Ricerca di soggetti qualificati
per un servizio di estensione e manutenzione dell'ecosistema digitale
di Muoversi in Toscana

Lotto unico - CIG: 8184985F48
ID gara: 7667842
Codice commessa: PROG/72

Allegato 2 – Specifiche Tecniche

0.1 Introduzione

a. Scopo e ambito applicativo del Sistema

L'ambito applicativo del sistema Muoversi In Toscana è il trasporto pubblico locale.
Il sistema raccoglie ed elabora i dati delle agenzie di trasporto che operano nella regione, produce contenuti redazionali relativi alla mobilità, ed espone e propri contenuti agli utenti finali attraverso applicazioni mobile e applicazioni web.
Questo documento descrive le caratteristiche del sistema e delle singole componenti software che lo compongono.

b. Descrizione dei principali moduli componenti

La piattaforma Muoversi In Toscana può essere rappresentata dal seguente diagramma.

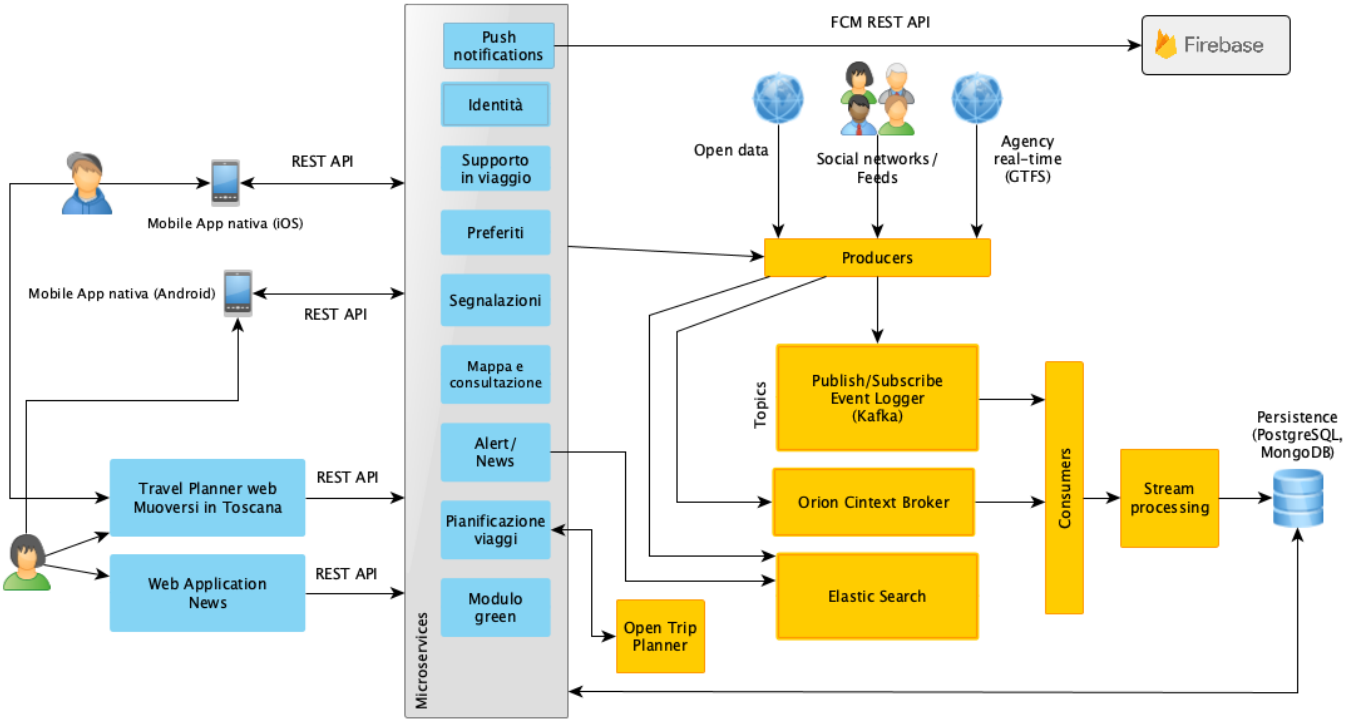

Fig. 1 - Architettura della piattaforma Muoversi In Toscana

L'intera piattaforma è composta da quattro componenti principali, che verranno di seguito introdotti.

1 Servizi di backend

Il sistema Muoversi In Toscana è stato progettato come un'architettura distribuita, basata sui microservizi e il paradigma dell'Event Sourcing. La logica applicativa è delegata ai singoli servizi, il cui isolamento rende il sistema più performante, resiliente e scalabile.
Nella sezione dedicata verranno descritti i componenti software coinvolti, in particolare:

- L'Identity Management (IdM) *Keycloak*.
- L'API Gateway *Kong*.
- La piattaforma di streaming Apache *Kafka*.
- Il travel planner *OpenTripPlanner*.

- Il componente per la gestione, consumazione e produzione dell'informazione di contesto su larga scala *Orion Context Broker*.
- Il motore per la gestione di container software *Docker*.
- L'orchestratore *Kubernetes*.

2 App per smartphone

Per la gestione delle app mobile Android e iOS è stato scelto il framework *NativeScript*, in grado di produrre applicazioni native per entrambi i sistemi utilizzando una unica base di codice.
NativeScript utilizza le API JavaScript esposte da Android e iOS per accedere alle funzionalità native dei sistemi di riferimento, e produce quindi app completamente native, in grado di utilizzare le SDK Android e iOS e i componenti messi a disposizioni dalle rispettive piattaforme.
Nella sezione di questo documento dedicata alla app mobile, verranno descritte in dettaglio le caratteristiche del framework e della app.

3 Travel planner web

Per il travel planner web è stata scelta la webapp open source *Digitransit UI*.
Digitranist UI richiede una versione di OpenTripPlanner in grado di supportare il query language GraphQL, che è stata appositamente sviluppata per la piattaforma.
URL: https://www.muoversintoscana.it/gw/travelplanner

4 App per la gestione delle News

L'applicazione web per l'inserimento e la gestione delle news riguardanti la mobilità locale (in particolare di treni e traghetti) è stata sviluppata usando il framework open source *Vue.js* (https://vuejs.org). L'applicazione web interagisce con gli stessi microservizi dedicati alla gestione delle news che vengono interrogati dalle app mobile.
URL: https://www.muoversintoscana.it/news

0.2 Descrizione funzionale

Muoversi in Toscana si occupa di:

- Importare i feed GTFS delle agenzie di trasporto pubblico che operano nella regione, che contengono i dati relativi a fermate, orari programmati, linee, corse.
- Interrogare i feed sugli orari in tempo reale messi a disposizione dalle suddette agenzie, quando presenti.
- Integrare nella piattaforma i feed GTFS e i dati real time per elaborare informazioni aggiornate sulla pianificazione viaggi e sugli orari dei mezzi pubblici.
- Produrre contenuti redazionali riguardanti la mobilità regionale, attraverso una web app gestionale dedicata, in grado anche di generare notifiche push per gli utenti della app mobile.
- Elaborare l'insieme delle informazioni raccolte.
- Pubblicarle l'insieme delle informazioni sul trasporto pubblico regionale attraverso la app mobile e il travel planner web.
- Raccogliere informazioni sui luoghi, fermate, linee, corse preferite dagli utenti della app mobile, costruendo attraverso queste informazioni un profilo utente.
- Inviare agli utenti della app mobile notifiche push personalizzate su ritardi o variazioni del trasporto pubblico basandosi sul profilo dei singoli utenti.
- Fornire soluzioni di viaggio aggiornate che - quando l'informazione è presente - tengono conto degli orari in tempo reale, sia attraverso la app mobile che il travel planner web.
- Raccogliere le segnalazioni degli utenti sul trasporto pubblico locale attraverso il modulo segnalazioni della app mobile.
- Assegnare un punteggio green agli utenti della app mobile analizzando le soluzioni di viaggio selezionate.
- Elaborare una classifica green generale, mensile e settimanale degli utenti iscritti alla app mobile.

a. Flussi di dati ingresso e integrazioni con sistemi di terze parti

Muoversi In Toscana integra i dati di dieci agenzie per quattro diverse modalità di trasporto: autobus, treno, tram e traghetto. Di seguito verranno elencate per ciascuna agenzia le modalità di integrazione sia degli orari programmati che - quando presenti - degli orari in tempo reale.

1 Linee di trasporto pubblico, corse programmate

La maggior parte degli orari programmati di ciascuna agenzia vengono recuperati nel formato GTFS dalla directory dedicata al trasporto pubblico della piattaforma *open data* di Regione Toscana: http://dati.toscana.it/dataset/rt-oraritb
La piattaforma *open data* di Regione Toscana è basata su CKAN (https://ckan.org/), un framework open source per la gestione di documenti che espone delle API REST per la lettura delle directory e il recupero dei file. La directory dedicata al trasporto pubblico non richiede credenziali di accesso e può essere quindi letta da qualsiasi client.
Le agenzie CTT Nord, CAP e COPIT forniscono direttamente un loro feed GTFS standard, con trip id compatibili con quelli del loro feed GTFS real time. I loro dati non vengono di conseguenza letti dalla directory open data, ma da URL web che fanno riferimento risorse presenti sui server delle rispettive agenzie.

Agenzia	Mezzo di trasporto	Modalità di integrazione
TFT	Treno	Feed GTFS dalla directory open data di RT

Trenitalia	Treno	Feed GTFS dalla directory open data di RT
GEST	Tram	Feed GTFS dalla directory open data di RT
Toremar	Traghetto	Feed GTFS dalla directory open data di RT
ATAF	Bus	Feed GTFS dalla directory open data di RT
Tiemme	Bus	Feed GTFS dalla directory open data di RT
CTT Nord	Bus	Feed GTFS sui server dell'agenzia
CAP	Bus	Feed GTFS sui server dell'agenzia
COPIT	Bus	Feed GTFS sui server dell'agenzia

I feed GTFS così raccolti vengono utilizzati, insieme al grafo stradale OpenStreetMap di Regione Toscana in formato pbf disponibile alla url:
http://geodati.fmach.it/gfoss_geodata/osm/output_osm_regioni/toscana.pbf per produrre un graph.object compatibile con OpenTripPlanner versione 1.2.2.

2 Linee di trasporto pubblico, dati in real time

Le agenzie che forniscono dati in tempo reale sono Ataf, CTT Nord, CAP, COPIT e Tiemme.
Di queste, CTT Nord, CAP e COPIT forniscono un feed GTFS Real time standard, immediatamente integrabile in Open Trip Planner, mentre Ataf e Tiemme forniscono le informazioni sui prossimi passaggi alla fermata tramite web service in un formato proprietario.

