificio, alla sua classe di vulnerabilità e alla descrizione del danneggiamento stesso. La possibilità, per gli operatori specializzati del rilievo macrosismico, di accedere ad un database di informazioni sul danneggiamento del terremoto redatto da una platea potenzialmente vastissima, darà la possibilità di definire meglio l'area del danneggiamento e di indirizzare in maniera più efficace le risorse. Il database costituirà esso stesso un contenitore di informazioni specifiche che potrà essere implementato nel corso del tempo costituendo, al tempo stesso, memoria storica dello stato di conservazione degli edifici stessi.
Ad oggi, nella pagina "Home" dell'applicazione (cfr. Figura 7.a), premendo il bottone "Compila il questionario" sul terremoto, è possibile raggiungere la schermata per riempire il questionario

con le informazioni sul risentimento di un terremoto. La prima schermata che viene visualizzata è molto simile a quella di aggiunta di una nuova galleria fotografica (Figura 35).

Figura 35 Schermata mostrata alla selezione del bottone "Compila il questionario sul terremoto" presente nella Home.

Figure 35 *Page shown at the selection of the button "Compila il questionario sul terremoto" (Complete the questionnaire about earthquake) from the Home.*

Visto che la classe in cui è stata implementata questa schermata è la stessa di quella per una nuova galleria fotografica, per scegliere quali componenti della pagina (e quindi quali funzioni rendere disponibili in dipendenza del contesto) si è scelto di passare al costruttore della classe, come parametro, una variabile booleana. Se il valore di questa variabile è *true* allora nella schermata sono presenti i componenti utili a raccogliere i dati sul risentimento del terremoto, viceversa sono rese disponibili le funzionalità per creare una nuova galleria fotografica. Prima di procedere alla compilazione del questionario, è necessario effettuare la raccolta dati indicando: un nome dell'osservazione, eventuali note, il soggetto (edificio, oggetto o ambiente) ed eventuale foto relativa al risentimento osservato. Successivamente, è possibile procedere verso la schermata contenente alcune domande specifiche alle quali l'utente può rispondere. In questa schermata, ma anche in quella precedente, è richiesta l'abilitazione della geolocalizzazione GPS. Questa seconda parte del questionario è divisa in due sezioni: una relativa agli effetti sulle persone e cose (Figura 36.a) e la successiva agli effetti sull'ambiente (Figura 36.b).

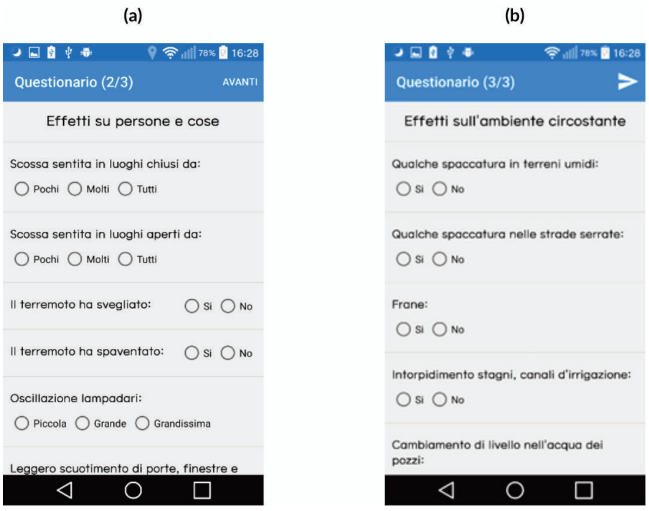


Figura 36 Questionario da compilare in caso di terremoto: effetti su persone e cose (a) e effetti sull'ambiente (b).

Figure 36 *Questionnaire to be complete in case of earthquake: effects on people and things (a) and effects on the environment (b).*

Dalla figura si può vedere che il questionario è composto soltanto da domande a risposta multipla, quindi per realizzare queste pagine è stato necessario utilizzare dei radiogroup. Visto che *Xamarin.Forms* non fornisce questo tipo di controlli, in tutte le sezioni in cui appaiono è stato fatto uso delle classi fornite dalla libreria *XLabs.Forms.Controls*.

