Alma Mater Studiorum - Università di Bologna

Scuola di Ingegneria e Architettura

Dipartimento di Informatica - Scienza e Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea
in Fondamenti di Informatica T-2

Progetto e sviluppo di una applicazione Android per la visualizzazione delle aree di attesa per emergenze

Candidato
Daniele Santini

Relatore
Prof. Enrico Denti

Correlatore
Ing. Ambra Molesini

Anno Accademico 2017/2018
Sessione III

Sommario

Ogni Comune italiano ha l'obbligo di realizzare un piano di emergenza per fronteggiare qualsiasi calamità. Al suo interno vengono indicate le aree di attesa in cui la popolazione si deve recare in caso di emergenza per ricevere informazioni e i primi generi di conforto.

Questi documenti, e di conseguenza la posizione delle aree di attesa, sono però spesso sconosciuti a gran parte della popolazione e difficili da trovare. Esistono alcuni progetti di mappatura digitale delle aree di attesa, ma sono tutti a carattere strettamente locale.

Lo scopo di questa tesi è sviluppare una applicazione per smartphone Android che permetta di visualizzare interattivamente le aree di attesa di emergenza previsti dal Piano comunale di Protezione Civile di una qualsiasi città e guidare l'utente verso essi.

Indice

Capitolo 1

Introduzione

Ogni Comune italiano ha l'obbligo di realizzare un piano di emergenza per fronteggiare qualsiasi calamità. Al suo interno vengono indicate le aree di attesa in cui la popolazione si deve recare in caso di emergenza per ricevere informazioni e i primi generi di conforto.

Questi documenti, e di conseguenza la posizione delle aree di attesa, sono però spesso sconosciuti a gran parte della popolazione e difficili da trovare. Esistono alcuni progetti di mappatura digitale delle aree di attesa, ma sono tutti a carattere strettamente locale.

Lo scopo di questa tesi è sviluppare una applicazione per smartphone Android che permetta di visualizzare interattivamente le aree di attesa di emergenza previsti dal Piano comunale di Protezione Civile di una qualsiasi città e guidare l'utente verso essi.

Il processo di sviluppo inizierà con la definizione ed una prima analisi dei requisiti (funzionali e non funzionali) dell'applicazione e proseguirà con la fase di analisi del problema, dove sarà indagato ogni aspetto del dominio applicativo e saranno valutate possibili scelte alternative (cap. 2).

Il processo di sviluppo proseguirà quindi con la fase di progettazione, nella quale, partendo dall'analisi condotta nella fase precedente, si definirà l'architettura del sistema che al meglio coniuga le esigenze espresse attraverso i requisiti funzionali e i vincoli specificati dai requisiti non funzionali (cap. 3). Tra i vincoli non funzionali che guidano la scelta dell'architettura sarà posta particolare attenzione specialmente ai vincoli di robustezza, manutenibilità ed estendibilità. Vista la presenza di un vincolo sulla piattaforma su cui implementare l'applicazione (Android), in fase di progettazione sarà appropriato adottare soluzioni per migliorare la portabilità, come la definizione di una architettura generica e di una più specifica per la piattaforma di destinazione.

Successivamente il sistema verrà implementato e collaudato. In particolare, prima verranno implementati e collaudati i singoli moduli (cap. 4), quindi verrà fatto un test di integrazione (cap. 5).

Capitolo 2

Analisi

Studio delle motivazioni che portano alla creazione dell'applicazione, analisi del dominio che verrà rappresentato e specifica di funzionalità, comportamenti e interfaccia che l'applicazione dovrà avere, senza descrivere una loro possibile realizzazione.

2.1 Definizione dei requisiti

Comportamento atteso dalla applicazione e requisiti che dovrà rispettare:

2.1.1 Requisiti funzionali

Visualizzazione delle aree di attesa

L'utente dovrà essere in grado di visualizzare una mappa che evidenzi le aree di attesa nelle sue vicinanze, con la possibilità di effettuare lo zoom e spostarsi nella mappa. L'applicazione non dovrà essere limitata a una sola città: dovrà essere utilizzabile da qualsiasi città (se i dati per quella città sono disponibili).

Visualizzazione dei dettagli di un'area di attesa

L'applicazione deve permettere all'utente di selezionare un'area di attesa e quindi visualizzare i dettagli di essa. I dettagli includono nome, descrizione, coordinate geografiche, foto se disponibile.

Supporto per più tipi di area di attesa

L'applicazione deve essere in grado di visualizzare per ogni area i tipi di emergenza per cui essa è pensata (terremoto, alluvione, ...).

Navigazione verso un'area di attesa

Una volta scelta un'area di attesa l'utente deve avere la possibilità di essere guidato verso esso.

Funzionamento offline

Essendo pensato per l'utilizzo in situazioni di emergenza, in cui probabilmente la rete cellulare sarà non funzionante, l'applicazione deve essere in grado di visualizzare le aree di attesa e i relativi dettagli anche in assenza di un collegamento alla rete internet.

2.1.2 Requisiti non funzionali

Piattaforma

L'applicazione deve essere realizzata sul sistema operativo Android.

2.2 Analisi del dominio

Studio della situazione attuale in cui l'applicazione andrà a inserirsi.

2.2.1 Mappa

Una **mappa** è una rappresentazione semplificata dello spazio che evidenzia relazioni tra componenti (oggetti, regioni) di quello spazio. Una mappa è comunemente una rappresentazione bidimensionale di uno spazio tridimensionale, come ad esempio una carta geografica.

Una mappa rispetta alcuni parametri di qualità come completezza dei particolari topografici, consistenza logica, accuratezza della posizione degli oggetti rappresentati, accuratezza tematica (corretta attribuzione delle classi degli oggetti) e accuratezza temporale.

Più in generale, le mappe possono essere usate per rappresentare qualsiasi proprietà locale del mondo o parte di esso, o qualsiasi altro spazio, anche mentale o concettuale [1].

Coordinate geografiche

Un punto P sulla superficie terrestre può essere rappresentato approssimando la terra con un elissoide (detto "elissoide di riferimento") orientato e quindi localizzando il punto su di esso con le *coordinate geografiche*, derivate dalle coordinate ellissoidiche [2][3][4]:

- **Latitudine**: l'angolo formato dalla normale (all'ellissoide) nel punto P con il piano equatoriale. Compresa fra -90°e +90°.
- **Longitudine**: l'angolo diedro formato dal piano meridiano passante per P ed il piano meridiano scelto come riferimento. Compresa fra -180°e +180°.

L'elissoide di riferimento orientato usato per approssimare la terra, la sua orientazione e il meridiano di riferimento vengono definiti da un *sistema di riferimento* geografico, detto anche "datum" [5].

Esistono numerosi sistemi di riferimento, ma quello attualmente più diffuso (utilizzato per esempio nei sistemi GPS) è WGS84. Esso si basa su una terna cartesiana XYZ a partire dalla quale vengono orientati l'elissoide di riferimento e definite le coordinate [6][7][8].

La terna cartesiana è definita in questo modo:

- origine nel centro di massa della terra (terna geocentrica)
- asse Z passante per il polo nord (corrispondente quindi all'asse di rotazione terrestre)
- asse X passante per il meridiano di Greenwich
- asse Y scelto per fare una terna destrorsa.

Definita questa terna possiamo definire le coordinate più semplicemente come:

- Latitudine: distanza angolare di un punto dall'equatore
- Longitudine: distanza angolare (attorno all'asse Z) di un punto dal meridiano di Greenwich

Una *proiezione* permette di rappresentare sulla mappa bidimensionale le coordinate dei punti (provenienti dalla superficie tridimensionale dell'elissoide).

Rappresentazione degli elementi

Per essere rappresentati su una mappa ogni oggetto necessiterà una **geometria** che ne descriva la forma. Alcuni esempi di possibili geometrie sono:

- **Nodo**: punto geografico, identificato da latitudine e longitudine

- **Percorso**: linea spezzata definita da una lista di nodi che ne definiscono i vertici
- **Poligono**: area geografica delimitata da una linea spezzata chiusa (primo e ultimo nodo corrispondono, esistono almeno 4 nodi)

Per rappresentare elementi geografici più complessi è possibile fare riferimento a standard internazionali appositi.

Un possibile standard è ISO 19107:2003, realizzato da ISO/TC211, il comitato tecnico della International Organization for Standardization che si occupa di informazioni geografiche e geomatica [9]. La norma definisce schemi concettuali per la descrizione delle caratteristiche spaziali degli elementi geografici, ed un insieme di operazioni nello spazio coerenti con tali schemi[10]. ISO 19107 è stato adattato dall'Ente Italiano di Normazione nella norma UNI EN ISO 19107:2005 [11]. ISO/TC211 sta attualmente sviluppando ISO/FDIS 19107, lo standard che andrà a sostituire ISO 19107.

Un altro possibile standard è RFC 7946, realizzato dalla Internet Engineering Task Force [12]. RFC 7946 definisce il formato GeoJSON per la codifica e il trasporto di dati geografici utilizzando JSON. Oltre a Point e Polygon (corrispondenti a Nodo e Poligono, con la differenza che in Polygon alla fine della lista dei vertici viene aggiunto il vertice iniziale) vengono definite altre geometrie come MultiPoint, LineString, MultiLineString, MultiPolygon, GeometryCollection.

Confine

Un **confine** è il limite di un territorio, di un terreno, di una regione geografica o di uno stato [13].

Bounding box L'esempio più basilare di confine è il **bounding box** (abbreviabile con '*bbox*'), il quale ha come limite esterno una latitudine minima e massima e una longitudine minima e massima. In termini più semplici questi indicano i limiti est, ovest, sud e nord.

I limiti dovranno rispettare alcuni vincoli [14]:

- -90 $\leq$ Latitudine $\leq$ +90
- Latitudine nord $>$ Latitudine sud
- -180 $\leq$ Longitudine $\leq$ +180
- Longitudine est != Longitudine ovest (se Latitudine ovest $<$ Latitudine est vorrà dire che il bounding box si sovrappone al meridiano opposto a quello di Greenwich, detto antimeridiano)
- |Longitudine ovest - Longitudine est| $\leq$ 360 (la condizione è già garantita dal fatto che la longitudine è compresa fra -180 e 180)

Visualizzazione mappa

Per visualizzare le mappe nell'applicazione è possibile usare un toolkit che faciliti la visualizzazione dei dati fornendo la mappa di sfondo su cui sovrapporre le geometrie. Alcuni dei più noti strumenti di questo tipo disponibili per Android sono:

Toolkit	Costo
Google Maps SDK	Gratuito su Android se non si utilizza StreetView [15][16].
TomTom Maps SDK	Gratis fino a 2500 transazioni al giorno, poi si paga una cifra che parte da 0,50 € ogni 1000 transazioni e scende fino ad arrivare a 0,42 € ogni 1000 transazioni una volta superate 10'000'000 transazioni [17][18].
HERE SDK	Gratis fino a 250'000 transazioni e 5000 utenti al mese, poi 1€ ogni 1000 transazioni [19][20].
Mapbox Maps SDK	Gratis fino a 50'000 utenti al mese, poi 0,50 € ogni 500 utenti [21][22].

2.2.2 Emergenze

Un'**emergenza** può essere definita come qualsiasi condizione critica che si manifesta in conseguenza del verificarsi di un evento di protezione civile, ovvero un fenomeno di origine naturale o antropica in grado di arrecare danno alla popolazione, alle attività, alle strutture e infrastrutture del territorio [23][24].

La legge n. 225 del 1992 all'art. 2 individua tre tipi di eventi di protezione civile:

- eventi naturali o connessi con l'attività dell'uomo che possono essere fronteggiati con interventi di singoli enti e amministrazioni in via ordinaria
- eventi naturali o connessi con l'attività dell'uomo che, per loro natura ed estensione, comportano l'intervento coordinato di più enti e amministrazioni in via ordinaria
- calamità naturali, catastrofi o altri eventi che, per intensità ed estensione, devono essere fronteggiati con mezzi e poteri straordinari

Fra i possibili esempi di emergenza possiamo citare terremoti, incendi, alluvioni, frane, tsunami, tornado, incidenti chimici, etc. . . .

Protezione Civile

Il Dipartimento della Protezione Civile è una struttura della Presidenza del Consiglio dei Ministri. Nasce nel 1982 per dotare il Paese di un organismo capace di mobilitare e coordinare tutte le risorse nazionali utili ad assicurare assistenza alla popolazione in caso di grave emergenza.

Con la legge n. 225 del 1992 il Dipartimento diventa il punto di raccordo del Servizio Nazionale della Protezione Civile, con compiti di indirizzo, promozione e coordinamento dell'intero sistema. Il Servizio nazionale , dal 2 gennaio 2018, è disciplinato dal Codice della Protezione civile (decreto legislativo. n.1 del 2 gennaio 2018) , con il quale è stata riformata tutta la normativa in materia, e ha come componenti tutti i livelli di governo: lo Stato, le Regioni, le Province autonome, e gli Enti locali.

Il Dipartimento, operando in stretto raccordo con le componenti, si occupa di tutte le attività volte alla previsione, alla prevenzione e alla mitigazione dei rischi, al soccorso e all'assistenza delle popolazioni colpite da calamità, al contrasto e al superamento dell'emergenza [24][25][26].

Piano comunale di Protezione Civile

Il Piano comunale di Protezione Civile è un piano di emergenza (un insieme di procedure operative di intervento per fronteggiare una qualsiasi calamità) redatto dai comuni per gestire adeguatamente un'emergenza ipotizzata nel proprio territorio, sulla base degli indirizzi regionali, come indicato dal DLgs. 112/1998. Tiene conto dei vari scenari di rischio considerati nei programmi di previsione e prevenzione stabiliti dai programmi e piani regionali [27][28].

In base al Decreto Legge n. 59/2012 ogni comune avrebbe dovuto approvare e adottare il proprio Piano di emergenza entro il 12 ottobre 2012 [29].

2.2.3 Aree di emergenza

Le **aree di emergenza** sono aree destinate, in caso di emergenza, ad uso di protezione civile. Esse devono essere preventivamente individuate nella pianificazione di emergenza e possono essere di tre tipi [30]:

- Aree di attesa della popolazione
- Aree di ammassamento soccorritori e risorse
- Aree di accoglienza o di ricovero della popolazione

Per definizione la geometria di un'area di emergenza è un poligono, ma è anche possibile conoscerne solo un nodo che la rappresenti (per esempio se il piano di emergenza rappresenta solo un punto interno all'area invece di tutta l'area).

Ogni area di emergenza sarà caratterizzata da:

- Nome
- Descrizione (opzionale)
- Gestore, cioè il comune che la gestisce (opzionale)
- Tipi di emergenza supportati (opzionale)
- **Fotografie** dell'area, possibilmente accompagnate da un titolo (opzionali)

Analizziamo i tipi di area di emergenza:

Aree di attesa della popolazione

Le **aree di attesa** sono i luoghi di prima accoglienza per la popolazione; possono essere utilizzate piazze, slarghi, parcheggi, spazi pubblici o privati non soggetti a rischio (frane, alluvioni, crollo di strutture attigue, etc.), raggiungibili attraverso un percorso sicuro. Il numero delle aree da scegliere è funzione della capacità ricettiva degli spazi disponibili e del numero degli abitanti. In tali aree la popolazione riceve le prime informazioni sull'evento e i primi generi di conforto. Le Aree di Attesa della popolazione saranno utilizzate per un periodo di tempo compreso tra poche ore e qualche giorno [31].

Un'area di attesa, in aggiunta alle caratteristiche tipiche di ogni area di emergenza, può avere una capienza massima e può essere al coperto (invece che all'aperto).

Aree di ammassamento soccorritori e risorse

Luoghi, in zone sicure rispetto alle diverse tipologie di rischio, dove dovranno trovare sistemazione idonea i soccorritori e le risorse necessarie a garantire un razionale intervento nelle zone di emergenza. Tali aree dovranno essere facilmente raggiungibili attraverso percorsi sicuri, anche con mezzi di grandi dimensioni, e ubicate nelle vicinanze di risorse idriche, elettriche ed con possibilità di smaltimento delle acque reflue. Il periodo di permanenza in emergenza di tali aree è compreso tra poche settimane e qualche mese [32].

Aree di accoglienza o di ricovero della popolazione

Sono luoghi, individuati in aree sicure rispetto alle diverse tipologie di rischio e poste nelle vicinanze di risorse idriche, elettriche e fognarie, in cui vengono installati i primi insediamenti abitativi per alloggiare la popolazione colpita. Dovranno essere facilmente raggiungibili anche da mezzi di grandi dimensioni per consentirne l'allestimento e la gestione. Rientrano nella definizione di aree di accoglienza o di ricovero anche le strutture ricettive (hotel, residence, camping, etc.) [33].

Mappatura delle aree di emergenza

Un elenco delle aree di emergenza deve essere inserito nel Piano comunale di Protezione Civile, generalmente rilasciato sotto forma di file PDF.

Il sito web della Protezione Civile nazionale fornisce un elenco dei comuni dotati del Piano di emergenza comunale, ma per visualizzare le aree di emergenza è necessario cercare e analizzare i singoli documenti [34].

Alcune amministrazioni e associazioni di Protezione Civile hanno realizzato progetti per la mappatura interattiva delle aree di attesa della popolazione, tra cui:

- Città di Palermo e Regione Sicilia (realizzate da Open Data Sicilia) [35][36].
- Calderara di Reno (realizzata dai Volontari Protezione Civile) [37].
- Comune di Morano Calabro [38].
- Provincia di Livorno (realizzato dalla Protezione Civile) [39].

Si tratta però di progetti pensati per un ambito locale, circoscritto a una singola città o regione. Attualmente non esiste nessun sistema di mappatura digitale ufficiale omogenea delle aree di emergenza.

2.2.4 Utilizzo offline

Le aree di emergenza scaricate possono essere rese disponibili per l'utilizzo offline salvandole nell'archiviazione del dispositivo.

Ogni volta che l'utente si sposta su una porzione di mappa l'applicazione deve provvedere a verificare se le aree di emergenza in quella porzione di mappa sono già state scaricate. Vista questa necessità le aree di emergenza dovranno essere salvate in **gruppi di aree** composti da un confine e da tutte le aree che ne fanno parte.

L'applicazione potrà quindi verificare se le aree in una porzione di mappa sono già state scaricate semplicemente verificando se essa è completamente inclusa nel confine di un gruppo di aree salvato in precedenza. In caso negativo è necessario scaricare le aree di emergenza, visualizzarle e renderle disponibili per l'utilizzo offline. In caso affermativo l'applicazione visualizzerà le aree contenute in quel gruppo.

Per evitare di mantenere inutilmente aree duplicate in memoria conviene verificare al momento del download per ciascuno dei gruppi scaricati in precedenza se esso è completamente compreso nel nuovo gruppo e in questo caso eliminare il vecchio gruppo.

Lo scaricamento delle aree di emergenza e la gestione dei gruppi scaricati può essere fatto in più modi:

- Soluzione completamente automatica: Se le aree di emergenza nella porzione di mappa non sono già state scaricate l'applicazione provvede automaticamente a scaricarle. Per scaricare offline le aree di emergenza di una certa porzione di mappa è quindi sufficiente per l'utente spostarsi su di essa.

  Per evitare di scaricare inutilmente gruppi di aree dalle dimensioni troppo grandi è possibile scaricare le aree solo oltre un certo livello di zoom o limitando l'area complessiva che può occupare un gruppo.

  - **Vantaggi:** Essendo trasparente non richiede alcuna azione all'utente, permettendo l'utilizzo anche alle persone più impacciate con la tecnologia (dettaglio notevole, vista l'importanza della conoscenza di queste aree per tutta la popolazione).
  - **Svantaggi:** Esplorando la mappa vengono scaricate molte aree di emergenza a cui l'utente non è interessato. Non essendo possibile gestire le aree scaricate questo causerebbe un consumo inutile di memoria.

- Soluzione completamente manuale: L'utente decide manualmente quali gruppi di aree scaricare, aggiornare o eliminare attraverso un'apposita interfaccia. Per scaricare le aree di emergenza di una certa porzione di mappa è necessario che l'utente acceda all'apposita interfaccia e scarichi l'area Se le aree di emergenza nella porzione di mappa non sono già state scaricate l'applicazione avvisa l'utente che prima di visualizzarle deve scaricare le aree di attesa.

  - **Vantaggi:** L'utente ha un controllo completo delle aree da scaricare e scaricate.
  - **Svantaggi:** L'applicazione diventa più complessa per l'utente, e la necessita di scaricare manualmente una porzione di mappa prima di poter usare quella porzione di mappa potrebbe scoraggiarlo. Aggiunge complessità all'applicazione, con ovvie ripercussioni su possibilità di bug e manutenibilità.

- Soluzione semi-automatica: Lo scaricamento avviene in automatico (come nella soluzione completamente automatica) ma l'utente ha a disposizione un'interfaccia per aggiornare o eliminare le aree di attesa scaricate. Per evitare di scaricare inutilmente gruppi di aree dalle dimensioni troppo grandi le aree andranno scaricate solo oltre un certo livello di zoom.

  - **Vantaggi:** Il download delle aree è trasparente, quindi permette l'utilizzo anche alle persone più impacciate con la tecnologia. In caso l'applicazione occupi troppa memoria l'utente è in grado di eliminare i gruppi a cui non è interessato.
  - **Svantaggi:** Aggiunge complessità all'applicazione.

La terza soluzione è un ottimo compromesso che combina i vantaggi delle precedenti e sembra la più appropriata.

Gestione aggiornamento gruppi

È possibile che con l'avanzare del tempo un comune modifichi le proprie aree di emergenza o che la sorgente dati perfezioni i dati a sua disposizione. Per questo motivo è opportuno aggiornare regolarmente i gruppi di aree scaricati. L'interfaccia di gestione conterrà quindi dei comandi per aggiornare un gruppo o tutti i gruppi.

È opportuno aggiornare i gruppi anche se l'utente non li aggiorna manualmente dall'interfaccia di gestione. Per permettere all'applicazione di controllare che le aree di emergenza da visualizzare siano state scaricate sufficientemente recentemente ogni gruppo di aree conterrà anche l'istante in cui è stato scaricato (*timestamp*).

Ogni qual volta l'utente si sposterà su una porzione di mappa già scaricata l'applicazione mostrerà le aree di emergenza e soltanto dopo controllerà se il gruppo al quale appartengono è troppo vecchio (meglio delle aree vecchie che nessuna area!). Nel caso questo risulti troppo vecchio sarà necessario scaricarlo di nuovo e mostrare le nuove aree.

Gestione preferiti

Come già accennato in caso l'applicazione occupi troppa memoria l'utente è in grado di eliminare i gruppi a cui non è interessato. Per semplificare questa operazione è possibile aggiungere all'interfaccia di gestione un comando per aggiungere un gruppo ai preferiti, uno per rimuovere un gruppo dai preferiti e uno per eliminare tutti i gruppi tranne i preferiti.

2.2.5 Sorgenti dati

Come già evidenziato in precedenza un elenco delle aree di emergenza deve essere inserito nel Piano comunale di Protezione Civile, generalmente rilasciato sotto forma di file PDF. Allo stato attuale non esiste un database digitale ufficiale delle aree di emergenza da cui attingere.

OpenStreetMap

Una possibile fonte da cui ottenere delle aree di attesa è OpenStreetMap, un database geografico libero e open-source [40][41].

Esso è realizzato dalla OpenStreetMap Foundation e da una comunità di volontari che contribuiscono mappando e mantenendo i dati su strade, sentieri, punti di interesse, stazioni ferroviarie e molto altro, in tutto il mondo [40][42].

Tutte le informazioni geografiche inserite dalla comunità sono disponibili per l'utilizzo con licenza Open Data Commons Open Database License (ODbL). Questa licenza permette di copiare, distribuire, trasmettere e adattare i dati a patto di attribuirli a OpenStreetMap e ai suoi contributori. Se invece si distribuisce pubblicamente una versione adattata o modificata del database OSM è necessario offrire anche quella Banca Dati adattata sotto la licenza ODbL [43][44].

Tutti gli oggetti inseriti nel database di OpenStreetMap appartengono a uno di queste tipologie [45]:

- **Nodo:** Punto geografico caratterizzato da latitudine e longitudine.
- **Percorso:** Lista ordinata di Nodi che compongono una linea. Se l'ultimo Nodo corrisponde al primo, l'elemento viene detto Percorso chiuso.
- **Relazione:** Lista ordinata di Nodi e Percorsi, ognuno con una etichetta che indica il suo ruolo.

Le coordinate geografiche dei nodi sono espresse nel sistema di riferimento WGS84, di cui spesso viene utilizzata la proiezione EPSG3857 nella visualizzazione[46][47].

Gli elementi possono essere caratterizzati da uno o più 'tag', ovvero coppie `Chiave=Valore` che forniscono informazioni su di esso [48]. Nei prossimi capitoli useremo la notazione `chiave=*` per indicare una chiave senza specificare un valore.

La Comunità si accorda su determinate combinazioni chiave-valore per i tag di uso più comune, che fungono da standard e che vengono definiti nell'OpenStreetMap Wiki [49].

È possibile rappresentare un'area applicando uno o più tag adeguati (come `area=*`,`landuse=*`, ...) ad un percorso chiuso o ad una relazione composta da percorsi che incolonnati compongono una linea chiusa. È inoltre possibile rappresentare aree con dei "buchi" attraverso una relazione con il tag `type=multipolygon` contenente l'area esterna con ruolo `outer` e l'area interna con ruolo `inner`.

Utilizzo di OpenStreetMap per le emergenze

OpenStreetMap è già usato per progetti legati alle emergenze, tra cui:

- Crowd-mapping (mappatura collaborativa) di edifici e strade in aiuto ai soccorritori in seguito a alluvioni, esondazioni, epidemie e terremoti [50][51].

  Per esempio:

  - HOTOSM (Humanitarian OpenStreetMap Team) è un'organizzazione che mette in atto progetti internazionali dedicati all'azione umanitaria e allo sviluppo delle comunità attraverso la mappatura. Quando si verifica un disastro migliaia di volontari si uniscono (online e sul sposto) per realizzare una mappatura libera che permetta ai soccorritori di raggiungere chi necessita del loro aiuto. La mappatura viene coordinata con un software apposito chiamato Tasking Manager, il quale suddivide l'area da mappare in molte aree ("*task*") e tiene traccia delle stato di ciascuna di esse (non mappato, mappatura in corso, mappato, validato) [52][53].
  - In occasione del Terremoto del 2016 in centro Italia la comunità italiana OSM si è mobilitata e 160 utenti da tutto il mondo hanno collaborato per realizzare ed aggiornare in tempo reale una mappatura delle aree terremotate [54].
  - V-IOLA (Volunteer International On Line Asset) è un progetto europeo che vede il coinvolgimento di diversi partner importanti come il Dipartimento di Protezione Civile DPC, Fondazione CIMA (centro di competenza del DPC), le Croci Rosse Italiana, Rumena, Montenegrina, l'Associazione Nazionale Pubbliche Assistenze e il Corpo italiano di Soccorso dell'Ordine di Malta, che insieme a Wikimedia Italia studieranno e sperimenteranno sul campo un percorso di coinvolgimento di volontari online per la mappatura di zone interessate da emergenze, focalizzandosi in particolare sul territorio balcanico. Obiettivo del progetto sarà coinvolgere il maggior numero possibile di volontari online a supporto di risorse attive sul campo, mostrando le potenzialità della mappatura da remoto attraverso strumenti tecnologie e applicazioni di semplice utilizzo per la raccolta collaborativa di dati geografici (ad esempio il Tasking Manager) e l'utilizzo di immagini satellitari [55][56].
  - Missing Maps: Organizzazione fondata da HOTOSM, Medici Senza Frontiere e varie Società Nazionali di Croce Rossa con l'obbiettivo di mappare tramite OpenStreetMap le zone più vulnerabili dei paesi in via di sviluppo in modo da permettere alle ONG locali di rispondere più prontamente alle crisi che colpiscono l'area [57].

- Prevenzione e valutazione del rischio di inondazioni attraverso la mappatura di idranti, scantinati, attività commerciali e altre caratteristiche rilevanti nell'eventualità di un'alluvione [58].

- Realizzazione e simulazione dei piani di evacuazione attraverso tool basati sui dati geografici OSM che permettano di eliminare punti di congestione [59].

Di particolare interesse per questo progetto è la possibilità di usare OpenStreetMap come database delle aree di attesa attraverso il tag `emergency=assembly_point`, che nello standard di OpenStreetMap indica un luogo sicuro designato dove le persone possono radunarsi o devono fare rapporto durante un'emergenza o un'esercitazione antincendio. Questo tag quindi può rappresentare le aree di attesa della popolazione [60].
Questo tag può essere affiancato da altre informazioni, come:

- tipi di emergenza per cui il punto è pensato [60]

  - Terremoti: `assembly_point:earthquake=yes/no`
  - Incendio: `assembly_point:fire=yes/no`

  - Alluvioni: `assembly_point:flood=yes/no`
  - Frane: `assembly_point:landslide=yes/no`
  - Tornado: `assembly_point:tornado=yes/no`
  - Tsunami: `assembly_point:tsunami=yes/no/<altezza in m>`
- nome (`name=*`)[61]
- operatore che ha istituito il punto (`operator=*`)[62]
- capienza (`capacity=*`)[63]

Overpass API

Esiste un'interfaccia che permette di eseguire query read-only sul database OpenStreetMap: si tratta di Overpass, una API REST che accetta chiamate HTTP GET aventi come parametro la query e che permette di ricevere risposte in formato XML, JSON o CSV [64][65].

Android supporta nativamente il parsing di XML e JSON, quindi per semplicità conviene scegliere uno di questi [66][67].

La query va scritta in uno dei due linguaggi appositi: Overpass XML QL e Overpass QL. Entrambi permettono la scrittura di richieste anche molto elaborate sugli elementi del database OSM [68][69][70].

È possibile ottenere in formato JSON tutte le aree di attesa in un certo bounding box con una query Overpass QL come quella sotto (A.1). Il risultato JSON ottenuto in risposta alla query è visibile nel codice A.2. La stessa query può essere fatta richiedendo i dati in formato XML (codice A.3), ottenendo in risposta il risultato visibile nel codice A.4. È inoltre possibile effettuare richieste usando come confine della ricerca un confine amministrativo con una query come quella nel codice A.5.

Esistono varie istanze di Overpass che offrono endpoint per le richieste. Ogni istanza mette in atto delle politiche di limitazione del carico di lavoro a cui ciascun utente la può sottoporre.

L'istanza principale (con endpoint `https://overpass-api.de/api/interpreter`) supporta globalmente circa 1'000'000 query al giorno, quindi la documentazione ufficiale suggerisce indicativamente di non effettuare più di 10'000 query al giorno e non scaricare più di 5 Gb in risposte al giorno [71].

2.2.6 Navigazione

Una volta scelta un'area di emergenza l'utente può chiedere di essere guidato verso di essa. La navigazione può essere inserita nell'applicazione in due modi:

- Integrazione all'interno dell'applicazione di un sistema di navigazione completo con cui guidare l'utente.
  - **Realizzabilità:** Realizzare un sistema di navigazione richiederebbe una sorgente dati per il grafo stradale, un sistema di routing che trovi il miglior percorso dalla nostra posizione a quella del punto di emergenza attraverso il grafo stradale e un sistema di navigazione che guidi l'utente sul percorso, con relativa interfaccia utente.
  - **Vantaggi:** rende l'applicazione indipendente da altre applicazioni.
  - **Svantaggi:** Progetta da zero un task estremamente complesso come la navigazione, il quale è già stato implementato da altre applicazioni specializzate che ne hanno perfezionato il funzionamento e l'interfaccia.
- Ricerca di un'applicazione sul dispositivo in grado di svolgere la navigazione verso il punto desiderato e delegare la navigazione a quest'ultima.
  - **Realizzabilità:** In Android è possibile delegare la navigazione verso un certo punto geografico sfruttando il sistema degli Intent (oggetti che descrivono azioni da intraprendere da un'altra applicazione) e in particolare l'Intent `android.intent.action.VIEW`. Esso richiama un'applicazione in grado di supportare un URI che gli è stato passato al momento della creazione. La navigazione verso una certa destinazione può essere chiesta attraverso l'URI `google.navigation` [72][73][74].

I dettagli della navigazione desiderata vengono specificati con parametri separati da `&`. La destinazione deve essere specificata con il parametro `q=latitudine,longitudine` (indicazioni verso il punto con le coordinate indicate) o `q=nome+destinazione` (indicazioni verso la destinazione con il nome indicato).

Il metodo di viaggio può essere indicato con il parametro `mode=d` (in macchina), `mode=w` (a piedi) o `mode=c` (in bici). Le strade da evitare possono essere indicate con il parametro `avoid=t` (evita le strade a pedaggio), `avoid=h` (evita le autostrade) e/o `avoid=f` evita i traghetti.

Un codice di esempio della delega della navigazione può essere trovato in A.6.

Attualmente l'URI `google.navigation` è supportato da numerose app quali Google Maps, Waze, Here WeGo, TomTom Go e OsmAnd.

- **Vantaggi:** sfrutta le applicazioni che hanno già perfezionato la navigazione, riducendo anche la probabilità di bug e migliorando la manutenibilità.
- **Svantaggi:** rende l'applicazione dipendente dalla presenza di applicazioni in grado di svolgere la navigazione; per essere guidato offline l'utente deve aver scaricato la mappa anche nell'applicazione che userà per la navigazione.

Nella maggioranza dei dispositivi Android è installata una applicazione che supporti la delegazione della navigazione (per esempio fino a Ottobre 2018 Google Maps era preinstallato in qualsiasi dispositivo con servizi Google [75]). Vista l'abbondanza di queste applicazioni e considerati i vantaggi della delegazione a esse, la seconda soluzione sembra la più appropriata.

2.2.7 Aspetti legali

Attribuzione dei dati

Come già accennato nella sezione 2.2.5 OpenStreetMap distribuisce i propri dati sotto licenza Open Data Commons Open Database License (ODbL). Questa licenza permette di copiare, distribuire, trasmettere e adattare i dati a patto di attribuirli a OpenStreetMap e ai suoi contributori [43][44].

In base alle linee guida di OpenStreetMap questo significa che sarà necessario informare l'utente della sorgente dei dati mostrando la stringa `©OpenStreetMap contributors` con un link a *openstreetmap.org* [43].

Gli SDK accennati nella sezione 2.2.1 forniscono strumenti per semplificare l'inserimento dell'attribuzione all'interno della mappa. In particolare Mapbox Maps SDK, sfruttando mappe di sfondo basate su OpenStreetMap, provvede all'attribuzione da default [76][77].

Trattamento dei dati personali

In Europa il trattamento dei dati personali è disciplinato dal Regolamento UE 2016/679 del 27 Aprile 2016, comunemente noto come GDPR (General Data Protection Regulation). Dal 25 Maggio 2018 il GDPR è diventato esecutivo in tutti i paesi membri a prescindere dalla presenza o meno di un decreto di recepimento (essendo regolamento europeo e quindi fonte primaria non ne ha bisogno) il quale, in Italia, è stato emanato solamente ad agosto 2018. [78].

In Italia il trattamento dei dati personali è disciplinato dal d.lgs. 196 del 30 Giugno 2003, comunemente noto come Codice della Privacy, il quale ha subito successive modificazioni con il d.lgs. 101 del 10 Agosto 2018 di recepimento e adeguamento al regolamento europeo 679 (il GDPR).

Un dato personale è qualsiasi informazione riguardante una persona fisica identificata o identificabile («interessato»). Si considera identificabile la persona fisica che può essere identificata, direttamente o indirettamente, con particolare riferimento a un identificativo come il nome, un numero di identificazione, dati relativi all'ubicazione, un identificativo online o a uno o più elementi caratteristici della sua identità fisica, fisiologica, genetica, psichica, economica, culturale o sociale (art. 4, punto 1).

Un dato personale può fornire informazioni sulle sue caratteristiche, le sue abitudini, il suo stile di vita, le sue relazioni personali, il suo stato di salute, la sua situazione economica, ecc... [79].

I dati personali includono:

- dati che permettono l'identificazione diretta (dati anagrafici, immagini, ...)

- dati che permettono l'identificazione indiretta (codice fiscale, indirizzo IP, numero di targa, ...)
- dati cosiddetti "sensibili" (etnia, convinzioni religiose o filosofiche, opinioni politiche, appartenenza sindacale, stato di salute, dati genetici, dati biometrici, orientamento sessuale, ...). Per essi si applicano regole particolarmente stringenti.(art. 9)
- dati cosiddetti "giudiziari", che possono rivelare l'esistenza di provvedimenti giudiziari soggetti ad iscrizione nel casellario giudiziario.(art. 10)

Per trattamento si intende qualsiasi operazione o insieme di operazioni, compiute con o senza l'ausilio di processi automatizzati e applicate a dati personali o insiemi di dati personali, come la raccolta, la registrazione, l'organizzazione, la strutturazione, la conservazione, l'adattamento o la modifica, l'estrazione, la consultazione, l'uso, la comunicazione mediante trasmissione, diffusione o qualsiasi altra forma di messa a disposizione, il raffronto o l'interconnessione, la limitazione, la cancellazione o la distruzione. (art. 4, punto 2)

Il Regolamento impone che i soggetti che procedono al trattamento dei dati personali altrui adottino particolari misure per garantire il corretto e sicuro utilizzo dei dati. In particolare i dati devono essere raccolti per finalità determinate, esplicite e legittime. Essi devono inoltre essere limitati nella quantità (deve essere trattato solo quanto necessario rispetto alle finalità) e nel tempo (devono essere conservati in forma che consenta l'identificazione degli interessati per un arco di tempo non superiore al conseguimento delle finalità per le quali sono trattati)(art. 5).