Ataf

Ataf fornisce informazioni sui prossimi passaggi in tempo reale da una singola fermata, interrogando una URL http che prende come parametro il codice della fermata stessa, così come scritto nel feed GTFS programmato.
Ad esempio, con una chiamata GET a questa URL:

http://www.temporealeataf.it/WebServicePrevisioni/ServicePrevisioni.asmx/GetPrevisioniATAF?**codiceFermata=FM0165**

si ottengono i prossimi passaggi in tempo reale dalla fermata con codice FM0165, in formato xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfPrevisioneATAF xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://simonweb/WebServicePrevisioni/">
   <PrevisioneATAF>
      <Linea>27</Linea>
      <Destinazione>VINGONE</Destinazione>
      <Teorica>false</Teorica>
      <PrevisionePartenza>false</PrevisionePartenza>
      <OraArrivo>18:00:56</OraArrivo>
      <PrevisioneArrivo>5'</PrevisioneArrivo>
      <DescrizionePercorso>PONTIGNALE PACE MONDIALE-&gt;VINGONE</DescrizionePercorso>
      <CodiceBarrata>_</CodiceBarrata>
```

```
        <NumeroSociale>283</NumeroSociale>
    </PrevisioneATAF>
    <PrevisioneATAF>
        <Linea>27</Linea>
        <Destinazione>VINGONE</Destinazione>
        <Teorica>false</Teorica>
        <PrevisionePartenza>false</PrevisionePartenza>
        <OraArrivo>18:14:32</OraArrivo>
        <PrevisioneArrivo>19'</PrevisioneArrivo>
        <DescrizionePercorso>PONTIGNALE PACE MONDIALE-&gt;VINGONE</DescrizionePercorso>
        <CodiceBarrata>_</CodiceBarrata>
        <NumeroSociale>281</NumeroSociale>
    </PrevisioneATAF>
</ArrayOfPrevisioneATAF>
```

Attraverso il web service esposto da Ataf si ottengono le stesse informazioni che vengono mostrate nelle paline informative presenti in alcune fermate. Queste informazioni non possono essere integrate direttamente in OpenTripPlanner, non avendo nessun riferimento diretto al *trip,* ma solo alla linea. Vengono quindi mostrate direttamente nella app mobile come informazione aggiuntiva rispetto all'orario programmato. Nella pianificazione dei viaggi non viene tenuto conto degli orari in tempo reale forniti da Ataf.

Tiemme

Tiemme gestisce le linee urbane ed extraurbane delle province di Piombino, Grosseto, Siena e Arezzo. Le informazioni sugli orari in tempo reale di Tiemme vengono fornite attraverso web service di tipo SOAP. Ogni provincia ha un servizio dedicato per il tempo reale. Rispetto al servizio di Ataf, il web service Tiemme è più preciso nel fornire informazioni, in quanto indica nella risposta anche l'orario programmato delle corse, oltre a quello in tempo reale, rendendo possibile identificare con esattezza la corsa di riferimento e calcolare l'eventuale ritardo o anticipo sull'orario programmato. Allo stesso tempo, i web service Tiemme sono più complessi da interrogare, perché richiedono di risalire alla provincia di riferimento della fermata e vogliono come parametro anche un orario di riferimento (mentre Ataf assume che l'orario di riferimento sia sempre quello in cui viene effettuata la chiamata).
I web service Tiemme richiedono che l'indirizzo IP del client che effettua la chiamata sia inserito in una loro *white list*. E' pertanto possibile effettuare chiamate di prova solo da server autorizzati di Regione Toscana. Per una documentazione completa sulle modalità di chiamata e di risposta dei servizi in tempo reale Tiemme, si rimanda all'Allegato B - Siri User Manual 003.pdf.

CTT Nord, CAP, COPIT

CTT, CAP e COPIT forniscono gli orari in tempo reale utilizzando lo standard *GTFS Realtime,* e sono al momento della stesura di questo documento, le uniche agenzie a farlo.
I feed GTFS realtime possono essere integrati direttamente in OpenTripPlanner inserendo le loro URL nel file di configurazione del grafo utilizzato (nel caso di Muoversi In Toscana il file router-config-json all'interno della directory graphs/toscana).
Il vantaggio dei feed GTFS in formato real time, oltre alla loro immediata integrazione, è che le loro informazioni vengono incluse nel calcolo della pianificazione viaggi. Per le soluzioni di viaggio che coinvolgono autobus delle agenzie CTT Nord, CAP e COPIT vengono quindi restituite soluzioni che tengono conto dei passaggi in tempo reale alla fermate.
Perché le informazioni real time funzionino correttamente è necessario che ci sia una perfetta identità tra i *trip id* (ovvero gli identificativi dei singoli viaggi) del feed GTFS standard e di quello real time. Nel caso di CTT questa identità si trova solo se si usa il feed GTFS standard fornito direttamente dall'agenzia, e non quello elaborato da Regione Toscana e pubblicato nella directory open data. Questo rende necessario recuperare i feed GTFS standard di CTT, CAP e COPIT direttamente dai server dell'agenzia, a queste URL:

ID Agenzia	URL feed GTFS standard
CTT	https://avmdata-cttn.cttcompany.it/download/gtfs/estrazione_gtfs.zip
CAP	https://avmdata-cap.cttcompany.it/download/gtfs/estrazione_gtfs.zip
COPIT	https://avmdata-copit.cttcompany.it/download/gtfs/estrazione_gtfs.zip

Come nel caso di Tiemme, l'accesso a queste URL è consentito solo dagli IP inseriti nella *white list* delle rispettive agenzie. Di conseguenza è possibile scaricare questi feed solo dai server di Regione Toscana.

Anche per l'accesso ai feed real time è necessario effettuare la chiamata da un indirizzo IP abilitato. Per il resto, è sufficiente configurare le URL real-time corrette nel file di configurazione di OpenTripPlanner (per i dettagli sul file *router-config.json* si rimanda alla documentazione ufficiale di OpenTripPlanner stesso: http://docs.opentripplanner.org/en/latest/).

ID Agenzia	URL feed GTFS real time
CTT Nord	https://avmdata-cttn.cttcompany.it/download/gtfs/gtfs-rt-trip-updates.pb
CAP	https://avmdata-cap.cttcompany.it/download/gtfs/gtfs-rt-trip-updates.pb
COPIT	https://avmdata-copit.cttcompany.it/download/gtfs/gtfs-rt-trip-updates.pb

Perché i dati real-time vengano correttamente interpretati è necessario che ci sia corrispondenza tra tra gli identificativi dell'agenzia del feed standard e di quello real-time.

3 Informazioni e news di carattere redazionale

Le news della piattaforma Muoversi In Toscana sono alimentate da due fonti:

- La webapp nella quale la redazione di Muoversi In Toscana scrive le notizie riguardanti i treni e i traghetti attivi nella regione.
- Il canale Twitter ufficiale di Muoversi in Toscana (https://twitter.com/muoversintoscan)

Le news del canale Twitter vengono mostrate solo nella app mobile, quelle redazionali vengono pubblicate anche nel portale web Muoversi In Toscana (http://www.regione.toscana.it/speciali/muoversi-in-toscana)

Ciascuna delle due fonti viene gestita da un servizio dedicato.
Elastic Search viene usato come storage per entrambi i flussi informativi.
Il servizio *Kraken* si occupa di leggere il feed Twitter e recuperare le informazioni aggiornate.
Il servizio *Inquirer* si occupa di salvare, pubblicare, aggiornare ed eliminare le news redazionali.
Entrambi i servizi restituiscono le news paginate e consentono di operare un ricerca libera sul loro contenuto.

Per una documentazione sulle API dei suddetti servizi si rimanda all'***Appendice A - Manuale di riferimento delle API REST.***

4 Altri flussi e/o servizi di cui viene fatto uso

Servizio di geocoding e reverse geocoding

Sia la app mobile che il travel planner web di Muoversi In Toscana richiedono funzionalità di *geocoding* e *reverse geocoding* per la selezione dei luoghi preferiti (ad esempio Casa/Lavoro) che per la ricerca degli indirizzi.
Per il servizio di geocoding da app mobile è essenziale la funzionalità di *autocomplete*, ovvero la capacità di suggerire immediatamente indirizzi fin dalle prime lettere digitate nel campo di ricerca. Una funzione di *autocomplete* che abbia performance sufficienti richiede risorse software hardware che non erano disponibili tra i servizi di Regione Toscana (che offre servizi per il geocoding molto accurati, ma senza autocompletamento). Si è deciso così di appoggiarsi al servizio di geocoding offerto da Mapbox (https://www.mapbox.com/).