6. Conclusioni e Sviluppi futuri

TECH4DA rappresenta, per quanto a noi noto, uno strumento innovativo in grado di consentire un'ampia partecipazione di utenti all'acquisizione di informazioni sugli effetti prodotti da eventi estremi, sfruttando strumenti tecnologici largamente diffusi nella popolazione. In questo lavoro abbiamo presentato i risultati della prima fase dello sviluppo del sistema, che ha portato alla progettazione e realizzazione di una versione prototipale, composta da tre componenti principali: Web Service, Applicazione Mobile e motore di ricostruzione 3D. Lo sviluppo è stato effettuato durante il lavoro di tre tesi di laurea svolte in collaborazione con il Dipartimento di Matematica e Informatica dell'Università della Calabria.
A valle dello sviluppo sono state delineate le possibili applicazioni per le quali, con un necessario adeguamento, *TECH4DA* può portare dei benefici in termini di aggiornamento ed operatività (HSIT e QUEST). Le potenziali applicazioni di *TECH4DA* possono essere rivolte a categorie diverse di utenti: l'estensione relativa ad HSIT è rivolta ad un pubblico più ampio possibile, e non-esperto; difatti, è destinata a raccogliere il maggior numero possibile di informazioni sì da renderle statisticamente significative. D'altro canto, la sezione dedicata al supporto del rilievo macrosismico (come da immagini rappresentative nelle Figure 35 e 36) è riservata al personale esperto che interviene sul campo in seguito ad un evento. La realizzazione di queste estensioni potrebbe ampliare l'offerta tecnologica dell'applicazione che, pur incompleta, è di fatto già funzionante, assumendo un ruolo strategico in occasione di eventi estremi. Ad esempio, attraverso una delle molteplici funzionalità, *TECH4DA* potrebbe supportare le operazioni di rilievo ed archiviazione dei dati sul campo (immagini e testi), rappresentando un utile supporto alla rapida mappatura dell'intensità macrosismica. Sebbene *TECH4DA* non è stata progettata unicamente per operare su danni legati alla attività sismica sul patrimonio pubblico e privato è palese che, per le peculiarità dell'INGV, l'analisi dei danni generati da un terremoto sono di prioritario interesse per l'Ente. Infatti, collaborando con il personale specializzato e seguendo specifiche esigenze, possono essere realizzati format digitali che il singolo operatore può compilare ed inviare al server con una semplice connessione dati. L'applicazione di *TECH4DA* può inoltre garantire l'acquisizione di immagini del territorio e dei centri urbani in qualunque momento, contribuendo alla crescita della memoria digitale e fornendo, all'occorrenza, informazioni propedeutiche alle fasi di recupero, restauro e ricostruzione.
Alcuni possibili sviluppi futuri già pianificati saranno: (1) la realizzazione della cartografia per IOS; (2) la registrazione di nuovi utenti direttamente dall'interfaccia *TECH4DA*; (3) la possibilità di operare off-line per un invio differito dei dati; (4) lo sviluppo dei sistemi di restituzione dei dati e delle relative elaborazioni, sia in ambiente GIS che attraverso un visualizzatore tridimensionale; (5) l'invio di notifiche "push" utilizzata per inviare le informazioni per aumentare il coinvolgimento nell'uso dell'applicazione anche quando l'utente non usa attivamente l'applicazione di destinazione. Quest'ultima caratteristica, cioè il coinvolgimento e la cura del "cittadino" richiede un sostanziale cambio di punto di vista: il cittadino non come destinatario finale degli sviluppi scientifici e tecnologici ma comprimario degli stessi enfatizzando il percorso bidirezionale che una comunicazione consapevole deve contemplare.
In considerazione dello stato dello sviluppo del prototipo, sarà resa disponibile su richiesta l'applicazione mobile per il sistema *Android* (contattare gli Autori).