Il trattamento dei dati è lecito solo se l'interessato ha espresso il consenso al trattamento dei propri dati personali per una o più specifiche finalità oppure il trattamento è necessario all'esecuzione di un contratto di cui l'interessato è parte oppure è necessario per adempiere un obbligo legale al quale è soggetto il titolare del trattamento.(art.6)

Se l'applicazione trattasse dati personali sarebbe quindi necessario richiedere l'accettazione da parte dell'utente di una informativa conforme al GDPR, in cui specificare le finalità e l'ambito del trattamento.

Download aree di attesa Le comunicazioni via rete fatte dall'applicazione sono caratterizzate dall'indirizzo IP del dispositivo. Di conseguenza se l'applicazione inviasse dei dati che permettessero di individuare la posizione dell'utente questi sarebbe indirettamente riconducibili alla persona e diventerebbero un dato personale regolato dal GDPR.

L'unico dato in uscita dall'applicazione via rete è il confine di cui scaricare le aree di emergenza. Se esso fosse sempre centrato sulla localizzazione dell'utente allora sarebbe sufficiente per individuare la posizione della persona.

Questo però non è il caso: il confine può essere centrato sulla posizione dell'utente ma può anche essere una porzione di mappa arbitraria che l'utente ha visualizzato. I due casi sono irriconoscibili dalla sorgente dati (o da un eventuale man-in-the-middle) e quindi nessun dato personale viene inviato via rete.

Archiviazione delle aree di attesa L'applicazione salva sul dispositivo i gruppi scaricati. Per questi valgono le stesse osservazioni precedenti: non è possibile riconoscere i confini centrati su una posizione dell'utente da quelli scaricati esplorando la mappa, quindi nessun dato personale viene salvato nell'archiviazione locale.

In conclusione, non essendo salvato nessun dato personale nell'archiviazione locale e non essendo inviato nessun dato personale durante il download, la gestione delle aree di attesa non ricade nell'ambito del GDPR.

Mapbox Maps SDK L'SDK per la visualizzazione delle mappe di Mapbox, descritto in 2.2.1, contiene un modulo in grado di raccogliere dati anonimi sull'utilizzo della mappa come:[80]

- Utilizzo iniziale dell'applicazione che usa la mappa
- Caricamento della mappa
- Spostamento della mappa (pan)
- Click sulla mappa (tap)

- Posizione, se si utilizza la geo-localizzazione

Nonostante questi dati vengano inviati in forma anonima rientrano comunque nei dati personali, in quanto sono identificati dall'IP e da attributi che vengono inviati assieme alla telemetria [80].
Questi parametri includono:

- Identificatore della sessione
- Identificatore dell'applicazione che usa la mappa
- Modello del dispositivo
- Tipo di connessione internet
- Timestamp degli eventi

Mapbox utilizza i dati ottenuti per le seguenti finalità:[81]

- Diagnostica interna
- Miglioramento dei prodotti e servizi di mappatura
- Fornire servizi agli utenti finali
- Generare statistiche dell'utilizzo anonimizzate e aggregate

La telemetria può essere abilitata o disabilitata dallo sviluppatore che utilizza il SDK con la funzione `TelemetryDefinition::setUserTelemetryRequestState(boolean enabled)` (come nel codice in A.7) [82].
A default la telemetria è abilitata, quindi se lo sviluppatore non la disabilita completamente sarà suo compito implementare le misure necessarie per rispettare le leggi sul trattamento dei dati personali:[83]

- Disabilitare la telemetria se l'utente ha meno di 13 anni
- Ottenere il consenso dell'utente prima di utilizzare la geo-localizzazione (la notifica automatica di Android è sufficiente)
- Informare l'utente dei dati che verranno trattati da Mapbox tramite Mapbox Telemetry e delle finalità del trattamento e chiedere il permesso esplicito, disabilitandola in caso di mancato consenso
- Non cercare di identificare individui, veicoli o entità intercettando i dati della telemetria

2.3 Glossario

area di attesa Area di emergenza destinata alla prima accoglienza, in cui la popolazione riceve le prime informazioni sull'evento e i primi generi di conforto. Può avere una capienza e può essere al coperto. 8

area di emergenza Area destinata, in caso di emergenza, ad uso di protezione civile. Caratterizzata dalla geometria e dal nome; può avere una descrizione, un gestore e/o delle fotografie. 7–10, 12

bounding box Confine costituito da una latitudine minima e massima e da una longitudine minima e massima. 6

confine Limite di un territorio, di un terreno, di una regione geografica o di uno stato. 6, 9

emergenza Fenomeno di origine naturale o antropica in grado di arrecare danno alla popolazione, alle attività, alle strutture e infrastrutture del territorio. Caratterizzata dal nome. 7, 8

fotografia Immagine che ritrae un'area di emergenza. Può avere un titolo. 7

geometria Rappresentazione di un oggetto sulla mappa. 5, 7

gruppo di aree Un confine, tutte le aree di emergenza che ne fanno parte e il momento in cui sono state scaricate. 9, 10, 18

latitudine Distanza angolare di un punto dall'equatore. 5, 6

longitudine Distanza angolare di un punto dal meridiano di Greenwich. 5, 6

mappa Rappresentazione semplificata dello spazio.. 5, 9

nodo Punto geografico, identificato da latitudine e longitudine. 5, 7

poligono Area geografica delimitata da una linea spezzata chiusa (i vertici sono quindi una lista ordinata di almeno tre nodi). 5, 7

2.4 Specifica dei requisiti

2.4.1 Casi d'uso

Figura 2.1: Diagramma dei casi d'uso

Visualizza mappa

Visualizzazione della mappa contenente le aree di attesa.

- **Scenario principale:** L'applicazione visualizza una mappa in cui sono evidenziate le aree di emergenza. Se le aree di emergenza presenti nell'area visualizzata non sono già state scaricate l'applicazione provvede a scaricarle. Se le aree sono già state scaricate ma sono troppo vecchie l'applicazione le visualizza e poi prova a scaricarle nuove. Se l'utente non ha installata una app per la navigazione viene avvisato che non sarà in grado di avviare la navigazione; altrimenti si avverte l'utente di rendere disponibile offline anche la navigazione se si ha intenzione di usarla.
- **Scenario alternativo:** Se il download delle aree di attesa fallisce l'utente viene avvisato.

Sposta mappa

Cambiamento della porzione di mappa visualizzata.

- **Scenario principale:** La porzione di mappa visualizzata cambia, mostrando la posizione e lo zoom scelti dall'utente. Se le aree di emergenza presenti nell'area visualizzata non sono già state scaricate l'applicazione provvede a scaricarle. Se le aree sono già state scaricate ma sono troppo vecchie l'applicazione le visualizza e poi prova a scaricarle nuove.
- **Scenario alternativo:** se il download delle aree di attesa fallisce l'utente viene avvisato.

Mostra posizione attuale

Uguale a *Sposta mappa*, ma la posizione mostrata è quella attuale dell'utente.

- **Scenario principale:** La porzione di mappa visualizzata cambia, mostrando la posizione attuale dell'utente. Se le aree di emergenza presenti nell'area visualizzata non sono già state scaricate l'applicazione provvede a scaricarle.
- **Scenari alternativi:**
  - se la posizione attuale dell'utente non è disponibile nessuna azione viene intrapresa
  - se il download delle aree di attesa fallisce l'utente viene avvisato

Visualizza dettagli area

Visualizzazione dei dettagli dell'area di attesa scelta dall'utente.

- **Scenario principale:** Vengono mostrati tutti i dettagli disponibili dell'area di attesa scelta dall'utente. Questi includono nome, descrizione se disponibile, coordinate geografiche, foto se disponibile.

Naviga verso area

Navigazione verso l'area di attesa.

- **Scenario principale:** Viene avviata una app di navigazione che guida l'utente all'area scelta.
- **Scenari alternativi:** Non è disponibile un'app per la navigazione, l'utente viene avvisato.

Visualizza gruppi scaricati

Nasconde le aree di emergenza e al loro posto mostra i confini dei gruppi di aree già scaricati, con la possibilità di selezionarne uno alla volta.

Aggiorna gruppo

Ri-scarica il gruppo selezionato.

- **Precondizioni:** Un gruppo deve essere selezionato.
- **Scenario principale:** Viene scaricato un gruppo con lo stesso confine di quello selezionato e quello vecchio viene rimosso dall'archiviazione.
- **Scenari alternativi:** Se lo scaricamento del nuovo gruppo fallisce quello vecchio non viene rimosso.

Aggiorna tutti i gruppi

Come *Aggiorna gruppo*, ma su tutti i gruppi scaricati.

- **Scenario principale:** Per ogni gruppo viene scaricato un gruppo con lo stesso confine e quello vecchio viene rimosso dall'archiviazione.
- **Scenari alternativi:** Se lo scaricamento di un nuovo gruppo fallisce il corrispondente vecchio gruppo non viene rimosso e l'utente viene avvisato.

Elimina gruppo

Rimuove il gruppo selezionato dall'archiviazione.

- **Precondizioni:** Un gruppo deve essere selezionato.
- **Scenario principale:** Il gruppo selezionato viene rimosso dall'archiviazione.

Elimina tutti i gruppi

Come *Elimina gruppo*, ma su tutti i gruppi scaricati.

- **Scenario principale:** Tutti i gruppi vengono rimossi dall'archiviazione.

Aggiungi gruppo ai preferiti

Aggiunge il gruppo selezionato ai preferiti.

- **Precondizioni:** Un gruppo deve essere selezionato.
- **Scenario principale:** Il gruppo selezionato viene aggiunto ai preferiti.

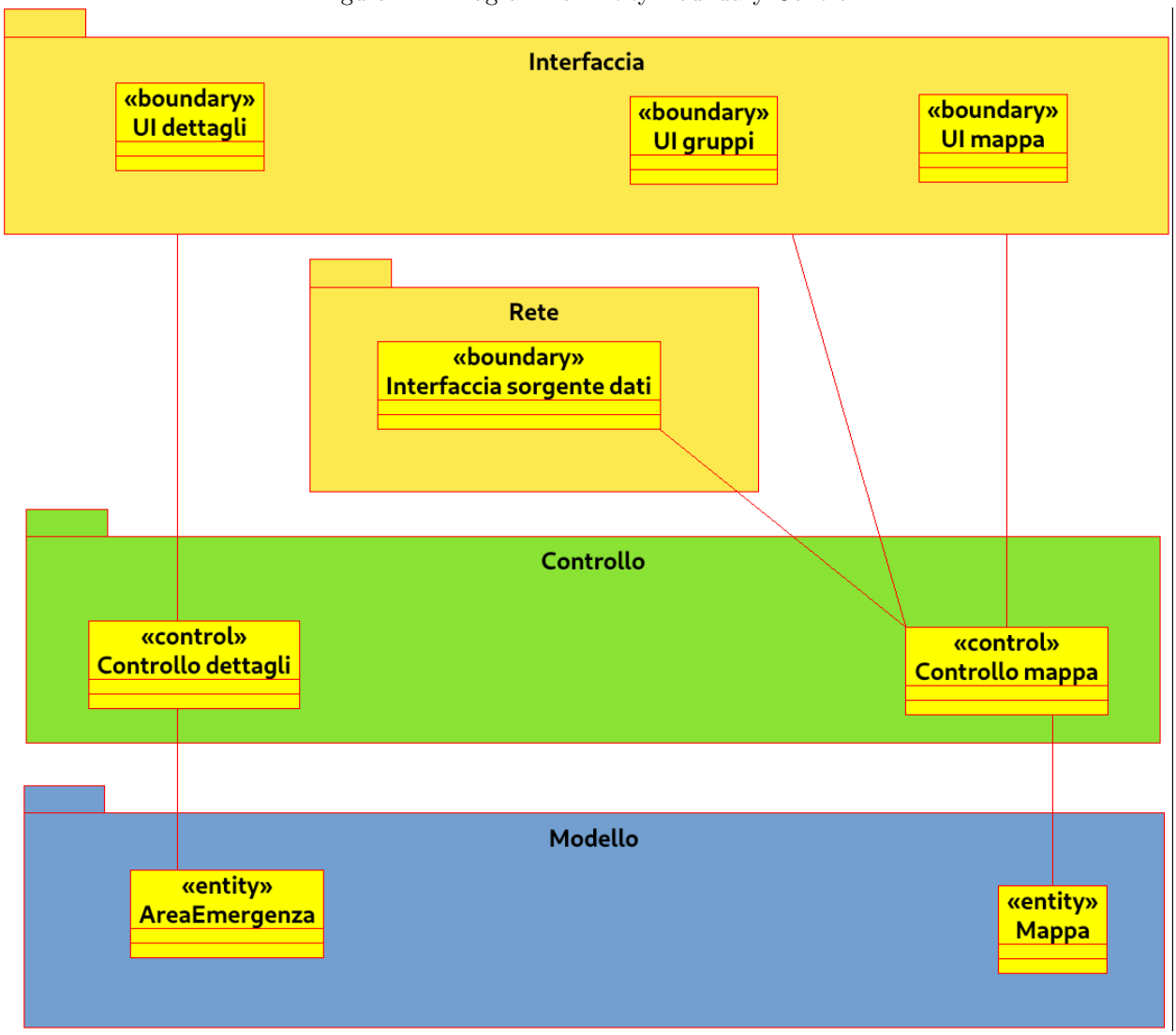
Rimuovi gruppo dai preferiti

Rimuove il gruppo selezionato dai preferiti.

- **Precondizioni:** Un gruppo deve essere selezionato.
- **Scenario principale:** Il gruppo selezionato viene rimosso dai preferiti.

2.4.2 Architettura logica

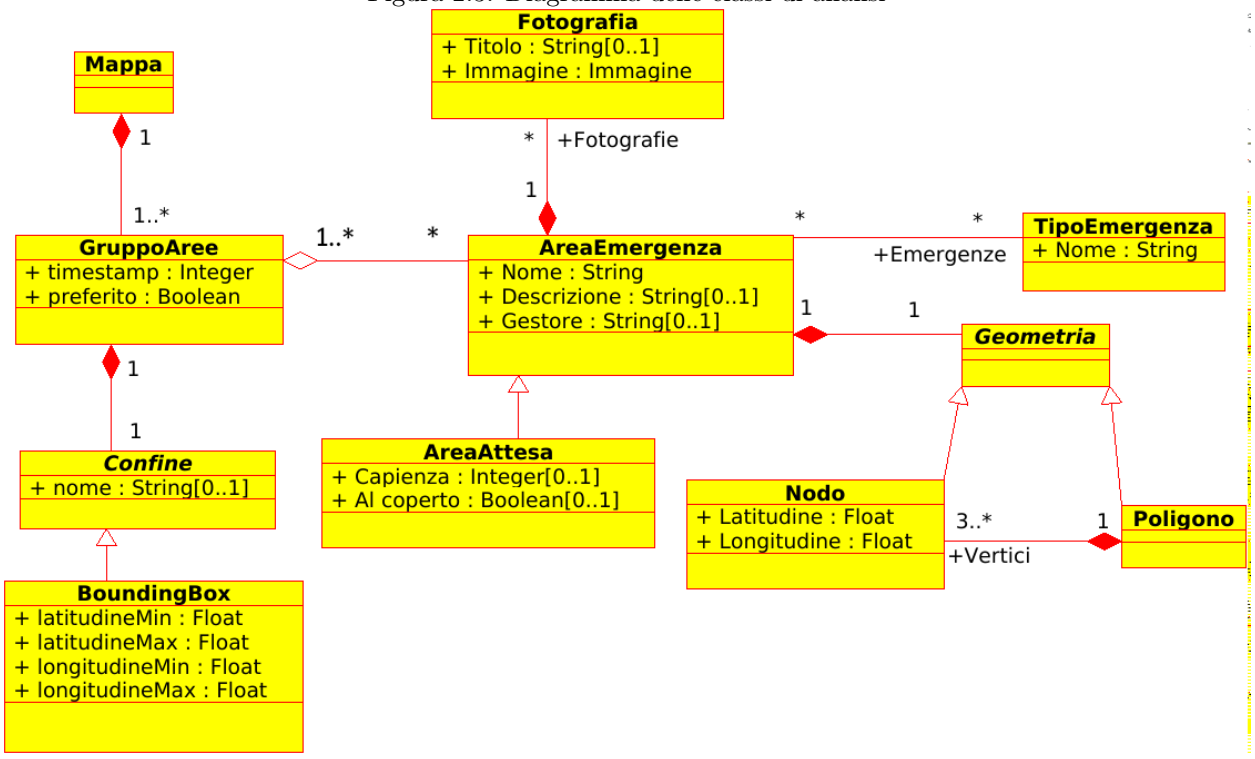
Figura 2.2: Diagramma Entity-Boundary-Control



Il contenuto del package di Modello deriverà dal modello dei dati (figura 2.3).

2.4.3 Modello dei dati

Figura 2.3: Diagramma delle classi di analisi



2.4.4 Modello dinamico

Figura 2.4: Diagramma degli stati dell'utente

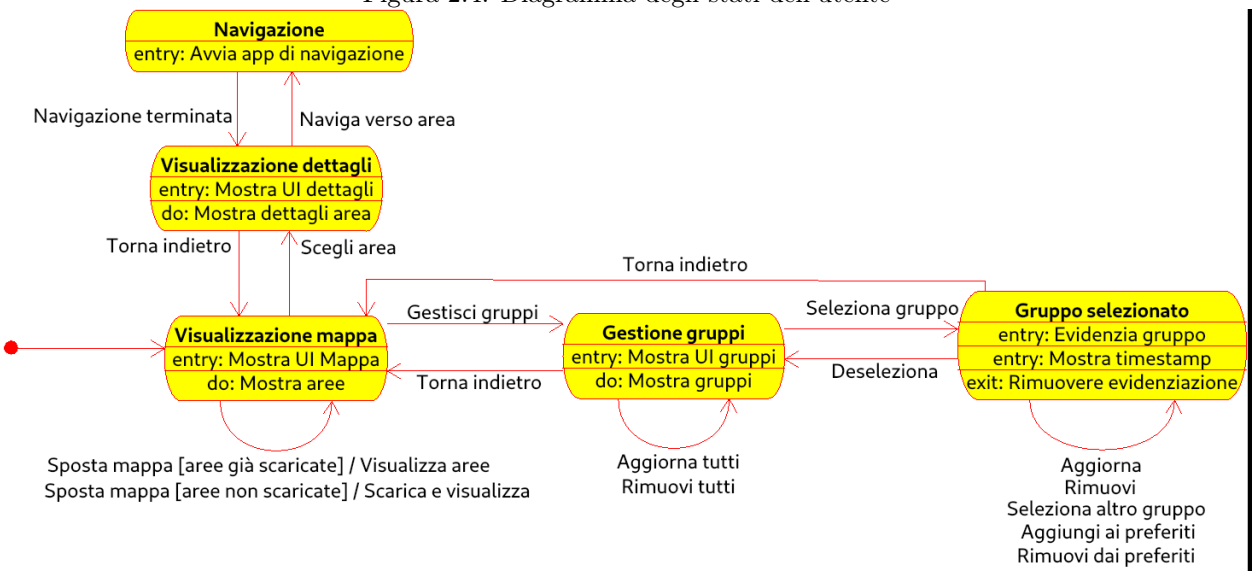


Figura 2.5: Diagramma degli stati del gruppo di aree

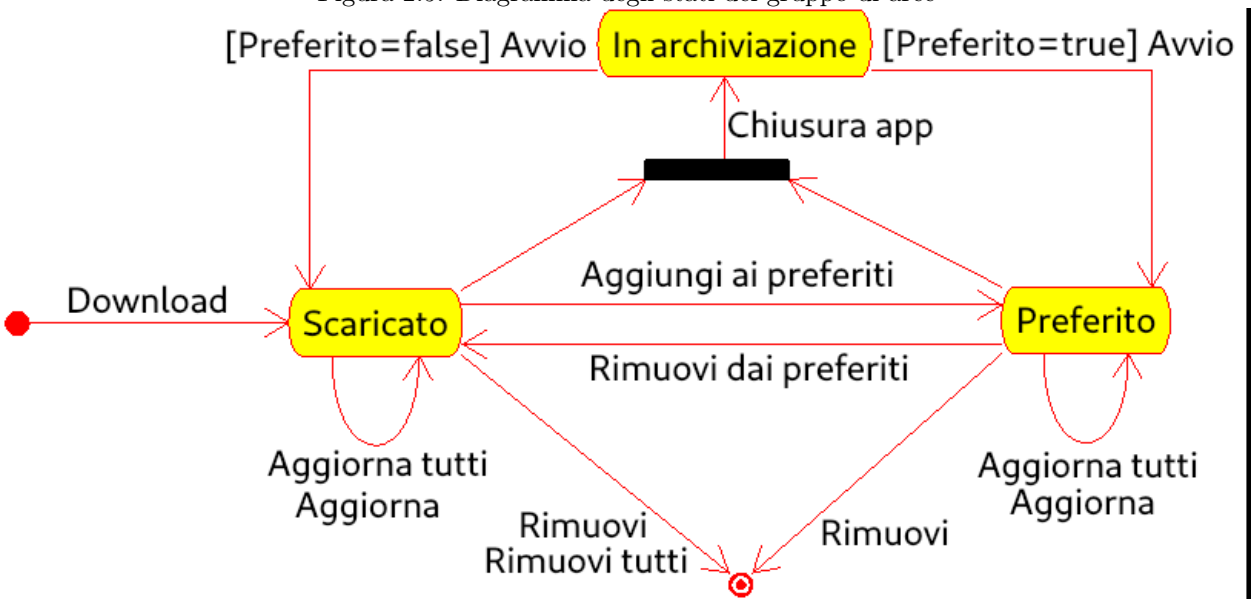


Figura 2.6: Inizializzazione (caricamento aree salvate)

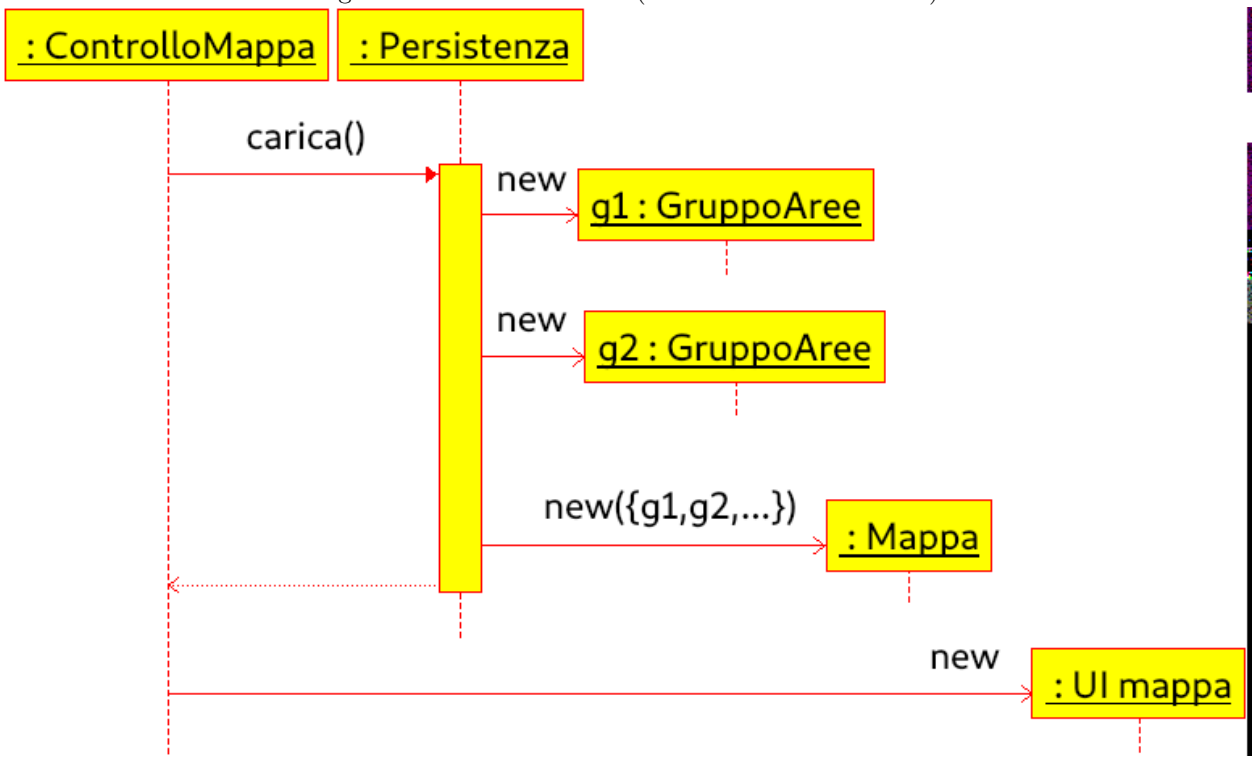


Figura 2.7: Spostamento mappa nel caso in cui le aree della nuova porzione sono già scaricate

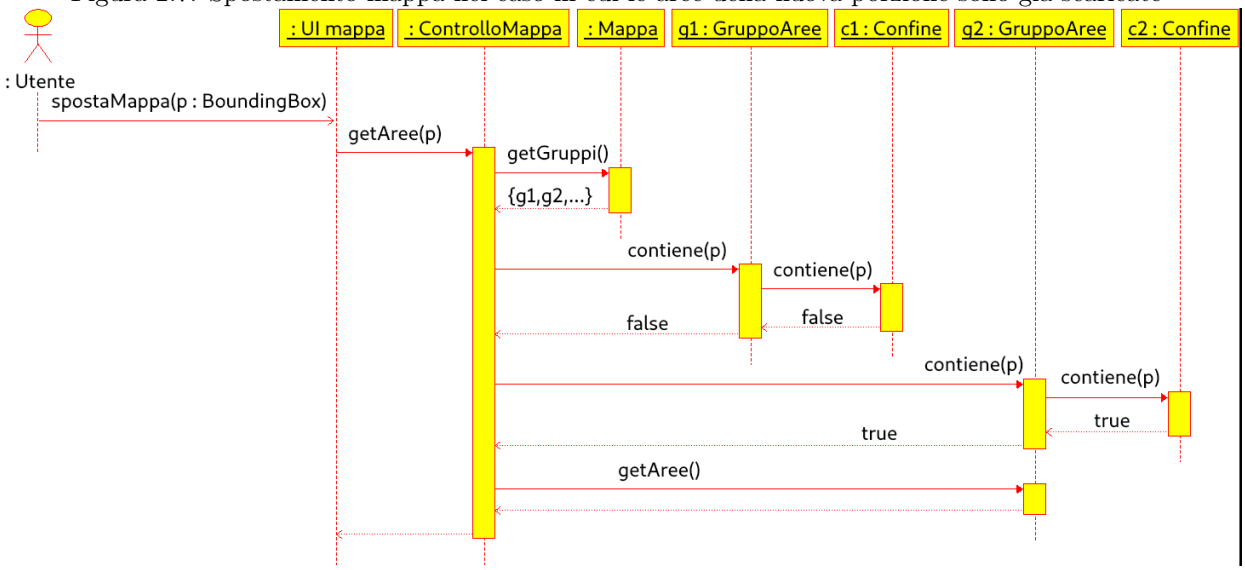


Figura 2.8: Spostamento mappa nel caso in cui le aree della nuova porzione non siano ancora scaricate

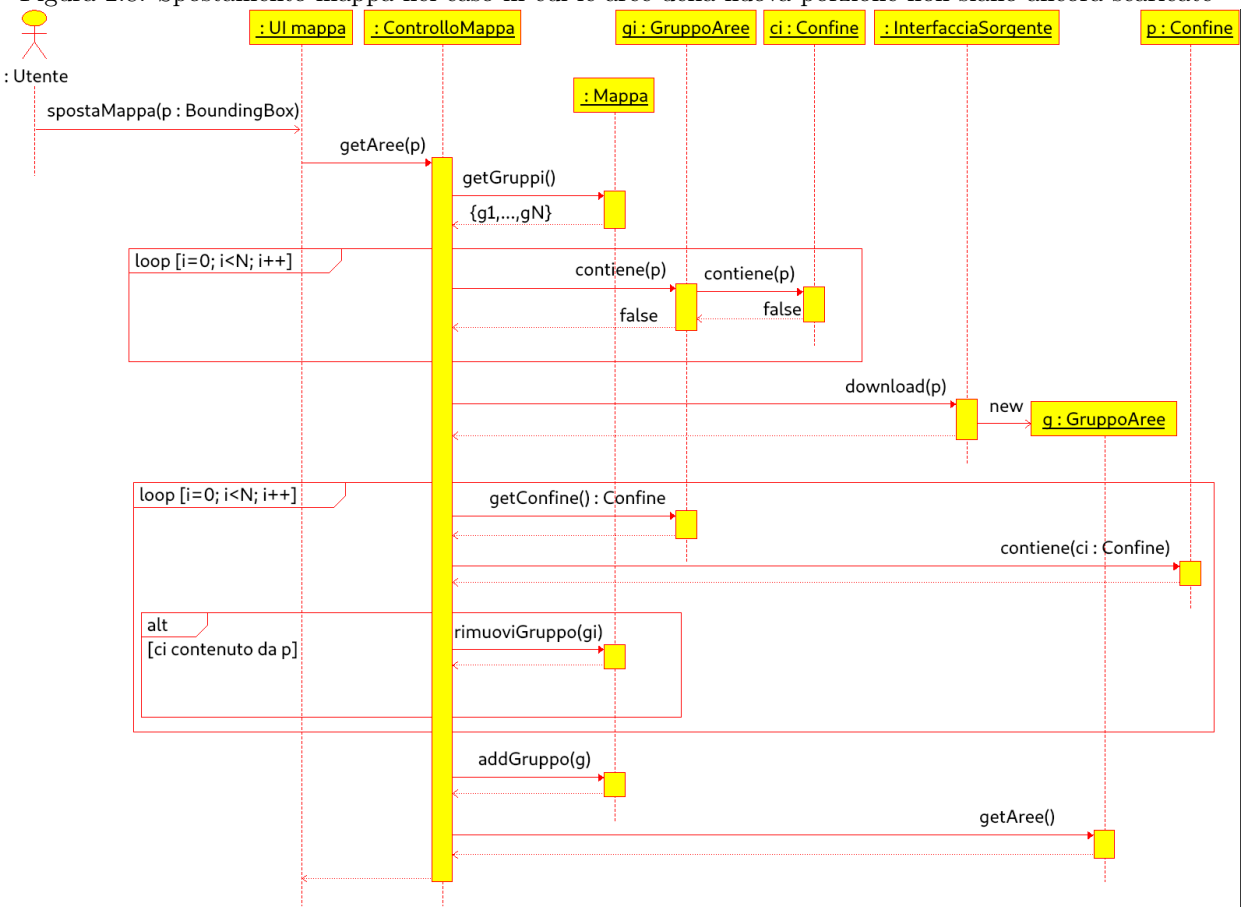


Figura 2.9: Aggiornamento di un gruppo di aree

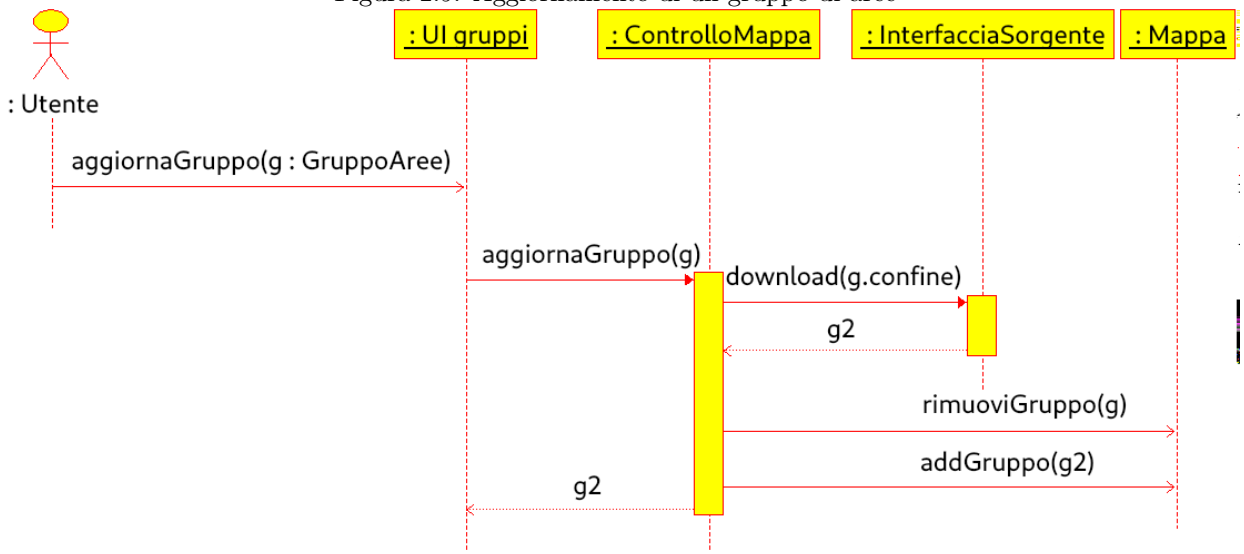


Figura 2.10: Rimozione di un gruppo di aree

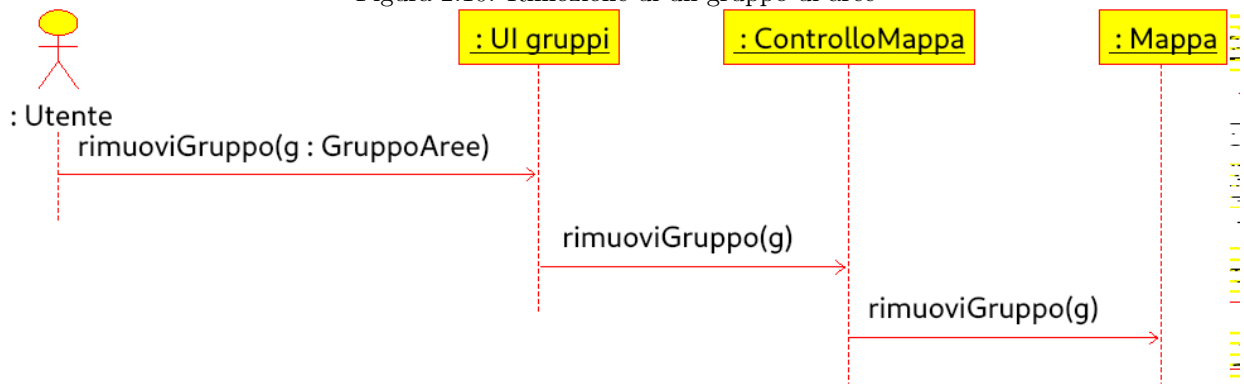
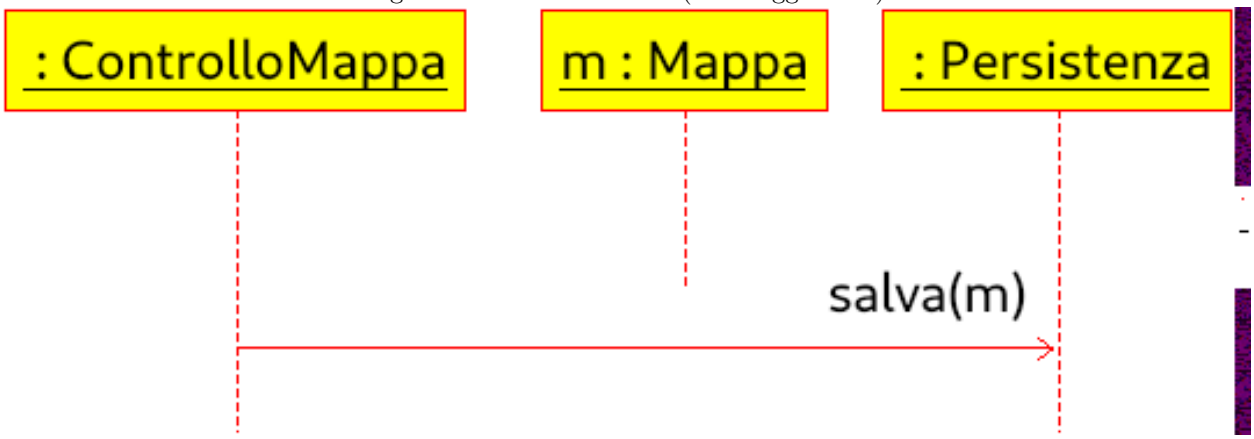


Figura 2.11: Terminazione (salvataggio aree)



Capitolo 3

Progettazione

Definizione dei moduli che comporranno l'applicazione e di come interagiranno.

Il processo decisionale per la progettazione dell'applicazione terrà necessariamente conto del requisito che pone come obbiettivo la piattaforma Android, analizzando l'architettura più adatta alle librerie fornite. Nonostante questo bisognerà cercare di mantenere il progetto più indipendente possibile dalla piattaforma, in modo da facilitare una eventuale estensione futura a più piattaforme.

3.1 Background

3.1.1 Pattern Model-View-ViewModel

Il pattern architetturale Model-View-ViewModel oltre a disaccoppiare le funzionalità di Logica di business, controllo e visualizzazione si pone l'obbiettivo di mantenere l'UI il più semplice possibile per renderla più facilmente gestibile [84].

MVVM suddivide i componenti in View e Model in modo simile a MVP e MVC, ma lo strato che li collega cambia:

- **Model:** Contiene i dati del dominio e la logica di business.

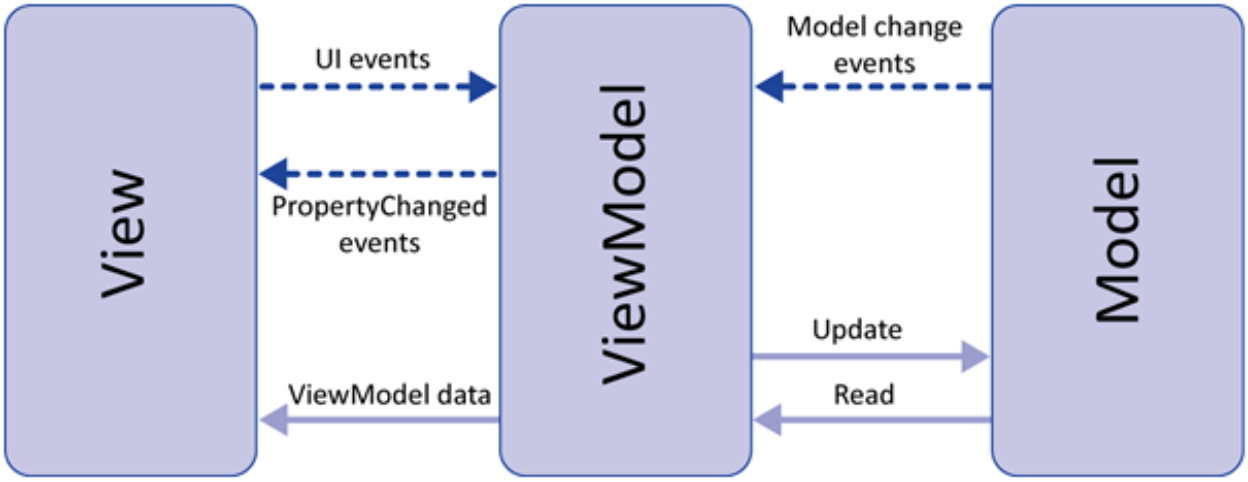
  Deve essere possibile per altre classi ricevere notifiche in caso di modifica del modello, per esempio tramite il pattern Observer.

- **ViewModel:** Interagisce con il modello e prepara i dati per essere osservabili dalla View. Espone degli handler per gli eventi che la View gli invierà.

  Il ViewModel deve essere disaccoppiato dalla View, quindi non dovrebbe mai avere un riferimento diretto ad essa.

- **View:** Osserva i dati (si iscrive alle notifiche delle modifiche) e modifica l'UI di conseguenza. Se necessario notifica il ViewModel delle interazioni dell'utente tramite gli handler che esso espone.

Figura 3.1: Pattern architetturale Model-View-ViewModel (MVVM) [84].



3.1.2 Pattern architetturali in Android

Android offre all'interno del toolkit Jetpack una collezione di librerie (chiamata *Android Architecture Components*) per creare applicazioni robuste, testabili e manutenibili organizzando strutturalmente l'applicazione [85][86].

In particolare il package `android.arch.lifecycle` fornisce un insieme di componenti che permettono di gestire il ciclo di vita dell'applicazione modificando il proprio comportamento o eseguendo azioni in risposta al cambio di stato di altri componenti come attività e frammenti ("*lifecycle-aware*").

Questo permette di evitare la classica organizzazione del codice Android all'interno delle Activity in metodi del ciclo di vita (`onCreate()`, `onStart()`, `onPause()`, `onResume()`, ...), organizzando invece il codice con un pattern strutturale e garantendo la separazione degli interessi [87][88].

Android Architecture Components

Per gestire il ciclo di vita vengono messe a disposizione delle classi che permettono di monitorare i cambi stato tramite pattern comportamentale Observer:

- `LifecycleObserver`: interfaccia che espone i callback per i vari eventi del ciclo di vita tramite l'annotazione `OnLifecycleEvent` [89][90].
- `Lifecycle`: classe che definisce un oggetto con un ciclo di vita [91].
- `LifecycleOwner`: classe che possiede un `Lifecycle`, ottenibile con `getLifecycle()` [92].

Per permettere di mantenere la visualizzazione dei dati al passo dei dati vengono messe da disposizione delle classi lifecycle-aware che monitorano le variazioni dei dati tramite il pattern Observer, permettendo ad altre classi di eseguire azioni in risposta al cambiamento del valore [93].

- `Observer<T>`: interfaccia che espone il callback `onChange(T)` [94].
- `LiveData<T>`: classe astratta data holder dotata di getter che al variare del dato notifica gli Observer registrati se questi sono attivi [95].
- `MutableLiveData<T>`: sottoclasse di `LiveData<T>` che espone i setter `setValue(T)` (sincrono) e `postValue(T)` (asincrono) [96].
- `MediatorLiveData<T>`: sottoclasse di `LiveData<T>` che implementa il pattern Mediator svolgendo il ruolo di Mediator, mentre `LiveData<T>` svolge il ruolo di Colleague. `MediatorLiveData` osserva quindi altri `LiveData` aggiunti tramite `addSource()` e reagisce ai cambiamenti del loro valore. Si occupa anche di propagare le variazioni del Lifecycle [97][98][99].

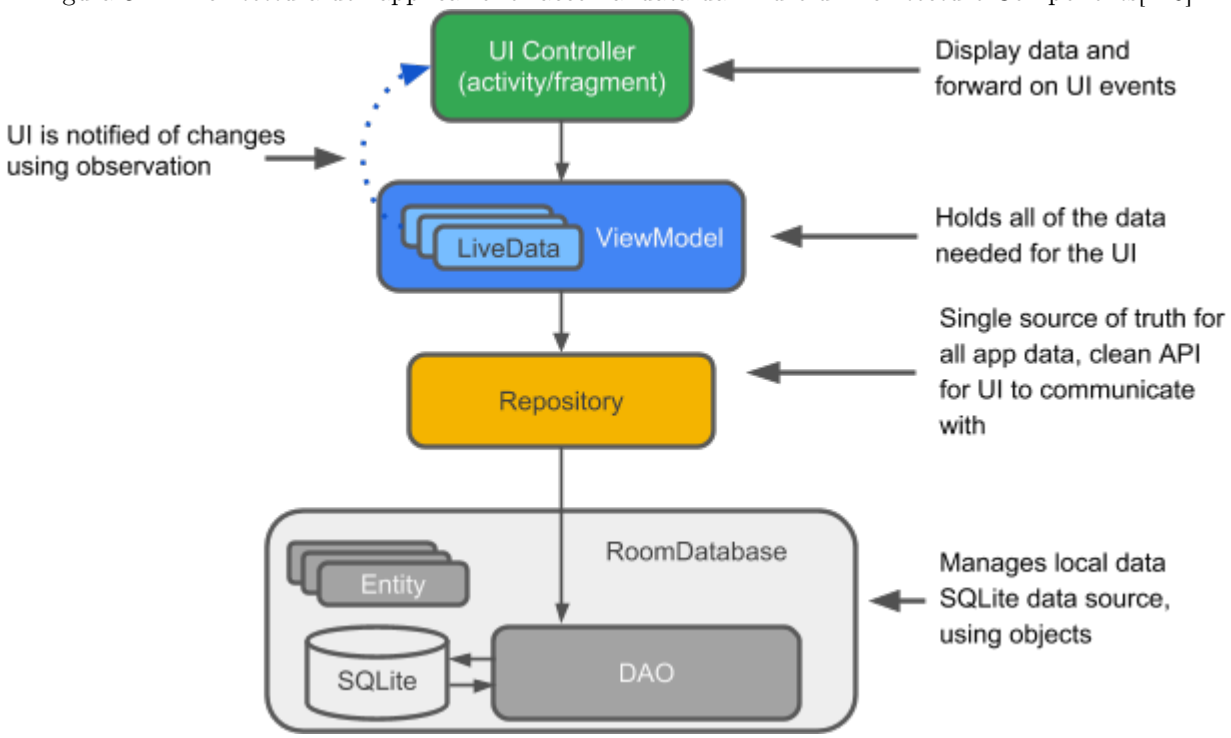
Android Architecture Components adotta il pattern architetturale Model-View-ViewModel (MVVM) e fornisce una serie di strumenti per realizzarlo:

- `ViewModel`: classe lifecycle-aware pensata per facilitare la realizzazione di un ViewModel che sopravviva al ciclo di vita della UI [100][101].
- `ViewModelProviders`: classe che permette di ottenere i ViewModel (come nel codice A.9) [102].
- Data Binding Library: facilita la connessione di elementi della UI (View, realizzata da una Activity o un Fragment) ai dati (Model osservabile come LiveData) [103][104][105].
- `AndroidViewModel`: Sottoclasse di `ViewModel` pensata per quei ViewModel che necessitano del Context. Normalmente questo sarebbe impossibile perchè è sconsigliato mantenere un riferimento alle View, ma questa sottoclasse è lifecycle-aware e si occupa di rimuovere il riferimento quando necessario [106].

Per utilizzare questa libreria è necessario inserire nel file `build.gradle` dell'applicazione le sue dipendenze come nel codice A.8 [107].

Un ulteriore package, `android.arch.persistence.room`, fornisce un sistema di persistenza che permette di caricare e salvare oggetti dalla archiviazione del dispositivo attraverso il pattern DAO sfruttando un database SQLite [108][109].

Figura 3.2: Architettura dell'applicazione raccomandata da Android Architecture Components[110]



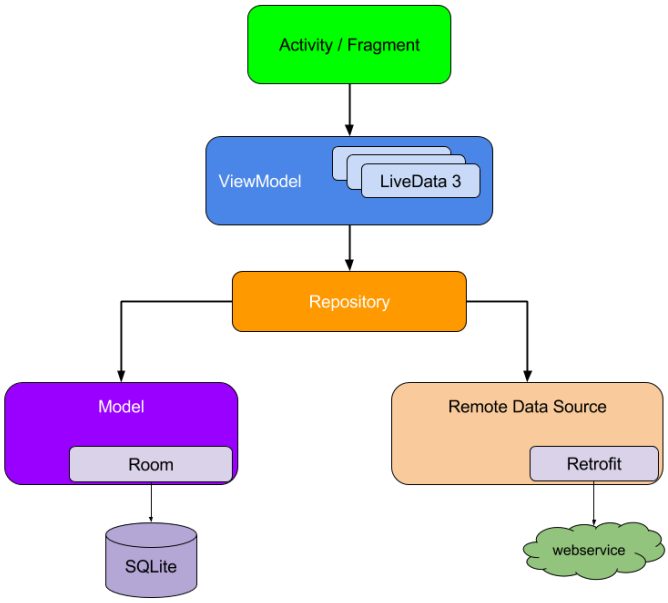
Pattern e anti-pattern

Gli sviluppatori di Android Architecture Components suggeriscono alcune regole da seguire e alcuni anti-pattern da evitare [111][110][112][113]:

- Le classi di ViewModel e Model non dovrebbero utilizzare classi da pacchetti `android.*`, con l'eccezione di `android.arch.*`.

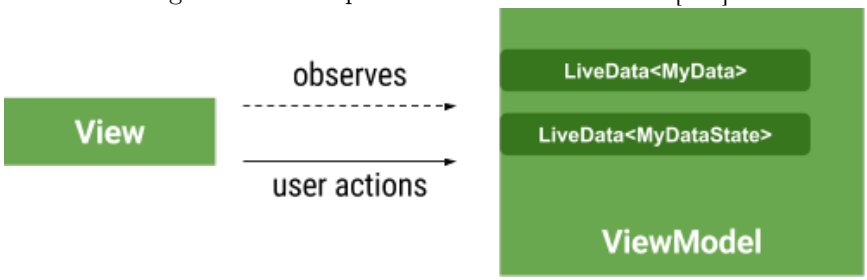
- La view è difficile da testare automaticamente (servono tool complessi come *Roboelectric*) quindi la logica nelle Activity e nei Fragment (che comporranno la View) deve essere ridotta al minimo indispensabile. Questo obbiettivo coincide con la scelta di MVVM di lasciare alla View solo il compito di visualizzare i dati del Model forniti dal ViewModel.
- Un ViewModel non dovrebbe avere riferimenti alla View
- Ricordarsi di mantenere la separazione degli interessi e il principio di singola responsabilità nel ViewModel. Se un ViewModel mantiene troppe responsabilità conviene intervenire per ridistribuirle:
  - muovendo parte della logica in un presenter con lo stesso scope del ViewModel, comunicando con altre parti dell'applicazione e aggiornando i `LiveData<>` del ViewModel
  - introducendo uno strato "Domain layer" e adottando la "Clean Architecture" [114]
- Se l'applicazione ha più fonti dati (archiviazione, rete, cache, ...) introdurre uno strato "Data layer" contenente una repository che funzioni da unico punto di accesso ai dati.

Figura 3.3: Esempio di architettura con repository [110].



- Se i dati sono caratterizzati da uno stato (dati non ancora caricati/dati caricati/errore di rete/...) conviene esporre lo stato dal ViewModel come osservabile.

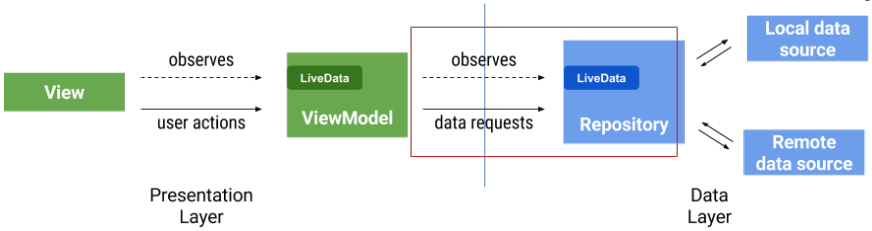
Figura 3.4: Esempio di ViewModel con stato [111].



- Le Activity e i Fragment vengono distrutti per qualsiasi cambio di configurazione, per esempio in caso di rotazione del display. I ViewModel possono essere distrutti in caso il sistema sia a corto di memoria e quindi chiuda l'applicazione. Di conseguenza se la UI ha uno stato che va ripristinato in seguito a un cambio di configurazione o a uno stop è necessario che questo stato venga mantenuto nel ViewModel e salvato nell'archiviazione [115].

- Quando l'app viene chiusa le View vengono rimosse e i ViewModel, non più osservati, idealmente dovrebbero essere distrutti dal garbage collector. Tuttavia se la Repository mantiene un riferimento (per esempio di callback per le notifiche di modifica) a un ViewModel questo non verrà distrutto fino alla eventuale terminazione del processo, causando quindi un memory leak. Per evitare questo è possibile esporre i dati che necessitano di essere osservati della Repository attraverso `LiveData<>`, che essendo lifecycle-aware è in grado di riconoscere lo stato in cui si trova l'Observer e eliminare i riferimenti se questo è nello stato terminale.

Figura 3.5: Utilizzo di `LiveData<>` nella Repository per evitare memory leak [111].



- Non bisogna mettere nel ViewModel codice critico per salvare i dati in uno stato coerente perché ogni chiamata del ViewModel può essere l'ultima.

3.1.3 Librerie Android rilevanti

Room

La libreria Room fornisce un sistema di persistenza che permette di caricare e salvare oggetti dalla archiviazione del dispositivo attraverso il pattern DAO sfruttando un database SQLite [108][116].

Per essere utilizzate richiede l'implementazione di tre componenti [117][118]:

- Classi del modello annotate con `@Entity` per rappresentare le tabelle del database (come nel codice A.11 e A.12) [119]. Le chiavi primarie si indicano tramite l'annotazione `@PrimaryKey` [119]. È possibile inserire le relazioni fra entità tramite le annotazioni `@ForeignKey` e `@TypeConverter` [120][121]. Quando le entità presentano un polimorfismo si possono attuare attuare diverse strategie apposite, come creare una tabella relazionale per ogni classe concreta oppure un'unica tabella per tutte le sottoclassi di una classe [122][123].

- Un'interfaccia annotata con `Dao` che contiene i metodi necessari per accedere ai dati (come nel codice A.13) [109][124].

- Una sottoclasse astratta di `RoomDatabase` annotata con `@Database(version=...,entities=...)` (che elenca le entità associate) e che presenta un metodo astratto senza argomenti che restituisce la classe annotata con `@Dao` (come nel codice A.14) [125][126].

Sarà quindi possibile accedere ai dati instanziando il Database tramite `Room.databaseBuilder` (come nel codice A.15).

Room supporta anche funzionalità avanzate come Transazioni (tramite `@Transaction`), ereditarietà del Dao e traduzione automatica delle relazioni 1-N (tramite `@Relation`) [127].

È possibile effettuare Unit Test sul Database attraverso `Room::inMemoryDatabaseBuilder()` [128].

Per utilizzare questa libreria è necessario inserire nel file `build.gradle` dell'applicazione le sue dipendenze come nel codice A.10[118].

Retrofit

La libreria Android più diffusa per la fruizione di API REST è Retrofit [129][130][131]. Per utilizzare questa libreria è necessario:

- Definire i Plain Old Java Object in cui inserire i dati scaricati (come nel codice A.17). È possibile generare automaticamente queste classi attraverso il servizo web `jsonschema2pojo.org` [132].
- Dichiarare un'interfaccia attraverso cui Retrofit può interfacciarsi all'API (come nel codice A.18).
- Generare un client Retrofit con cui ottenere i dati dall'interfaccia precedentemente definita (come nel codice A.19).
- Utilizzare il client per ottenere effettivamente i dati (come nel codice A.20).

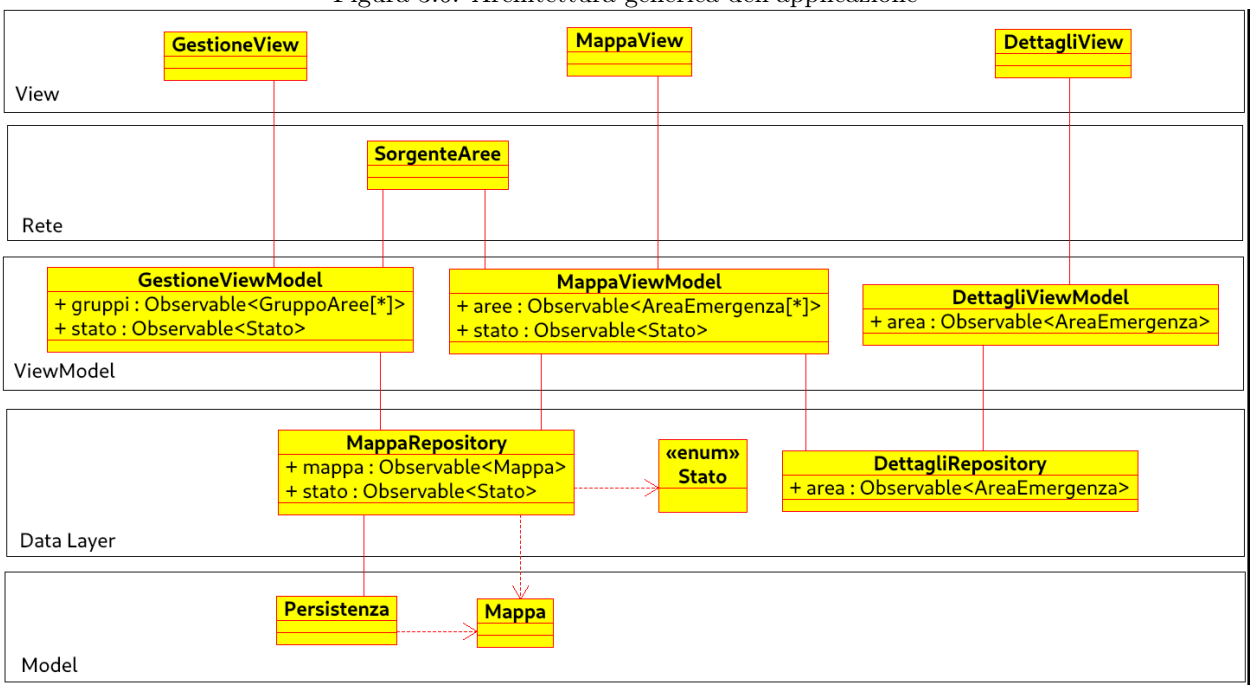
Per utilizzare questa libreria è necessario inserire nel file `build.gradle` dell'applicazione la sua dipendenza come nel codice A.16 [129]. Per creare i POJO partendo dalla risposta JSON sarà necessaria una libreria per la de-serializzazione. Un esempio è GSON, che richiede a sua volta l'aggiunta di una dipendenza, come nel codice A.16 [133].

3.2 Architettura

Visto il supporto estensivo fornita dal sistema Android utilizzeremo come pattern strutturale Model-View-ViewModel (MVVM).

Per prima cosa definiamo una architettura generica indipendente da Android, implementabile in qualsiasi linguaggio ad oggetti, frutto dell'evoluzione dell'architettura logica in MVVM tenendo in considerazione le linee guida già citate.

Figura 3.6: Architettura generica dell'applicazione



Definiamo ora i componenti dell'architettura, prima generici e poi specifici per Android. Questi ultimi saranno quelli utilizzati nell'implementazione dell'applicazione.

3.2.1 Model

Modello dei dati, derivato dal modello dei dati studiato in analisi (2.3).

Per garantire il principio di **inversione delle dipendenze** per ogni classe presente nel modello dei dati di analisi viene creata un'opportuna interfaccia che espone i metodi pubblici (getter e funzioni della logica di business) e di conseguenza una relativa classe di progetto che implementa tale interfaccia. Per semplificare la gestione e minimizzare la probabilità di errori le classi sono progettate come immutabili e pertanto non presentano setter.

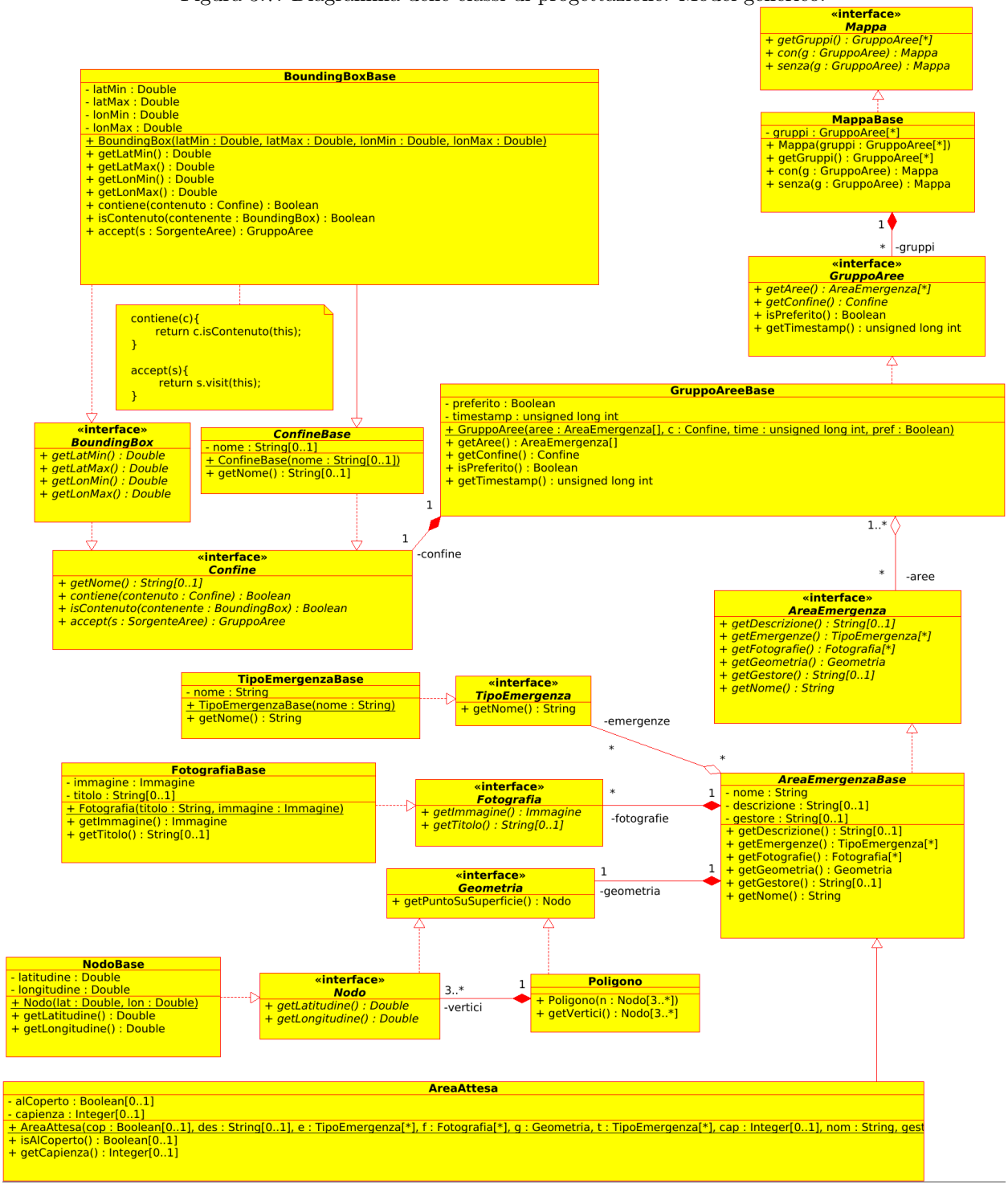
L'implementazione di `Confine::contiene(contenuto:Confine)` potrebbe essere contenuta interamente al suo interno. In questo caso però l'inserimento di una nuovo tipo di confine comporterebbe la modifica della funzione `contiene()` dei tipi di `Confine` pre-esistenti per aggiungere la gestione dei nuovi tipi.

Per evitare questo si può applicare il pattern Visitor in modo che Confine svolga sial il ruolo di Element che di Visitor. In particolare questo viene realizzato attraverso `contiene(contenuto:Confine)` che corrisponde alla `accept()` e `isContenuto(contenente:BoundingBox)` che corrisponde alla `visit()`.

L'inserimento di un nuovo tipo di Confine richiederà quindi l'aggiunta ai tipi pre-esistenti di una nuova funzione `isContenuto(contenente:...)` per il nuovo tipo. In compenso non sarà necessaria la modifica delle funzioni di comparazione pre-esistenti.

Per ridurre al minimo l'imprecisione delle coordinate viene scelto il double come tipo di dato in virgola mobile per latitudine e longitudine.

Figura 3.7: Diagramma delle classi di progettazione: Model generico.



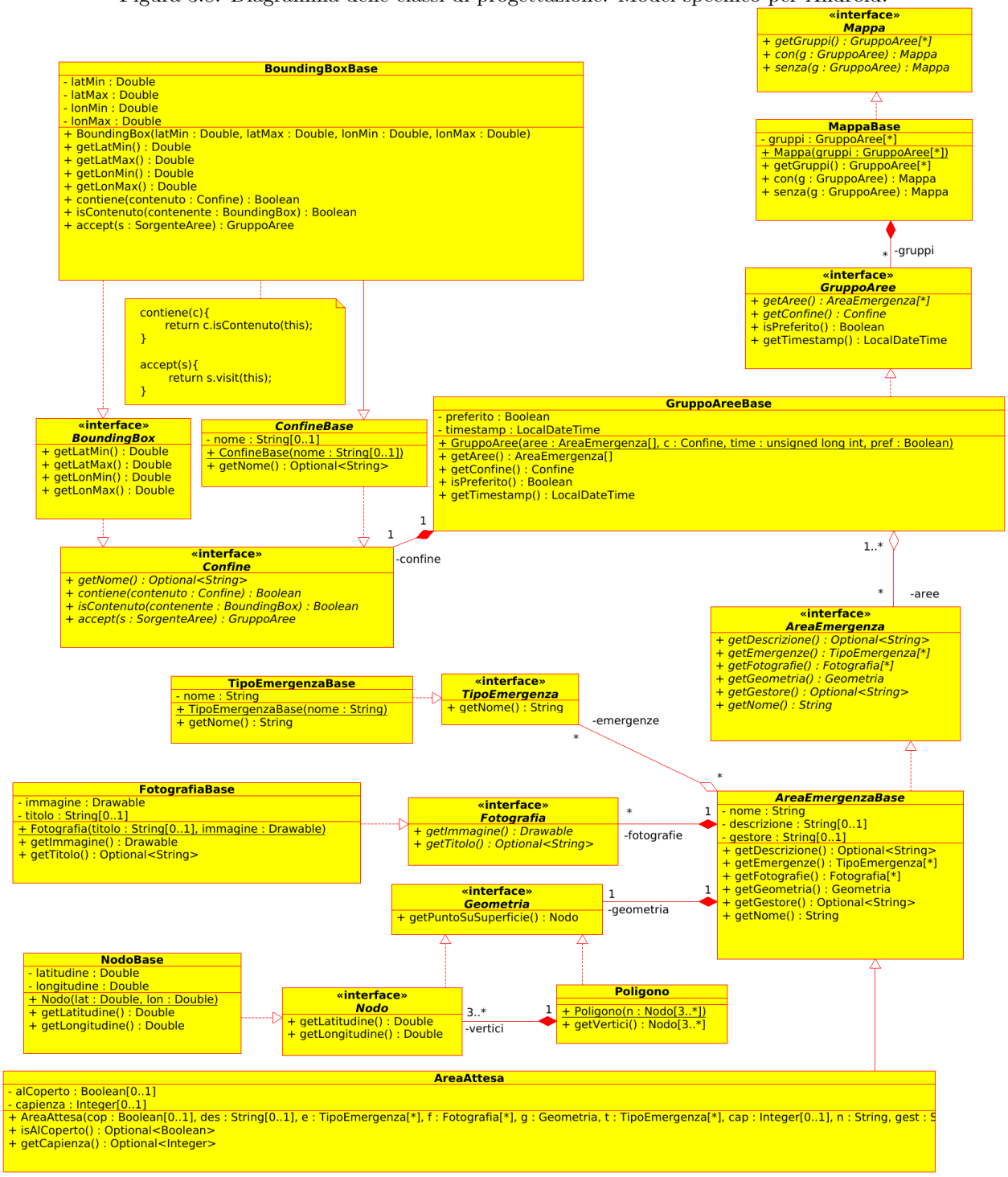
Nell'implementazione Android il tipo di dato *Immagine* (restituito da `Fotografia::getImmagine()`) sarà sostituito dalla classe `android.graphics.drawable.Drawable` [134].

Inoltre il tipo di dato `unsigned long int` restituito da `GruppoAree::getTimestamp()` verrà sostituito da `LocalDateTime` [135].

Le proprietà opzionali degli elementi (`String[0..1]`, `Integer[0..1]`, ...) vengono restituite sotto forma

di `Optional<T>`.

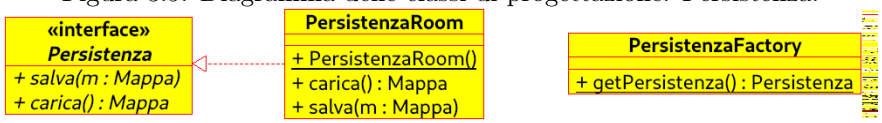
Figura 3.8: Diagramma delle classi di progettazione: Model specifico per Android.



3.2.2 Persistenza

Come definito in analisi, le classi di persistenza permetteranno di caricare e salvare i gruppi di aree di emergenza. A questo scopo verrà sfruttata la libreria Room, analizzata in 3.1.3.

Figura 3.9: Diagramma delle classi di progettazione: Persistenza.

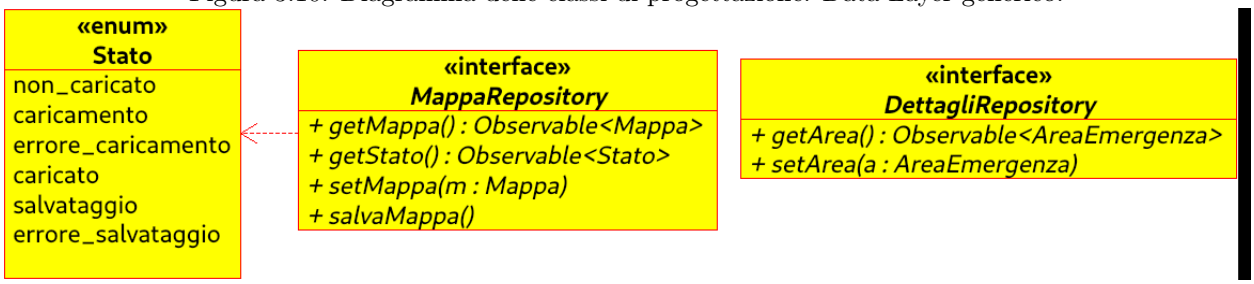


Agli elementi del Model verranno quindi aggiunte le annotazioni appropriate per guidare il Dao di Room. Alcune entità hanno già valori utilizzabili come Primary Key (per esempio nel caso di Nodo la combinazione latitudine+longitudine), nelle altre verrà inserita una chiave surrogata (con nome "id") e i relativi getter.

3.2.3 Data layer

Lo strato Data layer conterrà le istanze osservabili del modello, sulle quali opereranno i diversi ViewModel.

Figura 3.10: Diagramma delle classi di progettazione: Data Layer generico.

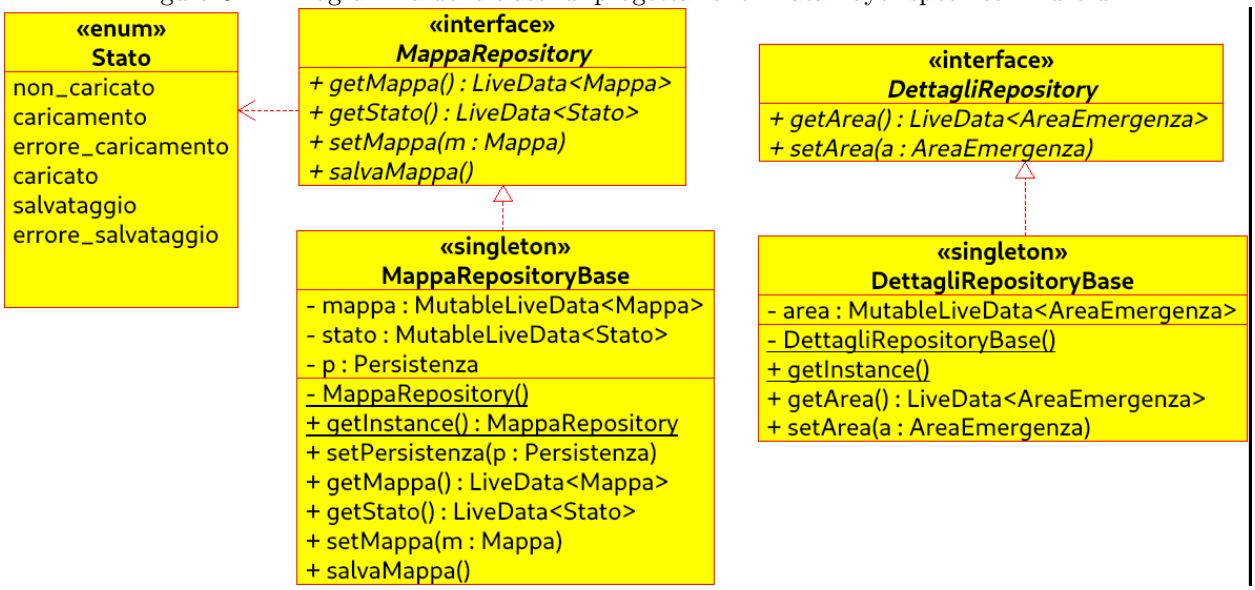


Sarà necessario per più ViewModel ottenere un riferimento alla stessa Repository, per esempio MappaRepository dovrà essere condivisa dai ViewModel che gestiscono GestioneView e MappaView.

Come accennato in 3.1.2 i ViewModel vengono creati attraverso `ViewModelProviders` (come nel codice A.9) e non vi si possono passare argomenti.

Per passare a più ViewModel la stessa repository è quindi necessario trasformare le Repository in Singleton.

Figura 3.11: Diagramma delle classi di progettazione: Data Layer specifico Android.