Le API di Mapbox per il geocoding richiedono la registrazione di un API Key e al momento della stesura di questo documento è gratuito fino a 100.000 richieste mensili.

L'unico parametro richiesto per le richieste di geocoding è il testo di ricerca. Le richieste vengono effettuate con una chiamata http di tipo GET in questa forma:

`/geocoding/v5/{endpoint}/{search_text}.json`

Nel contesto di Muoversi In Toscana sono importanti altri parametri facoltativi, in particolare:

bbox	Il *bounding box* (ovvero il rettangolo) entro il quale devono essere compresi i risultati della ricerca. Il *bounding box* viene indicato con quattro numeri separati da virgola: minLon, minLat, maxLon, maxLat. Nel caso di Muoversi In Toscana vengono richiesti risultati limitati al rettangolo in cui è iscritto il territorio della regione.
language	Specifica il linguaggio in cui si vogliono ricevere i risultati. Specificando italiano (IT) si riceve *piazza* invece di *square* e così via.
proximity	Ordina i risultati di risposta dando la precedenza a quelli più vicini al punto indicato come *proximity*, nella forma di una coppia di coordinate *longitudine, latitudine* separate da virgola. Nel caso di richieste provenienti da app mobile il valore di *proximity* è valorizzato con la posizione dell'utente.

Il *reverse geocoding* è utile per ricavare un indirizzo da un punto geografico. Nel caso della app mobile, le richieste di reverse geocoding vengono effettuate quando un utente fa un'azione di *tap* sulla mappa per impostare un luogo preferito. Nel caso del travel planner web l'utente può cliccare sulla mappa per impostare il luogo di partenza e di arrivo, e in questo caso con una richiesta di reverse geocoding si aggiunge una label al punto selezionato.
Le richieste di reverse geocoding vengono effettuate in questa forma:

`/geocoding/v5/{endpoint}/{longitude},{latitude}.json`

b. *Funzioni svolte dal Sistema*

1 Geolocalizzazione delle informazioni in ingress

Le informazioni geolocalizzate recepite e gestite dal sistema Muoversi In Toscana si possono dividere in tre categorie:

1. **Informazioni fornite dei feed GTFS delle agenzie di trasporto**
   Queste informazioni comprendono le posizioni di fermate e stazioni (fornite come punti) e i percorsi delle linee (fornite come *polyline*).

2. **Informazioni fornite dagli utenti**
   Questi dati comprendono i luoghi preferiti impostati dall'utente (ad esempio la posizione dell'abitazione o del luogo di lavoro), le richieste di pianificazione viaggi, la selezione di una soluzione di viaggio, e in generale la posizione in cui si trova l'utente nel momento in cui effettua particolari azioni, come ad esempio l'invio di una segnalazione.

3. **Informazioni presenti nelle news**
   Le news, sia social che redazionali, possono contenere informazioni su una particolare posizione (espressa come coppia di coordinate in formato WGS84) o riguardare una particolare linea.

2 Ricerca e scelta dei percorsi, visualizzazione

Per la funzionalità di pianificazione viaggi e in generale per tutte le richieste che riguardano i feed GTFS del trasporto pubblico è stato implementato un micro-servizio dedicato che fa da proxy verso OpenTripPlanner. Questo consente di rendere stabili le API indipendentemente dalla versione di OpenTripPlanner utilizzata e di gestire in modo univoco le richieste provenienti dai client web e mobile.
Tutte le richieste riguardanti routing, fermate, linee e corse vengono elaborate usando OpenTripPlanner, ma le risposte vengono estese con informazioni aggiuntive che OpenTripPlanner non sarebbe in grado di produrre, come ad esempio il coefficiente *green* delle soluzioni di viaggio.

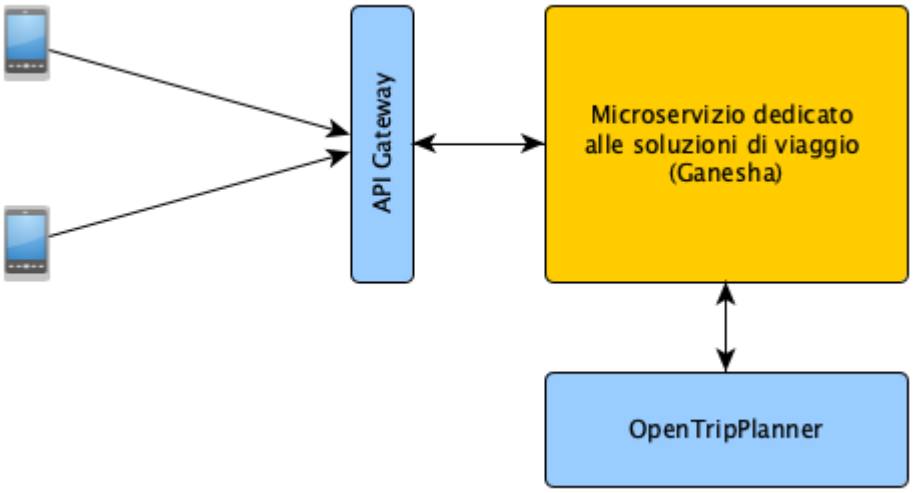

Fig. 2 - Servizio dedicato alla pianificazione viaggi

Il micro-servizio dedicato alla pianificazione viaggi e alle informazioni sul trasporto pubblico (Ganesha) è in grado di elaborare le seguenti richieste.

a. Pianificazione di un viaggio

Il client deve inviare come parametri posizione di partenza e di arrivo, come coppia di coordinate in formato WGS84 e la data di inizio viaggio.
Il servizio risponde con un elenco di soluzioni di viaggio fino a un massimo di cinque.
Nel caso di Muoversi In Toscana, le soluzioni di viaggio sono limitate per design a quelle che riguardano il trasporto pubblico (bus, treno, tram, traghetto).
La risposta è in formato JSON e contiene una lista di itineraries.
Ogni itinerary ha i seguenti attributi:

`co2EmissionsTotal`	Emissioni totali di CO2 del viaggio.
`co2SavedEmissionsPercentage`	Emissioni di CO2 risparmiate rispetto a uno stesso viaggio fatto in auto, in percentuale.
`co2SavedEmissionsTotal`	Il valore totale di CO2 risparmiata rispetto a uno stesso viaggio fatto in auto.
`duration`	La durata complessiva del viaggio.
`elevationGained`	Eventuale differenza di altezza del punto di arrivo rispetto al punto di partenza.
`elevationLost`	Eventuale differenza di altezza del punto di partenza rispetto al punto di arrivo.
`endDateTime`	Data e ora di fine viaggio
`endTime`	Orario di fine viaggio, in millisecondi
`greenPoints`	Punti green del viaggio
`greenScore`	Coefficiente green del viaggio
`legs`	Singole tratte che compongono il viaggio (vedi tabella successiva)
`startDateTime`	Data e ora di inizio viaggio
`startTime`	Orario di inizio viaggio, in millisecondi
`transfers`	Numero di cambi mezzi richiesti dalla soluzione di viaggio
`transitTime`	Tempo effettivo di viaggio sui mezzi pubblici
`waitingTime`	Tempi di attesa richiesto
`walkDistance`	Distanza da percorrere piedi
`walkLimitExceeded`	(boolean) Indica se il limita di distanza a piedi indicato nella configurazione di OpenTripPlanner è stato superato
`walkTime`	Tempo di percorrenza a piedi richiesto dalla soluzione di viaggio

Ogni soluzione di viaggio è composta da varie *leg*, ovvero da varie tratte.

Ogni tratta ha le seguenti proprietà:

`agencyTimeZoneOffset`	Eventuale differenza il fuso orario usato dall'agenzia e quello locale (naturalmente non si applica nel contesto Muoversi In Toscana)
`arrivalDelay`	Ritardo rispetto all'ora di arrivo programmata (valorizzato solo se sono presenti dati GTFS realtime)
`co2Emissions`	Emissioni CO2
`co2SavedEmissions`	Emissioni CO2 risparmiate rispetto alla stessa tratta percorsa in auto
`departureDelay`	Ritardo rispetto all'ora di partenza programmata (valorizzato solo se sono presenti dati GTFS realtime)
`distance`	Distanza della tratta
`duration`	Durata della tratta
`endDateTime`	Data e ora di fine tratta
`endTime`	Ora di fine tratta (in millisecondi)
`from`	Posizione di partenza, come oggetto che contiene un Point in formato WGS84 ed eventuali informazioni sulla fermata/stazione
`legGeometry`	Geometria della tratta (polilinea in formato binario)
`mode`	Modalità di trasporto utilizzata (nel contesto di Muoversi In Toscana può essere WALK, BUS, RAIL, TRAM, FERRY)
`realtime`	(boolean) Indica se gli orari sono in tempo reale
`route`	Eventuale linea del trasporto pubblico utilizzata
`startDateTime`	Data e ora di inizio tratta
`startTime`	Ora di inizio tratta (in millisecondi)
`steps`	Indicazioni puntuali per la tratta (solo per tratte a piedi)
`to`	Posizione di arrivo, come oggetto che contiene un Point in formato WGS84 ed eventuali informazioni sulla fermata/stazione
`transitLeg`	(boolean) Indica se la tratta è percorso con un mezzo pubblico (*false* in caso di tratte a piedi)

Questo insieme di informazioni viene visualizzato sia nella app mobile che nel travel planner web come un elenco di leg con relativi orari e mezzi di trasporto, affiancato da una rappresentazione del viaggio sulla mappa.

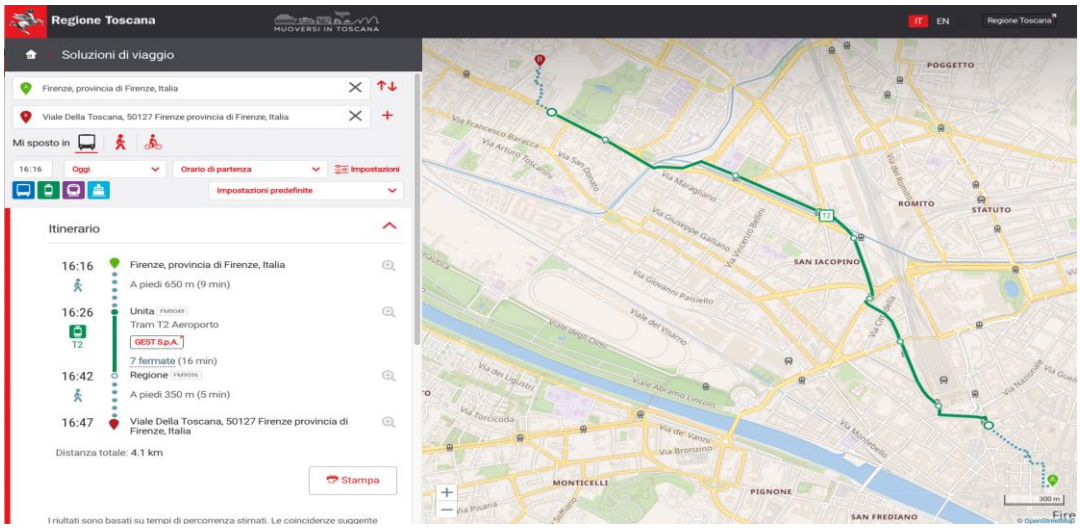

Fig. 3 - Soluzione di viaggio visualizzata nel travel planner web

b. Recupero delle fermate da bounding box

Il servizio di pianificazione viaggi restituisce le fermate del trasporto pubblico presenti una data area geografica, prendendo come parametri la coppia di coordinate (minLat, minLon, maxLat, maxLon) che definiscono il *bounding box* in cui si intendono visualizzare le fermate.
Questa richiesta viene eseguita alla prima apertura della mappa nella app mobile, e ad ogni evento di spostamento o di zoom della mappa stessa. La lista delle fermate viene quindi aggiornata ad ogni movimento della mappa, caricando sempre (e solo) le fermate che rientrano nell'area geografica inquadrata.

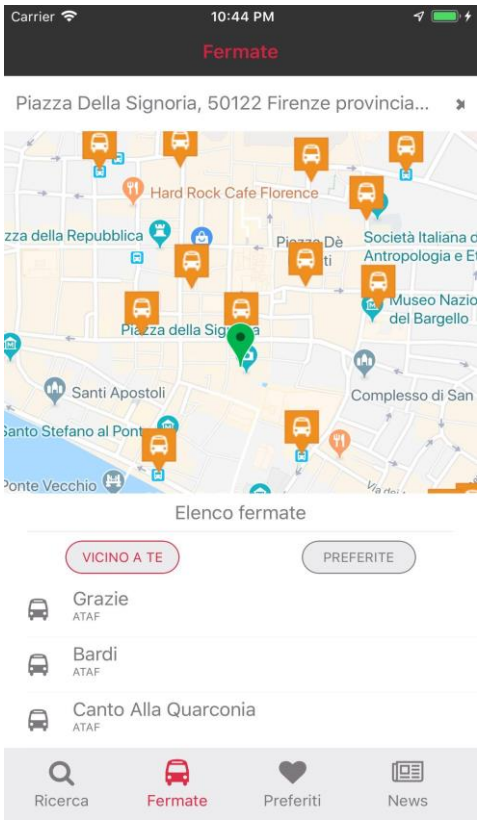

Fig. 4 - Fermate nella mappa della app mobile

c. Prossime partenze da una specifica fermata

Selezionando una fermata dalla mappa si esegue una richiesta di informazioni dettagliate per quella specifica fermata. Questa chiamata ha come parametro unico l'identificativo della fermata (che ha forme diverse da agenzia ad agenzia ma che è consistente con quanto importato dai loro feed GTFS). Il servizio risponde con le informazioni di base della fermata stessa (nome, codice, agenzia, posizione) e con una lista di *StopTime* (partenze dalla fermata) che comprendono:

- identificativo della linea (o della corsa);
- fermata di arrivo della corsa;
- orario di partenza programmato
- orario di partenza in tempo reale (se presente)

La lista di queste informazioni viene rappresentata come elenco nella app mobile, con gli orari in tempo reale evidenziati da una grafica diversa.

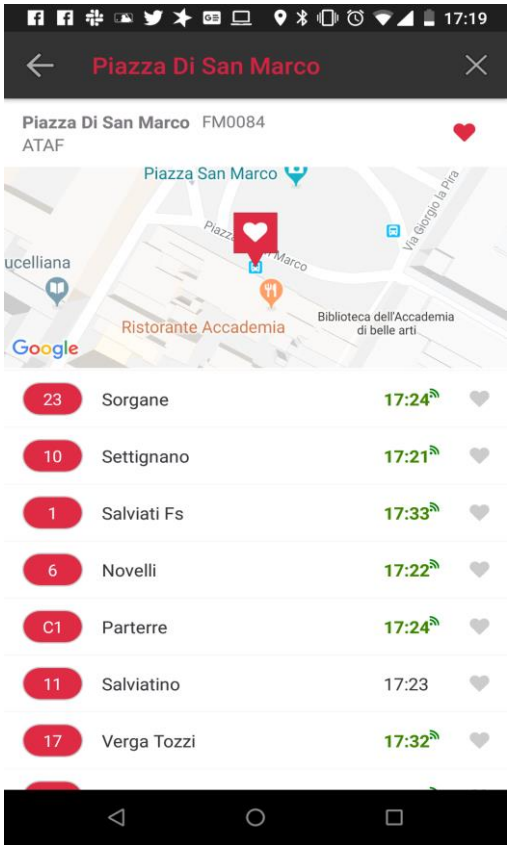

Fig. 5 - Prossime partenze da una fermata

d. Dettaglio di un trip

Selezionando una specifica corsa dalla lista delle prossime partenze, si effettua una richiesta di informazioni dettagliate sul trip (corsa) in questione. La richiesta riceve come unico parametro il *trip id*, ovvero l'identificativo di quella specifica corsa, così come viene fornito dai feed GTFS delle agenzie. La risposta contiene il percorso del trip (come polilinea in formato binario) e la lista delle fermate toccate dal trip con i relativi orari di partenza. Da notare che nel caso del trasporto ferroviario, il trip corrisponde a uno specifico treno.

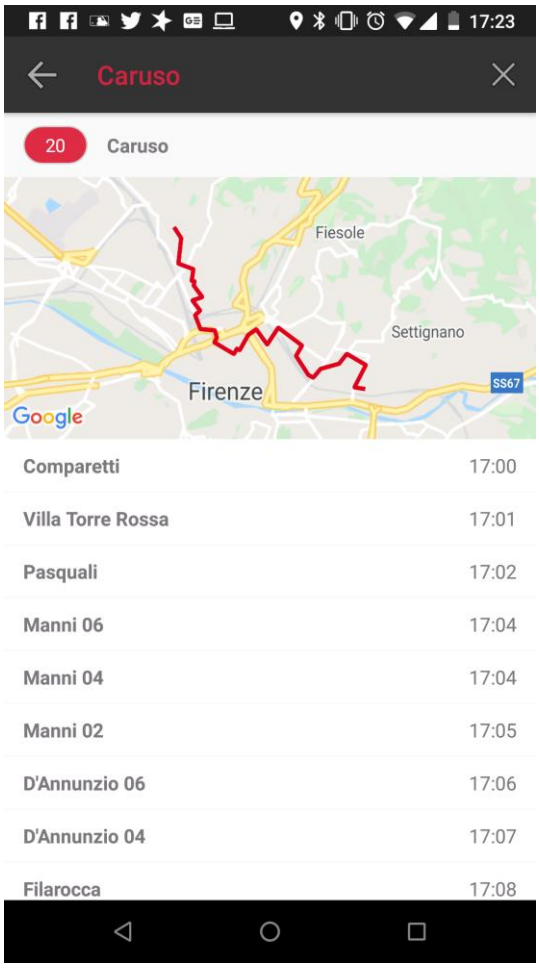

Fig. 6 - Dettaglio di un trip

Per una documentazione esaustiva sulle API esposte dal servizio di pianificazione viaggi si rimanda all'***Appendice A - Manuale di riferimento delle API REST***.

3 Registrazione e profilo utenti

Il processo di registrazione e autenticazione degli utenti viene gestito in Muoversi In Toscana dall'*identity and access manager Keycloak*, opportunamente configurato per permettere l'autenticazione utente tramite Facebook e Google, oltre che la registrazione e l'autenticazione tramite email e password appositamente creati per le applicazioni di Muoversi in Toscana. In quest'ultimo caso il corretto flusso di registrazione viene gestito prevedendo la procedura di verifica dell'email scelta, oltre alle operazioni di recupero e reimpostazione della password.
Essendo Keycloak conforme allo standard al momento più aggiornato per i servizi di Identity Management, ovvero OpenID Connect, garantisce l'interoperabilità con una moltitudine di servizi esterni, e risulta di conseguenza una scelta valida anche in prospettiva futura.
La costituzione di un unico punto di gestione dell'identità consente di avere un'unica banca dati degli utenti di Muoversi in Toscana, che possono essere associati a diversi ruoli applicativi.
Le categorie di utenti principali sono due: gli utenti comuni delle applicazioni sia mobile che web, con autorizzazioni per la consultazione delle informazioni, il salvataggio dei preferiti e l'invio delle segnalazioni; e gli utenti di back-office, che sono a loro volta suddivisi in utenti con permessi di inserimento e modifica delle News (per i treni e/o per i traghetti), e utenti che possono accedere ai sistemi di monitoraggio.
Questo sistema di single-sign-on riconosce l'utente fino a che il token di autenticazione sarà valido; quindi, ad esempio, un utente che ha eseguito l'accesso sul sito web di Muoversi in Toscana, verrà riconosciuto in

automatico all'interno dell'applicativo per la gestione delle News. La centralizzazione dell'autenticazione e della configurazione delle autorizzazioni, inoltre, alleggerisce l'onere a carico dei microservizi, in quanto il controllo della validità del token viene fatto a livello di API Gateway (il componente open source Kong controlla l'autorità emittente, in questo caso l'IdM Keycloak di Muoversi in Toscana, e la validità temporale del token), lasciando come unico compito del servizio il controllo del ruolo applicativo.

Gli utenti autenticati hanno la possibilità di inviare segnalazioni, inserire luoghi preferiti e selezionare fermate, linee o corse preferite. Con l'introduzione del modulo *green* gli utenti registrati possono anche salvare soluzioni di viaggio e acquisire un punteggio green che li fa concorrere nella classifica green della piattaforma.

4 Distribuzione delle notifiche

L'insieme di informazioni sopra descritto costruisce il *profilo dell'utente*. Ad ogni azione di selezione di un preferito, l'evento viene registrato nell'*event logger Apache Kafka* (descritto più in dettaglio nella sezione successiva), che lo rende persistente assegnandolo a uno specifico *topic*. I topic a cui l'utente è registrato determinano le notifiche personalizzate che l'utente potrà ricevere. Quando, ad esempio, il sistema registra un ritardo di uno specifico treno, tutti gli utenti che hanno indicato quel treno come preferito riceveranno una notifica push sul ritardo (meno che non hanno disabilitato la ricezione di notifiche, naturalmente).
Dall'applicazione web per la redazione di news è inoltre possibile assegnare un attributo *broadcast* a una singola notizia. Le notizie di tipo broadcast producono una immediata notifica push per tutti gli utenti dell'app al momento della loro pubblicazione. Si tratta di una funzionalità che viene usata in caso di scioperi o di disservizi di estensione tale da essere ritenuti di interesse generale.

0.3 Descrizione architetturale

a. Back-end

1 Architettura software complessiva

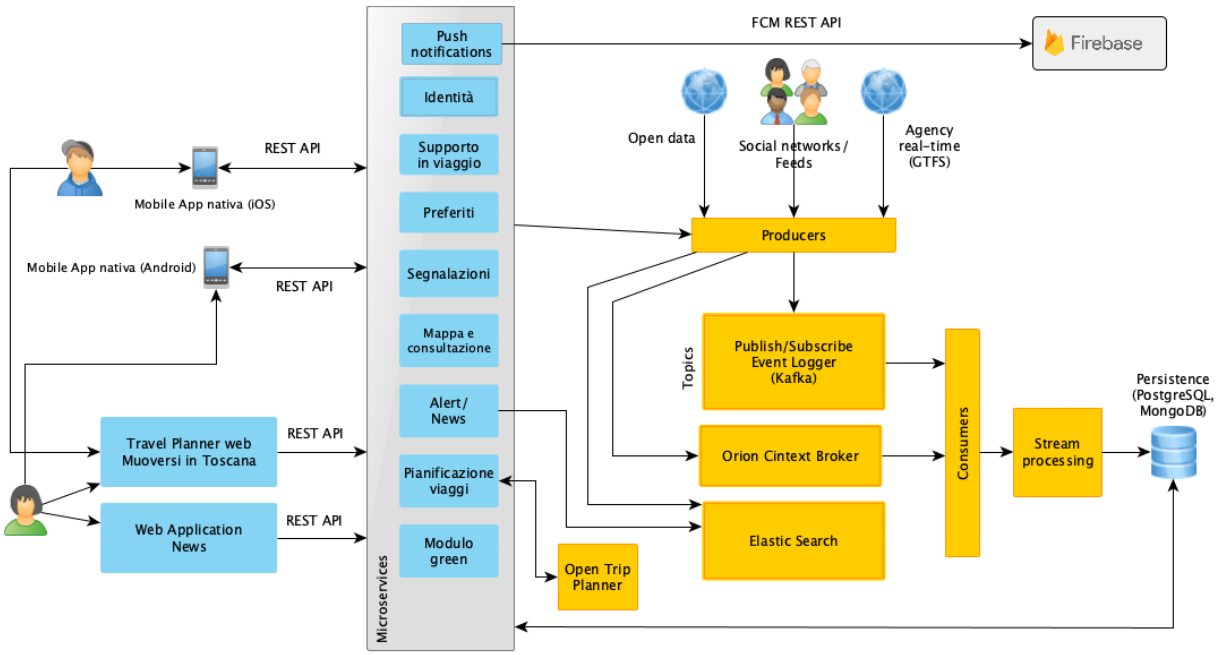

Fig. 7 - Architettura della piattaforma Muoversi In Toscana

La soluzione architetturale realizzata è incentrata sui microservizi e sulla piattaforma di streaming Apache Kafka, che costituisce l'*Event Logger*.

Ogni microservizio, alla ricezione di un comando da parte dell'esterno (applicazione o altro servizio), lo elabora generando un evento (comportandosi così da *producer*), che viene persistito su un topic di Kafka. Ogni consumer registrato sul topic potrà recuperare l'evento ed elaborarlo a sua volta. Questo meccanismo disaccoppia i produttori e i consumatori, permettendone l'evoluzione indipendente. Inoltre, la presenza di tutti gli eventi nell'event logger, permette l'operazione di log-replay per ricostruire informazioni rielaborando eventi passati, o implementare in un secondo momento elaborazioni non preventivate al momento della messa in produzione del sistema.
Abbracciando totalmente il paradigma dell'*event sourcing*, è stato implementato anche il pattern CQRS (*Command Query Responsibility Segregation*), in modo da separare i comandi che si traducono in richieste di scrittura dalle query (richieste di lettura).
Il microservizio, alla ricezione di un comando di scrittura, genera un evento, salvato sull'event logger. L'evento viene consumato da un consumer, che si occupa a sua volta di aggiornare lo stato dell'applicativo. Quando il servizio deve rispondere ad una richiesta di lettura, accede direttamente al motore di persistenza che contiene lo stato corrente, senza bisogno di ricostruirlo rielaborando tutti gli eventi.

Questo approccio garantisce delle ottime performance anche quando c'è un forte sbilanciamento tra richieste di lettura e di scrittura e pertanto è particolarmente indicato per un contesto applicativo come quello di Muoversi In Toscana, dove le interrogazioni generate dagli utenti tramite le applicazioni (consultazione mappa, orari, pianificazione) sono numericamente molto maggiori delle richieste di scrittura (impostazione preferiti, segnalazioni, salvataggio viaggi). L'incremento nel *throughput* che si ottiene, abbinato anche alla possibilità di distribuire gli event logger, garantisce una notevole scalabilità al sistema.

L'architettura logica basata sull'erogazione del servizio tramite microservizi, che vengono fruiti sia dalle "mobile application" (iOS e Android) che dalle web application (compatibili con i moderni browser) necessita di essere integrata con dei componenti software in grado di migliorare le funzionalità di monitoraggio del sistema e di analisi della grande mole di dati raccolti, in particolare quelli riguardanti l'interazione degli utenti con la piattaforma, le loro abitudini di viaggio, le loro preferenze riguardo ai mezzi di trasporto e i loro effettivi spostamenti. La soluzione architetturale implementate coinvolge i seguenti componenti software.

Keycloak IDM

Per l'autenticazione al sistema Muoversi In Toscana è stato scelto il protocollo di rete aperto e standard *OAuth 2.0* con lo strato di autenticazione *OpenID Connect (OIDC)*.
Con lo stesso IDM si può accedere alla app mobile o alla web app gestionale, sia registrando un account Muoversi In Toscana che usando un account esistente Google o Facebook.
Come flusso di autenticazione è stato scelto l'*authorization code flow* - uno standard per le app mobile - rappresentato dal seguente diagramma.

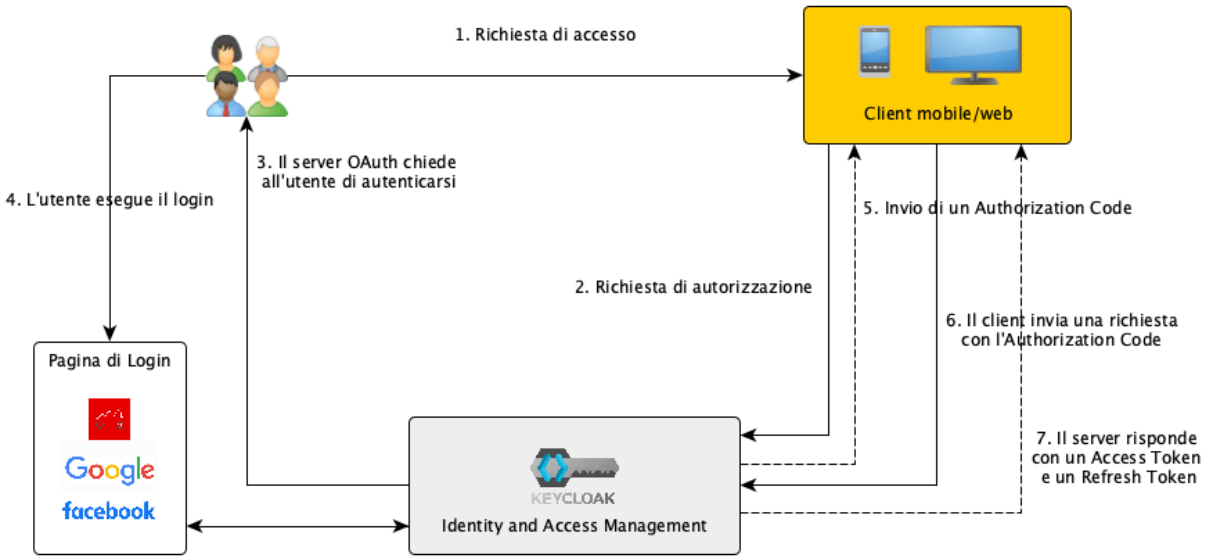

Fig. 8 - Authorization Code Flow

L'*authorization code flow* è una procedura standard per l'autenticazione da app mobile con *single sign on* e prevede questi passaggi:

- Quando un utente cerca di accedere a una risorsa protetta, il client invia una richiesta di autorizzazione all'IDM.
- L'IDM chiede all'utente di autenticarsi.
- L'utente esegue il Login (nel caso di Muoversi In Toscana è possibile registrare un account Muoversi In Toscana e autenticarsi con quello oppure usare un account Google o Facebook esistente).
- Se il login va a buon fine l'IDM invia un *authorization code* al client.
- Il client può usare l'*authorization code* che ha appena ricevuto per completare il processo di autenticazione.
- Quando riceve l'*authorization code* l'IDM risponde con un *Access Token* e un *Refresh Token*, con i quali il client può effettuare da quel momento in poi tutte le richieste che richiedono autenticazione.

L'*access token* è il token che serve ad effettuare tutte le richieste, e per motivi di sicurezza è opportuno che abbia una durata breve. Nel caso di Muoversi In Toscana, l'access token resta valido per 5 minuti dal momento della sua emissione.

Il *refresh token* serve a chiedere all'IDM un nuovo *access token* quando il vecchio *access token* è scaduto. Insieme all'access token, viene in questo caso restituito anche un nuovo refresh token, con scadenza aggiornata. La durata del *refresh token* è quindi quella che determina la durata della sessione utente: fino a che il client ha un *refresh token* valido, l'utente non ha bisogno di effettuare un nuovo login. Dal momento in cui non è più disponibile un *refresh token* valido, è necessario riavviare la procedura di *authorization code flow* ed eseguire un nuovo login.
Visto che dai dispositivi mobili la procedura di login può risultare complicata, è opportuno che il refresh token abbia una lunga durata. Nel caso di Muoversi In Toscana la scadenza del refresh token è impostata a 60 giorni dal momento della sua emissione. Questo vuol dire che se un utente apre la app almeno una volta ogni 60 giorni, non avrà mai bisogno di effettuare un nuovo login.
L'*access token*, oltre che a garantire l'autenticazione, contiene anche informazioni sul profilo utente (nome, email, ruolo ecc.). Se un utente è autenticato, il sistema è quindi in grado di registrare ogni sua richiesta e associarla al suo profilo, rendendo sempre più definito il profilo utente ad ogni utilizzo dell'app.
https://www.keycloak.org/

Apache Kafka

Già descritto nelle sezioni precedenti, l'event logger Kafka è un sistema open source di messaggistica istantanea, che consente la gestione di un elevato numero di operazioni in tempo reale da migliaia di client, sia in lettura che in scrittura. Apache Kafka permette operazioni di log-replay per ricostruire informazioni rielaborando eventi passati, o implementare in un secondo momento elaborazioni non preventivate al momento della messa in produzione del sistema.
https://kafka.apache.org/

OpenTripPlanner

Nell'attuale release di Muoversi In Toscana viene utilizzata una versione custom di **OpenTripPlanner** che fornisce le informazioni sulla pianificazione dei viaggi e sugli orari del trasporto pubblico sia alle app mobile che al travel planner web (realizzato a partire dalla componente open source Digitransit UI).
Questo consente di recepire i miglioramenti a livello di sicurezza, di performance e funzionali, come ad esempio la possibilità di configurare feed GTFS real time, che possono essere interrogati per fornire gli orari in tempo reale dei mezzi pubblici.
Per esempio, OpenTripPlanner importa in modo diretto i feed GTFS Real Time delle agenzie CTT, CAP e COPIT. Questo, oltre che mostrare all'utente gli orari in tempo reale dei mezzi pubblici delle suddette agenzie, consente di tenere conto dei dati real time anche nella pianificazione dei viaggi.
La natura modulare dell'architettura di Muoversi In Toscana e la presenza di uno strato intermedio di API lascia spazio anche a valutazioni future riguardanti l'eventuale sostituzione del componente OpenTripPlanner con soluzioni alternative che possano risultare più adatte allo scopo, senza richiedere alcuna modifica delle applicazioni che utilizzano il servizio.
https://www.opentripplanner.org/

OTP Graph Builder

Il travel planner per poter fornire informazioni sempre aggiornate e pertinenti ha bisogno di essere alimentato con una banca dati attualizzata.
E' stato quindi implementato il componente software **OTP Graph Builder**, un'applicazione Java che utilizza le librerie di OpenTripPlanner e si occupa in maniera programmata e monitorata di aggiornare:

- il grafo stradale sul quale calcolare i percorsi;
- la banca dati GTFS per le informazioni sul trasporto pubblico (agenzie di trasporto, fermate, orari), recuperando quotidianamente le informazioni dalla piattaforma OpenData di Regione Toscana (http://dati.toscana.it/dataset/rt-oraritb) o direttamente dalle agenzie di trasporto, come descritto nella sezione di questo documento dedicata ai feed GTFS.

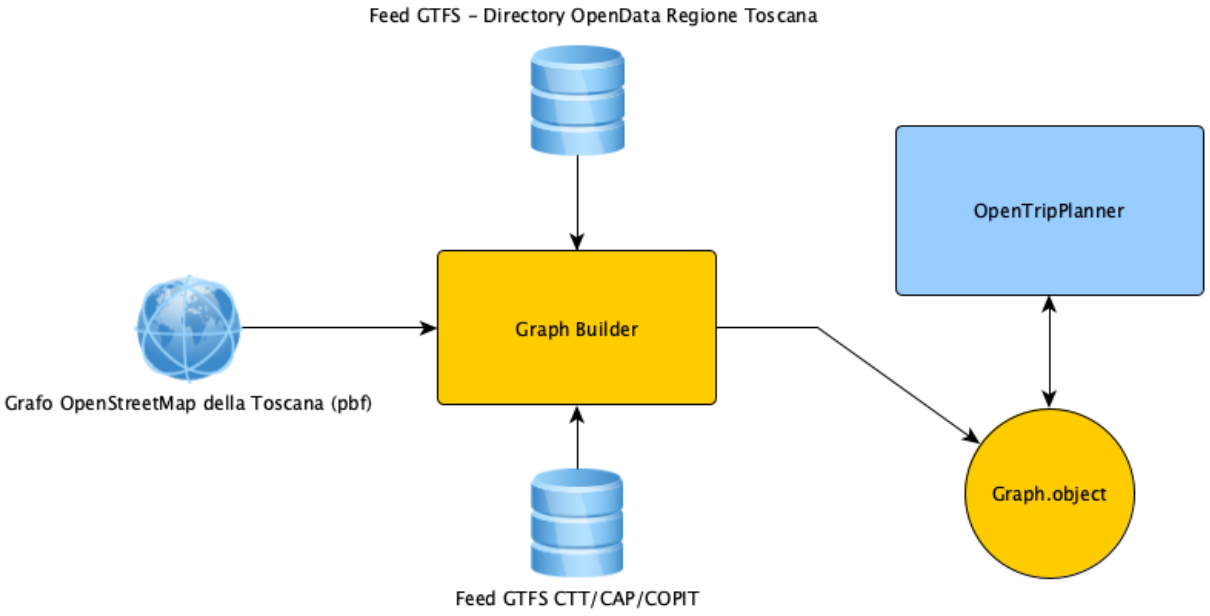

Fig. 9 - OTP Graph Builder

OTP Graph Builder recupera i feed GTFS standard dalle fonti disponibili e il grafo stradale OpenStreetMap della Toscana, li elabora e li unisce un unico file *graph.object* pronto per essere letto da OpenTripPlanner. Durante le operazioni di importazione, OTP Graph Builder verifica la validità e la consistenza dei dati importati, per evitare di produrre un graph.object che risulti poi inutilizzabile. Non è improbabile, infatti, che nella directory open data o nei server delle agenzie vengano pubblicati feed GTFS che contengono errori (ad esempio ID sbagliati o relazioni inesistenti). In questi casi è preferibile fornire all'utente un grafo meno aggiornato piuttosto che un grafo incompleto (anche perché l'aggiornamento è quotidiano).
In ogni caso, il monitoraggio e la frequenza di aggiornamento risultano fondamentali per evitare di fornire all'utente informazioni non attuali. L'operazione di costruzione del grafo è molto onerosa in termini di tempo e di risorse ed è programmata per essere eseguita nelle ore notturne.

Orion Context Broker

In Muoversi In Toscana è stato introdotto un *context broker* per dare alla piattaforma la possibilità di immagazzinare lo stato dei dati di contesto relativi ai ritardi dei trip, seguendo uno standard comune di interoperabilità. Il context broker è basato sul pattern *publish/subscribe*, dove i vari servizi che generano il flusso di dati possono essere visti come *pubblicatori*, mentre i sistemi e i servizi interessati ai dati come *consumatori*.
Il componente scelto per permettere la gestione, consumazione e produzione dell'informazione di contesto in larga scala è *Orion Context Broker*, sponsorizzato dalla fondazione FIWARE. L'informazione di contesto è rappresentata attraverso valori assegnati ad attributi che caratterizzano le entità che sono rilevanti nelle nostre applicazioni. Orion Context Broker permette di gestire queste informazioni utilizzando API standard secondo il paradigma REST.

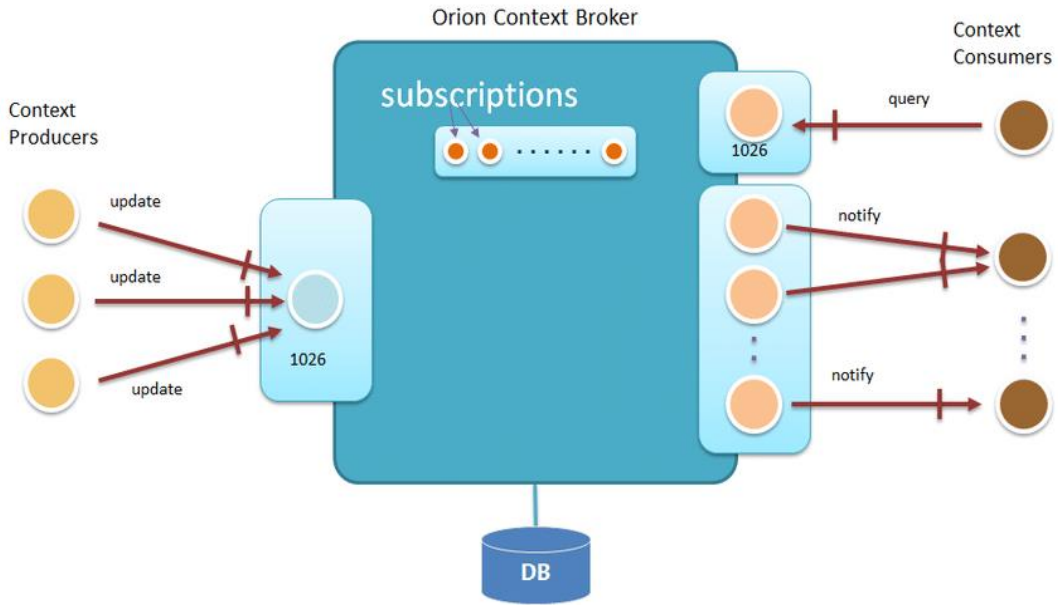

Fig. 10 - OTP Graph Builder

Una delle caratteristiche più importanti del Context Broker è che permette di modellare e ottenere l'accesso a informazioni di contesto in modo totalmente indipendente dalla sorgente dell'informazione stessa. Usando un'interfaccia standard, chiamata NGSI-10, basata su uno standard dell'*Open Mobile Alliance*, i client possono effettuare varie operazioni:

- registrare applicazioni che producono dati di contesto;
- aggiornare l'informazione di contesto;
- ricevere notifiche quando avvengono modifiche delle informazioni di contesto (ad esempio ritardi nei mezzi pubblici, incidenti o interruzioni);
- interrogare l'informazione di contesto presente in Orion Context Broker.

In modo simile ad Apache Kafka, Orion Context Broker viene utilizzato come un hub centrale dove i produttori di dati si registrano e pubblicano le loro informazioni, mentre i consumatori si sottoscrivono per ottenere le informazioni per loro rilevanti. Il ruolo centrale di Orion è quello di mantenere l'ultimo aggiornamento disponibile per ogni singola entità. A differenza di Apache Kafka, che ha lo scopo di raccogliere gli eventi nel loro ordine temporale e necessita di eventuali servizi dedicati per raccogliere lo stato di una specifica informazione in uno specifico momento (con le modalità descritte nelle sezioni precedenti), Orion gestisce in modo nativo lo stato delle sue entità e consente di fare interrogazioni dirette sui dati.
I dati di contesto di Orion vengono usati sia per dare informazioni in tempo reale all'interno dell'app che per inviare notifiche in caso di ritardi dei treni o delle linee preferite.
https://fiware-orion.readthedocs.io/en/master/

Elastic Search

ElasticSearch è un server di ricerca basato su Lucene, con capacità Full Text e supporto ad architetture distribuite. Tutte le funzionalità sono esposte tramite interfaccia RESTful e le informazioni sono gestite come documenti JSON. Nella piattaforma Muoversi In Toscana *ElasticSearch* viene usato per gestire il flusso informativo di notizie sia social (Twitter) che redazionali, sfruttando le sue caratteristiche di filtraggio, indicizzazione e ricerca delle informazioni.

Kong API Gateway

Un API gateway è un componente software che si occupa di garantire la sicurezza nell'accesso alle API del sistema. In sistemi distribuiti in cloud come Muoversi In Toscana, l'API gateway ha un ruolo cruciale nell'intercettare tutte le chiamate verso i servizi REST e garantire che passino solo i client autorizzati.
Kong è stato scelto come API Gateway per la piattaforma Muoversi In Toscana: si tratta di un software open source le cui caratteristiche principali sono la velocità, la scalabilità e la modularità, che permette la configurazione di plugin aggiuntivi per aumentarne le funzionalità. Il gateway opera in stretto contatto con il sistema di gestione dell'identità (L'IDM Keycloak): esso infatti valida i token di autenticazione forniti dai client, che sono basati sullo standard JWT, verificando che siano stati emessi da un'autorità fidata e che siano validi: questo controllo, eseguito a livello di gateway, permette di alleggerire il carico e la responsabilità dei microservizi, che si limitano, quando necessario, a effettuare un controllo sui ruoli applicativi, ma non hanno necessità di verificare l'autenticità dei token.
https://konghq.com/kong/

Docker

In Muoversi In Toscana viene utilizzato Docker come tecnologia standard per i container. Ogni microservizio viene pubblicato in un container Docker che contiene tutte le informazioni per l'esecuzione dell'applicativo in modo indipendente dal sistema in cui si trova. Lo stesso container viene quindi usato per il deploy in ambiente di staging o di produzione, senza che venga richiesta alcuna modifica al momento dell'operazione di *build.* Saranno i sistemi ospitanti, eventualmente, ad avere valori diversi per variabili d'ambiente richieste dai diversi servizi.
https://www.docker.com

Kubernetes

Per automatizzare il dispiegamento, la scalabilità e la gestione del ciclo di vita delle applicazioni che girano nei container, si utilizza il componente open source Kubernetes che, oltre ad essere il più diffuso per questo tipo di applicazioni, permette in maniera quasi trasparente di installare la piattaforma sia su cloud privato che pubblico.
https://kubernetes.io

2 Integrazione di sistema

Modalità di integrazione dei sistemi di terze parti

L'integrazione con i sistemi di terze parti, ad esempio i feed GTFS, gli orari in tempo reale, i ritardi dei treni, avviene importando i dati attraverso operazioni di *stream processing* in *Apache Kafka* o, con modalità analoghe, in *Orion Context Broker.*
I dati di input non elaborati vengono importati in *topics* e quindi aggregati, arricchiti o altrimenti trasformati in nuovi *topics* per ulteriori utilizzi o ulteriori elaborazioni. Ad esempio, una pipeline di elaborazione per l'importazione di notizie dai canali social sottopone a scansione il contenuto degli articoli dai feed RSS Twitter di Muoversi In Toscana e lo pubblica sul topic "twitter".

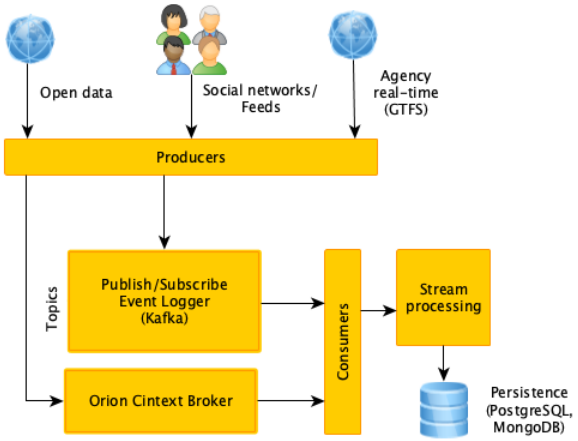

Fig. 11 - Stream processing dei dati di terze parti

Nel caso degli orari in tempo reale, il dato grezzo viene importato in opportuni *topics* e viene arricchito con informazioni relative a fermate/stazioni, linee e corse tenendo conto degli identificativi recuperati dai feed GTFS. Questo rende possibile l'associazione dei singoli ritardi ai *topics* che riguardano questi elementi. Di conseguenza un utente che ha scelto come preferito uno specifico treno in partenza sa una specifica stazione (e che quindi è stato iscritto ai *topics* che riguardano quel treno e quella stazione) potrà ricevere notifiche push quando eventi di ritardi riguardano questi particolari elementi.

Modalità di integrazione dei canali di output (p.es. notifiche push)

Sono stati sviluppati due microservizi dedicati, uno per la registrazione dei dispositivi e uno per l'invio delle notifiche.
Al momento dell'apertura della app mobile Muoversi In Toscana l'identificativo del dispositivo viene registrato su Firebase per la ricezione di notifiche dal servizio dedicato. Firebase restituisce una chiave univoca per ogni dispositivo.
Quando un utente esegue l'accesso, il suo profilo viene automaticamente associato alla chiave del suo dispositivo (o dei suoi dispositivi, nel caso di più dispositivi registrati con uno stesso account).

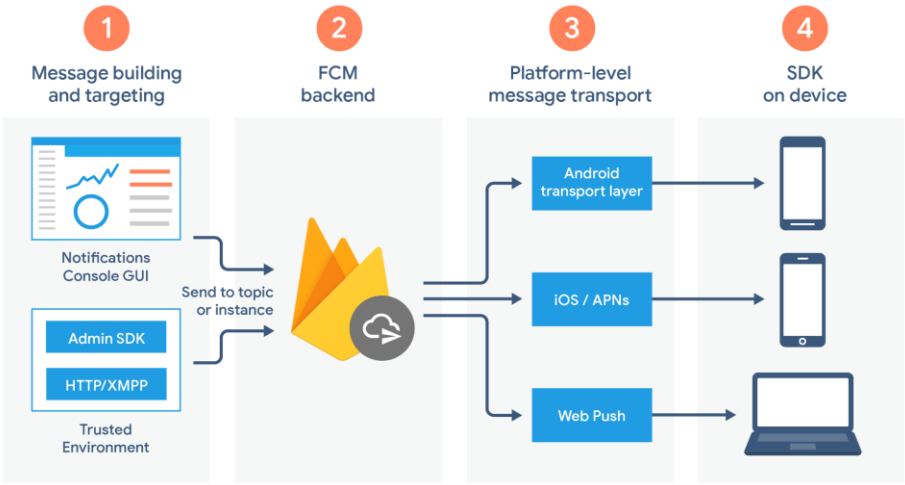

Fig. 12 - Firebase Cloud Messaging

Quando il sistema stabilisce, ad esempio, che tutti gli utenti che hanno messo come preferito uno specifico treno debbano ricevere una notifica, il servizio dedicato all'invio di notifiche push recupera i *device ID* associati a quegli account e invia una richiesta a Firebase, utilizzando le sue API rest, contenente

l'identificativo del dispositivo e il testo del messaggio. Firebase si occupa di recapitare il messaggio in tempo reale, o comunque alla prima occasione disponibile.
Perché Firebase sia in grado di raggiungere i dispositivi, è necessario registrare due app, una per Android e una per iOS, e configurare le due versioni delle app con la chiave server fornita da Firebase stesso.

Di seguito, esempio di invio notifica attraverso una chiamata POST verso l'endopoint https://fcm.googleapis.com/fcm/send (è evidenziatol'identificativo del device regitrato da Firebase):

```
{
  "to":
"eBLS6PqCw1w:APA91bHgGUD0sgz1GBqjCNJ4ZwRjbLWW5Lqlh7Hm9aT1tYkO0_DfKDuk_LkNs3bdVBYhq
V95VexaVQbxYaQZeL2t-tNex1GmsNBeRzZhYiqn2b4133Q90xESTuF7bLmcO7DP9Zyalmwf",
  "data": {
    "category": "news"
  },
  "notification": {
    "title": "Ritardo",
    "text": "Il treno 11657 in arrivo a Firenze Campo di Marte è in ritardo di 8
minuti."
  }
}
```

Dalla app di gestione new è possibile assegnare l'attributo booleano *broadcast* a una news per notizie di particolare importanza e di interesse generale. Se una news è broadcast viene inviata automaticamente una notifica push a tutti i dispositivi, indipendentemente dalle caratteristiche degli utenti. In questo caso viene eseguita una chiamata POST analoga a quella descritta sopra, ma per tutti i dispositivi registrati (senza quindi passare dai *topics*).

b. *App per smartphone*

1 Framework utilizzato e architettura software

Per lo sviluppo della app mobile Muoversi In Toscana è stato scelto il framework NativeScript (https://www.nativescript.org) che consente di produrre app native per le piattaforme Android e iOS partendo da una base di codice comune. Le interfacce utente delle applicazioni generate con NativeScript sono prodotte utilizzando i componenti UI nativi forniti dai sistemi Android e iOS e le API messe a disposizione dai dispositivi sono totalmente accessibili. La caratteristica più importante che differenzia NativeScript da altri framework che consentono di sviluppare app per più piattaforma usando una unica base di codice è che non fa uso di *web view*, a differenza di Cordova o altri progetti simili che essenzialmente permettono di costruire una app mobile come se si trattasse di una app per il web, eseguendola poi sui dispositivi all'interno di un browser web privo di controlli e barra degli indirizzi. Applicazioni di questo tipo sono definite comunemente app ibride e, pur offrendo alcuni vantaggi in fase di sviluppo, hanno prestazioni notevolmente inferiori alle app native e un accesso molto limitato alle API di sistema. NativeScript (analogamente ai framework Xamarin o ReactNative) produce invece app totalmente native.

Caratteristiche di NativeScript

NativeScript permette di sviluppare applicazioni *cross-paltform* utilizzando TypeScript o moderno JavaScript e, opzionalmente, Angular o Vue.js. Attraverso JavaScript garantisce un accesso completo alle API native e

riutilizza librerie da NPM (Node Package Manager), CocoaPods e Gradle. NativeScript è un progetto open source supportato da Progress.

Sito ufficiale del progetto open source:	https://www.nativescript.org
Presentazione del framework:	https://www.progress.com/nativescript

Chi sviluppa con NativeScript può creare interfacce utente in xml. Tutti gli elementi definiti in xml vengono poi tradotti in componenti UI nativi del sistema di riferimento.
NativeScript supporta anche in sottoinsieme (molto ampio) di regole css per definire lo stile delle pagine e dei loro elementi. E' possibile quindi utilizzare id e classi per assegnare uno stile specifico agli elementi, oltre a scrivere delle regole di stile in linea direttamente come attributi dell'xml.
Grazie a queste caratteristiche, chi ha esperienza con i più moderni linguaggi per lo sviluppo di applicazioni web è in grado in brevissimo tempo di essere produttivo con NativeScript.

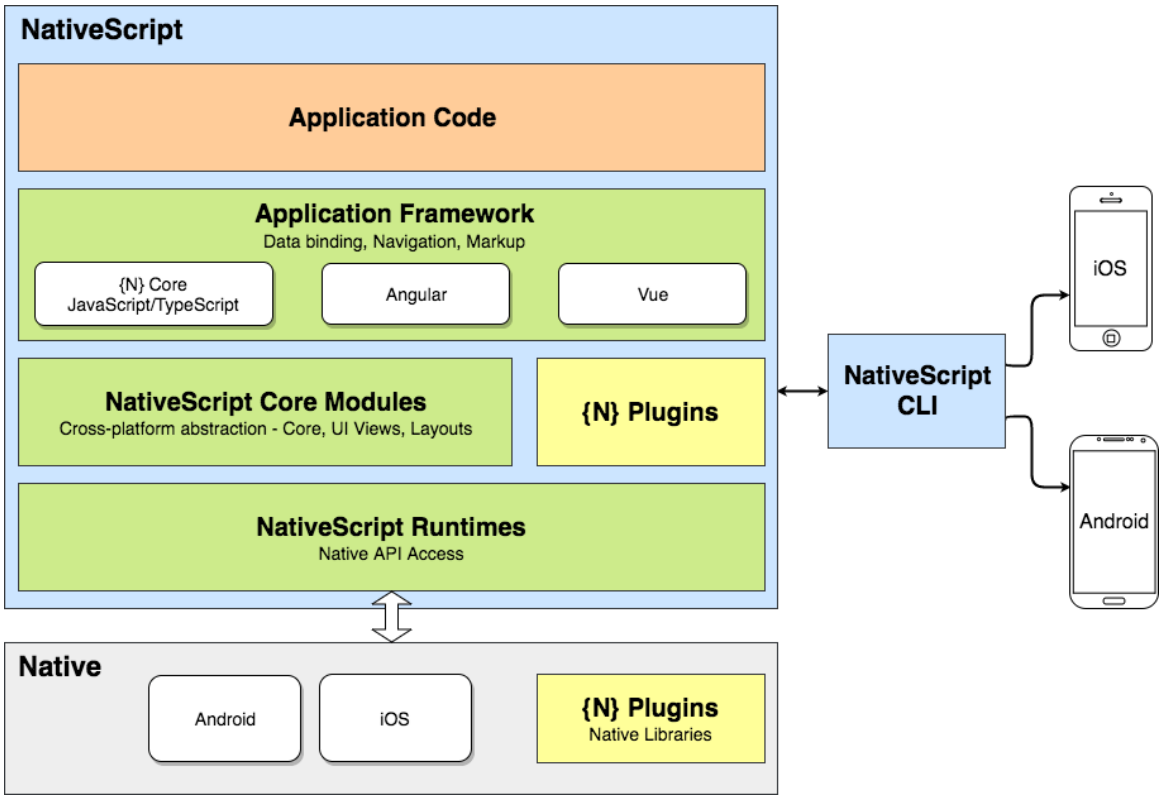

Fig. 13 - Architettura di NativeScript

Per lo lo sviluppo di Muoversi in Toscana è stato scelto NativeScript Core con Typescript invece del semplice JavaScript. Data la solidità delle funzioni *core* del frameworks non è stato rilevato alcun particolare vantaggio nell'appoggiarsi per lo sviluppo mobile a frameworks come Angular o Vue.js, che avrebbero aggiunto ulteriore complicazione alla gestione di plugin.

Supporto a nuove versioni dei sistemi operativi

NativeScript supporta immediatamente nuove versioni dei sistemi operativi. NativeScript esegue JavaScript sui sistemi nativi che lo ospitano, di conseguenza per supportare una nuova versione di Android o iOS è sufficiente rimappare le API native, cose che il framework fa al momento dell'operazione di build.

2 Specificità per Android e per iOS

Il framework NativeScript consente l'accesso immediato a molti componenti native sia per la costruzione della UI che per l'implementazione di specifiche funzionalità (come ad esempio le chiamate HTTP o la gestione delle immagini). Quando è necessario accedere a funzionalità native non disponibili nel *framework core* si fa ricorso ai plugin. Un plugin ampiamente utilizzato nella app Muoversi In Toscana e *NativeScript Google Maps SDK* (https://github.com/dapriett/nativescript-google-maps-sdk) , che mette a disposizione le SDK native di Google Maps per Android e iOS e un set di API per svolgere le operazioni più comuni sulle mappe.

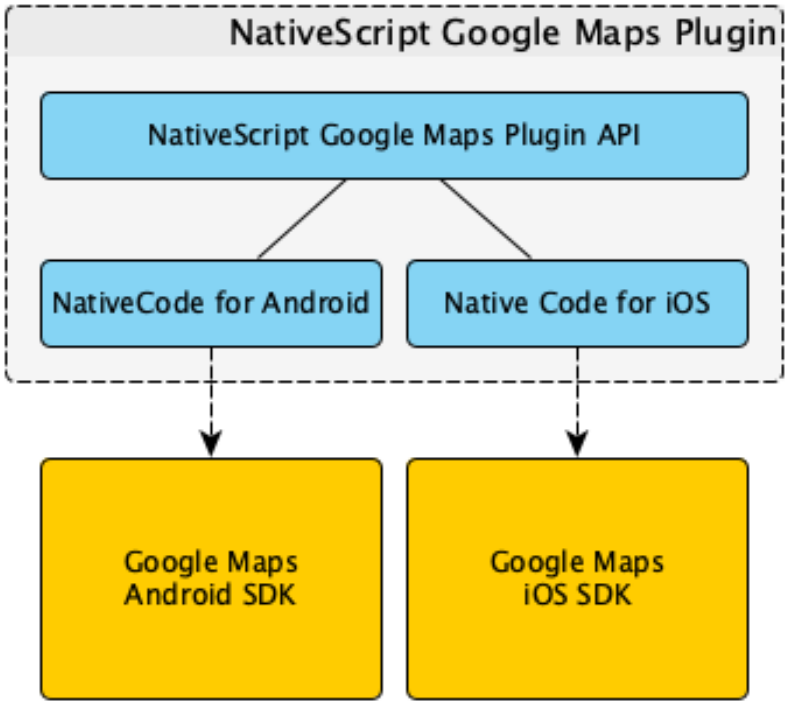

Fig. 14 - Architettura di un plugin NativeScript

Per le funzionalità di *autocomplete* degli indirizzi o di accesso alla app attraverso l'IDM centralizzato, sono stati invece sviluppati da Phoops specifici plugin, che hanno le loro specificità per Android o per iOS. Per la pagina di login, nel caso di Android viene utilizzata la *Advanced WebView* nativa di sistema con le *ChromeTabs.* Nel caso di iOS viene utilizzata la classe *ASWebAuthenticationSession* (https://developer.apple.com/documentation/authenticationservices/aswebauthenticationsession) come suggerito dalle linee guida di Apple stessa. Questa differenza nell'accesso alle risorse native viene comunque mascherata dal plugin, che mette a disposizione un'unica interfaccia per entrambi i sistema e poi implementa i metodi in modo diverso.

c. Sito web (Travel Planner)

La versione web del *travel planner* è stata sviluppata utilizzando il compnente software open source *Digitransit UI* (https://github.com/HSLdevcom/digitransit-ui).
Digitransit UI è il framework per la pubblicazione web della piattaforma Digitransit, un sistema integrato che, a partire da OpenTripPlanner, fornisce informazioni sul trasporto pubblico locale nell'area di Helsinki. Il team di sviluppo di Digitransit contribuisce in modo attivo allo sviluppo di OpenTripPlanner e mette a pubblica su GitHub i componenti della piattaforma. Phoops ha a sua volta contribuito allo sviluppo di OpenTripPlanner e ha creato una versione personalizzata di Digitransit UI per muoversi in Toscana: https://www.muoversintoscana.it/gw/travelplanner/

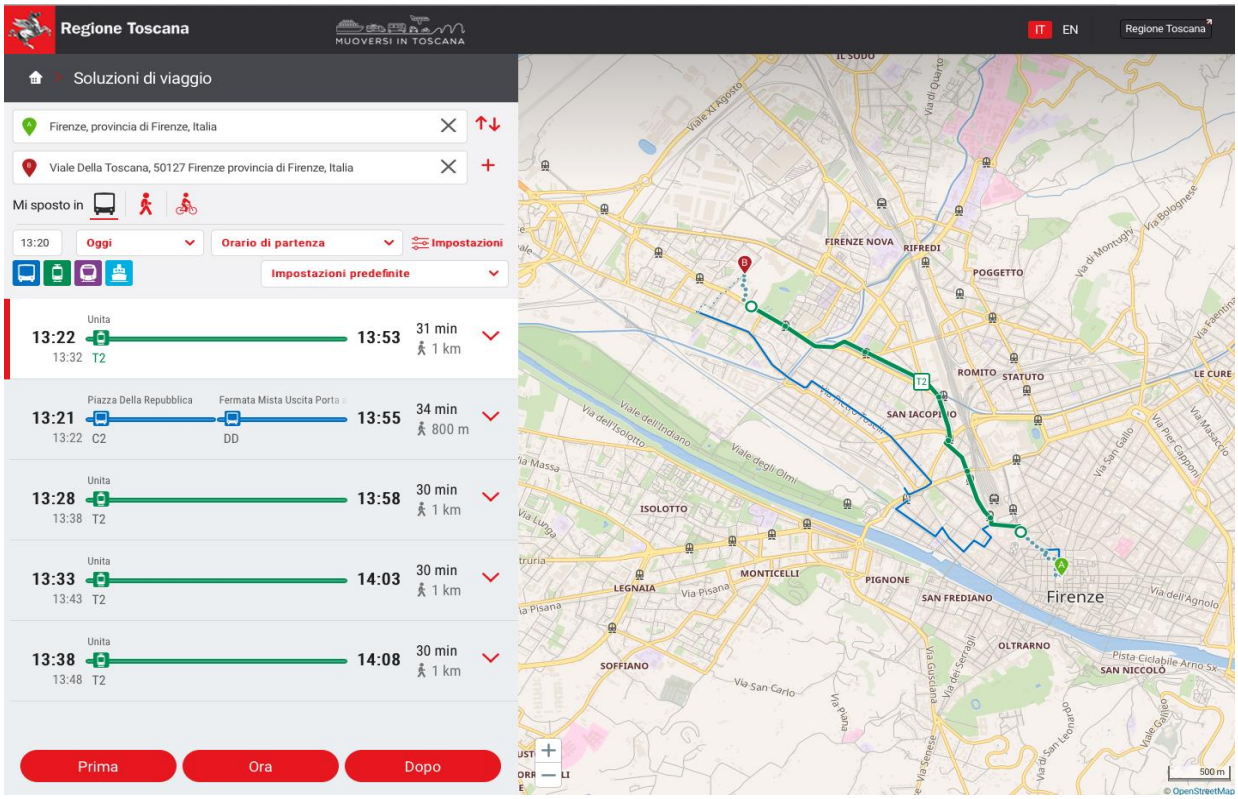

Fig. 15 - Travel Planner web di Muoversi In Tiscana

1. Requisiti tecnici di integrazione

Il travel planner web viene pubblicato come puro client HTML5/Css/JavaScript e può essere quindi integrato in qualsiasi sistema come webapp auto-contenuta e autosufficiente. Tutto il codice del travel planner web gira in fatti sul client (è sufficiente un moderno browser web) e non richiede nessuna interazione con il sistema ospitante.

2. Architettura software

Il progetto Digitransit-UI si basa su JavaScript e *React*. La sua architettura può essere rappresentata dal seguente diagramma:

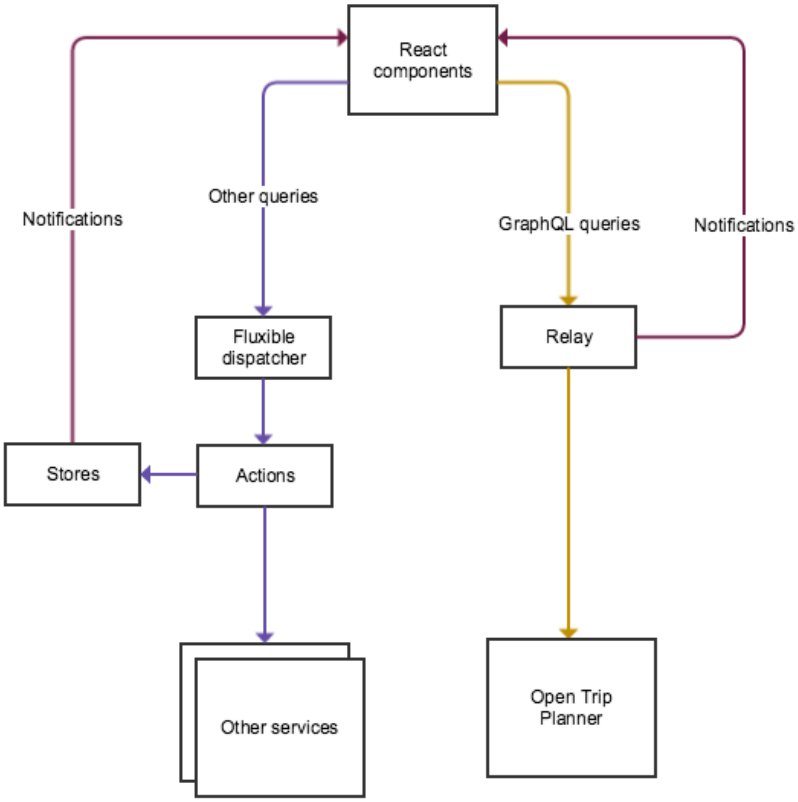

Fig. 16 - Architettura di Digitransit UI

Per una dettagliata documentazione sull'architettura di *Digitransit-UI* e su altri aspetti della piattaforma Digitransit si rimanda all'eccellente pagina GitHub del progetto: https://github.com/HSLdevcom/digitransit-ui/blob/master/docs/Architecture.md

0.4 Dispiegamento

a. Dispiegamento dei moduli componenti

Si utilizzano descrittori standard compatibili con Kubernetes e pertanto il deploy viene effettuato tramite il tool *kubectl*.

0.5 Appendice A
Manuale di riferimento della API REST

Le API REST dei servizi del sistema Muoversi In Toscana sono descritte nei seguenti documenti PDF, uno per ciascun servizio:

Servizio per routing, orari trasporto pubblico e pianificazione viaggi:
File “07_api-documentation.pdf”

Servizio per la gestione di sondaggi e questionari
File “02_api-documentation.pdf”

Servizio dedicato ai ritardi Trenitalia
File “01_api-documentation.pdf”

Servizio dedicato alla segnalazioni
File “06_api-documentation.pdf”

Servizio per gestione dei preferiti legati al GTFS (fermate, linee, corse)
File “03_api-documentation.pdf”

Servizio per la gestione dei luoghi preferiti (casa/lavoro ecc.)
File “05_api-documentation.pdf”

Servizio dedicato al modulo green
File “04_api-documentation.pdf”

Per accettazione
Luogo, data

_____