Ringraziamenti

Il lavoro presentato è stato sviluppato a seguito dell'esperienza relativa al PON 01_02710 "MASSIMO" (Monitoraggio in Area Sismica di SIstemi MOnumentali), finanziato dal Ministero dell'Istruzione, Università e Ricerca. Si ringraziano la Dr.ssa Maria Fabrizia Buongiorno, responsabile scientifico e coordinatrice del progetto "MASSIMO", e il Dr. Fawzi Doumaz, responsabile della sede INGV di Rende (CS) e del sistema informatico di gestione dei dati e prodotti dello stesso progetto. Gli autori ringraziano l'editor e i revisori per i loro commenti e suggerimenti che hanno consentito di migliorare significativamente il manoscritto.

Bibliografia

Alonso G., Casati F., Kuno H., Machiraju V., (2004). *Web Services. Concepts, Architectures and Applications*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-10876-5.

Costanzo A., Falcone S. Bisignano R., Straface M., Ritacco G., La Piana C., Musacchio M., Calimeri F., (2018). *A Smartphone Application for Supporting the Data Collection and Analysis of the Cultural Heritage Damaged during Natural Disasters*. Proceedings 2018, 2, 121.

Larman C., (2005). *Applying UML and patterns - An Intro to OOA/D and Iterative Development*. Prentice Hall, pp. 736, ISBN 9780131489066.

Montuori A., Costanzo A., Gaudiosi I., Vecchio A., Pannaccione Apa M.I., Gervasi A., Falcone S., La Piana C., Minasi M., Stramondo S., Buongiorno M.F., Doumaz F., Musacchio M., Casula G., Caserta A., Speranza F., Bianchi M.G., Guerra I., Porco G., Compagnone L., Cuomo M., De Marco M., (2016). *The MASSIMO system for the safeguarding of historic buildings in a seismic area: operationally-oriented platforms*. European Journal of Remote Sensing, 2016, 49, 397-415. http://dx.doi.org/10.5721/EuJRS20164922.

Mousavi V., Khosravi M., Ahmadi M., Noori N., Hosseini Naveh A., Varshosaz M., (2015). *The performance evaluation of multi-image 3D reconstruction software with different sensors*. International Conference on Sensors & Models in Remote Sensing & Photogrammetry, 23-25 Nov 2015, Kish, Island, Iran.

Wu C., Agarwal S., Curless B. and Seitz S.M., (2011). *Multicore Bundle Adjustment - IEEE Conference on Computer Vision and Pattern Recognition* (CVPR 2011), 20-25 June 2011, Colorado Springs, CO, USA.

Wu C., Agarwal S., Curless B., Seitz S.M., (2012). *Schematic Surface Reconstruction, IEEE Conference on Computer Vision and Pattern Recognition* (CVPR 2012), 16-21 June 2012, Providence, RI, USA.

Wu C., (2013). *Towards Linear-time Incremental Structure From Motion - International Conference on 3D Vision*, 29-30 June 2013, Seattle, USA.

Sitografia

http://ccwu.me/vsfm/
https://developer.android.com/guide/index.html
https://developer.xamarin.com/api/
https://developer.xamarin.com/guides/android/platform_features/maps_and_location/maps/part_2_-_maps_api/
https://docs.microsoft.com/it-it/dotnet/csharp/
https://msdn.microsoft.com/it-it/library/bb399567(v=vs.110).aspx
http://nlog-project.org/
https://www.asp.net/mvc
https://www.asp.net/web-api
https://www.di.ens.fr/cmvs/
https://www.di.ens.fr/pmvs/
http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

http://www.meshlab.net/
https://www.microsoft.com/it-it/sql-server/sql-server-editions-express
https://www.visualstudio.com/it/
https://www.un.org/sustainabledevelopment/cities/
http://www.questingv.it/