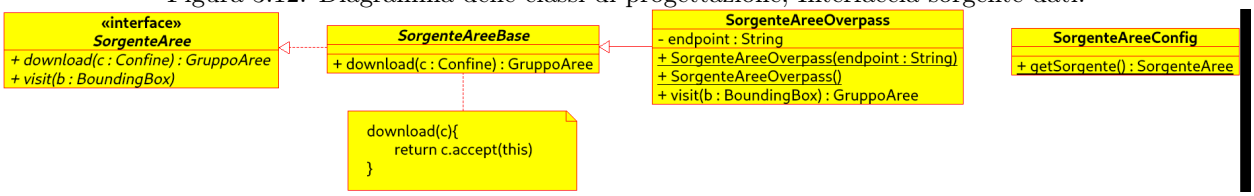


3.2.4 Rete

Come osservato in 2.2.5 l'unica sorgente dati digitale unificata contenente le aree di attesa è OpenStreetMap, sul quale ricade quindi la scelta come sorgente dati. Per ottenere i dati da essa sfrutto l'API Overpass. Come formato di output viene scelto JSON, il quale può essere facilmente analizzato tramite Retrofit con l'adattatore Gson, analizzati in 3.1.3.

Nella progettazione dell'interfaccia sorgente dati applico il pattern Visitor (con SorgenteAree che fa da visitor e client mentre Confine fa da element) per rendere sostenibile l'aggiunta di nuovi tipi di Confine.

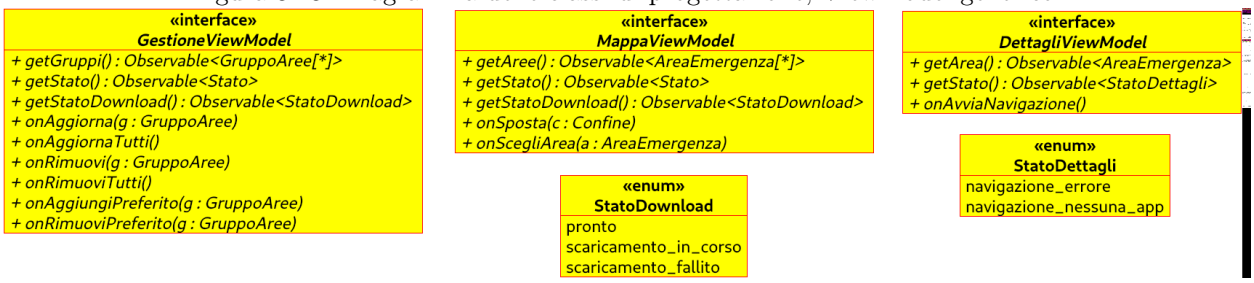
Figura 3.12: Diagramma delle classi di progettazione, Interfaccia sorgente dati.



3.2.5 ViewModel

Le classi ViewModel forniranno alle View le funzioni getter per ottenere i dati osservabili (pattern Observer) e le funzioni da chiamare in seguito agli eventi della UI.

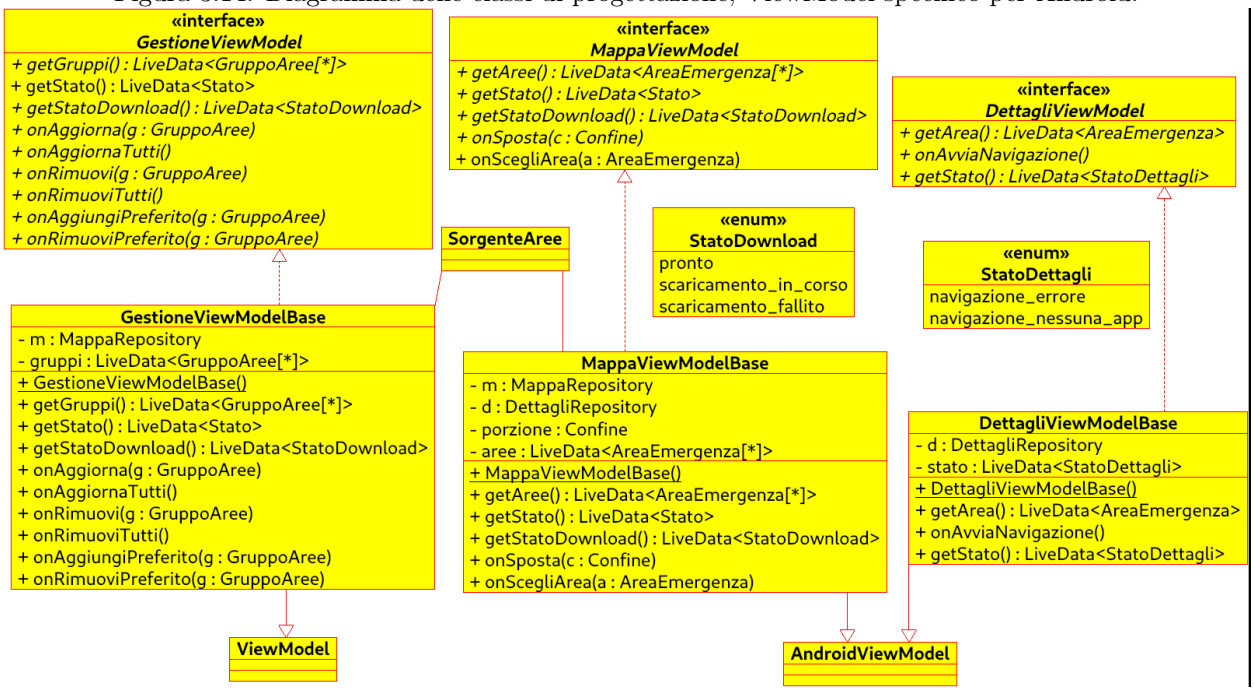
Figura 3.13: Diagramma delle classi di progettazione, ViewModel generico.



Nella progettazione del caso specifico di Android verrà sfruttato Android Architecture Components. In particolare il pattern Observer verrà applicato attraverso `LiveData<T>` e le classi del ViewModel estenderanno la classe `ViewModel`.

In particolare `MappaViewModel` e `DettagliViewModel` necessitano del `Context` per lanciare rispettivamente le altre View e la navigazione e per questo estenderanno `AndroidViewModel` [106][136].

Figura 3.14: Diagramma delle classi di progettazione, ViewModel specifico per Android.



Le classi concrete da utilizzare come Persistenza e SorgenteDati saranno definite in un file di configurazione.

3.2.6 View

Il ruolo delle View sarà semplicemente visualizzare i dati esposti tramite pattern Observer dai ViewModel.

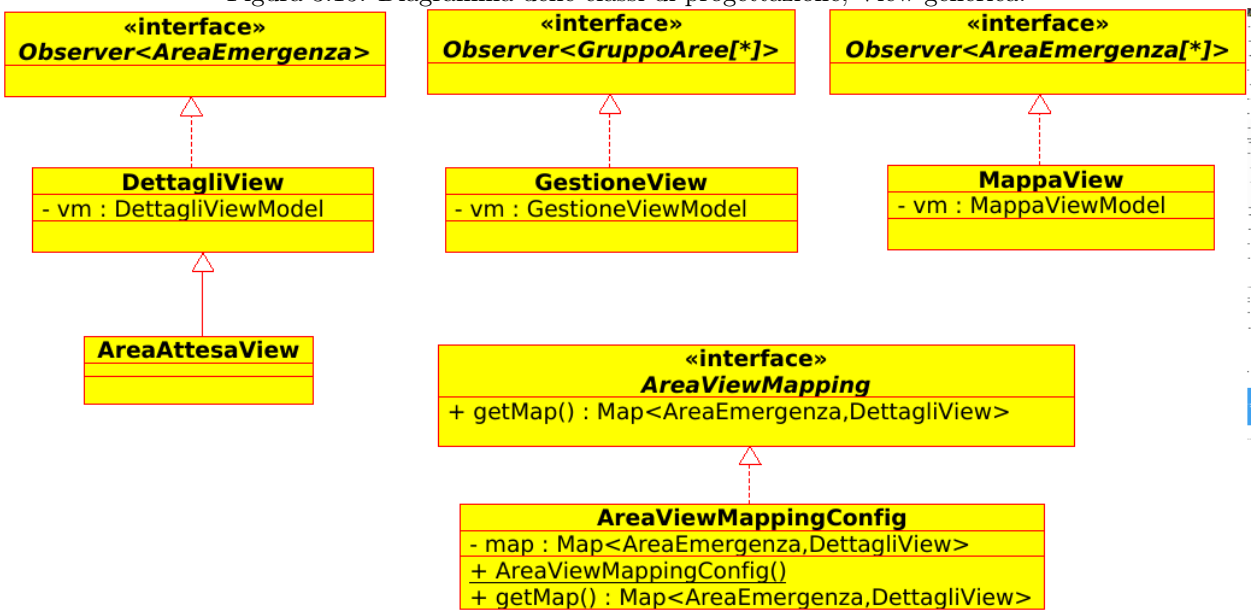
Ogni sottoclasse di AreaEmergenza che espone nuove caratteristiche necessiterà di una sottoclasse di DettagliView che la visualizzi opportunamente.

La necessità di scegliere quale DettagliView utilizzare in base al tipo di AreaEmergenza da visualizzare ci pone di fronte a un nuovo problema: come strutturare la scelta della DettagliView in modo che l'aggiunta di un nuovo tipo di AreaEmergenza non comporti una modifica del codice pre-esistente?

Purtroppo è impossibile risolvere questo problema esclusivamente con il codice se non violando vincoli strutturali (per esempio sarebbe possibile inserendo nell'`AreaEmergenza` una funzione che restituisca la classe della `DettagliView` da istanziare, ma inserire la gestione della View nel Model inquina il corretto funzionamento di MVVM).

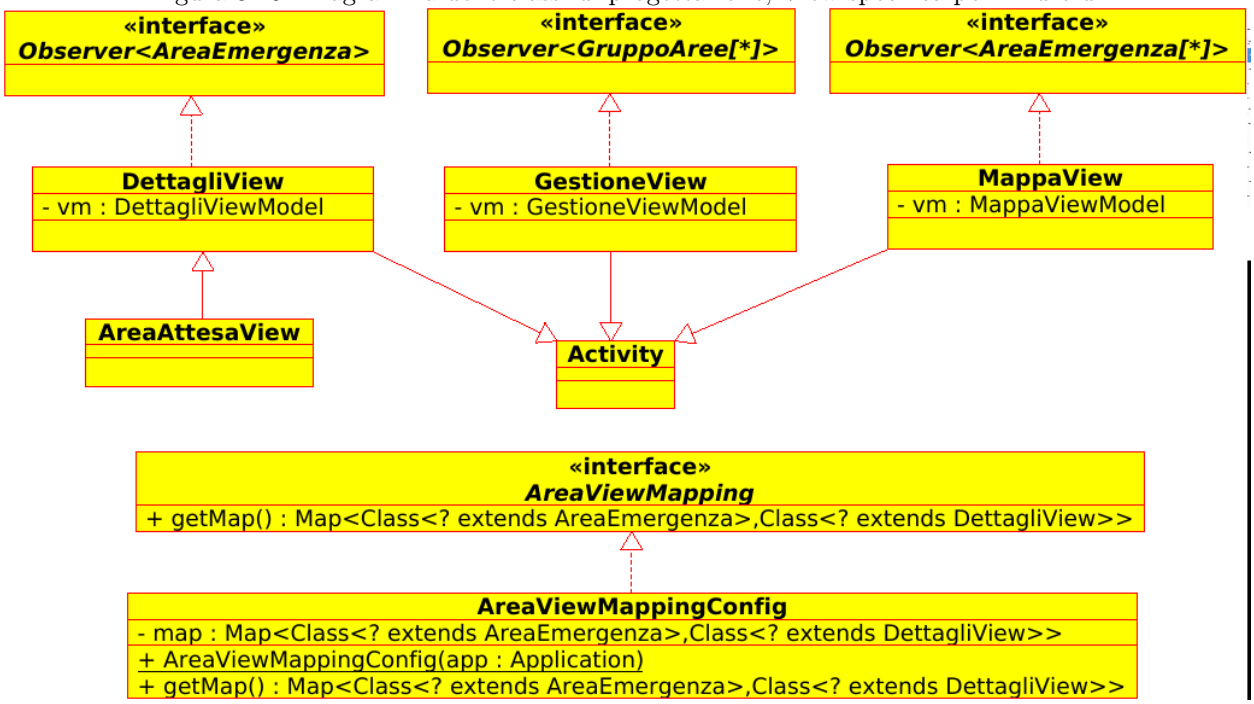
È quindi necessario utilizzare un file di configurazione che provveda a mappare le sottoclassi di `AreaEmergenza` alle sottoclassi di `DettagliView` e inserire una classe che legga il file e generi una corrispondente mappa delle classi.

Figura 3.15: Diagramma delle classi di progettazione, View generica.



In Android le attività vengono lanciate sfruttando gli Intent e `Class<T>` della Activity da lanciare (come nel codice A.21), quindi la mappatura in Android avverrà attraverso una `Map<Class<? extends AreaEmergenza>,Class<? extends DettagliView»`.

Figura 3.16: Diagramma delle classi di progettazione, View specifica per Android.



3.3 Interazioni

Specifica del comportamento dinamico delle classi.

Figura 3.17: Diagramma di sequenza, Spostamento della mappa.

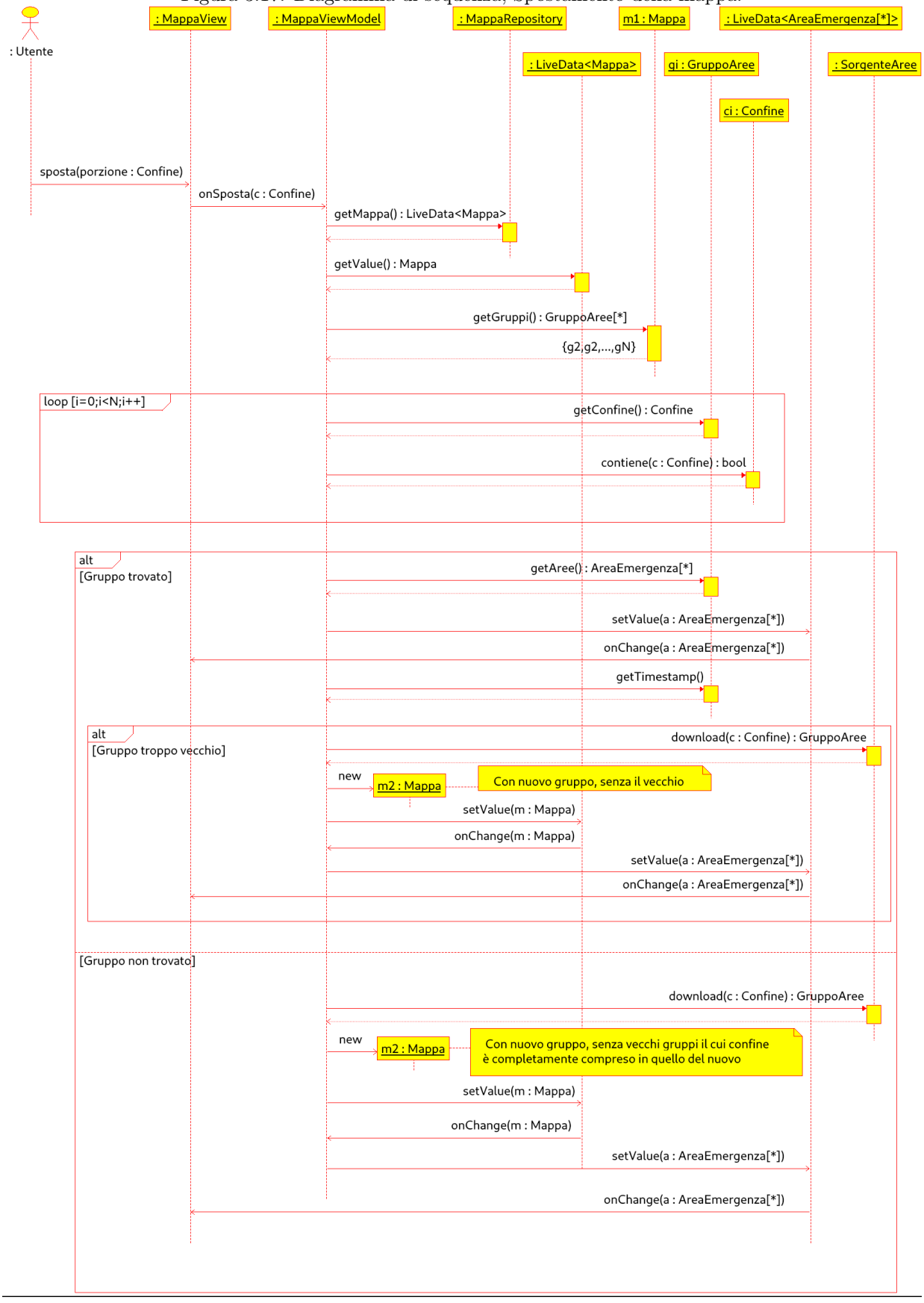


Figura 3.18: Diagramma di sequenza, Rimozione di un gruppo di aree.

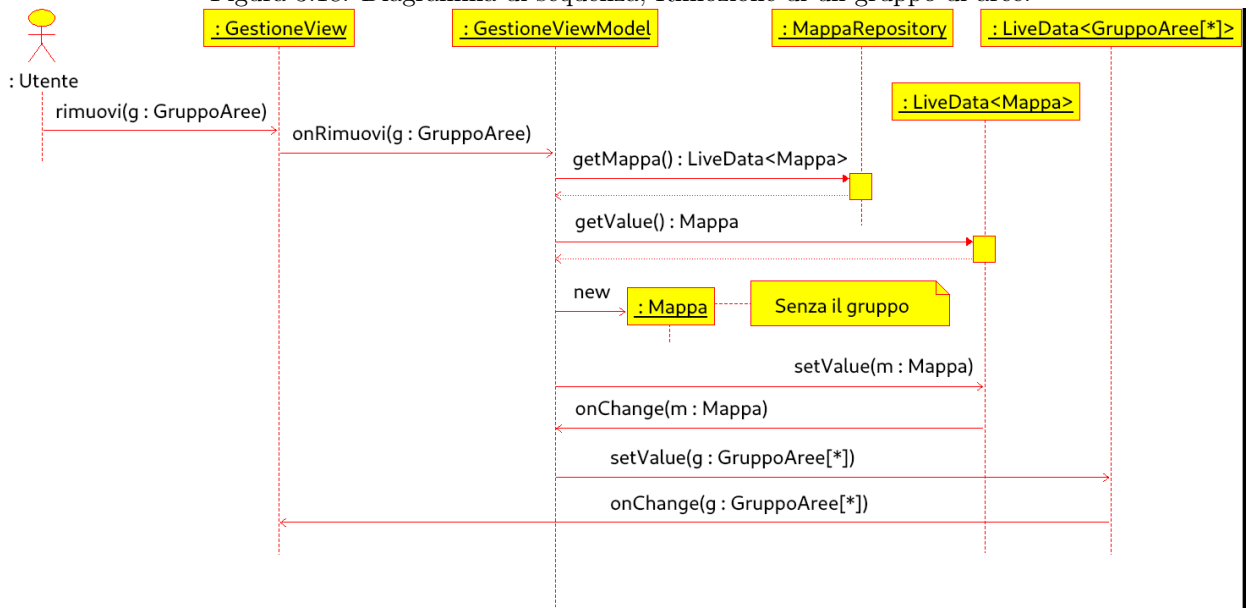


Figura 3.19: Diagramma di sequenza, Aggiornamento di un gruppo di aree.

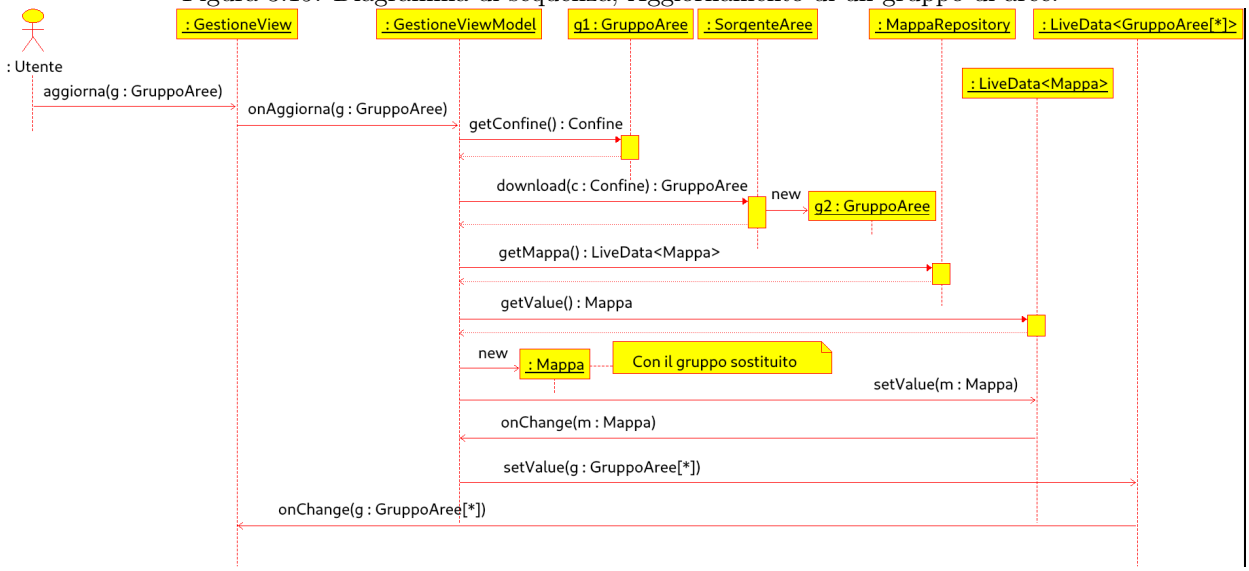
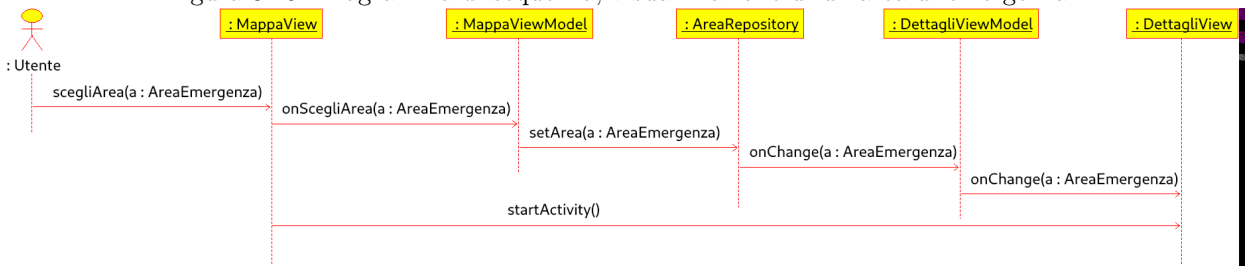


Figura 3.20: Diagramma di sequenza, visualizzazione di un'area di emergenza.



L'operazione `startActivity()` in quest'ultimo diagramma non sta a indicare una vera operazione ma l'insieme delle operazioni che vengono messe in atto per avviare la nuova attività.

Capitolo 4

Realizzazione e collaudo dei moduli

Realizzazione e collaudo dei moduli definiti nella fase di progettazione.

4.1 Scelte implementative

4.1.1 Visualizzazione mappa

Come osservato in 2.2.1 Google Maps SDK è lo strumento più economico per la visualizzazione della mappa, e per questo viene scelto nella realizzazione di MappaView e GestioneView. Per utilizzarlo viene inserita in build.gradle la seguente dipendenza:

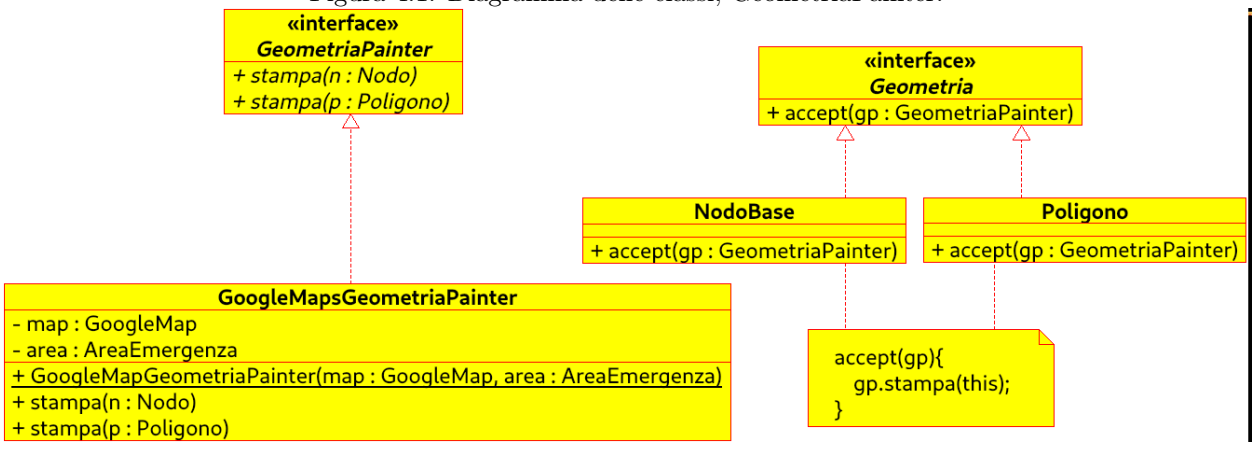
```
com.google.android.gms:play-services-maps:16.1.0
```

La mappa viene rappresentata da un oggetto di tipo `GoogleMap` [137].

Per disegnare un nodo sulla mappa con i relativi dettagli bisogna usare la funzione `GoogleMap::addMarker()` come nel codice A.22 [138][139][140]. Per disegnare un poligono con i relativi dettagli bisogna invece usare la funzione `GoogleMap::addPolygon()` come nel codice A.23 [141][142].

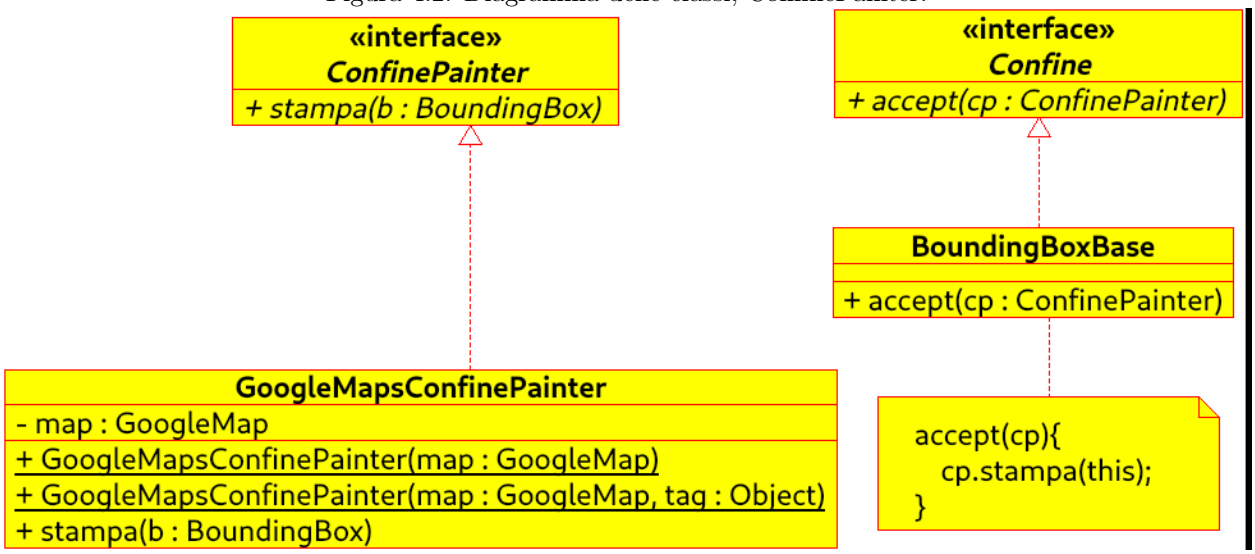
Per evitare di dover modificare la funzione di stampa esistente al momento dell'aggiunta di nuove `Geometria` è necessario introdurre una nuova interfaccia `GeometriaPainter` che implementa il pattern Visitor con Geometria (dove quest'ultimo svolge il ruolo di element con la nuova funzione `accept()` mentre `GeometriaPainter` svolge il ruolo di visitor con la `stampa()`).

Figura 4.1: Diagramma delle classi, GeometriaPainter.



Allo stesso modo, per evitare di dover modificare la funzione di stampa esistente al momento dell'aggiunta di nuovi `Confine` viene inserita l'interfaccia `ConfinePainter` che implementa il pattern Visitor con Confine.

Figura 4.2: Diagramma delle classi, ConfinePainter.



Per quanto riguarda la visualizzazione dell'attribuzione a OpenStreetMap, su Google Maps SDK è possibile inserirla creando una TextView apposita nel layout e inserendo il testo dell'attribuzione, come nel codice A.24 [76].

4.2 Realizzazione

Per l'applicazione sono necessarie le seguenti autorizzazioni:

- Per visualizzare la posizione è necessaria l'autorizzazione per la posizione
- Per scaricare i dati è necessario l'accesso a internet

Quindi aggiungo a `AndroidManifest.xml` le seguenti richieste:

Codice 4.1: Permessi in AndroidManifest

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

L'autorizzazione per la posizione verrà chiesta a runtime tramite `ActivityCompat.requestPermissions()` e il corrispondente callback `onRequestPermissionsResult()` [143].

4.2.1 Model

Il codice di realizzazione del Model può essere trovato nella sezione A.3.1.

4.2.2 Data layer

Il codice di realizzazione del Data Layer può essere trovato nella sezione A.3.3.

4.2.3 Rete

Il codice di realizzazione dello strato di Rete può essere trovato nella sezione A.3.4

4.2.4 ViewModel

Il codice di realizzazione del ViewModel può essere trovato nella sezione A.3.5. La configurazione della Persistenza e della Sorgente dati può essere trovata nella sezione A.3.7.

4.2.5 View

Il codice di realizzazione della View può essere trovato nella sezione A.3.6. La mappatura AreaEmergenza-DettagliView può essere trovata nella sezione A.3.7.

4.3 Collaudo

Il collaudo dei moduli viene effettuato attraverso gli Unit Test. Questi si dividono in due tipi:

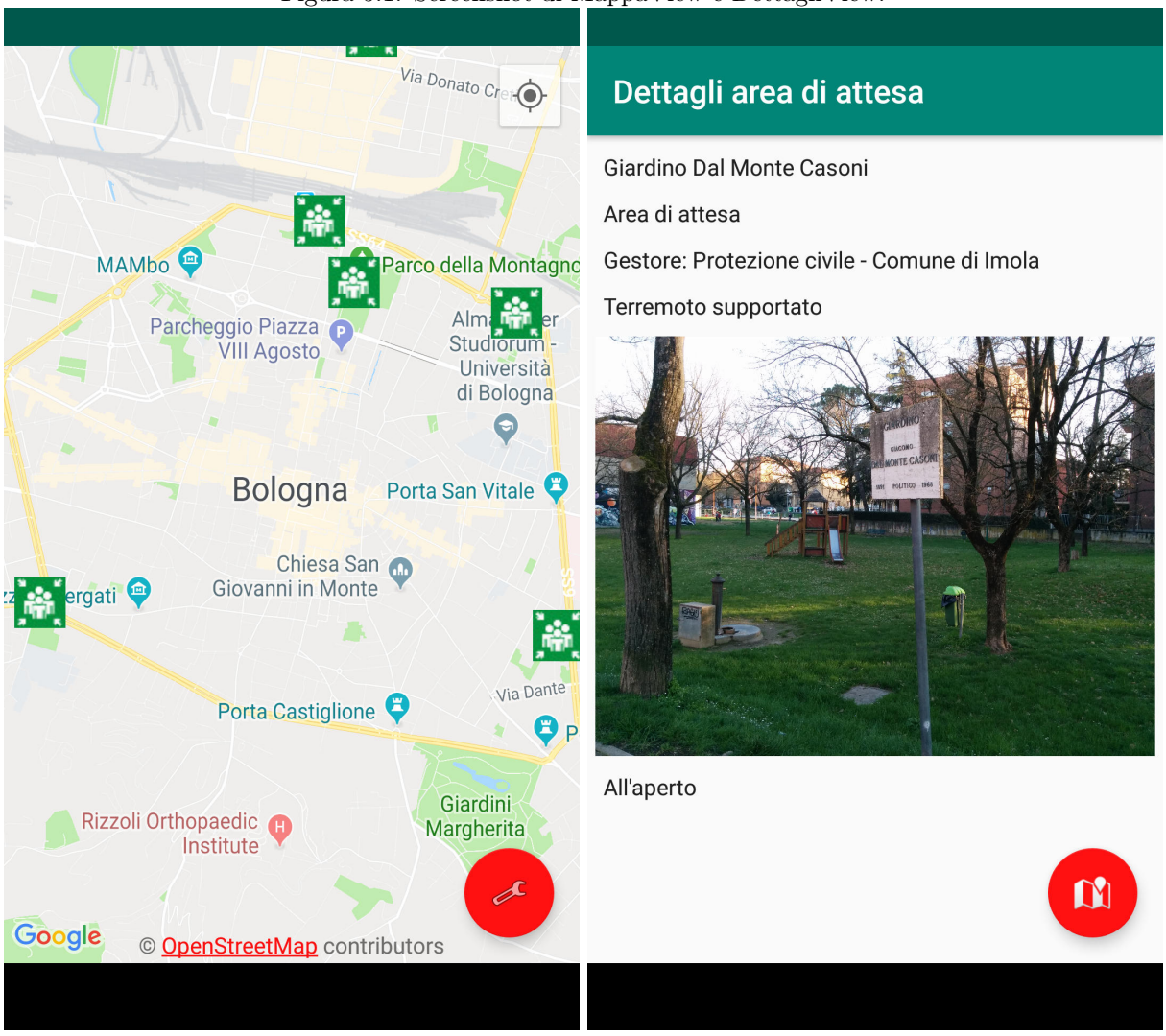
- **Unit Test**: non richiedono un `Context` o le `Resources` dell'applicazione, ma solo le classi da testare e eventuali librerie utilizzate da esse. È il caso delle classi del Model (che non dipendono da alcun contesto), del Data Layer (a cui si può passare un mock della Persistenza) e della rete (che in questo caso non è stata sottoposta a Unit test, ma sarebbe potuta essere testata simulando localmente il server remoto). Il codice di questi test può essere trovato nella sezione A.4.1.
- **Instrumented Unit Test**: richiedono un `Context` e pertanto necessitano di essere eseguiti sopra un dispositivo. È il caso del ViewModel e di AreaViewMapping. In particolare il `Context` viene simulato ed è possibile ottenerlo attraverso la classe `InstrumentationRegistry` [144]. Il codice di questi test può essere trovato nella sezione A.4.2.

Capitolo 5

Integrazione del sistema

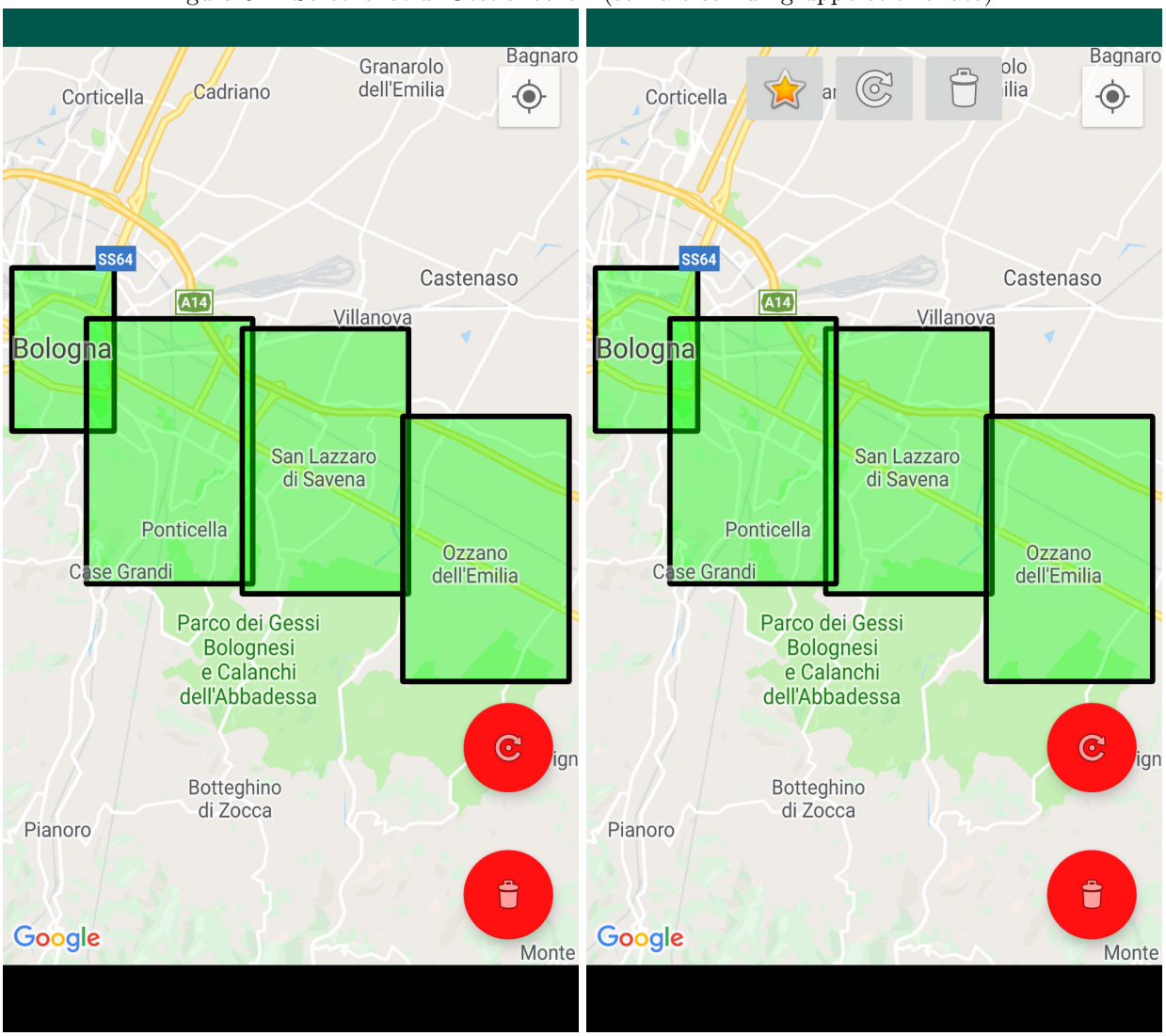
Una volta terminata la realizzazione dei moduli procediamo all'integrazione dei moduli e al collaudo dell'applicazione completa. Nella sezione A.5 è possibile trovare il codice di `build.gradle` con le dipendenze e `AndroidManifest.xml` con tutti i permessi richiesti e le attività registrate.

Figura 5.1: Screenshot di MappaView e DettagliView.



L'applicazione rispetta correttamente i requisiti specificati nella sezione di analisi, come visibile negli screenshot.

Figura 5.2: Screenshot di GestioneView (senza e con un gruppo selezionato).



Capitolo 6

Conclusioni

Un'attenta analisi del dominio in cui opera l'applicazione e una progettazione realizzata prestando particolare attenzione alla manutenibilità e all'estensibilità del codice hanno permesso di creare un'applicazione che può essere estesa senza necessità di modificare funzioni pre-esistenti (con la possibile eccezione della persistenza):

- I moduli dell'applicazione sono stati progettati applicando il **Dependency Inversion Principle**, permettendo alle classi degli altri moduli di fare riferimento a loro direttamente solo se strettamente necessario, facendo invece riferimento a loro attraverso le interfacce che implementano. La conseguenza di questa scelta è che le classi sono in grado di gestire le classi degli altri moduli indipendentemente dalla classe concreta, permettendo di estenderli senza richiedere modifiche al codice esistente.
- Le classi del Model rispettano il **pattern strutturale Private Class Data** mantenendo tutte le proprietà come private, esponendole tramite una funzione getter. Questo migliora la manutenibilità delle classi disaccoppiandole e riducendo le potenziali cause di errore. Tutte le classi del model sono inoltre immutabili, semplificando la gestione dei dati e eliminando numerose potenziali cause di errore.
- I moduli sono stati progettati tenendo in considerazione il **Open/Closed Principle**, permettendo l'estensione senza richiedere la modifica del codice pre-esistente:
  - Grazie all'applicazione del **pattern comportamentale Visitor** a tutte le classi che dipendono dalle implementazioni di `Confine` è possibile aggiungere nuove implementazioni senza dover modificare le funzioni pre-esistenti.
  - Grazie all'applicazione del **pattern comportamentale Visitor** a tutte le classi che dipendono dalle implementazioni di `Geometria` è possibile aggiungere nuove implementazioni senza dover modificare le funzioni pre-esistenti.
  - Grazie all'utilizzo del **pattern architetturale Model-View-ViewModel** è possibile creare nuove View senza dover modificare nulla negli altri livelli. A titolo esemplificativo, se volessimo aggiungere una attività che svolga le stesse funzioni di `MappaView` ma con un SDK per la visualizzazione delle mappe diverso (per esempio Mapbox Map SDK) basterebbe creare una nuova View che si interfacci con `MappaViewModel` allo stesso modo di `MappaView` e registrarla in `AndroidManifest.xml`, senza dover modificare nessun'altra classe pre-esistente.
- Lo strato di ViewModel rispetta il **Single Responsibility Principle** grazie alla scelta di separare la gestione dei dati, spostata allo strato di Data Layer. La conseguenza di questa scelta è che le classi del ViewModel e Data Layer svolgono compiti ben definiti e limitati, disaccoppiando le funzionalità e facilitando la manutenzione.
- L'applicazione del **Single Responsibility Principle** è evidente anche nelle sottoclassi di `Confine` e `Geometria`, che non contengono il codice per essere stampate sulla mappa, lasciando questo compito a `ConfinePainter` e `GeometriaPainter`.

In conclusione, un'attenta analisi e progettazione hanno permesso di realizzare un'applicazione facilmente manutenibile e estendibile.

Bibliografia

[1] Mappa. URL https://it.wikipedia.org/wiki/Mappa.

[2] Sistemi di riferimento e coordinate - parte 1, 01 2016. URL http://geomappando.com/2016/01/26/sistemi-di-riferimento-coordinate-1parte/.

[3] Latitude, longitude and coordinate system grids, 02 2018. URL https://gisgeography.com/latitude-longitude-coordinates/.

[4] Piero Calvini. Coordinate. 02 2012. URL https://www.ge.infn.it/~calvini/scamb/0227a.pdf.

[5] Coordinate e datum. Università di Perugia. URL http://labtopo.ing.unipg.it/files_sito/compiti/georef.pdf.

[6] Paolo Zatelli. Sistemi di riferimento in uso in italia. 2009. URL http://www.ing.unitn.it/~zatelli/cartografia_numerica/slides/Sistemi_di_riferimento_italiani.pdf#Outline0.3.

[7] Sistemi di riferimento e coordinate - parte 2, 07 2016. URL https://geomappando.com/2016/07/07/sistemi-riferimento-coordinate-2-parte/.

[8] Aurelio Stoppini. Il servizio di posizionamento con le reti gnss. 2009. URL http://labtopo.ing.unipg.it/files_sito/aurelio/Tutorial%20Reti%20GNSS_1_finale.pdf#page=26.

[9] ISO 19107. Geographic information — spatial schema, 05 2003.

[10] Iso geometry. URL http://docs.geotools.org/latest/userguide/library/opengis/geometry.html.

[11] UNI EN ISO 19107. Informazioni geografiche - schema dei dati spaziali, 05 2005. URL http://store.uni.com/catalogo/index.php/uni-en-iso-19107-2005.html.

[12] H. Butler, M. Daly, A. Doyle, Sean Gillies, T. Schaub, and T. Schaub. The GeoJSON Format. RFC 7946, August 2016. URL https://rfc-editor.org/rfc/rfc7946.txt.

[13] Treccani.it - vocabolario treccani on line, 2011.

[14] Jason Davies. Geographic bounding boxes. URL https://www.jasondavies.com/maps/bounds/.

[15] Google maps sdk for android, . URL https://developers.google.com/maps/documentation/android-sdk/intro.

[16] Prezzi per maps, routes e places, . URL https://cloud.google.com/maps-platform/pricing/sheet/.

[17] Tomtom maps sdk for android, . URL https://developer.tomtom.com/maps-sdk-android.

[18] Maps api pricing, . URL https://developer.tomtom.com/store/maps-api.

[19] Here android sdk, . URL https://developer.here.com/documentation/android-starter/dev_guide/topics/quick-start.html.

[20] Plans and pricing, . URL https://developer.here.com/plans.

[21] Mapbox maps sdk, . URL https://www.mapbox.com/android-docs/map-sdk/overview/.

[22] Plans & pricing, . URL https://www.mapbox.com/pricing/.

[23] Glossario di protezione civile: evento. URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13475.

[24] *Legge n. 225 del 24 febbraio 1992: istituzione del Servizio Nazionale della Protezione Civile*, 02 1992. URL http://www.protezionecivile.gov.it/jcms/it/view_prov.wp?contentId=LEG1602.

[25] Dipartimento della protezione civile. URL http://www.protezionecivile.gov.it/jcms/it/dipartimento.wp.

[26] *Decreto Legislativo n.1 del 2 gennaio 2018: Codice della protezione civile*, 01 2018. URL http://www.protezionecivile.gov.it/jcms/it/view_prov.wp?contentId=LEG67433.

[27] Glossario di protezione civile: Piano comunale di protezione civile, . URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13508.

[28] Descrizione del piano di emergenza, . URL http://www.protezionecivile.gov.it/jcms/it/piano_emergenza.wp.

[29] *Decreto-legge n. 59 del 15 maggio 2012 convertito dalla legge n. 100 del 12 luglio 2012: disposizioni urgenti per il riordino della protezione civile*, 05 2012. URL http://www.protezionecivile.gov.it/jcms/it/view_prov.wp?contentId=LEG34388.

[30] Glossario di protezione civile, . URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13367.

[31] Glossario di protezione civile: Aree di attesa, . URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13370.

[32] Glossario di protezione civile: Aree di ammassamento, . URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13369.

[33] Glossario di protezione civile: Aree di accoglienza, . URL http://www.protezionecivile.gov.it/jcms/it/glossario.wp?contentId=GLO13371.

[34] Piani di emergenza comunale, . URL http://www.protezionecivile.gov.it/jcms/it/piani_di_emergenza_comuna.wp.

[35] Mappa aree di emergenza palermo, . URL http://palermohub.opendatasicilia.it/piano_prot_civ.html.

[36] Mappa aree di emergenza regione sicilia, . URL https://siciliahub.github.io/mappe/aree_emergenza/about.html.

[37] Mappa aree di attesa comune di calderara di reno, . URL https://www.google.com/maps/d/u/0/viewer?mid=13bqNzYKUpZ0wxaQM3Of8FZCwGXg.

[38] Mappa aree di attesa comune di morano calabro, . URL http://www.comunemoranocalabro.it/Protezione_Civile/Full.aspx.

[39] Protezione Civile Provincia di Livorno. Mappa aree di attesa provincia di livorno, 09 2013. URL http://www.protezionecivileprovincialivorno.it/index.php?option=com_content&view=article&id=382.

[40] Openstreetmap, . URL https://www.openstreetmap.org/about.

[41] Pascal Neis and Dennis Zielstra. Recent developments and future trends in volunteered geographic information research: The case of openstreetmap. *Future Internet*, 6(1):76–106, 2014. ISSN 1999-5903. doi: 10.3390/fi6010076. URL http://www.mdpi.com/1999-5903/6/1/76.

[42] Openstreetmap foundation, . URL https://wiki.osmfoundation.org/wiki/Main_Page.

[43] Copyright e licenza. URL https://www.openstreetmap.org/copyright.

[44] Odbl in breve. URL https://it.okfn.org/odbl-riassunto/.

[45] OpenStreetMap Wiki. Elementi, . URL https://wiki.openstreetmap.org/wiki/IT:Elementi.

[46] . URL https://wiki.openstreetmap.org/wiki/Converting_to_WGS84.

[47] Projections/spatial reference systems, . URL http://openstreetmapdata.com/info/projections.

[48] OpenStreetMap Wiki. Etichette (tag), . URL https://wiki.openstreetmap.org/wiki/IT:Etichette.

[49] OpenStreetMap Wiki. Map features, . URL https://wiki.openstreetmap.org/wiki/Map_Features.

[50] Wikimedia Italia. Openstreetmap: uno strumento a servizio della comunità in occasione di catastrofi naturali, 01 2017. URL https://www.wikimedia.it/openstreetmap-uno-strumento-servizio-della-comunita-occasione-disastri-naturali/.

[51] Alessandro Palmas. Dati e openstreetmap per arginare il rischio ambientale, 10 2018. URL http://www.ingenium-magazine.it/dati-openstreetmap-arginare-rischio-ambientale/.

[52] Humanitarian openstreetmap team. URL https://www.hotosm.org/what-we-do.

[53] OpenStreetMap Wiki. Openstreetmap tasking manager, . URL https://wiki.openstreetmap.org/wiki/OSM_Tasking_Manager.

[54] Lorenza Castagneri. Con la mappa open source un aiuto ai paesi terremotati. *La Stampa*, 08 2016. URL http://www.lastampa.it/2016/08/25/tecnologia/con-la-mappa-open-source-un-aiuto-ai-paesi-terremotati-Cq8dsfBzEMGBJzKcomi0dP/pagina.html.

[55] V-iola project, 2017. URL http://violaproject.eu/.

[56] Wikimedia Italia. Il nostro contributo al progetto v-iola: il volontariato online come risorsa, 08 2018. URL https://www.wikimedia.it/nostro-contributo-al-progetto-v-iola-volontariato-online-risorsa/.

[57] About missing maps. URL https://www.missingmaps.org/about/.

[58] Wikimedia Italia. Protezione civile e openstreetmap: prove di collaborazione a chiavari su rischio inondazioni, 09 2017. URL https://www.wikimedia.it/protezione-civile-openstreetmap-prove-collaborazione-chiavari-rischio-inondazioni/.

[59] Peter Lawrence, Lazaros Filippidis, Anand Veeraswamy, and E.R. Galea. Utilising openstreetmap for urban evacuation analysis. 03 2016.

[60] OpenStreetMap Wiki. Tag:emergency=assembly_point, . URL https://wiki.openstreetmap.org/wiki/Tag:emergency=assembly%20point.

[61] OpenStreetMap Wiki. Key:name, . URL https://wiki.openstreetmap.org/wiki/IT:Key:name.

[62] OpenStreetMap Wiki. Key:operator, . URL https://wiki.openstreetmap.org/wiki/IT:Key:operator.

[63] OpenStreetMap Wiki. Key:capacity, . URL https://wiki.openstreetmap.org/wiki/Key:capacity.

[64] Overpass api. URL http://overpass-api.de/.

[65] OpenStreetMap Wiki. Overpass api, . URL https://wiki.openstreetmap.org/wiki/Overpass_API.

[66] *Parse XML data*. Google, . URL https://developer.android.com/training/basics/network-ops/xml. Android Developers Documentation.

[67] *JsonReader*. Google, . URL https://developer.android.com/reference/android/util/JsonReader. Android Developers Documentation.

[68] OpenStreetMap Wiki. *Overpass API language guide*, . URL https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide.

[69] OpenStreetMap Wiki. *Overpass API language examples*, . URL https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_API_by_Example.

[70] Quick guide to extracting data from osm using overpass turbo. URL https://github.com/mapbox/mapping/wiki/Overpass-Guide.

[71] OpenStreetMap Wiki. Public overpass api instances, . URL https://wiki.openstreetmap.org/wiki/Overpass_API#Public_Overpass_API_instances.

[72] *Intents*. Google, . URL https://developers.google.com/actions/reference/rest/intents.

[73] *ACTION VIEW*. Google, . URL https://developer.android.com/reference/android/content/Intent#ACTION_VIEW. Android Developers Documentation.

[74] *Google Maps Intents for Android*. Google, . URL https://developers.google.com/maps/documentation/urls/android-intents#launch_turn-by-turn_navigation. Google Maps Platform Documentation.

[75] Stefano Bontempi. Google, i riflessi della sentenza ue: le app preinstallate costeranno fino a 40 dollari, 10 2018. URL https://android.hdblog.it/2018/10/19/google-app-ue-costi-40-dollari-licenza/.

[76] Displaying attributions, . URL https://developers.google.com/places/android-sdk/attributions.

[77] Attribution - mapbox maps sdk for android, . URL https://docs.mapbox.com/help/how-mapbox-works/attribution/#mapbox-maps-sdk-for-android.

[78] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, May 2016. URL http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.

[79] Garante per la protezione dei dati personali. Cosa intendiamo per dati personali? URL https://www.garanteprivacy.it/home/diritti/cosa-intendiamo-per-dati-personali.

[80] Mapbox telemetry. URL https://www.mapbox.com/telemetry/.

[81] URL https://www.mapbox.com/privacy/#[AaMSWwdt].

[82] Antonio Zugaldia. Mapbox telemetry and gdpr, 06 2018. URL https://github.com/mapbox/mapbox-navigation-android/issues/963#issuecomment-395541147.

[83] Mapbox terms of service. URL https://www.mapbox.com/tos/#[TuMSTMMS].

[84] Hazem Saleh. Mvvm architecture, viewmodel and livedata (part 1), 5 2017. URL https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1.

[85] Android jetpack. URL https://developer.android.com/jetpack/. Android Developers Documentation.

[86] *Android Architecture Components*. Google, . URL https://developer.android.com/topic/libraries/architecture/. Android Developers Documentation.

[87] *Handling Lifecycles with Lifecycle-Aware Components*. Google, . URL https://developer.android.com/topic/libraries/architecture/lifecycle. Android Developers Documentation.

[88] *android.arch.lifecycle*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/package-summary. Android Developers Documentation.

[89] *LifecycleObserver*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/LifecycleObserver. Android Developers Documentation.

[90] *OnLifecycleEvent*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/OnLifecycleEvent. Android Developers Documentation.

[91] *Lifecycle*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/Lifecycle. Android Developers Documentation.

[92] *LifecycleOwner*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/LifecycleOwner. Android Developers Documentation.

[93] *LiveData Overview*. Google, . URL https://developer.android.com/topic/libraries/architecture/livedata. Android Developers Documentation.

[94] *android.arch.lifecycle.Observer*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/Observer. Android Developers Documentation.

[95] *LiveData*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/LiveData. Android Developers Documentation.

[96] *android.arch.lifecycle.MutableLiveData*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/MutableLiveData. Android Developers Documentation.

[97] Mediatorlivedata. URL https://developer.android.com/reference/android/arch/lifecycle/MediatorLiveData.html. Android Developers Documentation.

[98] Jose Alcérreca. Livedata beyond the viewmodel - reactive patterns using transformations and mediatorlivedata, 07 2018. URL https://medium.com/androiddevelopers/livedata-beyond-the-viewmodel-reactive-patterns-using-transformations-and-mediatorlivedata-fda520b

[99] Fred Porciúncula. Mediatorlivedata to the rescue, 07 2018. URL https://proandroiddev.com/mediatorlivedata-to-the-rescue-5d27645b9bc3.

[100] *ViewModel Overview*. Google, . URL https://developer.android.com/topic/libraries/architecture/viewmodel. Android Developers Documentation.

[101] *ViewModel*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/ViewModel. Android Developers Documentation.

[102] *ViewModelProviders*. Google, . URL https://developer.android.com/reference/android/arch/lifecycle/ViewModelProviders. Android Developers Documentation.

[103] Data binding library, . URL https://developer.android.com/topic/libraries/data-binding/#java. Android Developers Documentation.

[104] Data binding - get started, . URL https://developer.android.com/topic/libraries/data-binding/start. Android Developers Documentation.

[105] Husayn Hakeem. Android by example: Mvvm + data binding, 09 2017. URL https://medium.com/@husayn.hakeem/android-by-example-mvvm-data-binding-introduction-part-1-6a7a5f388bf7.

[106] Androidviewmodel. URL https://developer.android.com/reference/android/arch/lifecycle/AndroidViewModel. Android Developers Documentation.

[107] Lifecycle. URL https://developer.android.com/jetpack/androidx/releases/lifecycle. Android Developers Documentation.

[108] *Room Persistence Library*. Google, . URL https://developer.android.com/topic/libraries/architecture/room. Android Developers Documentation.

[109] *android.arch.persistence.room*. Google, . URL https://developer.android.com/reference/android/arch/persistence/room/package-summary. Android Developers Documentation.

[110] Guide to app architecture. URL https://developer.android.com/jetpack/docs/guide. Android Developers Documentation.

[111] Jose Alcérreca. Viewmodels and livedata: Patterns + antipatterns, 9 2017. URL https://medium.com/androiddevelopers/viewmodels-and-livedata-patterns-antipatterns-21efaef74a54.

[112] Anubhav Gupta. Building an android app using android architecture components: Room, viewmodel, and livedata, 1 2019. URL https://proandroiddev.com/building-an-android-app-using-android-architecture-components-room-viewmodel-and-livedata-702a0af8

[113] Mario Sanoguera de Lorenzo. The death of presenters and the rise of viewmodels (aac), 3 2018. URL https://proandroiddev.com/the-death-of-presenters-and-the-rise-of-viewmodels-aac-f14d54b419a.

[114] Robert C. Martin. The clean architecture, 08 2012. URL https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

[115] Viewmodels: Persistence, onsaveinstancestate(), restoring ui state and loaders. URL https://medium.com/androiddevelopers/viewmodels-persistence-onsaveinstancestate-restoring-ui-state-and-loaders-fc

[116] Android room with a view - introduction, . URL https://codelabs.developers.google.com/codelabs/android-room-with-a-view/. Android Developers Documentation.

[117] Save data in a local database using room, . URL https://developer.android.com/training/data-storage/room/#java. Android Developers Documentation.

[118] Florina Muntenescu. 7 steps to room, 07 2017. URL https://medium.com/androiddevelopers/7-steps-to-room-27a5fe5f99b2.

[119] *Defining data using Room entities*. Google, . URL https://developer.android.com/training/data-storage/room/defining-data. Android Developers Documentation.

[120] Brijesh Thumar. Room: Database relationships, 12 2017. URL http://androidkt.com/database-relationships/.

[121] Referencing complex data using room, . URL https://developer.android.com/training/data-storage/room/referencing-data. Android Developers Documentation.

[122] Mark Murphy. Polymorphic room relations. URL https://commonsware.com/AndroidArch/previews/polymorphic-room-relations.

[123] Danail Alexiev. Polymorphic entities in room. URL https://stackoverflow.com/a/52113714/2347196.

[124] *Accessing data using Room DAOs*. Google, . URL https://developer.android.com/training/data-storage/room/accessing-data. Android Developers Documentation.

[125] *Database*. Google, . URL https://developer.android.com/reference/androidx/room/Database. Android Developers Documentation.

[126] *RoomDatabase*. Google, . URL https://developer.android.com/reference/androidx/room/RoomDatabase. Android Developers Documentation.

[127] Florina Muntenescu. 7 pro-tips for room, 11 2017. URL https://medium.com/androiddevelopers/7-pro-tips-for-room-fbadea4bfbd1.

[128] Testing your database, . URL https://developer.android.com/training/data-storage/room/testing-db. Android Developers Documentation.

[129] Retrofit. URL https://square.github.io/retrofit/.

[130] Anupam Chugh. Retrofit android example tutorial. URL https://www.journaldev.com/13639/retrofit-android-example-tutorial.

[131] David Weiser Lars Vogel, Simon Scholz. Using retrofit 2.x as rest client - tutorial, 06 2018. URL https://www.vogella.com/tutorials/Retrofit/article.html.

[132] jsonschema2pojo. URL http://www.jsonschema2pojo.org/.

[133] Norman Peitek. Retrofit 2 - adding and customizing the gson converter, 10 2016. URL https://futurestud.io/tutorials/retrofit-2-adding-customizing-the-gson-converter.

[134] *Drawable*. Google, . URL https://developer.android.com/reference/android/graphics/drawable/Drawable. Android Developers Documentation.

[135] Localdatetime. URL https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html. Android Developers Documentation.

[136] Mvvm pattern and startactivity. URL https://stackoverflow.com/a/46827625/2347196.

[137] Googlemap. URL https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.

[138] Shapes. URL https://developers.google.com/maps/documentation/android-sdk/shapes.

[139] Marker, . URL https://developers.google.com/android/reference/com/google/android/gms/maps/model/Marker.

[140] Markeroptions, . URL https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions.

[141] Polygon, . URL https://developers.google.com/android/reference/com/google/android/gms/maps/model/Polygon.

[142] Polygonoptions, . URL https://developers.google.com/android/reference/com/google/android/gms/maps/model/PolygonOptions.

[143] Request app permissions. URL https://developer.android.com/training/permissions/requesting. Android Developers Documentation.

[144] Instrumentationregistry. URL https://developer.android.com/reference/android/support/test/InstrumentationRegistry. Android Developers Documentation.

[145] Room, . URL https://developer.android.com/jetpack/androidx/releases/room. Android Developers Documentation.

[146] Start another activity. URL https://developer.android.com/training/basics/firstapp/starting-activity. Android Developers Documentation.

Appendice A

Codice

A.1 Analisi

A.1.1 Overpass API

Codice A.1: Esempio di query Overpass

```
[out:json][timeout:25];

// Ottieni e metti in output tutte le aree di attesa nel bounding box
// {{bbox}} = <latitudine min>,<longitudine min>,<lat max>,<lon max>
(
  node["emergency"="assembly_point"
      ](44.48655583488967,11.311197280883789,44.49589310341622,11.32965087890625);
  way["emergency"="assembly_point"
      ](44.48655583488967,11.311197280883789,44.49589310341622,11.32965087890625);
);
out body;

// Ottieni e metti in output i nodi che compongono le way selezionate
>;
out skel qt;
```

Per tornare al testo: 2.2.5

Codice A.2: Esempio di risultato JSON di una query Overpass

```
{
  "version": 0.6,
  "generator": "Overpass API 0.7.55.5 2ca3f387",
  "osm3s": {
    "timestamp_osm_base": "2019-01-21T16:49:02Z",
    "copyright": "The data included in this document is from www.openstreetmap.org.
       The data is made available under ODbL."
  },
  "elements": [

{
  "type": "node",
  "id": 5777070727,
  "lat": 44.4924601,
  "lon": 11.3115013,
  "tags": {
    "emergency": "assembly_point",
    "name": "Piazza della Pace",
```

```
    "operator": "Protezione Civile",
    "ref": "9",
    "source": "Piano comunale di Protezione Civile"
  }
},
{
  "type": "node",
  "id": 5777070734,
  "lat": 44.4895375,
  "lon": 11.3291396,
  "tags": {
    "emergency": "assembly_point",
    "name": "Giardino di Villa Cassarini",
    "operator": "Protezione Civile",
    "ref": "7",
    "source": "Piano comunale di Protezione Civile"
  }
}

  ]
}
```

Per tornare al testo: 2.2.5

Codice A.3: Esempio di query Overpass

```
[out:xml][timeout:25];

// Ottieni e metti in output tutte le aree di attesa nel bounding box
// {{bbox}} = <latitudine min>,<longitudine min>,<lat max>,<lon max>
(
  node["emergency"="assembly_point"
      ](44.48655583488967,11.311197280883789,44.49589310341622,11.32965087890625);
  way["emergency"="assembly_point"
      ](44.48655583488967,11.311197280883789,44.49589310341622,11.32965087890625);
);
out body;

// Ottieni e metti in output i nodi che compongono le way selezionate
>;
out skel qt;
```

Per tornare al testo: 2.2.5

Codice A.4: Esempio di risultato XML di una query Overpass

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass␣API␣0.7.55.5␣2ca3f387">
<note>The data included in this document is from www.openstreetmap.org. The data is
    made available under ODbL.</note>
<meta osm_base="2019-01-21T17:02:03Z"/>

  <node id="5777070727" lat="44.4924601" lon="11.3115013">
    <tag k="emergency" v="assembly_point"/>
    <tag k="name" v="Piazza␣della␣Pace"/>
    <tag k="operator" v="Protezione␣Civile"/>
    <tag k="ref" v="9"/>
    <tag k="source" v="Piano␣comunale␣di␣Protezione␣Civile"/>
  </node>
  <node id="5777070734" lat="44.4895375" lon="11.3291396">
    <tag k="emergency" v="assembly_point"/>
    <tag k="name" v="Giardino␣di␣Villa␣Cassarini"/>
```

```
    <tag k="operator" v="Protezione Civile"/>
    <tag k="ref" v="7"/>
    <tag k="source" v="Piano comunale di Protezione Civile"/>
  </node>

</osm>
```

Per tornare al testo: 2.2.5

Codice A.5: Esempio di query Overpass

```
[out:json][timeout:25];

// Trova tutti i poligoni che circondano il punto
// {{center}} = <latitudine>,<longitudine>
is_in(44.3534055,11.7141310);

// Filtra solo i confini amministrativi comunali
// Documentazione dei valori per admin_level:
// https://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative#10
    _admin_level_values_for_specific_countries
area._[boundary=administrative][admin_level=8]->.a;

// Metti in output il confine selezionato
.a;
out qt;

// Ottieni e metti in output tutte le aree di attesa nella regione selezionata
(
node["emergency"="assembly_point"](area.a);
way["emergency"="assembly_point"](area.a);
);
out qt;

// Ottieni e metti in output i nodi che compongono le way selezionate
>;
out skel qt;
```

Per tornare al testo: 2.2.5

A.1.2 Navigazione Android

Codice A.6: Esempio di richiesta di navigazione in Android

```
float lat = ..., // Latitudine
        lon = ...; // Longitudine
Uri uri = Uri.parse("google.navigation:q=" + lat + "," + lon);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

Per tornare al testo: 2.2.6

A.1.3 Mapbox Telemetry

Codice A.7: Disabilitazione Mapbox Telemetry

```
TelemetryDefinition telemetry = Mapbox.getTelemetry();
if (telemetry != null) {
        telemetry.setUserTelemetryRequestState(false);
}
```

Per tornare al testo: 2.2.7

A.2 Progettazione

A.2.1 Android Architecture Components

Codice A.8: Dipendenze gradle per Android Architecture Components [107]

```
dependencies {
        def lifecycle_version = "1.1.1"
        implementation "android.arch.lifecycle:viewmodel:$lifecycle_version"
        implementation "android.arch.lifecycle:extensions:$lifecycle_version"
        annotationProcessor "android.arch.lifecycle:compiler:$lifecycle_version"
        testImplementation "android.arch.core:core-testing:$lifecycle_version"
}
```

Per tornare al testo: 3.1.2

Codice A.9: Esempio di utilizzo di `ViewModelProviders` per creare un `ViewModel` [100]

```
public class MyActivity extends AppCompatActivity {
    public void onCreate(Bundle savedInstanceState) {
        // Create a ViewModel the first time the system calls an activity's onCreate
            () method.
        // Re-created activities receive the same MyViewModel instance created by
            the first activity.

        MyViewModel model = ViewModelProviders.of(this).get(MyViewModel.class);
        model.getUsers().observe(this, users -> {
            // update UI
        });
    }
}
```

Per tornare al testo: 3.1.2

A.2.2 Room

Codice A.10: Dipendenze gradle per Room [145]

```
dependencies {
        def room_version="1.1.1"
        implementation "android.arch.persistence.room:runtime:$room_version"
        annotationProcessor "android.arch.persistence.room:compiler:$room_version"
        testImplementation "android.arch.persistence.room:testing:$room_version"
}
```

Per tornare al testo: 3.1.3

Codice A.11: Esempio di Entity con primary key semplice [117]

```
@Entity(tableName = "users")
public class User {
    @PrimaryKey
    public int id;

    @ColumnInfo(name = "first_name")
    public String firstName;

    @ColumnInfo(name = "last_name")
    public String lastName;
}
```

Per tornare al testo: 3.1.3

Codice A.12: Esempio di Entity con primary key composta [117]

```
@Entity(primaryKeys = {"firstName", "lastName"})
public class User {
    public String firstName;
    public String lastName;
}
```

Per tornare al testo: 3.1.3

Codice A.13: Esempio di Dao [117]

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

Per tornare al testo: 3.1.3

Codice A.14: Esempio di Database [117]

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}
```

Per tornare al testo: 3.1.3

Codice A.15: Esempio di utilizzo di Room [117]

```
AppDatabase db = Room.databaseBuilder(getApplicationContext(), AppDatabase.class, "
    database-name").build();
```

Per tornare al testo: 3.1.3

A.2.3 Retrofit

Codice A.16: Dipendenze gradle per Retrofit [129][133]

```
dependencies {
        def retrofit_version = "2.5.0"
        implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
        implementation "com.squareup.retrofit2:converter-gson:$retrofit_version"
}
```

Per tornare al testo: 3.1.3

Codice A.17: Esempio di POJO per Retrofit [130]

```
package com.journaldev.retrofitintro.pojo;

import com.google.gson.annotations.SerializedName;
```

```
import java.util.ArrayList;
import java.util.List;

public class MultipleResource {

    @SerializedName("page")
    public Integer page;
    @SerializedName("per_page")
    public Integer perPage;
    @SerializedName("total")
    public Integer total;
    @SerializedName("total_pages")
    public Integer totalPages;
    @SerializedName("data")
    public List<Datum> data = null;

    public class Datum {

        @SerializedName("id")
        public Integer id;
        @SerializedName("name")
        public String name;
        @SerializedName("year")
        public Integer year;
        @SerializedName("pantone_value")
        public String pantoneValue;

    }
}
```

Per tornare al testo: 3.1.3

Codice A.18: Esempio di interfaccia per Retrofit [130]

```
package com.journaldev.retrofitintro;

import com.journaldev.retrofitintro.pojo.MultipleResource;
import com.journaldev.retrofitintro.pojo.User;
import com.journaldev.retrofitintro.pojo.UserList;

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.Field;
import retrofit2.http.FormUrlEncoded;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.Query;

interface APIInterface {

    @GET("/api/unknown")
    Call<MultipleResource> doGetListResources();

    @POST("/api/users")
    Call<User> createUser(@Body User user);

    @GET("/api/users?")
    Call<UserList> doGetUserList(@Query("page") String page);
```

```
    @FormUrlEncoded
    @POST("/api/users?")
    Call<UserList> doCreateUserWithField(@Field("name") String name, @Field("job")
        String job);
}
```

Per tornare al testo: 3.1.3

Codice A.19: Esempio di client per Retrofit [130]

```
package com.journaldev.retrofitintro;

import com.journaldev.retrofitintro.pojo.MultipleResource;

import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

class APIClient {

    private static Retrofit retrofit = null;

    static Retrofit getClient() {

        HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient client = new OkHttpClient.Builder().addInterceptor(interceptor)
            .build();


        retrofit = new Retrofit.Builder()
                .baseUrl("https://reqres.in")
                .addConverterFactory(GsonConverterFactory.create())
                .client(client)
                .build();



        return retrofit;
    }

}
```

Per tornare al testo: 3.1.3

Codice A.20: Esempio di utilizzo del client per Retrofit [130]

```
APIInterface apiInterface = APIClient.getClient().create(APIInterface.class);
Call<MultipleResource> call = apiInterface.doGetListResources();
call.enqueue(new Callback<MultipleResource>() {
        @Override
        public void onResponse(Call<MultipleResource> call, Response<
            MultipleResource> response) {
                Log.d("TAG",response.code()+"");

                String displayResponse = "";

                MultipleResource resource = response.body();
                Integer text = resource.page;
                Integer total = resource.total;
```

```
                Integer totalPages = resource.totalPages;
                List<MultipleResource.Datum> datumList = resource.data;

                // Utilizzo dei dati
        }

        @Override
        public void onFailure(Call<MultipleResource> call, Throwable t) {
                call.cancel();
        }
});
```

Per tornare al testo: 3.1.3

A.2.4 Lancio attività

Codice A.21: Esempio di avvio di una Activity [146]

```
public class MainActivity extends AppCompatActivity {
    public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.editText);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

Per tornare al testo: 3.2.6

A.2.5 Google Maps Android SDK

Codice A.22: Esempio di utilizzo GoogleMap::addMarker()

```
GoogleMap map = ...
Area area = ...
Nodo n = (Nodo) area.getGeometria();
LatLng pos = new LatLng(n.getLatitudine(), n.getLongitudine());
BitmapDescriptor icon = BitmapDescriptorFactory.fromResource(R.drawable.attesa);
MarkerOptions mOpt = new MarkerOptions()
                        .draggable(false)
                        .position(pos)
                        .icon(icon)
                        .title(area.getNome());
Marker m = map.addMarker(mOpt);
m.setTag(area);
```

Codice A.23: Esempio di utilizzo GoogleMap::addPolygon()

```
GoogleMap map = ...
Area area = ...
Poligono p = (Poligono) area.getGeometria();
PolygonOptions pOpt = new PolygonOptions();
for(Nodo n : p.getVertici())
```

```
        pOpt.add(new LatLng(n.getLatitudine(), n.getLongitudine()));
Polygon gmp = map.addPolygon(pOpt);
gmp.setTag(area);
```

Codice A.24: Esempio di visualizzazione dell'attribuzione

```
String thirdPartyAttributions = "Listings by <a href=\"https://www.example.com/\">
    Example Company</a>";
TextView attributionsText = (TextView) findViewById(R.id.attributions);
attributionsText.setText(thirdPartyAttributions);
```

Per tornare al testo: 4.1.1

A.3 Realizzazione moduli

A.3.1 Model

Fotografia

Codice A.25: Fotografia

```
package it.dsantini.areediattesa.model;

import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;

import java.util.Optional;

public interface Fotografia {
    /**
     * @return Immagine
     */
    @NonNull
    Drawable getImmagine();

    /**
     * @return Titolo della fotografia
     */
    @NonNull
    Optional<String> getTitolo();
}
```

Codice A.26: FotografiaBase

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.arch.persistence.room.Ignore;
import android.arch.persistence.room.PrimaryKey;
import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import java.util.Optional;

import it.dsantini.areediattesa.model.Fotografia;

@Entity
public class FotografiaBase implements Fotografia {
    @PrimaryKey
    private final int id;
```

```
    public static final int DEFAULT_ID = Integer.MIN_VALUE;

    @NonNull
    private final Drawable immagine;
    @Nullable
    private final String titolo;

    public FotografiaBase(int id, @NonNull Drawable immagine, @Nullable String
        titolo) {
        if (immagine == null)
            throw new IllegalArgumentException("immagine == null");

        this.id = id;
        this.immagine = immagine;
        this.titolo = titolo;
    }

    @Ignore
    public FotografiaBase(@NonNull Drawable immagine, @Nullable String titolo) {
        this(DEFAULT_ID, immagine, titolo);
    }

    public int getId() {
        return id;
    }

    @NonNull
    @Override
    public Drawable getImmagine() {
        return immagine;
    }

    @NonNull
    @Override
    public Optional<String> getTitolo() {
        return Optional.ofNullable(titolo);
    }
}
```

Geometria

Codice A.27: Geometria

```
package it.dsantini.areediattesa.model;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.view.helper.GeometriaPainter;

public interface Geometria {
    /**
     * @return Un Nodo che si trova sulla superficie della Geometria
     */
    @NonNull
    Nodo getPuntoSuSuperficie();

    /**
     * Pattern Visitor per la stampa della geometria (funzione accept)
     * Implementazione tipica: return gp.stampa(this)
     *
```

```
     * @param gp Painter con cui stampare la geometria
     */
    void accept(GeometriaPainter gp);
}
```

Codice A.28: Nodo

```
package it.dsantini.areediattesa.model;

public interface Nodo extends Geometria {
    /**
     * @return Latitudine del nodo
     */
    double getLatitudine();

    /**
     * @return Longitudine del nodo
     */
    double getLongitudine();
}
```

Codice A.29: NodoBase

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.view.helper.GeometriaPainter;

@Entity(primaryKeys = {"latitudine", "longitudine"})
public class NodoBase implements Nodo {
    private final double latitudine, longitudine;

    public NodoBase(double latitudine, double longitudine) {
        if (latitudine < -90)
            throw new IllegalArgumentException("latitudine = " + latitudine + " <
                -90");
        if (latitudine > 90)
            throw new IllegalArgumentException("latitudine = " + latitudine + " > 90
                ");
        if (longitudine < -180)
            throw new IllegalArgumentException("longitudine = " + longitudine + " <
                -180");
        if (longitudine > 180)
            throw new IllegalArgumentException("longitudine = " + longitudine + " >
                180");

        Double Latitudine = latitudine, Longitudine = longitudine;
        if (Latitudine.isNaN())
            throw new IllegalArgumentException("latitudine == NaN");
        if (Latitudine.isInfinite())
            throw new IllegalArgumentException("latitudine == infinto");
        if (Longitudine.isNaN())
            throw new IllegalArgumentException("longitudine == NaN");
        if (Longitudine.isInfinite())
            throw new IllegalArgumentException("longitudine == infinito");

        this.latitudine = latitudine;
```

```
        this.longitudine = longitudine;
    }

    @Override
    public double getLatitudine() {
        return latitudine;
    }

    @Override
    public double getLongitudine() {
        return longitudine;
    }

    @NonNull
    @Override
    public Nodo getPuntoSuSuperficie() {
        return this;
    }

    @Override
    public void accept(GeometriaPainter gp) {
        gp.stampa(this);
    }

    @Override
    public String toString() {
        return "NodoBase, lat=" + getLatitudine() + ", lon=" + getLongitudine();
    }
}
```

Codice A.30: Poligono

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.arch.persistence.room.Ignore;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Geometria;
import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.view.helper.GeometriaPainter;

@Entity
public class Poligono implements Geometria {
    @PrimaryKey
    private final int id;
    public static final int DEFAULT_ID = Integer.MIN_VALUE;

    @NonNull
    private final Nodo[] vertici;

    public Poligono(int id, @NonNull Nodo[] vertici) {
        if (vertici == null)
            throw new IllegalArgumentException("vertici == null");

        if (vertici.length < 3)
            throw new IllegalArgumentException("vertici ha meno di 3 elementi");
```

```
        for (int i = 0; i < vertici.length; i++) {
            if (vertici[i] == null)
                throw new IllegalArgumentException("vertici[" + i + "] == null");

            for (int j = i + 1; j < vertici.length; j++)
                if (vertici[i] == vertici[j])
                    throw new IllegalArgumentException("vertici[" + i + "] ==
                        vertici[" + j + "]");
        }

        this.id = id;
        this.vertici = vertici;
    }

    @Ignore
    public Poligono(@NonNull Nodo[] vertici) {
        this(DEFAULT_ID, vertici);
    }

    public int getId() {
        return id;
    }

    @NonNull
    public Nodo[] getVertici() {
        return vertici;
    }

    @NonNull
    @Override
    public Nodo getPuntoSuSuperficie() {
        return vertici[0];
    }

    @Override
    public void accept(GeometriaPainter gp) {
        gp.stampa(this);
    }
}
```

Codice A.31: NodoLatLng

```
package it.dsantini.areediattesa.model.impl.googleMaps;

import android.support.annotation.NonNull;

import com.google.android.gms.maps.model.LatLng;

import it.dsantini.areediattesa.model.impl.NodoBase;

public class NodoLatLng extends NodoBase {
    public NodoLatLng(@NonNull LatLng ll) {
        super(ll.latitude, ll.longitude);
    }
}
```

TipoEmergenza

Codice A.32: TipoEmergenza

```
package it.dsantini.areediattesa.model;
```

```java
import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;

import java.util.Optional;

public interface Fotografia {
    /**
     * @return Immagine
     */
    @NonNull
    Drawable getImmagine();

    /**
     * @return Titolo della fotografia
     */
    @NonNull
    Optional<String> getTitolo();
}
```

Codice A.33: FotografiaBase

```java
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.TipoEmergenza;

@Entity
public class TipoEmergenzaBase implements TipoEmergenza {
    public static final TipoEmergenza TERREMOTO = new TipoEmergenzaBase("Terremoto")
        ,
            ALLUVIONE = new TipoEmergenzaBase("Alluvione"),
            INCENDIO = new TipoEmergenzaBase("Incendio");

    @PrimaryKey
    @NonNull
    private final String nome;

    public TipoEmergenzaBase(@NonNull String nome) {
        if (nome == null)
            throw new IllegalArgumentException("nome == null");

        if (nome.isEmpty())
            throw new IllegalArgumentException("nome vuoto");

        this.nome = nome;
    }


    @Override
    @NonNull
    public String getNome() {
        return nome;
    }
}
```

AreaEmergenza

Codice A.34: AreaEmergenza

```
package it.dsantini.areediattesa.model;

import android.support.annotation.NonNull;

import java.util.Optional;

public interface AreaEmergenza {
    /**
     * @return Descrizione dell'area
     */
    @NonNull
    Optional<String> getDescrizione();

    /**
     * @return Tipi di emergenza per cui Ã¨ pensata l'area
     */
    @NonNull
    TipoEmergenza[] getEmergenze();

    /**
     * @return Foto dell'area
     */
    @NonNull
    Fotografia[] getFotografie();

    /**
     * @return Geometria geografica dell'area
     */
    @NonNull
    Geometria getGeometria();

    /**
     * @return Ente gestore dell'area
     */
    @NonNull
    Optional<String> getGestore();

    /**
     * @return Nome dell'area
     */
    @NonNull
    String getNome();
}
```

Codice A.35: AreaEmergenzaBase

```
package it.dsantini.areediattesa.model.impl;

import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import java.util.Optional;

import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Fotografia;
import it.dsantini.areediattesa.model.Geometria;
```

```java
import it.dsantini.areediattesa.model.TipoEmergenza;

public class AreaEmergenzaBase implements AreaEmergenza {
    @NonNull
    private final String nome;
    @Nullable
    private final String descrizione, gestore;
    @NonNull
    private final TipoEmergenza[] emergenze;
    @NonNull
    private final Fotografia[] fotografie;
    @NonNull
    private final Geometria geometria;

    public AreaEmergenzaBase(@NonNull String nome,
                             @Nullable String descrizione,
                             @Nullable String gestore,
                             @NonNull TipoEmergenza[] emergenze,
                             @NonNull Fotografia[] fotografie,
                             @NonNull Geometria geometria) {
        if (nome == null)
            throw new IllegalArgumentException("nome == null");

        if (emergenze == null)
            throw new IllegalArgumentException("emergenze == null");

        for (int i = 0; i < emergenze.length; i++)
            if (emergenze[i] == null)
                throw new IllegalArgumentException("emergenze[" + i + "] == null");

        if (nome == null)
            throw new IllegalArgumentException("fotografie == null");

        for (int i = 0; i < fotografie.length; i++)
            if (fotografie[i] == null)
                throw new IllegalArgumentException("fotografie[" + i + "] == null");

        for (int i = 0; i < fotografie.length; i++)
            for (int j = 0; j < fotografie.length; j++)
                if (i != j && fotografie[i] == fotografie[j])
                    throw new IllegalArgumentException("fotografie[" + i + "] ==
                        fotografie[" + j + "]");

        if (geometria == null)
            throw new IllegalArgumentException("geometria == null");

        this.nome = nome;
        this.descrizione = descrizione;
        this.gestore = gestore;
        this.emergenze = emergenze;
        this.fotografie = fotografie;
        this.geometria = geometria;
    }

    @NonNull
    @Override
    public Optional<String> getDescrizione() {
        if (descrizione == null)
            return Optional.empty();
```

```
        else
            return Optional.of(descrizione);
    }

    @NonNull
    @Override
    public TipoEmergenza[] getEmergenze() {
        return emergenze;
    }

    @NonNull
    @Override
    public Fotografia[] getFotografie() {
        return fotografie;
    }

    @NonNull
    @Override
    public Geometria getGeometria() {
        return geometria;
    }

    @NonNull
    @Override
    public Optional<String> getGestore() {
        if (gestore == null)
            return Optional.empty();
        else
            return Optional.of(gestore);
    }

    @NonNull
    @Override
    public String getNome() {
        return nome;
    }
}
```

Codice A.36: AreaAttesa

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.arch.persistence.room.Ignore;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import java.util.Optional;

import it.dsantini.areediattesa.model.Fotografia;
import it.dsantini.areediattesa.model.Geometria;
import it.dsantini.areediattesa.model.TipoEmergenza;

@Entity
public class AreaAttesa extends AreaEmergenzaBase {
    @PrimaryKey
    private final int id;
    public static final int DEFAULT_ID = Integer.MIN_VALUE;
```

```
    @Nullable
    private final Integer capienza;
    @Nullable
    private final Boolean alCoperto;

    public AreaAttesa(int id,
                      @NonNull String nome,
                      @Nullable String descrizione,
                      @Nullable String gestore,
                      @NonNull TipoEmergenza[] emergenze,
                      @NonNull Fotografia[] fotografie,
                      @NonNull Geometria geometria,
                      @Nullable Integer capienza,
                      @Nullable Boolean alCoperto) {
        super(nome, descrizione, gestore, emergenze, fotografie, geometria);

        if (capienza != null && capienza <= 0)
            throw new IllegalArgumentException("capienza <= 0");

        this.id = id;
        this.capienza = capienza;
        this.alCoperto = alCoperto;
    }

    @Ignore
    public AreaAttesa(@NonNull String nome,
                      @Nullable String descrizione,
                      @Nullable String gestore,
                      @NonNull TipoEmergenza[] emergenze,
                      @NonNull Fotografia[] fotografie,
                      @NonNull Geometria geometria,
                      @Nullable Integer capienza,
                      @Nullable Boolean alCoperto) {
        this(DEFAULT_ID, nome, descrizione, gestore, emergenze, fotografie,
            geometria, capienza, alCoperto);
    }

    public int getId() {
        return id;
    }

    @NonNull
    public Optional<Integer> getCapienza() {
        return Optional.ofNullable(capienza);
    }

    @NonNull
    public Optional<Boolean> isAlCoperto() {
        return Optional.ofNullable(alCoperto);
    }
}
```

Confine

Codice A.37: Confine

```
package it.dsantini.areediattesa.model;

import android.support.annotation.NonNull;
```

```java
import java.util.Optional;

import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.SorgenteAree;
import it.dsantini.areediattesa.view.helper.ConfinePainter;

public interface Confine {
    /**
     * @return Nome del confine
     */
    @NonNull
    Optional<String> getNome();

    /**
     * Pattern Visitor per il confronto fra confini (funzione accept)
     * Implementazione tipica: return contenuto.isContenuto(this)
     *
     * @param contenuto Il confine da confronatare
     * @return Se il confine specificato Ã¨ contenuto in questo
     */
    boolean contiene(@NonNull Confine contenuto);

    /**
     * Pattern Visitor per il confronto fra confini (funzione visit)
     *
     * @param contenente Il confine da confrontare
     * @return Se il confine specificato contiene questo
     */
    boolean isContenuto(@NonNull BoundingBox contenente);

    /**
     * Pattern Visitor per download gruppo di aree dato il confine (funzione accept)
     * Implementazione tipica: return s.visit(this)
     *
     * @param s Sorgente da cui scaricare il gruppo di aree
     * @return Gruppo di aree scaricato
     */
    @NonNull
    GruppoAree accept(@NonNull SorgenteAree s) throws DownloadException;

    /**
     * Pattern Visitor per la stampa del confine (funzione accept)
     * Implementazione tipica: return cp.stampa(this)
     *
     * @param cp Painter con cui stampare il confine
     */
    void accept(@NonNull ConfinePainter cp);
}
```

Codice A.38: BoundingBox

```java
package it.dsantini.areediattesa.model;

public interface BoundingBox extends Confine {
    /**
     * @return Latitudine minima (limite sud)
     */
    double getLatMin();
```

```
    /**
     * @return Latitudine massima (limite nord)
     */
    double getLatMax();

    /**
     * @return Longitudine minima (limite ovest)
     */
    double getLonMin();

    /**
     * @return Longitudine massima (limite est)
     */
    double getLonMax();
}
```

Codice A.39: ConfineBase

```
package it.dsantini.areediattesa.model.impl;

import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import java.util.Optional;

import it.dsantini.areediattesa.model.Confine;

public abstract class ConfineBase implements Confine {
    @Nullable
    private final String nome;

    ConfineBase(@Nullable String nome) {
        this.nome = nome;
    }

    @Override
    @NonNull
    public Optional<String> getNome() {
        return Optional.ofNullable(nome);
    }
}
```

Codice A.40: BoundingBoxBase

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import it.dsantini.areediattesa.model.BoundingBox;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.SorgenteAree;
import it.dsantini.areediattesa.view.helper.ConfinePainter;

@Entity(primaryKeys = {"latMin", "latMax", "lonMin", "lonMax"})
public class BoundingBoxBase extends ConfineBase implements BoundingBox {
```

```
    private final double latMin, latMax, lonMin, lonMax;

    /**
     * @param latMin Limite sud
     * @param latMax Limite nord
     * @param lonMin Limite est
     * @param lonMax Limite ovest
     * @param nome Nome del confine
     */
    public BoundingBoxBase(double latMin, double latMax, double lonMin, double
        lonMax, @Nullable String nome) {
        super(nome);

        //Log.v(TAG, latMin + "/" + latMin + "/" + latMax + "/" + lonMin + "/" +
            lonMax);

        if (latMin >= latMax)
            throw new IllegalArgumentException("latMin = " + latMin + " >= latMax = "
                + latMax);
        if (lonMin == lonMax)
            throw new IllegalArgumentException("lonMin == lonMax = " + lonMin);
        if (latMin < -90)
            throw new IllegalArgumentException("latMin = " + latMin + " < -90");
        if (latMax > 90)
            throw new IllegalArgumentException("latMax = " + latMax + " > +90");
        if (lonMin < -180)
            throw new IllegalArgumentException("lonMin = " + lonMin + " < -180");
        if (lonMax > 180)
            throw new IllegalArgumentException("lonMax = " + lonMax + " > +180");

        Double LatMin = latMin, LatMax = latMax, LonMin = lonMin, LonMax = lonMax;

        if (LatMin.isNaN())
            throw new IllegalArgumentException("latMin == NaN");
        if (LatMin.isInfinite())
            throw new IllegalArgumentException("latMin == infinito");
        if (LatMax.isNaN())
            throw new IllegalArgumentException("latMax == NaN");
        if (LatMax.isInfinite())
            throw new IllegalArgumentException("latMax == infinito");
        if (LonMin.isNaN())
            throw new IllegalArgumentException("lonMin == NaN");
        if (LonMin.isInfinite())
            throw new IllegalArgumentException("lonMin == infinito");
        if (LonMax.isNaN())
            throw new IllegalArgumentException("lonMax == NaN");
        if (LonMax.isInfinite())
            throw new IllegalArgumentException("lonMax == infinito");


        this.latMin = latMin;
        this.latMax = latMax;
        this.lonMin = lonMin;
        this.lonMax = lonMax;
    }

    @Override
    public double getLatMin() {
        return latMin;
```

```
    }

    @Override
    public double getLatMax() {
        return latMax;
    }

    @Override
    public double getLonMin() {
        return lonMin;
    }

    @Override
    public double getLonMax() {
        return lonMax;
    }

    @Override
    @NonNull
    public GruppoAree accept(@NonNull SorgenteAree s) throws DownloadException {
        if (s == null)
            throw new IllegalArgumentException("SorgenteAree s == null");

        return s.visit(this);
    }

    @Override
    public void accept(@NonNull ConfinePainter cp) {
        cp.stampa(this);
    }

    @Override
    public boolean contiene(@NonNull Confine contenuto) {
        if (contenuto == null)
            throw new IllegalArgumentException("contenuto == null");
        if (contenuto == this)
            return true;

        return contenuto.isContenuto(this);
    }

    @Override
    public boolean isContenuto(@NonNull BoundingBox contenente) {
        if (contenente == this)
            return true;

        if (contenente == null)
            throw new IllegalArgumentException("contenente == null");

        boolean contenutoSemplice =
                this.getLatMin() >= contenente.getLatMin() &&
                        this.getLatMax() <= contenente.getLatMax() &&
                        this.getLonMin() >= contenente.getLonMin() &&
                        this.getLonMax() <= contenente.getLonMax(),
                contenutoACavallo = false; // TODO se i bounding box sono a cavallo
                    del meridiano opposto a Greenwich

        return contenutoSemplice || contenutoACavallo;
    }
```

```
    @Override
    public String toString() {
        return "BoundingBoxBase, latMin=" + getLatMin() + ", latMax=" + getLatMax()
            + ", lonMin=" + getLonMin() + ", lonMax=" + getLonMax();
    }
}
```

Codice A.41: BoundingBoxLatLngBounds

```
package it.dsantini.areediattesa.model.impl.googleMaps;

import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import com.google.android.gms.maps.model.LatLngBounds;

import it.dsantini.areediattesa.model.impl.BoundingBoxBase;

public class BoundingBoxLatLngBounds extends BoundingBoxBase {

    public BoundingBoxLatLngBounds(@NonNull LatLngBounds bounds, @Nullable String
        nome) {
        super(bounds.southwest.latitude, bounds.northeast.latitude, bounds.southwest
            .longitude, bounds.northeast.longitude, nome);
    }
}
```

GruppoAree

Codice A.42: GruppoAree

```
package it.dsantini.areediattesa.model;

import android.support.annotation.NonNull;

import java.time.LocalDateTime;

public interface GruppoAree {
    /**
     * @return Aree di emergenza contenute nel gruppo di aree
     */
    @NonNull
    AreaEmergenza[] getAree();

    /**
     * @return Confine a cui appartengono le aree
     */
    @NonNull
    Confine getConfine();

    /**
     * @return Se il gruppo Ã¨ fra i preferiti dell'utente
     */
    boolean isPreferito();

    /**
     * @param p nuovo valore di preferito
     * @return Clone del gruppo, con preferito modificato
     */
```

```
    @NonNull
    GruppoAree withPreferito(boolean p);

    /**
     * @return Momento in cui Ã¨ stato scaricato il gruppo di aree
     */
    @NonNull
    LocalDateTime getTimestamp();
}
```

Codice A.43: GruppoAreeBase

```
package it.dsantini.areediattesa.model.impl;

import android.arch.persistence.room.Entity;
import android.arch.persistence.room.Ignore;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;

import java.time.LocalDateTime;

import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;

@Entity
public class GruppoAreeBase implements GruppoAree {
    @PrimaryKey
    private final int id;
    public static final int DEFAULT_ID = Integer.MIN_VALUE;

    @NonNull
    private final AreaEmergenza[] aree;

    @NonNull
    private final Confine confine;

    private final boolean preferito;

    @NonNull
    private final LocalDateTime timestamp;

    public GruppoAreeBase(int id,
                          @NonNull AreaEmergenza[] aree,
                          @NonNull Confine confine,
                          boolean preferito,
                          @NonNull LocalDateTime timestamp) {
        if (aree == null)
            throw new IllegalArgumentException("Aree nullo");
        if (confine == null)
            throw new IllegalArgumentException("Confine nullo");
        if (timestamp == null)
            throw new IllegalArgumentException("Timestamp nullo");

        for (int i = 0; i < aree.length; i++) {
            if (aree[i] == null)
                throw new IllegalArgumentException("aree[" + i + "] nullo");

            for (int j = 0; j < aree.length; j++)
```

```
                if (i != j && aree[i] == aree[j])
                    throw new IllegalArgumentException("aree[" + i + "] uguale a
                        aree[" + j + "]");
        }

        this.id = id;
        this.aree = aree;
        this.confine = confine;
        this.preferito = preferito;
        this.timestamp = timestamp;
    }

    @Ignore
    public GruppoAreeBase(@NonNull AreaEmergenza[] aree,
                          @NonNull Confine confine,
                          boolean preferito,
                          @NonNull LocalDateTime timestamp) {
        this(DEFAULT_ID, aree, confine, preferito, timestamp);

    }

    public int getId() {
        return id;
    }

    @Override
    @NonNull
    public AreaEmergenza[] getAree() {
        return aree;
    }

    @Override
    @NonNull
    public Confine getConfine() {
        return confine;
    }

    @Override
    public boolean isPreferito() {
        return preferito;
    }

    @Override
    @NonNull
    public LocalDateTime getTimestamp() {
        return timestamp;
    }

    @NonNull
    @Override
    public GruppoAree withPreferito(boolean p) {
        if (isPreferito() == p)
            return this;
        else
            return new GruppoAreeBase(getAree(), getConfine(), p, getTimestamp());
    }

    @Override
    public String toString() {
```

```
        StringBuilder b = new StringBuilder();
        b.append("GruppoAreeBase con ");
        b.append(getAree().length);
        b.append(" aree del ");
        b.append(getTimestamp());
        if (isPreferito())
            b.append(", preferito");
        return b.toString();
    }
}
```

Mappa

Codice A.44: Mappa

```
package it.dsantini.areediattesa.model;

import android.support.annotation.NonNull;

public interface Mappa {
    /**
     * @return Gruppi di aree di attesa contenuti nella mappa
     */
    @NonNull
    GruppoAree[] getGruppi();

    /**
     * @param g Gruppo di aree da aggiungere alla mappa
     * @return Nuova Mappa con il gruppo indicato
     */
    @NonNull
    Mappa con(@NonNull GruppoAree g);

    /**
     * @param g Gruppo di aree da rimuovere dalla mappa
     * @return Nuova Mappa senza il gruppo indicato
     */
    @NonNull
    Mappa senza(@NonNull GruppoAree g);
}
```

Codice A.45: MappaBase

```
package it.dsantini.areediattesa.model.impl;

import android.support.annotation.NonNull;
import android.util.Log;

import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;

public class MappaBase implements Mappa {
    private static final String TAG = "MappaBase";

    @NonNull
    private final GruppoAree[] gruppi;

    public MappaBase(@NonNull GruppoAree[] gruppi) {
        if (gruppi == null)
            throw new IllegalArgumentException("gruppi == null");
```

```
        for (int i = 0; i < gruppi.length; i++)
            if (gruppi[i] == null)
                throw new IllegalArgumentException("gruppi[" + i + "] == null");

        for (int i = 0; i < gruppi.length; i++)
            for (int j = 0; i < gruppi.length; i++)
                if (i != j && gruppi[i] == gruppi[j])
                    throw new IllegalArgumentException("gruppi[" + i + "] == gruppi[
                        " + j + "]");

        this.gruppi = gruppi;
    }

    @Override
    @NonNull
    public GruppoAree[] getGruppi() {
        return gruppi;
    }

    @NonNull
    @Override
    public Mappa con(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        GruppoAree[] nuovo = new GruppoAree[gruppi.length + 1];
        System.arraycopy(gruppi, 0, nuovo, 0, gruppi.length);
        nuovo[gruppi.length] = g;
        return new MappaBase(nuovo);
    }

    @NonNull
    @Override
    public Mappa senza(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        int pos = -1;
        for (int i = 0; i < gruppi.length; i++) {
            if (gruppi[i] == g) {
                pos = i;
                break;
            }
        }

        if (pos < 0) { // Gruppo non presente
            Log.e(TAG, "Gruppo da eliminare non presente");
            return this;
        } else {
            GruppoAree[] nuovo = new GruppoAree[gruppi.length - 1];
            System.arraycopy(gruppi, 0, nuovo, 0, pos);
            if (gruppi.length - 1 - pos > 0)
                System.arraycopy(gruppi, pos + 1, nuovo, pos, gruppi.length - 1 -
                    pos);
            return new MappaBase(nuovo);
        }
    }
```

```
    @Override
    public String toString() {
        return "Mappa con " + getGruppi().length + " gruppi";
    }
}
```

A.3.2 Persistenza

Codice A.46: Persistenza

```
package it.dsantini.areediattesa.persistenza;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Mappa;

public interface Persistenza {
    /**
     * @param m Mappa da salvare nell'archiviazione
     */
    void salva(@NonNull Mappa m);

    /**
     * @return Mappa caricata dall'archiviazione
     */
    @NonNull
    Mappa carica();
}
```

Codice A.47: PersistenzaRoom

```
package it.dsantini.areediattesa.persistenza.impl;

import android.arch.persistence.room.Room;
import android.content.Context;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.model.impl.MappaBase;
import it.dsantini.areediattesa.persistenza.Persistenza;

public class PersistenzaRoom implements Persistenza {
    @NonNull
    private final MappaDatabase db;

    public PersistenzaRoom(@NonNull Context c) {
        db = Room.databaseBuilder(c, MappaDatabase.class, "aree-di-attesa").build();
    }

    @Override
    public void salva(@NonNull Mappa m) {
        throw new UnsupportedOperationException(); // TODO
    }

    @NonNull
    @Override
    public Mappa carica() {
        GruppoAreeDao gad = db.gruppoAreeDao();
        Mappa mappa = new MappaBase(gad.loadAllGruppi());
        return mappa;
```

```
        //throw new UnsupportedOperationException(); // TODO
    }
}
```

Codice A.48: PersistenzaFallback

```
package it.dsantini.areediattesa.persistenza.mock;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.model.impl.MappaBase;
import it.dsantini.areediattesa.persistenza.Persistenza;

public class PersistenzaFallback implements Persistenza {
    @Override
    public void salva(@NonNull Mappa m) {

    }

    @NonNull
    @Override
    public Mappa carica() {
        return new MappaBase(new GruppoAree[0]);
    }
}
```

Per tornare al testo: 4.2.1

A.3.3 Data layer

Codice A.49: Stato

```
package it.dsantini.areediattesa.data;

public enum Stato {
    non_caricato,
    caricamento,
    errore_caricamento,
    caricato,
    salvataggio,
    errore_salvataggio;
}
```

MappaRepository

Codice A.50: MappaRepository

```
package it.dsantini.areediattesa.data;

import android.arch.lifecycle.LiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Mappa;

public interface MappaRepository {
    /**
     * @return Mappa osservabile
     */
```

```
    @NonNull
    LiveData<Mappa> getMappa();

    /**
     * @return Stato osservabile della mappa
     */
    @NonNull
    LiveData<Stato> getStato();

    /**
     * @param m Nuova mappa da impostare
     */
    void setMappa(Mappa m);

    /** Imposta la mappa da un altro thread
     * @param m Nuova mappa da impostare
     */
    void postMappa(Mappa m);

    /**
     * Salva la mappa nella persistenza
     */
    void salvaMappa();
}
```

Codice A.51: MappaRepositoryBase

```
package it.dsantini.areediattesa.data.impl;

import android.arch.lifecycle.LiveData;
import android.arch.lifecycle.MutableLiveData;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;

import it.dsantini.areediattesa.data.MappaRepository;
import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.model.impl.MappaBase;
import it.dsantini.areediattesa.persistenza.Persistenza;

public class MappaRepositoryBase implements MappaRepository {
    private static MappaRepositoryBase instance;

    @NonNull
    private final MutableLiveData<Mappa> mappa;
    @NonNull
    private final MutableLiveData<Stato> stato;

    @Nullable
    private Persistenza persistenza;

    private MappaRepositoryBase() {
        mappa = new MutableLiveData<>();
        stato = new MutableLiveData<>();
        stato.setValue(Stato.non_caricato);
    }
```

```
    public static MappaRepositoryBase getInstance() {
        if (instance == null)
            instance = new MappaRepositoryBase();

        return instance;
    }

    public void setPersistenza(@NonNull Persistenza p) {
        if (p == null)
            throw new IllegalArgumentException("Persistenza p null");

        this.persistenza = p;

        stato.setValue(Stato.caricamento);
        new CaricamentoAsincrono().start();
    }

    @NonNull
    @Override
    public LiveData<Mappa> getMappa() {
        return mappa;
    }

    @NonNull
    @Override
    public LiveData<Stato> getStato() {
        return stato;
    }

    @Override
    public void setMappa(Mappa m) {
        mappa.setValue(m);
    }

    @Override
    public void postMappa(Mappa m) {
        mappa.postValue(m);
    }

    @Override
    public void salvaMappa() {
        stato.setValue(Stato.salvataggio);
        new SalvataggioAsincrono().start();
    }

    private class CaricamentoAsincrono extends Thread {
        CaricamentoAsincrono() {
            super(new Runnable() {
                @Override
                public void run() {
                    final String TAG = "CaricamentoAsincrono::target::run";

                    if (persistenza == null)
                        Log.w(TAG, "Persistenza nulla");
                    else {
                        try {
                            Log.v(TAG, "Caricamento in corso");
                            mappa.postValue(persistenza.carica());
                            stato.postValue(Stato.caricato);
```

```
                        } catch (Exception e) {
                            Log.e(TAG, "Errore nel caricamento dei gruppi", e);
                            stato.postValue(Stato.errore_caricamento);
                            mappa.postValue(new MappaBase(new GruppoAree[0]));
                        }
                    }
                }
            });
        }
    }

    private class SalvataggioAsincrono extends Thread {
        SalvataggioAsincrono() {
            super(new Runnable() {
                @Override
                public void run() {
                    final String TAG = "SalvataggioAsincrono::target::run";

                    if (mappa.getValue() == null || persistenza == null)
                        Log.w(TAG, "Mappa o Persistenza nulla");
                    else {
                        try {
                            Log.v(TAG, "Salvataggio in corso");
                            // TODO In questo frangente il thread principale
                                potrebbe modifcare mappa.getValue() come null
                            persistenza.salva(mappa.getValue());
                            stato.postValue(Stato.caricato);
                        } catch (Exception e) {
                            Log.e(TAG, "Errore nel salvataggio", e);
                            stato.postValue(Stato.errore_salvataggio);
                        }
                    }
                }
            });
        }
    }
}
```

DettagliRepository

Codice A.52: AreaRepository

```
package it.dsantini.areediattesa.data;

import android.arch.lifecycle.LiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.AreaEmergenza;

public interface DettagliRepository {
    /**
     * @return Area di emergenza selezionata
     */
    @NonNull
    LiveData<AreaEmergenza> getArea();

    /**
     * @param a Area di emergenza da selezionare
     */
    void setArea(AreaEmergenza a);
```

```
}
```

Codice A.53: AreaRepositoryBase

```
package it.dsantini.areediattesa.data.impl;

import android.arch.lifecycle.LiveData;
import android.arch.lifecycle.MutableLiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.data.DettagliRepository;
import it.dsantini.areediattesa.model.AreaEmergenza;

public class DettagliRepositoryBase implements DettagliRepository {
    private static DettagliRepositoryBase instance;

    @NonNull
    private final MutableLiveData<AreaEmergenza> area;

    private DettagliRepositoryBase() {
        area = new MutableLiveData<>();
    }

    public static DettagliRepositoryBase getInstance() {
        if (instance == null)
            instance = new DettagliRepositoryBase();

        return instance;
    }

    @NonNull
    @Override
    public LiveData<AreaEmergenza> getArea() {
        return area;
    }

    @Override
    public void setArea(AreaEmergenza a) {
        area.setValue(a);
    }
}
```

Per tornare al testo: 4.2.2

A.3.4 Rete

Codice A.54: DownloadException

```
package it.dsantini.areediattesa.rete;

import android.support.annotation.NonNull;

public class DownloadException extends Exception {
    public DownloadException(int httpCode) {
        super("Codice HTTP " + httpCode);
    }

    public DownloadException(@NonNull String spiegazione, @NonNull Exception e){
        super(spiegazione,e);
    }
}
```

Codice A.55: SorgenteAree

```
package it.dsantini.areediattesa.rete;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.BoundingBox;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;

public interface SorgenteAree {
    /**
     * @param c Confine di cui scaricare le aree
     * @return Gruppo di aree con il confine specificato
     */
    @NonNull
    GruppoAree download(@NonNull Confine c) throws DownloadException;

    /**
     * Pattern Visitor per download gruppo di aree dato il confine (funzione visit)
     *
     * @param b Confine di cui scaricare le aree
     * @return Gruppo di aree con il confine specificato
     */
    @NonNull
    GruppoAree visit(@NonNull BoundingBox b) throws DownloadException;
}
```

Codice A.56: SorgenteAreeBase

```
package it.dsantini.areediattesa.rete.impl;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.SorgenteAree;

public abstract class SorgenteAreeBase implements SorgenteAree {
    @Override
    @NonNull
    public GruppoAree download(@NonNull Confine c) throws DownloadException {
        if (c == null)
            throw new IllegalArgumentException("c == null");

        return c.accept(this);
    }
}
```

Codice A.57: SorgenteAreeFallback

```
package it.dsantini.areediattesa.rete.mock;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.BoundingBox;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.impl.SorgenteAreeBase;
```

```
public class SorgenteAreeFallback extends SorgenteAreeBase {

    @NonNull
    @Override
    public GruppoAree visit(@NonNull BoundingBox b) throws DownloadException {
        throw new UnsupportedOperationException("Download non implementato (Sorgente
            aree fallback)");
    }
}
```

Retrofit + Overpass

Codice A.58: SorgenteAreeRetrofit.java

```
package it.dsantini.areediattesa.rete.impl;

import android.support.annotation.NonNull;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public abstract class SorgenteAreeRetrofit extends SorgenteAreeBase {
    @NonNull
    private final String endpoint;

    SorgenteAreeRetrofit(@NonNull String endpoint) {
        if (endpoint == null)
            throw new IllegalArgumentException("endpoint null");

        if (endpoint.isEmpty())
            throw new IllegalArgumentException("endpoint vuoto");

        this.endpoint = endpoint;
    }

    protected Retrofit getRetrofit() {
        return new Retrofit.Builder()
                .baseUrl(endpoint)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }
}
```

Codice A.59: SorgenteAreeOverpass.java

```
package it.dsantini.areediattesa.rete.impl;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;

import com.google.gson.JsonParseException;

import java.io.IOException;
import java.io.InputStream;
```

```
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Locale;
import java.util.Set;

import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.BoundingBox;
import it.dsantini.areediattesa.model.Fotografia;
import it.dsantini.areediattesa.model.Geometria;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.model.TipoEmergenza;
import it.dsantini.areediattesa.model.impl.AreaAttesa;
import it.dsantini.areediattesa.model.impl.FotografiaBase;
import it.dsantini.areediattesa.model.impl.GruppoAreeBase;
import it.dsantini.areediattesa.model.impl.NodoBase;
import it.dsantini.areediattesa.model.impl.Poligono;
import it.dsantini.areediattesa.model.impl.TipoEmergenzaBase;
import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.impl.overpass.Element;
import it.dsantini.areediattesa.rete.impl.overpass.OverpassData;
import it.dsantini.areediattesa.rete.impl.overpass.OverpassInterface;
import it.dsantini.areediattesa.rete.impl.overpass.Tags;
import retrofit2.Call;
import retrofit2.Response;

public class SorgenteAreeOverpass extends SorgenteAreeRetrofit {
    private static final String ENDPOINT_DEFAULT = "https://overpass-api.de/api/";

    public SorgenteAreeOverpass(@NonNull String endpoint) {
        super(endpoint);
    }

    public SorgenteAreeOverpass() {
        this(ENDPOINT_DEFAULT);
    }

    @Override
    @NonNull
    public GruppoAree visit(@NonNull BoundingBox b) throws DownloadException {
        final String TAG = "SorgenteAreeOverpass::visit";

        OverpassInterface s = getRetrofit().create(OverpassInterface.class);
        // [out:json][timeout:25]; (node[emergency=assembly_point]({s},{w},{n},{e});
            way[emergency=assembly_point]({s},{w},{n},{e}); ); out body; >; out skel
            qt;
        String bbox = String.format(Locale.US, "%f,%f,%f,%f", b.getLatMin(), b.
            getLonMin(), b.getLatMax(), b.getLonMax()),
                query = "[out:json][timeout:15];(node[emergency=assembly_point]("
                        + bbox
                        + ");way[emergency=assembly_point]("
                        + bbox
```

```
                    + "); ); out body qt; >; out skel qt;";
        Log.v(TAG, query);
        Call<OverpassData> call = s.jsonQuery(query);
        Response<OverpassData> response;

        try {
            response = call.execute();
        } catch (SocketTimeoutException e) {
            throw new DownloadException("Timeout della chiamata raggiunto", e);
        } catch (IOException e) {
            throw new DownloadException("Errore di I/O durante la chiamata per il
                download", e);
        } catch (JsonParseException e) {
            throw new DownloadException("Errore nell'interpretazione dei dati
                ricevuti", e);
        } catch (Exception e) {
            throw new DownloadException("Errore generico durante il download", e);
        }

        if (response.code() != 200)
            throw new DownloadException(response.code());

        AreaEmergenza[] aree;

        if (response.body() == null) {
            aree = new AreaEmergenza[0];
            Log.w(TAG, "Corpo risposta nullo");
        } else
            aree = getAree(response.body().getElements());

        return new GruppoAreeBase(aree, b, false, LocalDateTime.now());
    }

    @NonNull
    private AreaEmergenza[] getAree(@NonNull Collection<Element> elements) {
        final String TAG = "SorgenteAreeOverpass::getAree";

        Set<AreaEmergenza> aree = new HashSet<>();

        for (Element e : elements) {
            Tags t = e.getTags();
            if (t != null && t.getEmergency() != null && t.getEmergency().equals("
                assembly_point")) {
                Geometria geom = getGeometria(e, elements);
                if (geom != null) {
                    String name = e.getTags().getName();
                    if (name == null)
                        name = "";

                    TipoEmergenza[] emergenze = getTipiEmergenza(t);

                    Fotografia[] foto = getFotografie(t);

                    Integer capienza = null;
                    if (t.getCapacity() != null) {
                        try {
                            capienza = Integer.parseInt(t.getCapacity());
                        } catch (NumberFormatException ex) {
                            Log.e(TAG, "Errore nel parsing della capienza: " + t.
```

```
                            getCapacity(), ex);
                    }
                }
                Boolean coperto = null;
                if (t.getCovered() != null)
                    coperto = t.getCovered().equals("yes");

                try {
                    AreaEmergenza area = new AreaAttesa(name,
                            t.getDescription(),
                            t.getOperator(),
                            emergenze,
                            foto,
                            geom,
                            capienza,
                            coperto);
                    aree.add(area);
                } catch (IllegalArgumentException ex) {
                    Log.e(TAG, "Creazione AreaAttesa fallita", ex);
                }
            }
        }
    }

    return aree.toArray(new AreaEmergenza[0]);
}

@Nullable
private Geometria getGeometria(@NonNull Element e, @NonNull Collection<Element>
    elements) {
    final String TAG = "SorgenteAreeOverpass::getGeometria";

    Geometria geom = null;
    switch (e.getType()) {
        case "node": // Nodo
            geom = new NodoBase(e.getLat(), e.getLon());
            break;
        case "way":
            List<Long> nodi = e.getNodes();
            if (!nodi.get(0).equals(nodi.get(nodi.size() - 1))) // Ultimo nodo
                diverso dal primo -> Linea
                Log.e(TAG, "Trovata una linea, la scarto");
            else { // Ultimo nodo uguale al primo -> Poligono
                List<Nodo> vertici = new ArrayList<>();
                for (int i = 0; i < e.getNodes().size() - 1; i++) {
                    Long id = e.getNodes().get(i);
                    for (Element nodo : elements)
                        if (nodo.getId().equals(id) && nodo.getType().equals("
                            node"))
                            vertici.add(new NodoBase(nodo.getLat(), nodo.getLon
                                ()));
                }

                geom = new Poligono(vertici.toArray(new Nodo[0]));
            }
            break;
        default:
            Log.e(TAG, "Tipo sconosciuto trovato: " + e.getType());
    }
```

```
        return geom;
    }

    @NonNull
    private TipoEmergenza[] getTipiEmergenza(@NonNull Tags t) {
        Set<TipoEmergenza> emergenze = new HashSet<>();

        if (t.getAssemblyPointEarthquake() != null
                && t.getAssemblyPointEarthquake().equals("yes"))
            emergenze.add(TipoEmergenzaBase.TERREMOTO);
        if (t.getAssemblyPointFire() != null
                && t.getAssemblyPointFire().equals("yes"))
            emergenze.add(TipoEmergenzaBase.INCENDIO);
        if (t.getAssemblyPointFlood() != null
                && t.getAssemblyPointFlood().equals("yes"))
            emergenze.add(TipoEmergenzaBase.ALLUVIONE);
        //TODO altri tipi

        return emergenze.toArray(new TipoEmergenza[0]);
    }

    @NonNull
    private Fotografia[] getFotografie(@NonNull Tags t) {
        final String TAG = "SorgenteAreeOverpass::getFotografie";
        Set<Fotografia> foto = new HashSet<>();

        String image = t.getImage();
        if (image != null && (image.endsWith("jpg")
                || image.endsWith("jpeg")
                || image.endsWith("png"))) {
            try {
                URL url = new URL(image);
                HttpURLConnection connection = (HttpURLConnection) url.
                    openConnection();
                connection.setDoInput(true);
                connection.connect();
                InputStream in = connection.getInputStream();
                Bitmap bitmap = BitmapFactory.decodeStream(in);
                in.close();
                Drawable drawable = new BitmapDrawable(bitmap);
                foto.add(new FotografiaBase(drawable, null));

//                File directory= Environment.getExternalStorageDirectory();
//                String fileName = image.substring(image.lastIndexOf('/') + 1);
//                File imageFile = new File(directory, fileName);
//                FileOutputStream out = new FileOutputStream(imageFile);
//                bitmap.compress(Bitmap.CompressFormat.PNG, 50, out);
//                out.flush();
//                out.close();
            } catch (MalformedURLException e) {
                Log.e(TAG, "URL image malformato", e);
            } catch (IOException e) {
                Log.e(TAG, "Connessione fallita per la fotografia " + image, e);
            } catch (Exception e) {
                Log.e(TAG, "Download fallito per l'immagine " + image, e);
            }
        }

        // TODO wikimedia_commons=*
```

```
        return foto.toArray(new Fotografia[0]);
    }
}
```

Codice A.60: OverpassInterface.java

```
package it.dsantini.areediattesa.rete.impl.overpass;

import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Headers;
import retrofit2.http.Query;

public interface OverpassInterface {
    @GET("interpreter")
    @Headers("Accept:application/json")
    Call<OverpassData> jsonQuery(@Query("data") String data);
}
```

Codice A.61: OverpassData.java

```
package it.dsantini.areediattesa.rete.impl.overpass;

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

import java.util.List;

public class OverpassData {

    @SerializedName("version")
    @Expose
    private Double version;
    @SerializedName("generator")
    @Expose
    private String generator;
    @SerializedName("osm3s")
    @Expose
    private Osm3s osm3s;
    @SerializedName("elements")
    @Expose
    private List<Element> elements = null;

    public Double getVersion() {
        return version;
    }

    public void setVersion(Double version) {
        this.version = version;
    }

    public String getGenerator() {
        return generator;
    }

    public void setGenerator(String generator) {
        this.generator = generator;
    }
```

```
    public Osm3s getOsm3s() {
        return osm3s;
    }

    public void setOsm3s(Osm3s osm3s) {
        this.osm3s = osm3s;
    }

    public List<Element> getElements() {
        return elements;
    }

    public void setElements(List<Element> elements) {
        this.elements = elements;
    }

}
```

Codice A.62: Element.java

```
package it.dsantini.areediattesa.rete.impl.overpass;

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

import java.util.List;

public class Element {

    @SerializedName("type")
    @Expose
    private String type;
    @SerializedName("id")
    @Expose
    private Long id;
    @SerializedName("lat")
    @Expose
    private Double lat;
    @SerializedName("lon")
    @Expose
    private Double lon;
    @SerializedName("tags")
    @Expose
    private Tags tags;
    @SerializedName("nodes")
    @Expose
    private List<Long> nodes = null;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Long getId() {
```

```
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Double getLat() {
        return lat;
    }

    public void setLat(Double lat) {
        this.lat = lat;
    }

    public Double getLon() {
        return lon;
    }

    public void setLon(Double lon) {
        this.lon = lon;
    }

    public Tags getTags() {
        return tags;
    }

    public void setTags(Tags tags) {
        this.tags = tags;
    }

    public List<Long> getNodes() {
        return nodes;
    }

    public void setNodes(List<Long> nodes) {
        this.nodes = nodes;
    }

}
```

Codice A.63: Tags.java

```
package it.dsantini.areediattesa.rete.impl.overpass;

import android.view.ViewDebug;

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class Tags {

    @SerializedName("addr:city")
    @Expose
    private String addrCity;
    @SerializedName("addr:street")
    @Expose
    private String addrStreet;
```

```
    @SerializedName("emergency")
    @Expose
    private String emergency;
    @SerializedName("name")
    @Expose
    private String name;
    @SerializedName("assembly_point:earthquake")
    @Expose
    private String assemblyPointEarthquake;
    @SerializedName("assembly_point:fire")
    @Expose
    private String assemblyPointFire;
    @SerializedName("assembly_point:flood")
    @Expose
    private String assemblyPointFlood;
    @SerializedName("covered")
    @Expose
    private String covered;
    @SerializedName("capacity")
    @Expose
    private String capacity;
    @SerializedName("description")
    @Expose
    private String description;
    @SerializedName("operator")
    @Expose
    private String operator;
    @SerializedName("source")
    @Expose
    private String source;
    @SerializedName("area")
    @Expose
    private String area;
    @SerializedName("addr:postcode")
    @Expose
    private String addrPostcode;
    @SerializedName("image")
    @Expose
    private String image;

    public String getAddrCity() {
        return addrCity;
    }

    public void setAddrCity(String addrCity) {
        this.addrCity = addrCity;
    }

    public String getAddrStreet() {
        return addrStreet;
    }

    public void setAddrStreet(String addrStreet) {
        this.addrStreet = addrStreet;
    }

    public String getEmergency() {
        return emergency;
    }
```

```
    public void setEmergency(String emergency) {
        this.emergency = emergency;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getOperator() {
        return operator;
    }

    public void setOperator(String operator) {
        this.operator = operator;
    }

    public String getAssemblyPointEarthquake() {
        return assemblyPointEarthquake;
    }

    public void setAssemblyPointEarthquake(String assemblyPointEarthquake) {
        this.assemblyPointEarthquake = assemblyPointEarthquake;
    }

    public String getAssemblyPointFire() {
        return assemblyPointFire;
    }

    public void setAssemblyPointFire(String assemblyPointFire) {
        this.assemblyPointFire = assemblyPointFire;
    }

    public String getAssemblyPointFlood() {
        return assemblyPointFlood;
    }

    public void setAssemblyPointFlood(String assemblyPointFlood) {
        this.assemblyPointFlood = assemblyPointFlood;
    }

    public String getCovered() {
        return covered;
    }

    public void setCovered(String covered) {
        this.covered = covered;
```

```
    }

    public String getCapacity() {
        return capacity;
    }

    public void setCapacity(String capacity) {
        this.capacity = capacity;
    }

    public String getSource() {
        return source;
    }

    public void setSource(String source) {
        this.source = source;
    }

    public String getArea() {
        return area;
    }

    public void setArea(String area) {
        this.area = area;
    }

    public String getAddrPostcode() {
        return addrPostcode;
    }

    public void setAddrPostcode(String addrPostcode) {
        this.addrPostcode = addrPostcode;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

}
```

Per tornare al testo: 4.2.3

A.3.5 ViewModel

Codice A.64: MappaViewModel

```
package it.dsantini.areediattesa.viewModel;

public enum StatoDownload {
    scaricato,
    scaricamento_in_corso,
    scaricamento_fallito,
    area_troppo_grande;
}
```

Codice A.65: MappaViewModel

```
package it.dsantini.areediattesa.viewModel;

public enum StatoDettagli {
    pronto,
    navigazione_errore,
    navigazione_nessuna_app
}
```

MappaViewModel

Codice A.66: MappaViewModel

```
package it.dsantini.areediattesa.viewModel;

import android.arch.lifecycle.LiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Confine;

public interface MappaViewModel {
    /**
     * @return Aree di emergenza osservabili da visualizzare
     */
    @NonNull
    LiveData<AreaEmergenza[]> getAree();

    /**
     * @return Stato osservabile della mappa
     */
    @NonNull
    LiveData<Stato> getStato();

    /**
     * @return Stato osservabile dello scaricamento
     */
    @NonNull
    LiveData<StatoDownload> getStatoDownload();

    /**
     * @param porzioneDiMappa Porzione di mappa su cui spostarsi
     */
    void onSposta(@NonNull Confine porzioneDiMappa);

    /**
     * @param a Area di emergenza di cui visualizzare i dettagli
     */
    void onScegliArea(@NonNull AreaEmergenza a);


}
```

Codice A.67: MappaViewModelBase

```
package it.dsantini.areediattesa.viewModel.impl;

import android.app.Application;
import android.arch.lifecycle.AndroidViewModel;
```

```
import android.arch.lifecycle.LiveData;
import android.arch.lifecycle.MediatorLiveData;
import android.arch.lifecycle.MutableLiveData;
import android.arch.lifecycle.Observer;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;

import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;
import java.util.Map;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.data.DettagliRepository;
import it.dsantini.areediattesa.data.MappaRepository;
import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.data.impl.DettagliRepositoryBase;
import it.dsantini.areediattesa.data.impl.MappaRepositoryBase;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.persistenza.Persistenza;
import it.dsantini.areediattesa.persistenza.mock.PersistenzaFallback;
import it.dsantini.areediattesa.rete.SorgenteAree;
import it.dsantini.areediattesa.rete.mock.SorgenteAreeFallback;
import it.dsantini.areediattesa.view.DettagliView;
import it.dsantini.areediattesa.viewModel.MappaViewModel;
import it.dsantini.areediattesa.viewModel.StatoDownload;
import it.dsantini.areediattesa.viewModel.helper.AreaViewMapping;
import it.dsantini.areediattesa.viewModel.helper.impl.AreaViewMappingConfig;

public class MappaViewModelBase extends AndroidViewModel implements MappaViewModel,
    Observer<Mappa> {
    private static final String CLASS = "MappaViewModelBase";

    @NonNull
    private final MappaRepository m;
    @NonNull
    private final DettagliRepository d;
    @NonNull
    private final MutableLiveData<AreaEmergenza[]> aree;
    @NonNull
    private final LiveData<Stato> stato;
    @NonNull
    private final MutableLiveData<StatoDownload> statoDownload;
    @NonNull
    private final Map<Class<? extends AreaEmergenza>, Class<? extends DettagliView>>
        mapping;
    @NonNull
    private final SorgenteAree sorgente;

    private Confine porzione;

    public MappaViewModelBase(@NonNull Application application) {
        super(application);

        this.d = DettagliRepositoryBase.getInstance();
```

```
        MappaRepositoryBase m = MappaRepositoryBase.getInstance();
        this.m = m;

        MediatorLiveData<AreaEmergenza[]> aree = new MediatorLiveData<>();
        aree.addSource(m.getMappa(), this);
        this.aree = aree;

        LiveData<Stato> stato = m.getStato();
        String nomePersistenza = getApplication().getString(R.string.persistenza);
        Log.i(CLASS, "Istanzazione di " + nomePersistenza);
        try {
            Persistenza p = (Persistenza) Class.forName(nomePersistenza).
                getConstructor().newInstance();
            m.setPersistenza(p);
        } catch (Exception e) {
            Log.e(CLASS, "Errore nell'istanziazione della persistenza", e);
            m.setPersistenza(new PersistenzaFallback());
            stato = new MutableLiveData<>();
            ((MutableLiveData<Stato>) stato).setValue(Stato.errore_caricamento);
        }
        this.stato = stato;

        SorgenteAree sorgente;
        String nomeSorgente = getApplication().getString(R.string.sorgenteAree);
        Log.i(CLASS, "Istanzazione di " + nomeSorgente);
        try {
            sorgente = (SorgenteAree) Class.forName(nomeSorgente).getConstructor().
                newInstance();
        } catch (Exception e) {
            Log.e(CLASS, "Errore nell'istanziazione della sorgente", e);
            sorgente = new SorgenteAreeFallback();
        }
        this.sorgente = sorgente;

        this.statoDownload = new MutableLiveData<>();

        AreaViewMapping avm = new AreaViewMappingConfig(getApplication());
        this.mapping = avm.getMap();
    }

    @NonNull
    @Override
    public LiveData<AreaEmergenza[]> getAree() {
        return aree;
    }

    @NonNull
    @Override
    public LiveData<Stato> getStato() {
        return stato;
    }

    @NonNull
    @Override
    public LiveData<StatoDownload> getStatoDownload() {
        return statoDownload;
    }
```

```
    private void aggiornaAree() {
        final String TAG = CLASS + "::aggiornaAree";

        Mappa map = m.getMappa().getValue();
        if (map == null)
            Log.e(TAG, "Mappa nulla");
        else if (porzione == null)
            Log.e(TAG, "Porzione nulla");
        else { // Mappa non nulla
            GruppoAree gruppo = null;

            for (GruppoAree g : map.getGruppi()) {
                if (g.getConfine().contiene(porzione)) { // Porzione di mappa
                    contenuta nel confine del gruppo
                    gruppo = g;
                    break;
                }
            }

            if (gruppo != null) { // Gruppo trovato
                Log.v(TAG, "Trovato il gruppo con confine " + gruppo.getConfine());
                aree.setValue(gruppo.getAree());

                long eta = gruppo.getTimestamp().until(LocalDateTime.now(),
                    ChronoUnit.DAYS),
                        etaMassima = getApplication().getResources().getInteger(R.
                            integer.maxDays);
                if (eta >= etaMassima) { // Gruppo vecchio
                    new DownloadAsync(statoDownload, sorgente, gruppo.getConfine(),
                        m).start();
                }
            } else { // Gruppo non trovato
                Log.v(TAG, "Gruppo non trovato, download nuovo");

                new DownloadAsync(statoDownload, sorgente, porzione, m).start();
            }
        }
    }

    @Override
    public void onSposta(@NonNull Confine porzioneDiMappa) {
        final String TAG = CLASS + "::onSposta";

        this.porzione = porzioneDiMappa;
        Log.v(TAG, "Spostamento in " + porzioneDiMappa.toString());
        aggiornaAree();
    }

    @Override
    public void onScegliArea(@NonNull AreaEmergenza a) {
        d.setArea(a);

        Intent intent;
        if (mapping.containsKey(a.getClass()))
            intent = new Intent(getApplication().getApplicationContext(), mapping.
                get(a.getClass()));
        else
            intent = new Intent(getApplication().getApplicationContext(),
                DettagliView.class);
```

```
            getApplication().startActivity(intent);
        }

        @Override
        public void onChanged(@Nullable Mappa mappa) {
            Log.v("MappaViewModelBase::onChanged(Mappa)", "Mappa cambiata: " + mappa);
            aggiornaAree();
        }
}
```

DettagliViewModel

Codice A.68: DettagliViewModel

```
package it.dsantini.areediattesa.viewModel;

import android.arch.lifecycle.LiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.AreaEmergenza;

public interface DettagliViewModel {
    /**
     * @return Area di emergenza osservabile di cui mostrare i dettagli
     */
    @NonNull
    LiveData<AreaEmergenza> getArea();

    /**
     * @return Stato osservabile dei dettagli
     */
    @NonNull
    LiveData<StatoDettagli> getStato();

    /**
     * Event Handler per l'avvio della navigazione
     */
    void onAvviaNavigazione();
}
```

Codice A.69: DettagliViewModelBase

```
package it.dsantini.areediattesa.viewModel.impl;

import android.app.Application;
import android.arch.lifecycle.AndroidViewModel;
import android.arch.lifecycle.LiveData;
import android.arch.lifecycle.MutableLiveData;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.support.annotation.NonNull;
import android.util.Log;

import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Locale;

import it.dsantini.areediattesa.data.DettagliRepository;
import it.dsantini.areediattesa.data.impl.DettagliRepositoryBase;
```

```java
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.viewModel.DettagliViewModel;
import it.dsantini.areediattesa.viewModel.StatoDettagli;

public class DettagliViewModelBase extends AndroidViewModel implements
    DettagliViewModel {
    private static final String TAG = "DettagliViewModelBase";

    @NonNull
    private final DettagliRepository d;
    @NonNull
    private final MutableLiveData<StatoDettagli> stato;
    @NonNull
    private final NumberFormat format;

    public DettagliViewModelBase(@NonNull Application application) {
        super(application);
        d = DettagliRepositoryBase.getInstance();
        stato = new MutableLiveData<>();
        stato.setValue(StatoDettagli.pronto);

        format = DecimalFormat.getNumberInstance(Locale.US);
    }

    @NonNull
    @Override
    public LiveData<AreaEmergenza> getArea() {
        return d.getArea();
    }

    @NonNull
    @Override
    public LiveData<StatoDettagli> getStato() {
        return stato;
    }

    @Override
    public void onAvviaNavigazione() {
        AreaEmergenza a = getArea().getValue();
        if (a == null) {
            Log.e(TAG, "Richiesta navigazione ma area nulla");
            stato.setValue(StatoDettagli.navigazione_errore);
        } else {
            try {
                Nodo n = a.getGeometria().getPuntoSuSuperficie();

                String query = "google.navigation:q="
                        + format.format(n.getLatitudine())
                        + ","
                        + format.format(n.getLongitudine());

                Log.e(TAG, query);

                Uri uri = Uri.parse(query);
                Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                getApplication().startActivity(intent);
            } catch (ActivityNotFoundException e) {
                stato.setValue(StatoDettagli.navigazione_nessuna_app);
```

```
                Log.e(TAG, "Nessuna app disponibile per la navigazione", e);
            } catch (Exception e) {
                stato.setValue(StatoDettagli.navigazione_errore);
                Log.e(TAG, "Errore durante l'avvio della navigazione", e);
            }
        }
    }
}
```

GestioneViewModel

Codice A.70: GestioneViewModel

```
package it.dsantini.areediattesa.viewModel;

import android.arch.lifecycle.LiveData;
import android.support.annotation.NonNull;

import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.model.GruppoAree;

public interface GestioneViewModel {
    /**
     * @return Gruppi osservabili da visualizzare
     */
    @NonNull
    LiveData<GruppoAree[]> getGruppi();

    /**
     * @return Stato osservabile della mappa
     */
    @NonNull
    LiveData<Stato> getStato();

    /**
     * @return Stato osservabile dello scaricamento
     */
    @NonNull
    LiveData<StatoDownload> getStatoDownload();

    void onAggiornaTutti();

    /**
     * @param g Gruppo da aggiornare
     */
    void onAggiorna(@NonNull GruppoAree g);

    void onRimuoviTutti();

    /**
     * @param g Gruppo da rimuovere
     */
    void onRimuovi(@NonNull GruppoAree g);

    /**
     * @param g Gruppo da aggiungere ai preferiti
     */
    void onAggiungiPreferito(@NonNull GruppoAree g);

    /**
```

```
     * @param g Gruppo da rimuovere dai preferiti
     */
    void onRimuoviPreferito(@NonNull GruppoAree g);
}
```

Codice A.71: GestioneViewModelBase

```
package it.dsantini.areediattesa.viewModel.impl;

import android.arch.lifecycle.LiveData;
import android.arch.lifecycle.MediatorLiveData;
import android.arch.lifecycle.MutableLiveData;
import android.arch.lifecycle.Observer;
import android.arch.lifecycle.ViewModel;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;

import it.dsantini.areediattesa.data.MappaRepository;
import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.data.impl.MappaRepositoryBase;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.rete.DownloadException;
import it.dsantini.areediattesa.rete.SorgenteAree;
import it.dsantini.areediattesa.rete.impl.SorgenteAreeOverpass;
import it.dsantini.areediattesa.viewModel.GestioneViewModel;
import it.dsantini.areediattesa.viewModel.StatoDownload;

public class GestioneViewModelBase extends ViewModel implements GestioneViewModel,
    Observer<Mappa> {
    private static final String TAG = "GestioneViewModelBase";

    @NonNull
    private final MappaRepository m;
    @NonNull
    private final MediatorLiveData<GruppoAree[]> gruppi;
    @NonNull
    private final MutableLiveData<StatoDownload> statoDownload;
    @NonNull
    private final SorgenteAree sorgente;

    public GestioneViewModelBase() {
        gruppi = new MediatorLiveData<>();
        m = MappaRepositoryBase.getInstance();
        sorgente = new SorgenteAreeOverpass();
        statoDownload = new MutableLiveData<>();
        statoDownload.setValue(StatoDownload.scaricato);

        gruppi.addSource(m.getMappa(), this);
    }

    @Override
    public void onChanged(@Nullable Mappa mappa) {
        final String TAG = "GestioneViewModelBase::onChanged(Mappa)";

        if (mappa == null) {
            Log.e(TAG, "Mappa nulla");
```

```
            gruppi.setValue(null);
        } else {
            Log.v(TAG, "Mappa cambiata, mappa.gruppi.length=" + mappa.getGruppi().
                length);
            gruppi.setValue(mappa.getGruppi());
        }
    }

    @NonNull
    @Override
    public LiveData<GruppoAree[]> getGruppi() {
        return gruppi;
    }

    @NonNull
    @Override
    public LiveData<Stato> getStato() {
        return m.getStato();
    }

    @NonNull
    @Override
    public LiveData<StatoDownload> getStatoDownload() {
        return statoDownload;
    }

    @Override
    public void onAggiornaTutti() {
        Mappa mappa = m.getMappa().getValue();
        if (mappa != null) {
            for (GruppoAree g : mappa.getGruppi()) {
                Confine c = g.getConfine();
                new DownloadAsync(statoDownload, sorgente, c, m).start();
            }
        }
    }

    @Override
    public void onAggiorna(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        Mappa mappa = m.getMappa().getValue();
        if (mappa != null) {
            Confine c = g.getConfine();
            statoDownload.setValue(StatoDownload.scaricamento_in_corso);
            try {
                GruppoAree nuovo = sorgente.download(c);
                statoDownload.setValue(StatoDownload.scaricato);
                Mappa nuova = mappa.senza(g).con(nuovo);
                m.setMappa(nuova);
            } catch (DownloadException e) {
                statoDownload.setValue(StatoDownload.scaricamento_fallito);
                Log.e(TAG, "Download", e);
            }
        }
    }

    @Override
```

```
    public void onRimuoviTutti() {
        Mappa mappa = m.getMappa().getValue();
        if (mappa != null) {
            for (GruppoAree g : mappa.getGruppi())
                if (!g.isPreferito())
                    mappa = mappa.senza(g);

            m.setMappa(mappa);
        }
    }

    @Override
    public void onRimuovi(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        Mappa mappa = m.getMappa().getValue();
        if (mappa != null) {
            Mappa nuova = mappa.senza(g);
            m.setMappa(nuova);
        }
    }

    @Override
    public void onAggiungiPreferito(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        if (!g.isPreferito()) {
            Mappa mappa = m.getMappa().getValue();
            GruppoAree nuovo = g.withPreferito(true);
            if (mappa != null) {
                Mappa nuova = mappa.senza(g).con(nuovo);
                m.setMappa(nuova);
            }
        }
    }

    @Override
    public void onRimuoviPreferito(@NonNull GruppoAree g) {
        if (g == null)
            throw new IllegalArgumentException("GruppoAree g nullo");

        if (g.isPreferito()) {
            Mappa mappa = m.getMappa().getValue();
            GruppoAree nuovo = g.withPreferito(false);
            if (mappa != null) {
                Mappa nuova = mappa.senza(g).con(nuovo);
                m.setMappa(nuova);
            }
        }
    }
}
```

Per tornare al testo: 4.2.4

A.3.6 View

GeometriaPainter

Codice A.72: GeometriaPainter

```
package it.dsantini.areediattesa.view.helper;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.model.impl.Poligono;

public interface GeometriaPainter {
    void stampa(@NonNull Nodo n);

    void stampa(@NonNull Poligono p);
}
```

Codice A.73: GoogleMapAreaGeometriaPainter

```
package it.dsantini.areediattesa.view.helper.impl;

import android.graphics.Color;
import android.support.annotation.NonNull;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polygon;
import com.google.android.gms.maps.model.PolygonOptions;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Nodo;
import it.dsantini.areediattesa.model.impl.Poligono;
import it.dsantini.areediattesa.view.helper.GeometriaPainter;

public class GoogleMapAreaGeometriaPainter implements GeometriaPainter {
    private static BitmapDescriptor icon;

    @NonNull
    private final GoogleMap map;
    @NonNull
    private final AreaEmergenza area;

    public GoogleMapAreaGeometriaPainter(@NonNull GoogleMap map, @NonNull
        AreaEmergenza area) {
        this.map = map;
        this.area = area;

        if (icon == null)
            icon = BitmapDescriptorFactory.fromResource(R.drawable.attesa);
    }

    @Override
    public void stampa(@NonNull Nodo n) {
        LatLng pos = new LatLng(n.getLatitudine(), n.getLongitudine());
        MarkerOptions mOpt = new MarkerOptions()
                .draggable(false)
                .position(pos)
```

```
                .icon(icon)
                .title(area.getNome());
        Marker m = map.addMarker(mOpt);
        m.setTag(area);
    }

    @Override
    public void stampa(@NonNull Poligono p) {
        PolygonOptions pOpt = new PolygonOptions()
                .fillColor(Color.argb(100, 0, 255, 0))
                .clickable(true);
        for (Nodo n : p.getVertici())
            pOpt.add(new LatLng(n.getLatitudine(), n.getLongitudine()));
        Polygon gmp = map.addPolygon(pOpt);
        gmp.setTag(area);
    }
}
```

Per tornare al testo: 4.1.1

MappaView

Codice A.74: activity_mappa_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".view.MappaView">

    <!--
    <data>

        <variable
            name="viewModel"
            type="it.dsantini.areediattesa.viewModel.impl.MappaViewModelBase" />
    </data>
    -->

    <android.support.constraint.ConstraintLayout
        android:id="@+id/aree"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!-- https://developers.google.com/maps/documentation/android-sdk/map#
            using_xml_attributes -->

        <fragment
            android:id="@+id/mappaAree"
            android:name="com.google.android.gms.maps.SupportMapFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:cameraTargetLat="@dimen/latInit"
            app:cameraTargetLng="@dimen/lonInit"
            app:cameraZoom="@dimen/zoomInit" />

        <TextView
            android:id="@+id/attribuzione"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/attribution_margin"
```

```
            android:text='@string/attribution'
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            />

        <android.support.design.widget.FloatingActionButton
            android:id="@+id/fabGestioneGruppi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|end"
            android:layout_margin="@dimen/fab_margin"
            android:onClick="onGestioneGruppi"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:srcCompat="@android:drawable/ic_menu_manage" />
    </android.support.constraint.ConstraintLayout>
</layout>
```

Codice A.75: GestionePermessi

```
package it.dsantini.areediattesa.view.helper;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.support.v4.content.ContextCompat;
import android.util.Log;

public class GestionePermessi {
    private static final String COARSE_LOC = Manifest.permission.ACCESS_COARSE_LOCATION;
    private static final String FINE_LOC = Manifest.permission.ACCESS_FINE_LOCATION;
    private static final int PERMISSION_ID = 42;

    public static boolean HoPermessoLocalizzazione(Context context) {
        final String TAG = "GestionePermessi::HoPermessoLocalizzazione";

        boolean coarse = ContextCompat.checkSelfPermission(context, COARSE_LOC) == PackageM
                fine = ContextCompat.checkSelfPermission(context, FINE_LOC) == PackageMana

        Log.i(TAG, COARSE_LOC + (coarse ? " concesso" : " respinto"));
        Log.i(TAG, FINE_LOC + (fine ? " concesso" : " respinto"));

        return coarse || fine;
    }
}
```

Codice A.76: MappaView

```
package it.dsantini.areediattesa.view;

import android.Manifest;
import android.annotation.SuppressLint;
import android.arch.lifecycle.Observer;
import android.arch.lifecycle.ViewModelProviders;
import android.content.Intent;
```

```
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.text.method.LinkMovementMethod;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.Polygon;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.impl.googleMaps.BoundingBoxLatLngBounds;
import it.dsantini.areediattesa.view.helper.GestionePermessi;
import it.dsantini.areediattesa.view.helper.impl.GoogleMapAreaGeometriaPainter;
import it.dsantini.areediattesa.viewModel.StatoDownload;
import it.dsantini.areediattesa.viewModel.impl.MappaViewModelBase;

//import com.jakewharton.threetenabp.AndroidThreeTen;

public class MappaView extends FragmentActivity implements OnMapReadyCallback,
    Observer<AreaEmergenza[]> {
    private static final String COARSE_LOC = Manifest.permission.
        ACCESS_COARSE_LOCATION;
    private static final String FINE_LOC = Manifest.permission.ACCESS_FINE_LOCATION;
    private static final int PERMISSION_ID = 42;

    private MappaViewModelBase viewModel;
    private GoogleMap map;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //if (Build.VERSION.SDK_INT < 26)
        //    AndroidThreeTen.init(this);

        final String TAG = "MappaView::onCreate";

        if (getString(R.string.google_maps_key).equals("API_KEY")) {
            String error = getString(R.string.gmaps_api_key_error);
            Log.e(TAG, error);
            Toast.makeText(this, error, Toast.LENGTH_LONG).show();
        }

        setContentView(R.layout.activity_mappa_view);
```

```
        // Obtain the SupportMapFragment and get notified when the map is ready to
            be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
            getSupportFragmentManager()
                .findFragmentById(R.id.mappaAree);
        mapFragment.getMapAsync(this);

        viewModel = ViewModelProviders.of(this).get(MappaViewModelBase.class);

        viewModel.getStato().observe(this, new Observer<Stato>() {
            @Override
            public void onChanged(@Nullable Stato stato) {
                if (stato == null)
                    Log.w(TAG, "Stato nullo");
                else {
                    try {
                        int id = getResources().getIdentifier(stato.name(), "string"
                            , getPackageName());
                        Snackbar.make(findViewById(R.id.aree), getString(id),
                            Snackbar.LENGTH_LONG).show();
                    } catch (Exception e) {
                        Log.e(TAG, "Errore nella visualizzazione di " + stato.
                            toString(), e);
                        Snackbar.make(findViewById(R.id.aree), getString(R.string.
                            possibile_errore_persistenza), Snackbar.LENGTH_SHORT).
                            show();
                    }
                }
            }
        });
        viewModel.getStatoDownload().observe(this, new Observer<StatoDownload>() {
            @Override
            public void onChanged(@Nullable StatoDownload statoDownload) {
                if (statoDownload == null)
                    Log.w(TAG, "Stato download nullo");
                else {
                    try {
                        int id = getResources().getIdentifier(statoDownload.name(),
                            "string", getPackageName());
                        Snackbar.make(findViewById(R.id.aree), getString(id),
                            Snackbar.LENGTH_LONG).show();
                    } catch (Exception e) {
                        Log.e(TAG, "Errore nella visualizzazione di " +
                            statoDownload.toString(), e);
                        Snackbar.make(findViewById(R.id.aree), getString(R.string.
                            possibile_errore_rete), Snackbar.LENGTH_SHORT).show();
                    }
                }
            }
        });

        TextView v = findViewById(R.id.attribuzione);
        v.setMovementMethod(LinkMovementMethod.getInstance()); // Rende il link
            cliccabile

        checkNavigazione();
    }
```

```
    private void checkNavigazione() {
        String query = "google.navigation:q=0.0,0.0";
        Uri uri = Uri.parse(query);
        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
        if (intent.resolveActivity(getPackageManager()) == null) // Nessuna app
            disponibile per la navigazione
            Snackbar.make(findViewById(R.id.aree), getString(R.string.
                navigazione_nessuna_app), Snackbar.LENGTH_LONG).show();
    }


    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera.
         In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be
         prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the
         user has
     * installed Google Play services and returned to the app.
     */
    @SuppressLint("MissingPermission")
    @Override
    public void onMapReady(final GoogleMap googleMap) {
        final String TAG = "MappaView::onMapReady";

        this.map = googleMap;

        viewModel.getAree().observe(this, this);

        googleMap.setOnCameraIdleListener(new GoogleMap.OnCameraIdleListener() {
            @Override
            public void onCameraIdle() {
                LatLngBounds bounds = googleMap.getProjection().getVisibleRegion().
                    latLngBounds;
                try {
                    Confine porzione = new BoundingBoxLatLngBounds(bounds, null);
                    viewModel.onSposta(porzione);
                } catch (IllegalArgumentException e) {
                    Log.e(TAG, "Errore nella reazione allo spostamento della mappa",
                        e);
                    Snackbar.make(findViewById(R.id.mappaAree), "Errore nel
                        caricamento delle aree", Snackbar.LENGTH_LONG).show();
                }
            }
        });

        googleMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
            @Override
            public boolean onMarkerClick(Marker marker) {
                return onGeometriaClick(marker.getTag());
            }
        });

        googleMap.setOnPolygonClickListener(new GoogleMap.OnPolygonClickListener() {
            @Override
            public void onPolygonClick(Polygon polygon) {
```

```
                onGeometriaClick(polygon.getTag());
            }
        });

        /*
        googleMap.setOnInfoWindowClickListener(new GoogleMap.
            OnInfoWindowClickListener() {
            @Override
            public void onInfoWindowClick(Marker marker) {
                Object tag = marker.getTag();
                if (tag == null)
                    Log.e(TAG, "Tag non presente");
                else if (!(tag instanceof AreaEmergenza))
                    Log.e(TAG, "Tag non istanza di AreaEmergenza");
                else
                    viewModel.onScegliArea((AreaEmergenza) tag);
            }
        });
        */


        if (GestionePermessi.HoPermessoLocalizzazione(this))
            map.setMyLocationEnabled(true);
        else
            ActivityCompat.requestPermissions(this, new String[]{COARSE_LOC,
                FINE_LOC}, PERMISSION_ID);

    }

    public boolean onGeometriaClick(Object tag) {
        final String TAG = "MappaView::onGeometriaClick";
        boolean completato = false;

        if (tag == null)
            Log.e(TAG, "Tag non presente");
        else if (!(tag instanceof AreaEmergenza))
            Log.e(TAG, "Tag non istanza di AreaEmergenza");
        else {
            viewModel.onScegliArea((AreaEmergenza) tag);
            completato = true;
        }

        return completato;
    }

    @SuppressLint("MissingPermission")
    @Override
    public void onRequestPermissionsResult(int code, @NonNull String[] permessi,
        @NonNull int[] risultati) {
        final String TAG = "onRequestPermissionsRes";

        switch (code) {
            case PERMISSION_ID:
                if (risultati[0] == PackageManager.PERMISSION_GRANTED
                        || risultati[1] == PackageManager.PERMISSION_GRANTED) {
                    Log.d(TAG, "Permesso localizzazione concesso");
                    map.setMyLocationEnabled(true);
                } else
                    Log.w(TAG, "Permesso localizzazione negato");
```

```
                break;

            default:
                Log.w(TAG, "Codice richiesta di permessi sconosciuto");
        }
    }

    @Override
    public void onChanged(@Nullable AreaEmergenza[] aree) {
        final String TAG = "MappaView::onChanged";

        if (aree == null)
            Log.w(TAG, "Aree nullo");
        else {
            Log.v(TAG, "Aree cambiate, aree.length=" + aree.length);

            // Rimuovi le geometrie precedenti
            map.clear();

            // Aggiungi le nuove geometrie
            for (AreaEmergenza area : aree)
                area.getGeometria().accept(new GoogleMapAreaGeometriaPainter(map,
                    area));
        }
    }

    public void onGestioneGruppi(View sorgenteClick) {
        Intent intent = new Intent(this, GestioneView.class);
        Log.v("MappaView::onGestioneGruppi", "---------- PASSAGGIO ALLA
            GESTIONE DEI GRUPPI ----------");
        getApplication().startActivity(intent);
    }
}
```

Per tornare al testo: 4.2.5

ConfinePainter

Codice A.77: ConfinePainter

```
package it.dsantini.areediattesa.view.helper;

import android.support.annotation.NonNull;

import it.dsantini.areediattesa.model.BoundingBox;

public interface ConfinePainter {
    void stampa(@NonNull BoundingBox b);
}
```

Codice A.78: GoogleMapConfinePainter

```
package it.dsantini.areediattesa.view.helper.impl;

import android.graphics.Color;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Polygon;
```

```
import com.google.android.gms.maps.model.PolygonOptions;

import it.dsantini.areediattesa.model.BoundingBox;
import it.dsantini.areediattesa.view.helper.ConfinePainter;

public class GoogleMapConfinePainter implements ConfinePainter {
    @NonNull
    private final GoogleMap map;
    @Nullable
    private final Object tag;

    public GoogleMapConfinePainter(@NonNull GoogleMap map) {
        this(map, null);
    }

    public GoogleMapConfinePainter(@NonNull GoogleMap map, @Nullable Object tag) {
        if (map == null)
            throw new IllegalArgumentException("GoogleMap nulla");

        this.map = map;
        this.tag = tag;
    }

    @Override
    public void stampa(@NonNull BoundingBox box) {
        PolygonOptions pOpt = new PolygonOptions()
                .fillColor(Color.argb(100, 0, 255, 0))
                .clickable(true);
        pOpt.add(new LatLng(box.getLatMin(), box.getLonMin()),
                new LatLng(box.getLatMin(), box.getLonMax()),
                new LatLng(box.getLatMax(), box.getLonMax()),
                new LatLng(box.getLatMax(), box.getLonMin()));
        Polygon gmp = map.addPolygon(pOpt);
        if(tag!=null)
            gmp.setTag(tag);
        else
            gmp.setTag(box);
    }
}
```

Per tornare al testo: 4.1.1

GestioneView

Codice A.79: activity_gestione_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".view.GestioneView">

    <!--
    <data>

        <variable
            name="viewModel"
            type="it.dsantini.areediattesa.viewModel.impl.MappaViewModelBase" />
    </data>
    -->
```

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/gestione"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment xmlns:map="http://schemas.android.com/apk/res-auto"
        android:id="@+id/mappaGestione"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:cameraTargetLat="@dimen/latInit"
        app:cameraTargetLng="@dimen/lonInit"
        app:cameraZoom="@dimen/zoomInit" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/aggiornaTutti"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end"
        android:onClick="onAggiornaTutti"
        app:layout_anchor="@id/rimuoviTutti"
        app:layout_anchorGravity="top"
        app:srcCompat="@android:drawable/ic_menu_rotate"
        app:useCompatPadding="true" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/rimuoviTutti"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_marginEnd="@dimen/fab_margin"
        android:layout_marginBottom="@dimen/fab_margin"
        android:onClick="onRimuoviTutti"
        app:srcCompat="@android:drawable/ic_menu_delete"
        app:useCompatPadding="true" />

    <LinearLayout
        android:id="@+id/gestione_gruppo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        app:layout_anchor="@id/mappaGestione"
        app:layout_anchorGravity="top|center">

        <ImageButton
            android:id="@+id/aggiungi_preferito"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:contentDescription="@string/aggiungiPreferito"
            android:src="@android:drawable/btn_star_big_off"
            android:onClick="onAggiungiPreferito" />

        <ImageButton
            android:id="@+id/rimuovi_preferito"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:contentDescription="@string/rimuoviPreferito"
```

```
                android:src="@android:drawable/btn_star_big_on"
                android:onClick="onRimuoviPreferito" />

            <ImageButton
                android:id="@+id/aggiorna"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:contentDescription="@string/aggiornaGruppo"
                android:src="@android:drawable/ic_menu_rotate"
                android:onClick="onAggiorna" />

            <ImageButton
                android:id="@+id/rimuovi"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:contentDescription="@string/rimuoviGruppo"
                android:src="@android:drawable/ic_menu_delete"
                android:onClick="onRimuovi" />
        </LinearLayout>

    </android.support.design.widget.CoordinatorLayout>
</layout>
```

Codice A.80: GestioneView

```
package it.dsantini.areediattesa.view;

import android.annotation.SuppressLint;
import android.arch.lifecycle.Observer;
import android.arch.lifecycle.ViewModelProviders;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.Snackbar;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.View;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.Polygon;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.data.Stato;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.view.helper.GestionePermessi;
import it.dsantini.areediattesa.view.helper.impl.GoogleMapConfinePainter;
import it.dsantini.areediattesa.viewModel.GestioneViewModel;
import it.dsantini.areediattesa.viewModel.impl.GestioneViewModelBase;

public class GestioneView extends FragmentActivity implements OnMapReadyCallback,
    Observer<GruppoAree[]> {
    private static final String CLASS = "GestioneView";

    private GoogleMap map;
    private GestioneViewModel viewModel;
```

```
    private GruppoAree gruppoSelezionato = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        final String TAG = CLASS + "::onCreate";

        setContentView(R.layout.activity_gestione_view);
        // Obtain the SupportMapFragment and get notified when the map is ready to
            be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
            getSupportFragmentManager()
                .findFragmentById(R.id.mappaGestione);
        mapFragment.getMapAsync(this);

        viewModel = ViewModelProviders.of(this).get(GestioneViewModelBase.class);

        viewModel.getStato().observe(this, new StatoObserver());
    }

    private class StatoObserver implements Observer<Stato> {
        @Override
        public void onChanged(@Nullable Stato stato) {
            final String TAG = "StatoObserver::onChanged";

            if (stato == null)
                Log.w(TAG, "Stato nullo");
            else {
                try {
                    int id = getResources().getIdentifier(stato.name(), "string",
                        getPackageName());
                    Snackbar.make(findViewById(R.id.gestione), getString(id),
                        Snackbar.LENGTH_LONG).show();
                } catch (Exception e) {
                    Log.e(TAG, "Errore nella visualizzazione di" + stato.toString()
                        , e);
                    Snackbar.make(findViewById(R.id.gestione), getString(R.string.
                        possibile_errore_persistenza), Snackbar.LENGTH_SHORT).show();
                }
            }
        }
    }

    @SuppressLint("MissingPermission")
    @Override
    public void onMapReady(GoogleMap googleMap) {
        map = googleMap;

        viewModel.getGruppi().observe(this, this);

        map.setOnPolygonClickListener(new GestioneOnPolygonClickListener());

        if (GestionePermessi.HoPermessoLocalizzazione(this))
            map.setMyLocationEnabled(true);
    }

    private class GestioneOnPolygonClickListener implements GoogleMap.
        OnPolygonClickListener {
```

```
        @Override
        public void onPolygonClick(Polygon polygon) {
            final String TAG = "GestioneOnPolygonClickListener::onPolygonClick";

            final Object tag = polygon.getTag();
            Log.d(TAG, "Click sul poligono " + tag);

            if (tag instanceof GruppoAree) {
                Log.v(TAG, "tag GruppoAree, ggiiornamento bottoni di gestione");
                final GruppoAree gruppo = (GruppoAree) tag;
                gruppoSelezionato = gruppo;
                findViewById(R.id.aggiungi_preferito).setVisibility(gruppo.
                    isPreferito() ? View.GONE : View.VISIBLE);
                findViewById(R.id.rimuovi_preferito).setVisibility(gruppo.
                    isPreferito() ? View.VISIBLE : View.GONE);
                findViewById(R.id.gestione_gruppo).setVisibility(View.VISIBLE);
            } else
                Log.w(TAG, "Tipo tag sconosciuto, non faccio nulla");
        }
    }

    public void onAggiorna(View sorgenteClick) {
        if (gruppoSelezionato == null)
            Log.e("GestioneView::onAggiorna", "gruppoSelezionato nullo");
        else
            viewModel.onAggiorna(gruppoSelezionato);
    }

    public void onAggiornaTutti(View sorgenteClick) {
        viewModel.onAggiornaTutti();
    }

    public void onRimuovi(View sorgenteClick) {
        if (gruppoSelezionato == null)
            Log.e("GestioneView::onRimuovi", "gruppoSelezionato nullo");
        else
            viewModel.onRimuovi(gruppoSelezionato);
    }

    public void onRimuoviTutti(View sorgenteClick) {
        viewModel.onRimuoviTutti();
    }

    public void onAggiungiPreferito(View sorgenteClick) {
        if (gruppoSelezionato == null)
            Log.e("GestioneView::onRimuovi", "gruppoSelezionato nullo");
        else
            viewModel.onAggiungiPreferito(gruppoSelezionato);
    }

    public void onRimuoviPreferito(View sorgenteClick) {
        if (gruppoSelezionato == null)
            Log.e("GestioneView::onRimuovi", "gruppoSelezionato nullo");
        else
            viewModel.onRimuoviPreferito(gruppoSelezionato);
    }

    @Override
    public void onChanged(@Nullable GruppoAree[] gruppi) {
```

```
        final String TAG = "GestioneView::onChanged";

        findViewById(R.id.gestione_gruppo).setVisibility(View.INVISIBLE);
        if (gruppi == null)
            Log.e(TAG, "Elenco gruppi nullo");
        else {
            Log.v(TAG, "Gruppi cambiati, gruppi.length=" + gruppi.length);

            map.clear();

            for (GruppoAree gruppo : gruppi)
                gruppo.getConfine().accept(new GoogleMapConfinePainter(map, gruppo))
                    ;
        }
    }
}
```

Per tornare al testo: 4.2.5

DettagliView

Codice A.81: activity_dettagli_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".view.DettagliView">

    <data>

        <variable
            name="viewModel"
            type="it.dsantini.areediattesa.viewModel.impl.DettagliViewModelBase" />
    </data>

    <android.support.constraint.ConstraintLayout
        android:id="@+id/dettagli"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:padding="5dp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="1.0"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintVertical_bias="0.0">

                <TextView
                    android:id="@+id/nomeArea"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
```

```
                android:layout_margin="@dimen/text_margin"
                android:text="@{viewModel.area.nome, default=@string/nomeAssente
                    }"
                android:textAppearance="@android:style/TextAppearance.
                    DeviceDefault" />

            <TextView
                android:id="@+id/descArea"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="@dimen/text_margin"
                android:textAppearance="@android:style/TextAppearance.
                    DeviceDefault" />

            <TextView
                android:id="@+id/gestArea"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="@dimen/text_margin"
                android:textAppearance="@android:style/TextAppearance.
                    DeviceDefault" />

            <!--<ImageView
                android:id="@+id/img"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:adjustViewBounds="true"
                android:contentDescription="@string/img_content" />-->

            <LinearLayout
                android:id="@+id/elencoTipi"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"></LinearLayout>

            <LinearLayout
                android:id="@+id/elencoFoto"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"></LinearLayout>

            <LinearLayout
                android:id="@+id/elencoExtra"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"></LinearLayout>
        </LinearLayout>
    </ScrollView>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:onClick="onAvviaNavigazione"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
```

```
            app:srcCompat="@android:drawable/ic_dialog_map" />

    </android.support.constraint.ConstraintLayout>
</layout>
```

Codice A.82: DettagliView

```
package it.dsantini.areediattesa.view;

import android.arch.lifecycle.Observer;
import android.arch.lifecycle.ViewModelProviders;
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.Locale;
import java.util.Optional;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.databinding.ActivityDettagliViewBinding;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Fotografia;
import it.dsantini.areediattesa.model.TipoEmergenza;
import it.dsantini.areediattesa.viewModel.StatoDettagli;
import it.dsantini.areediattesa.viewModel.impl.DettagliViewModelBase;

public class DettagliView extends AppCompatActivity implements Observer<
    AreaEmergenza> {
    private static final String CLASS = "DettagliView";

    private DettagliViewModelBase viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        viewModel = ViewModelProviders.of(this).get(DettagliViewModelBase.class);

        setContentView(R.layout.activity_dettagli_view);

        ActivityDettagliViewBinding binding = DataBindingUtil.setContentView(this, R
            .layout.activity_dettagli_view);
        binding.setViewModel(viewModel);

        viewModel.getArea().observe(this, this);
        viewModel.getStato().observe(this, new StatoDettagliObserver());
    }

    private class StatoDettagliObserver implements Observer<StatoDettagli> {
        @Override
        public void onChanged(@Nullable StatoDettagli stato) {
```

```
            final String TAG = "DettagliView::onChanged(StatoDettagli)";

            if (stato == null)
                Log.w(TAG, "Stato nullo");
            else if (stato == StatoDettagli.pronto) {
                Log.v(TAG, "Pronto");
            } else {
                try {
                    int id = getResources().getIdentifier(stato.name(), "string",
                        getPackageName());
                    Snackbar.make(findViewById(R.id.dettagli), getString(id),
                        Snackbar.LENGTH_LONG).show();
                } catch (Exception e) {
                    Log.e(TAG, "Errore nella visualizzazione di" + stato.toString()
                        , e);
                    Snackbar.make(findViewById(R.id.dettagli), getString(R.string.
                        possibile_errore_dettagli), Snackbar.LENGTH_SHORT).show();
                }
            }
        }
    }

    @Override
    public void onChanged(@Nullable AreaEmergenza area) {
        final String TAG = "DettagliView::onChanged(AreaEmergenza)";

        if (area == null) {
            Log.e(TAG, "Area da visualizzare nulla");
            Snackbar.make(findViewById(R.id.dettagli), getString(R.string.
                impossibile_visualizzare_area), Snackbar.LENGTH_INDEFINITE).show();
        } else {
            stampaArea(area);
        }
    }

    protected void stampaArea(@NonNull AreaEmergenza area) {
        final String TAG = "DettagliView::stampaArea";

        if (area.getNome().trim().isEmpty()) {
            TextView nomeView = findViewById(R.id.nomeArea);
            nomeView.setText(getString(R.string.nomeAssente));
        }

        Optional<String> optDesc = area.getDescrizione(),
                optGestore = area.getGestore();

        TextView desc = findViewById(R.id.descArea);
        if (optDesc.isPresent()) {
            desc.setText(optDesc.get());
            desc.setEnabled(true);
        } else
            desc.setEnabled(false);

        TextView gestoreView = findViewById(R.id.gestArea);
        if (optGestore.isPresent()) {
            gestoreView.setText(String.format(Locale.getDefault(), getString(R.
                string.gestore), optGestore.get()));
            gestoreView.setEnabled(true);
        } else
```

```
            gestoreView.setEnabled(false);

        LinearLayout elencoTipi = findViewById(R.id.elencoTipi);
        elencoTipi.removeAllViews();
        TipoEmergenza[] tipi = area.getEmergenze();
        for (TipoEmergenza tipo : tipi) {
            TextView tipoView = (TextView) getLayoutInflater().inflate(R.layout.
                dettagli_text_view, elencoTipi, false);
            String tipoTradotto;
            try {
                tipoTradotto = getString(getResources().getIdentifier(tipo.getNome()
                    , "string", getPackageName()));
            } catch (Exception e) {
                Log.e(TAG, "Errore nella traduzione del tipo" + tipo.toString(), e);
                tipoTradotto = tipo.getNome();
            }
            tipoView.setText(String.format(getString(R.string.tipoSupportato),
                tipoTradotto));
            elencoTipi.addView(tipoView);
        }

        LinearLayout elencoFoto = findViewById(R.id.elencoFoto);
        for (Fotografia foto : area.getFotografie()) {
            View view;
            if (foto.getTitolo().isPresent()) {
                TextView fotoView = (TextView) getLayoutInflater().inflate(R.layout.
                    dettagli_text_view, elencoFoto, false);
                fotoView.setText(foto.getTitolo().get());
                fotoView.setCompoundDrawablesWithIntrinsicBounds(null, foto.
                    getImmagine(), null, null);
                view = fotoView;

            } else {
                ImageView fotoView = new ImageView(this);
                fotoView.setImageDrawable(foto.getImmagine());
                view = fotoView;
            }
            elencoFoto.addView(view);
        }

        LinearLayout elencoExtra = findViewById(R.id.elencoExtra);
        elencoExtra.removeAllViews();
    }

    public void onAvviaNavigazione(View sorgenteClick) {
        viewModel.onAvviaNavigazione();
    }
}
```

AreaAttesaView

Codice A.83: activity_dettagli_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="@dimen/text_margin"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault" />
```

Codice A.84: AreaAttesaView

```
package it.dsantini.areediattesa.view;

import android.support.annotation.NonNull;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.Locale;
import java.util.Optional;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.impl.AreaAttesa;

public class AreaAttesaView extends DettagliView {
    @Override
    protected void stampaArea(@NonNull AreaEmergenza area) {
        super.stampaArea(area);
        final String TAG = "AreaAttesaView::stampaArea";

        if (!(area instanceof AreaAttesa))
            Log.e(TAG, "Area non istanza AreaAttesa, non faccio nulla");
        else {
            AreaAttesa aa = (AreaAttesa) area;
            LinearLayout elencoExtra = findViewById(R.id.elencoExtra);

            Optional<Integer> optCapienza = aa.getCapienza();
            if (optCapienza.isPresent()) {
                TextView capienza = (TextView) getLayoutInflater().inflate(R.layout.
                    dettagli_text_view,elencoExtra);
                capienza.setText(String.format(Locale.getDefault(), getString(R.
                    string.capienza), optCapienza.get()));
                elencoExtra.addView(capienza);
            }

            Optional<Boolean> optCoperto = aa.isAlCoperto();
            if (optCoperto.isPresent()) {
                TextView coperto = (TextView) getLayoutInflater().inflate(R.layout.
                    dettagli_text_view,elencoExtra,false);
                coperto.setText(optCoperto.get() ? getString(R.string.al_coperto) :
                    getString(R.string.all_aperto));
                elencoExtra.addView(coperto);
            }
        }
    }
}
```

Per tornare al testo: 4.2.5

A.3.7 Configurazione

Codice A.85: config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools" tools:ignore="PxUsage">
    <!-- SORGENTE AREE DA UTILIZZARE -->
    <string name="sorgenteAree" translatable="false">it.dsantini.areediattesa.rete.
        impl.SorgenteAreeOverpass</string>
```

```
    <!--<string name="sorgenteAree" translatable="false">it.dsantini.areediattesa.
        rete.mock.SorgenteAreeMock</string>-->
    <!--<string name="sorgenteAree" translatable="false">it.dsantini.areediattesa.
        rete.mock.SorgenteAreeFallback</string>-->

    <!-- PERSISTENZA DA UTILIZZARE -->
    <string name="persistenza" translatable="false">it.dsantini.areediattesa.
        persistenza.impl.PersistenzaRoom</string>
    <!--<string name="persistenza" translatable="false">it.dsantini.areediattesa.
        persistenza.mock.PersistenzaMock</string>-->
    <!--<string name="persistenza" translatable="false">it.dsantini.areediattesa.
        persistenza.mock.PersistenzaFallback</string>-->

    <!-- ETA MASSIMA DEI GRUPPI DI AREE -->
    <integer name="maxDays">100</integer>

    <!-- MAPPA INIZIALE -->
    <!-- ROMA
    <dimen name="latInit">41.89331</dimen>
    <dimen name="lonInit">12.48293</dimen>
    <dimen name="zoomInit">12</dimen>
    -->
    <!-- BOLOGNA -->
    <dimen name="latInit">44.49427</dimen>
    <dimen name="lonInit">11.34265</dimen>
    <dimen name="zoomInit">14</dimen>
    <!-- ZOLINO
    <dimen name="latInit">44.36555</dimen>
    <dimen name="lonInit">11.69438</dimen>
    <dimen name="zoomInit">17</dimen>
    -->

</resources>
```

AreaViewMapping

Codice A.86: areaViewMapping.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <string-array name="areaViewMapping">
        <!-- <item>AREA-VIEW</item> -->
        <!-- Fallback
        <item>it.dsantini.areediattesa.model.impl.AreaAttesa-it.dsantini.
            areediattesa.view.DettagliView</item>
        -->
        <item>it.dsantini.areediattesa.model.impl.AreaAttesa-it.dsantini.
            areediattesa.view.AreaAttesaView</item>
    </string-array>
</resources>
```

Codice A.87: AreaViewMapping

```
package it.dsantini.areediattesa.viewModel.helper;

import android.support.annotation.NonNull;

import java.util.Map;

import it.dsantini.areediattesa.model.AreaEmergenza;
```

```
import it.dsantini.areediattesa.view.DettagliView;

public interface AreaViewMapping {
    @NonNull
    Map<Class<? extends AreaEmergenza>, Class<? extends DettagliView>> getMap();
}
```

Codice A.88: AreaViewMappingConfig

```
package it.dsantini.areediattesa.viewModel.helper.impl;

import android.content.Context;
import android.content.res.Resources;
import android.support.annotation.NonNull;
import android.util.Log;

import java.util.HashMap;
import java.util.Map;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.view.DettagliView;
import it.dsantini.areediattesa.viewModel.helper.AreaViewMapping;

public class AreaViewMappingConfig implements AreaViewMapping {
    private static final String TAG = "AreaViewMappingConfig";

    @NonNull
    private final Map<Class<? extends AreaEmergenza>, Class<? extends DettagliView>>
        map;

    public AreaViewMappingConfig(@NonNull Context context) {
        map = new HashMap<>();

        Resources r = context.getResources();
        String[] mapping = r.getStringArray(R.array.areaViewMapping);
        for (String s : mapping) {
            Log.v(TAG, s);
            String[] split = s.split("-");
            if (split.length == 2) {
                try {
                    Class<? extends AreaEmergenza> area = Class.forName(split[0]).
                        asSubclass(AreaEmergenza.class);
                    Class<? extends DettagliView> view = Class.forName(split[1]).
                        asSubclass(DettagliView.class);
                    map.put(area, view);
                } catch (ClassNotFoundException e) {
                    Log.e(TAG, "Mappatura area-view sbagliata (classe non trovata): "
                        + split[0] + " -> " + split[1], e);
                } catch (ClassCastException e) {
                    Log.e(TAG, "Mappatura area-view sbagliata (classe non
                        accettabile): " + split[0] + " -> " + split[1], e);
                }
            }
        }
    }

    @Override
    @NonNull
```

```
    public Map<Class<? extends AreaEmergenza>, Class<? extends DettagliView>> getMap
        () {
        return map;
    }
}
```

Per tornare al testo: 4.2.4

A.4 Collaudo moduli

A.4.1 Unit Test

Model

Codice A.89: MappaBaseTest

```
package it.dsantini.areediattesa.model;

import org.junit.Test;

import java.time.LocalDateTime;

import it.dsantini.areediattesa.model.AreaEmergenza;
import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.GruppoAree;
import it.dsantini.areediattesa.model.Mappa;
import it.dsantini.areediattesa.model.impl.BoundingBoxBase;
import it.dsantini.areediattesa.model.impl.GruppoAreeBase;
import it.dsantini.areediattesa.model.impl.MappaBase;

public class MappaBaseTest {
    private final Confine confineA, confineB;
    private final GruppoAree gruppoA, gruppoB;
    private final GruppoAree[] gruppi;

    public MappaBaseTest() {
        confineA = new BoundingBoxBase(0, 1, 0, 1, "TestA");
        confineB = new BoundingBoxBase(1, 2, 1, 2, "TestB");
        gruppoA = new GruppoAreeBase(new AreaEmergenza[]{}, confineA, false,
            LocalDateTime.now());
        gruppoB = new GruppoAreeBase(new AreaEmergenza[]{}, confineB, false,
            LocalDateTime.now());
        gruppi = new GruppoAree[]{gruppoA, gruppoB};
    }

    @Test
    public void ok() {
        new MappaBase(gruppi);
    }

    @Test(expected = IllegalArgumentException.class)
    public void gruppiNull() {
        new MappaBase(null);
    }

    @Test(expected = IllegalArgumentException.class)
    public void gruppoNull() {
        new MappaBase(new GruppoAree[]{gruppoA, null});
    }
```

```
    @Test(expected = IllegalArgumentException.class)
    public void gruppoRipetizione() {
        new MappaBase(new GruppoAree[]{gruppoA, gruppoB, gruppoA});
    }
}
```

Codice A.90: BoundingBoxBaseTest

```
package it.dsantini.areediattesa.model;

import junit.framework.Assert;

import org.junit.Test;

import it.dsantini.areediattesa.model.Confine;
import it.dsantini.areediattesa.model.impl.BoundingBoxBase;

public class BoundingBoxBaseTest {
    private final Confine contenuto, contenutoClone, contenente;

    public BoundingBoxBaseTest() {
        contenuto = new BoundingBoxBase(1, 2, 0, 1, "a");
        contenutoClone = new BoundingBoxBase(1, 2, 0, 1, "a");
        contenente = new BoundingBoxBase(0, 3, -1, 2, "b");
    }

    public void nomeNull() {
        Confine c = new BoundingBoxBase(0, 1, 0, 1, null);
        Assert.assertFalse(c.getNome().isPresent());
    }

    public void nomeNonNull() {
        Confine c = new BoundingBoxBase(0, 1, 0, 1, "Wololoooo");
        Assert.assertTrue(c.getNome().isPresent());
    }

    @Test(expected = IllegalArgumentException.class)
    public void latUguale() {
        new BoundingBoxBase(0, 0, 0, 1, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void lonUguale() {
        new BoundingBoxBase(0, 1, 0, 0, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void latInvertita() {
        new BoundingBoxBase(1, 0, 0, 1, "a");
    }

    @Test
    public void lonInvertita() {
        new BoundingBoxBase(0, 1, 1, 0, "a");
    }

    @Test
    public void latTotale() {
        new BoundingBoxBase(-90, 90, 0, 1, "a");
```

```
    }

    @Test
    public void lonTotale() {
        new BoundingBoxBase(0, 1, -180, 180, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void latAlta() {
        new BoundingBoxBase(0, 91, 0, 1, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void latBassa() {
        new BoundingBoxBase(-91, 0, 0, 1, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void lonAlta() {
        new BoundingBoxBase(0, 1, 0, 181, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void lonBassa() {
        new BoundingBoxBase(0, 1, -181, 0, "a");
    }

    @Test(expected = IllegalArgumentException.class)
    public void contieneNull() {
        contenuto.contiene(null);
    }

    @Test
    public void contieneUguale() {
        Assert.assertTrue(contenuto.contiene(contenuto));
    }

    @Test
    public void contieneSemplice() {
        Assert.assertTrue(contenente.contiene(contenuto));
    }

    @Test
    public void contenutoSemplice() {
        Assert.assertFalse(contenuto.contiene(contenente));
    }

    @Test
    public void contenutoUguale() {
        Assert.assertTrue(contenuto.contiene(contenutoClone));
        Assert.assertTrue(contenutoClone.contiene(contenuto));
    }
}
```

Codice A.91: NodoBaseTest

```
package it.dsantini.areediattesa.model;

import org.junit.Test;
```

```
import it.dsantini.areediattesa.model.impl.NodoBase;

public class NodoBaseTest {

    @Test
    public void posPos() {
        new NodoBase(10, 10);
    }

    @Test
    public void posNeg() {
        new NodoBase(10, -10);
    }

    @Test
    public void negPos() {
        new NodoBase(-10, 10);
    }

    @Test
    public void negNeg() {
        new NodoBase(-10, -10);
    }

    @Test(expected = IllegalArgumentException.class)
    public void latBassa() {
        new NodoBase(-91, 10);
    }

    @Test(expected = IllegalArgumentException.class)
    public void latAlta() {
        new NodoBase(91, 10);
    }

    @Test(expected = IllegalArgumentException.class)
    public void lonBassa() {
        new NodoBase(10, -181);
    }

    @Test(expected = IllegalArgumentException.class)
    public void lonAlta() {
        new NodoBase(10, 181);
    }
}
```

Codice A.92: PoligonoTest

```
package it.dsantini.areediattesa.model;

import org.junit.Test;

import it.dsantini.areediattesa.model.impl.NodoBase;
import it.dsantini.areediattesa.model.impl.Poligono;

public class PoligonoTest {
    private final Nodo a, b, c, d;

    public PoligonoTest() {
```

```
        a = new NodoBase(0,0);
        b = new NodoBase(0,1);
        c = new NodoBase(1,1);
        d = new NodoBase(1,0);
    }

    @Test(expected = IllegalArgumentException.class)
    public void nullo() {
        new Poligono(null);
    }

    @Test(expected = IllegalArgumentException.class)
    public void vuoto() {
        new Poligono(new Nodo[0]);
    }

    @Test(expected = IllegalArgumentException.class)
    public void uno() {
        new Poligono(new Nodo[]{a});
    }

    @Test(expected = IllegalArgumentException.class)
    public void due() {
        new Poligono(new Nodo[]{a,b});
    }

    @Test
    public void tre() {
        new Poligono(new Nodo[]{a,b,c});
    }

    @Test
    public void quattro() {
        new Poligono(new Nodo[]{a,b,c,d});
    }

    @Test(expected = IllegalArgumentException.class)
    public void ripetuto() {
        new Poligono(new Nodo[]{a,b,c,a});
    }

    @Test(expected = IllegalArgumentException.class)
    public void nodoNullo() {
        new Poligono(new Nodo[]{a,b,c,null});
    }


}
```

Per tornare al testo: 4.3

A.4.2 Instrumented Unit Test

AreaViewMappingConfig

Codice A.93: AreaViewMappingConfigTest

```
package it.dsantini.areediattesa.view;

import android.content.Context;
import android.support.annotation.NonNull;
```

```
import android.support.test.InstrumentationRegistry;
import android.util.Log;

import junit.framework.Assert;

import org.junit.Test;

import it.dsantini.areediattesa.R;
import it.dsantini.areediattesa.model.impl.AreaAttesa;
import it.dsantini.areediattesa.viewModel.helper.AreaViewMapping;
import it.dsantini.areediattesa.viewModel.helper.impl.AreaViewMappingConfig;

public class AreaViewMappingConfigTest {

    @NonNull
    private final Context context;
    @NonNull
    private final AreaViewMapping m;

    public AreaViewMappingConfigTest() {
        context = InstrumentationRegistry.getTargetContext();
        m = new AreaViewMappingConfig(context);
    }

    @Test
    public void conteggio() {
        final String TAG = "AreaViewMappingConfigTest::conteggio";
        String[] array = context.getResources().getStringArray(R.array.
            areaViewMapping);
        int mappa = m.getMap().size(), config = array.length;
        Log.i(TAG, "Elementi nella configurazione: " + config);
        Log.i(TAG, "Elementi nella mappa: " + mappa);
        Assert.assertEquals(config, mappa);
    }

    @Test
    public void attesa() {
        Assert.assertTrue(m.getMap().containsKey(AreaAttesa.class));
    }
}
```

Per tornare al testo: 4.3

A.5 Integrazione

Codice A.94: build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "it.dsantini.areediattesa"
        minSdkVersion 26
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
```

```
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), '
                proguard-rules.pro'
        }
    }
    dataBinding {
        enabled = true
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:design:27.1.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core
        :3.0.2'

    // Google Maps SDK
    implementation 'com.google.android.gms:play-services-maps:16.1.0'

    // Android Architecture Components
    def lifecycle_version = "1.1.1"
    implementation "android.arch.lifecycle:viewmodel:$lifecycle_version"
    implementation "android.arch.lifecycle:extensions:$lifecycle_version"
    annotationProcessor "android.arch.lifecycle:compiler:$lifecycle_version"
    testImplementation "android.arch.core:core-testing:$lifecycle_version"

    // Room - Libreria per la persistenza
    //def room_version = "2.1.0-alpha04"
    //implementation "androidx.room:room-runtime:$room_version"
    //testImplementation "androidx.room:room-testing:$room_version"
    def room_version="1.1.1"
    implementation "android.arch.persistence.room:runtime:$room_version"
    //annotationProcessor "android.arch.persistence.room:compiler:$room_version" //
        TODO scommentare dopo avere completato PersistenzaRoom
    testImplementation "android.arch.persistence.room:testing:$room_version"
    androidTestImplementation "android.arch.persistence.room:testing:$room_version"

    // Retrofit - Libreria per la fruizione di API REST
    def retrofit_version = "2.5.0"
    implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
    implementation "com.squareup.retrofit2:converter-gson:$retrofit_version"

    // ThreeTenABP - LocalDateTime backport per API level < 26
    //implementation 'com.jakewharton.threetenabp:threetenabp:1.1.2'

}
```

Codice A.95: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.dsantini.areediattesa">

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

```
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the
                 APK.
             You need a different API key for each encryption key, including the
                 release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/
                  and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".view.GestioneView"
            android:label="@string/title_activity_gestione_view"></activity>
        <activity
            android:name=".view.MappaView"
            android:label="@string/title_activity_mappa_view">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".view.DettagliView"
            android:label="@string/title_activity_dettagli_view" />
        <activity
            android:name=".view.AreaAttesaView"
            android:label="@string/title_activity_area_attesa_view" />
    </application>

</manifest>
```

Per tornare al testo: 5

Ringraziamenti

Vorrei ringraziare il prof. Enrico Denti e l'ing. Ambra Molesini, relatore e correlatrice di questa tesi, per la disponibilità e precisione dimostratemi durante tutto il periodo di stesura. Senza di Voi questo lavoro non avrebbe preso vita!

Un grande ringraziamento va a mia madre e mio padre che, con il loro dolce e instancabile sostegno, sia morale che economico, mi hanno permesso di arrivare fin qui, contribuendo alla mia formazione personale.

Tutti i miei amici (e con tutti, intendo proprio tutti!) hanno avuto un peso determinante nel conseguimento di questo risultato, punto di arrivo e contemporaneamente di partenza della mia vita. Grazie per aver condiviso con me in questi anni le esperienze più importanti, vi voglio bene!

In particolare voglio ringraziare Chiara per gli utili consigli che mi ha dato nella stesura della sezione sul GDPR.

Un sentito grazie a tutti!

Daniele Santini