QUADERNI di GEOFISICA

ISSN 1590-2595 http://istituto.ingv.it/it/le-collane-editoriali-ingv/quaderni-di-geofisica.html/

I QUADERNI DI GEOFISICA (QUAD. GEOFIS.) accolgono lavori, sia in italiano che in inglese, che diano particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari che necessitano di rapida diffusione nella comunità scientifica nazionale ed internazionale. Per questo scopo la pubblicazione on-line è particolarmente utile e fornisce accesso immediato a tutti i possibili utenti. Un Editorial Board multidisciplinare ed un accurato processo di peer-review garantiscono i requisiti di qualità per la pubblicazione dei contributi. I QUADERNI DI GEOFISICA sono presenti in "Emerging Sources Citation Index" di Clarivate Analytics, e in "Open Access Journals" di Scopus.

QUADERNI DI GEOFISICA (QUAD. GEOFIS.) welcome contributions, in Italian and/or in English, with special emphasis on preliminary elaborations of data, measures, and observations that need rapid and widespread diffusion in the scientific community. The on-line publication is particularly useful for this purpose, and a multidisciplinary Editorial Board with an accurate peer-review process provides the quality standard for the publication of the manuscripts. QUADERNI DI GEOFISICA are present in "Emerging Sources Citation Index" of Clarivate Analytics, and in "Open Access Journals" of Scopus.

RAPPORTI TECNICI INGV

ISSN 2039-7941 http://istituto.ingv.it/it/le-collane-editoriali-ingv/rapporti-tecnici-ingv.html/

I RAPPORTI TECNICI INGV (RAPP. TEC. INGV) pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico come manuali, software, applicazioni ed innovazioni di strumentazioni, tecniche di raccolta dati di rilevante interesse tecnico-scientifico. I RAPPORTI TECNICI INGV sono pubblicati esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. Un Editorial Board multidisciplinare ed un accurato processo di peer-review garantiscono i requisiti di qualità per la pubblicazione dei contributi.

RAPPORTI TECNICI INGV (RAPP. TEC. INGV) publish technological contributions (in Italian and/or in English) such as manuals, software, applications and implementations of instruments, and techniques of data collection. RAPPORTI TECNICI INGV are published online to guarantee celerity of diffusion and a prompt access to published data. A multidisciplinary Editorial Board and an accurate peer-review process provide the quality standard for the publication of the contributions.

MISCELLANEA INGV

ISSN 2039-6651 http://istituto.ingv.it/it/le-collane-editoriali-ingv/miscellanea-ingv.html

MISCELLANEA INGV (MISC. INGV) favorisce la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV. In particolare, MISCELLANEA INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli, ecc. La pubblicazione è esclusivamente on-line, completamente gratuita e garantisce tempi rapidi e grande diffusione sul web. L'Editorial Board INGV, grazie al suo carattere multidisciplinare, assicura i requisiti di qualità per la pubblicazione dei contributi sottomessi.

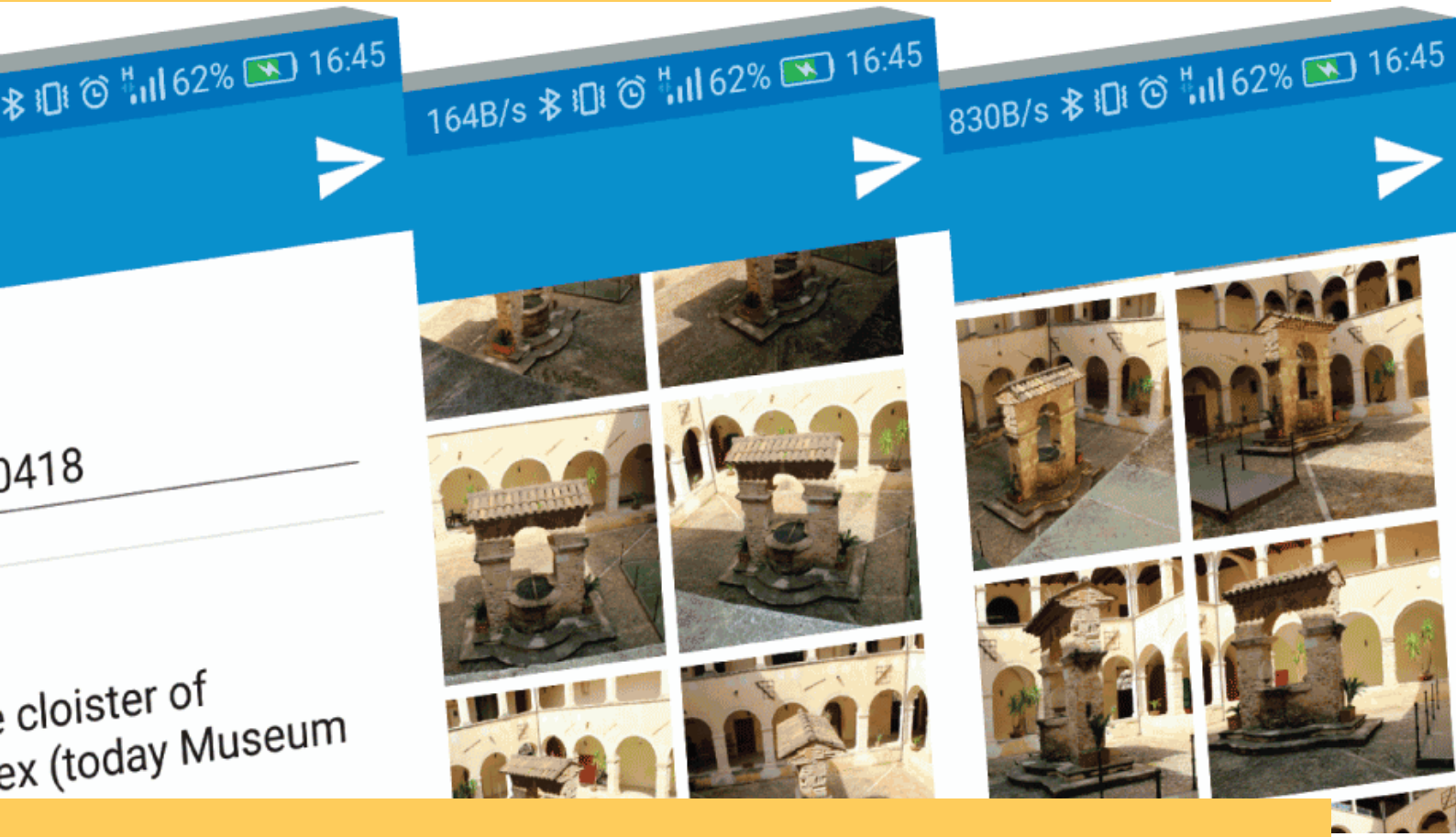
MISCELLANEA INGV (MISC. INGV) favours the publication of scientific contributions regarding the main activities carried out at INGV. In particular, MISCELLANEA INGV gathers reports of scientific projects, proceedings of meetings, manuals, relevant monographs, collections of articles etc. The journal is published online to guarantee celerity of diffusion on the internet. A multidisciplinary Editorial Board and an accurate peer-review process provide the quality standard for the publication of the contributions.

Coordinamento editoriale e impaginazione
Francesca DI STEFANO, Rossella CELI
Istituto Nazionale di Geofisica e Vulcanologia

Progetto grafico e impaginazione
Barbara ANGIONI
Istituto Nazionale di Geofisica e Vulcanologia

Istituto Nazionale di Geofisica e Vulcanologia
Via di Vigna Murata, 605
00143 Roma
t. +39 06518601

www.ingv.it



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA