Sii-Mobility

Supporto di Interoperabilità Integrato per i Servizi al Cittadino e alla Pubblica Amministrazione

Trasporti e Mobilità Terrestre, SCN_00112

Deliverable ID: DE3.21b
Titolo: Applicazioni / moduli campione dimostrativi

Data corrente	M18, Giugno 2017
Versione (solo il responsabile puo' cambiare versione	2
Stato (draft, final)	finale
Livello di accesso (solo consorzio, pubblico)	Pubblico
WP	OR3
Natura (report, report e software, report e HW..)	software
Data di consegna attesa	M18, Giugno 2017
Data di consegna effettiva	M18, Giugno 2017
Referente primario, coordinatore del documento	Simone Lucarelli, SOFTEC, simone.lucarelli@softecspa.it
Contributor	Alessandro Gronchi, QUESTIT, alessandrogronchi@gmail.com Giacomo Materozzi, SOFTEC, giacomo.materozzi@softecspa.it, Daniele Pasqui, SOFTEC,

	daniele.pasqui@softecspa.it
Coordinatore responsabile del progetto	Paolo Nesi, UNIFI, paolo.nesi@unifi.it

Indice generale

Introduzione e obiettivi

Questo documento è prodotto nel contesto del Obiettivo Realizzativo *OR3 - Sviluppo di prototipi applicativi verticali, sensori e attuatori* e in particolare della *Attività 3.5 - Applicazioni / moduli campione dimostrativi*.

La *Figura 1* mostra il ruolo dell'applicazione nell'architettura generale di Sii-Mobility. Complessivamente, i Sensori sviluppati nel progetto sono evidenziati nella prossima figura.

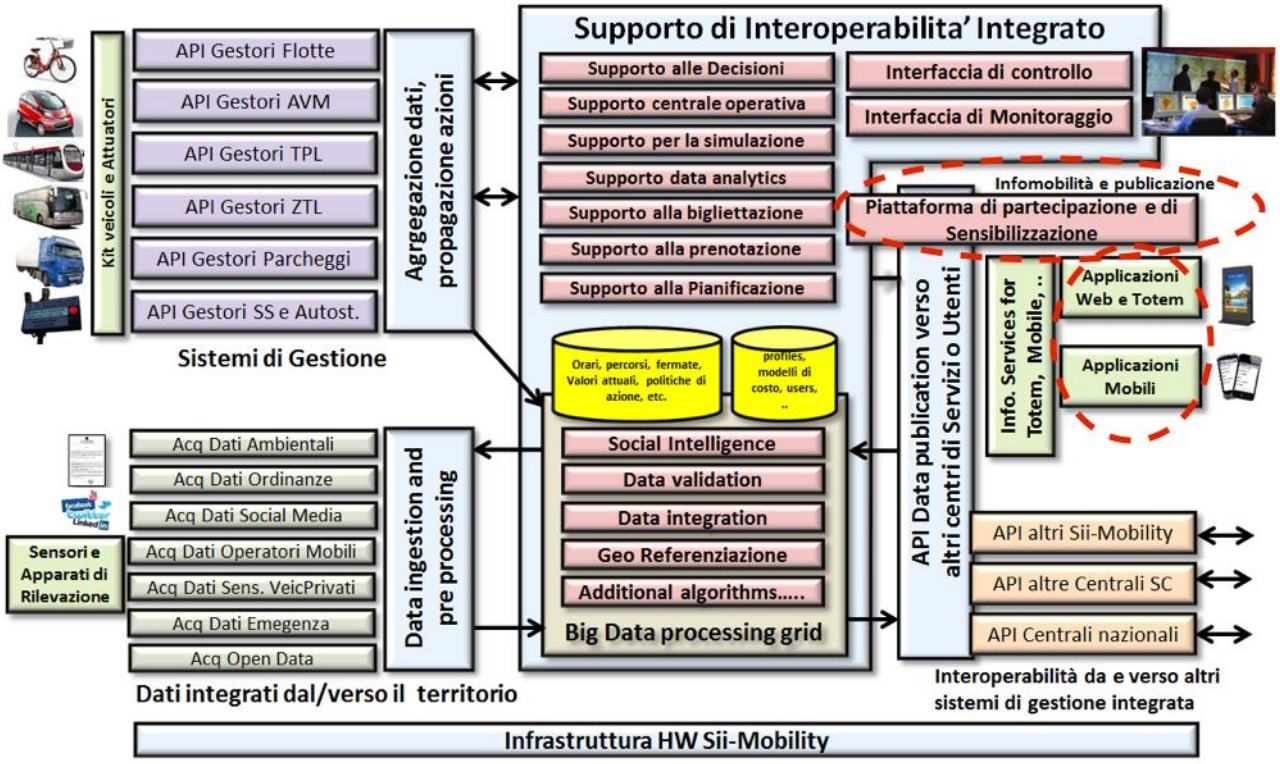


Figura 1 : Sensori nel contesto Sii-Mobiliy

1.1 Obiettivi

Realizzare moduli per l'applicazione mobile sulla base dell'applicazione "Toscana dove cosa".

Tutorial: sviluppare un modulo della app

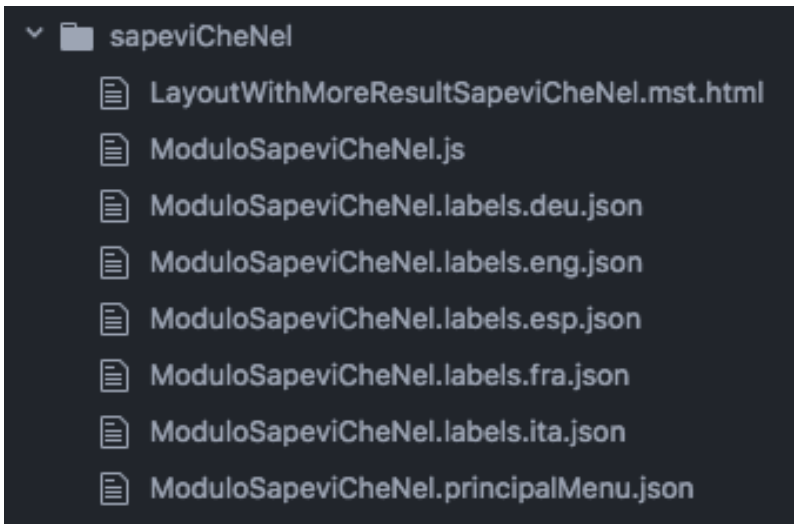
I paragrafi seguenti descrivono i passi principali da seguire per sviluppare un modulo della app Sii-Mobility: strumenti da utilizzare, convenzioni applicate, linee guide per la programmazione.

1.2 Cordova

I moduli della app Sii-Mobility poggiano su **Apache Cordova** (https://cordova.apache.org/), un framework di sviluppo open source che permette di scrivere codice eseguibile nella stessa versione sulle varie piattaforme mobile.

Un modulo Cordova della app Sii-Mobility è codificato utilizzando i linguaggi "standard" della programmazione web: **HTML, CSS, Javascript**. In maniera semplice, quindi, sarà possibile aggiungere un nuovo modulo alla app e successivamente implementare tutte le funzionalità richieste, senza dover effettuare porting del codice per sistemi operativi diversi. Lo stesso modulo Cordova, cioè, verrà eseguito indifferentemente su dispositivi Android, iOS o Windows Mobile.

1.3 Organizzazione dei file



Sull'esempio del modulo **"Sapevi che nel…"** (descritto dettagliatamente più oltre in questo documento), è qui illustrata l'organizzazione di riferimento dei file principali che concorrono a formare un modulo Cordova per la app mobile di Sii-Mobility.

I moduli del progetto si trovano nella directory **/www/js/modules** e sono suddivisi sull'esempio che segue:

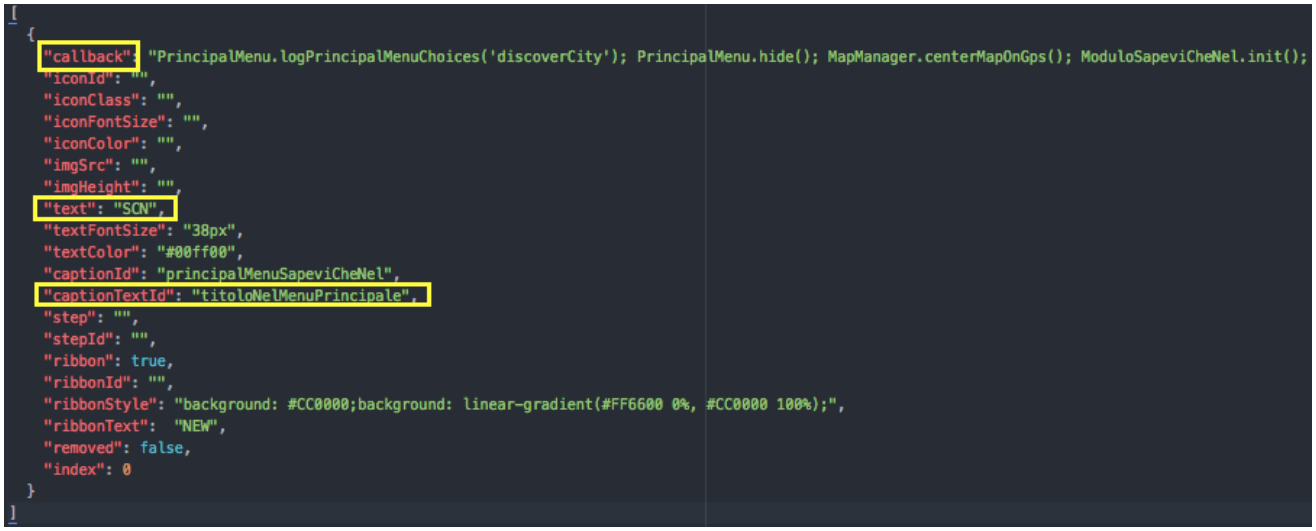
- **1 file {nomeModulo}.principalMenu.json**
  - *Descrive l'icona nel menu e le callback lanciate all'avvio del modulo.*
- **5 file {nomeModulo}.labels.{lingua}.json**
  - *Traduzioni delle label utilizzate nel modulo. Il numero di questi file può variare in relazione al numero di traduzioni previste.*

- **1 file {nomeModulo}.js**
  - *Funzioni di riferimento del modulo. Contiene le logiche di base attivabili dall'uso del modulo all'interno della app.*
- **1 file {nomeLayout}.mst.html**
  - *Descrive il layout del modulo, ossia la l'aspetto grafico con cui questo viene renderizzato sul device e presentato all'utente finale.*

1.4 Icona di lancio nel PrincipalMenu

Ogni modulo sarà attivabile dal menu principale della app. Per aggiungere una icona di lancio occorre creare un file in formato JSON, con la naming convention **{nomeModulo}.principalMenu.json**.

Il file sarà simile a quello qui sotto illustrato.



Il significato degli elementi che compongono l'oggetto JSON è facilmente intuibile; ciascun elemento viene valutato dal framework al momento di comporre la view del menu principale, disegnando le icone di lancio secondo quanto descritto nei JSON di ciascun modulo.

Particolare rilievo hanno gli elementi sopra evidenziati:

- **callback**
  - Descrive la catena esecutiva di azioni da intraprendere all'avvio del modulo. Nell'esempio fornito, a seguito del tap dell'utente sull'icona di lancio, viene nascosto il menu principale, centrata la mappa sulla geolocalizzazione del device ed eseguita la funzione init del modulo in oggetto (vedi oltre in questo documento).
- **text**

- La label principale dell'icona di lancio, quella più immediatamente visibile all'utente all'interno del menu principale della app. Nell'esempio, "SCN". 

- **captionTextId**
  - Recupera la traduzione della label relativa al nome del modulo da presentare nell'icona di lancio (vedi paragrafo successivo). Nell'esempio, "Sapevi che nel…"

1.5 Gestione delle traduzioni

Anche le label di traduzione sono descritte in file JSON, uno per lingua, nominati secondo la convenzione **{nomeModulo}.labels.{lingua}.json**. Nel modulo preso ad esempio e in riferimento alla lingua italiana, il file si chiamerà quindi **ModuloSapeviCheNel.labels.ita.json** ed avrà una struttura di partenza come quella che segue:

```
{
  "principalMenu": {
    "titoloNelMenuPrincipale": "Sapevi che nel..."
  },
  "menuSapeviCheNel": {
    "title": "Risultati"
  }
}
```

Riprendendo quanto illustrato nel paragrafo precedente, è quindi evidente che la traduzione del nome del modulo da mostrare nell'icona di lancio, viene recuperata da oggetto JSON, andando a leggere il valore dell'elemento **pricipalMenu.titoloNelMenuPrincipale**.

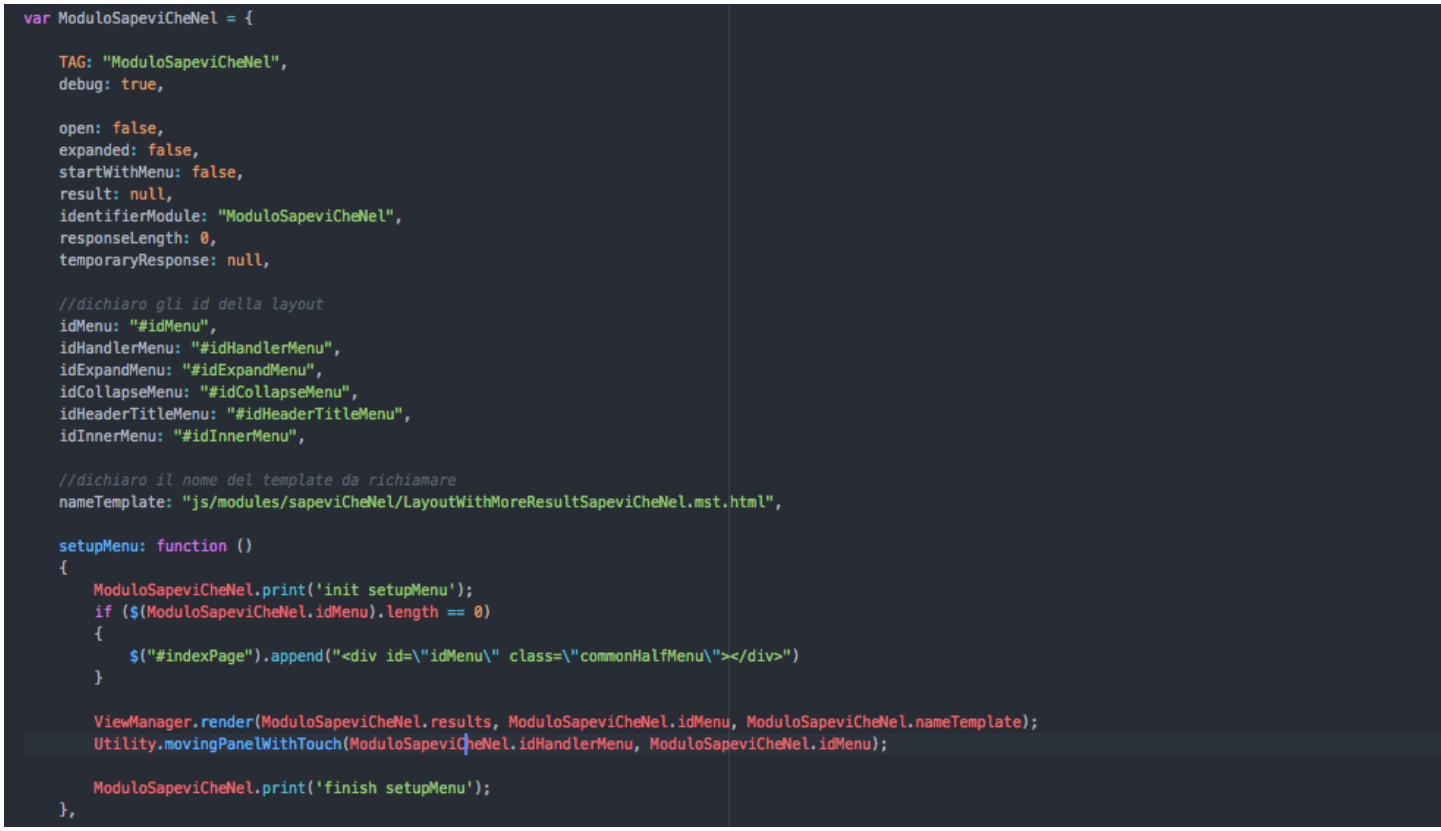
Le label tradotte in inglese, in francese o in altre lingue saranno quindi all'interno di oggetti JSON con la medesima struttura, e nominati ModuloSapeviCheNel.labels.ita.json, ModuloSapeviCheNel.labels.ita.json, ecc.

In maniera analoga, è possibile richiamare le traduzioni delle singole label presenti nei JSON, utilizzando l'oggetto globale delle traduzioni inizializzato all'avvio della app **Globalization.labels**. Nell'esempio precedente, per visualizzare la label tradotta per i risultati di ricerca, si potrà recuperare la stringa dall'elemento **Globalization.labels.menuSapeviCheNel.title**.

1.6 Funzioni del modulo Javascript

La logica di riferimento del modulo Sii-Mobility è raccolta nel file Javascript, nominato secondo la convenzione **{nomeModulo}.js**. Al suo interno, lo sviluppatore ha liberà di definire variabili,

funzioni e chiamate secondo i propri fini.



```
var ModuloSapeviCheNel = {

    TAG: "ModuloSapeviCheNel",
    debug: true,

    open: false,
    expanded: false,
    startWithMenu: false,
    result: null,
    identifierModule: "ModuloSapeviCheNel",
    responseLength: 0,
    temporaryResponse: null,

    //dichiaro gli id della layout
    idMenu: "#idMenu",
    idHandlerMenu: "#idHandlerMenu",
    idExpandMenu: "#idExpandMenu",
    idCollapseMenu: "#idCollapseMenu",
    idHeaderTitleMenu: "#idHeaderTitleMenu",
    idInnerMenu: "#idInnerMenu",

    //dichiaro il nome del template da richiamare
    nameTemplate: "js/modules/sapeviCheNel/LayoutWithMoreResultSapeviCheNel.mst.html",

    setupMenu: function ()
    {
        ModuloSapeviCheNel.print('init setupMenu');
        if ($(ModuloSapeviCheNel.idMenu).length == 0)
        {
            $("#indexPage").append("<div id=\"idMenu\" class=\"commonHalfMenu\"></div>")
        }

        ViewManager.render(ModuloSapeviCheNel.results, ModuloSapeviCheNel.idMenu, ModuloSapeviCheNel.nameTemplate);
        Utility.movingPanelWithTouch(ModuloSapeviCheNel.idHandlerMenu, ModuloSapeviCheNel.idMenu);

        ModuloSapeviCheNel.print('finish setupMenu');
    },
```

Sulla base del modulo di esempio, si pone attenzione sui due aspetti che seguono.

E' utile definire all'interno del modulo, come variabili utilizzabili dal modulo stesso, i riferimenti agli elementi HTML presenti nella vista mostrata all'utente finale. Nell'esempio sopra illustrato, è ad esempio utile associare alla variabile globale **idMenu** l'id del div HTML **#idMenu**, in modo da poterlo poi valutare in Javascript all'interno delle singole funzioni.

Un altro riferimento che è buona pratica variabilizzare all'interno del file .js del modulo è quello relativo al template (ossia alla vista graficizzata, cfr. paragrafi successivi) utilizzato dal modulo stesso. La variabile nameTemplate dell'esempio contiene quindi il path in cui la app va a recuperare la view da presentare all'utente finale (nel caso specifico, "**js/modules/sapeviCheNel/LayoutWithMoreResultSapeviCheNel.mst.html**").

Come già anticipato, a seguito del lancio da menu principale della app, la prima funzione interna al modulo che viene eseguita secondo quanto indicato nella callback è la **init()**.

```
init: function ()
{
    ModuloSapeviCheNel.print('on Init');
    ModuloSapeviCheNel.show();

    if (ModuloSapeviCheNel.startWithMenu)
    {
        ModuloSapeviCheNel.setupMenu();
        ModuloSapeviCheNel.showMenu();
    }
},
```

La funzione dell'esempio, in successione:

- stampa un messaggio di **debug** (disattivabile in ambiente di produzione tramite opportuna parametrizzazione)
- richiama la funzione **show()** del modulo stesso (vedi qui di seguito)
- se valorizzata la variabile **startWithMenu**, richiama le funzioni setupMenu() e showMenu()

Nel modulo di esempio, la funzione **show()** è così definita:

```
show: function ()
{
    //mostro il modulo
    application.resetInterface();
    ModuloSapeviCheNel.open = true;
    application.setBackButtonListener();
},
```

Essa, molto semplicemente:

- resetta l'interfaccia
- setta a true la variabile open (utile per essere poi valutata da altre funzioni javascript del modulo)
- valorizza in modo appropriato l'azione da associare al pulsante back della app

L'if nella funzione di init riferito alla valutazione della variabile **startWithMenu** è da sottolineare perché permette di gestire un aspetto particolare della app. Nella app Sii-Mobility, infatti, i vari moduli possono "interagire" fra di loro: potrebbe cioè essere utile che il modulo "Sapevi che nel…" sia richiamato da un altro modulo della app, con dei parametri in ingresso. In questo caso,

invocando dall'esterno il modulo "Sapevi che nel…" con il parametro startWithMenu valorizzato a true, si avrebbe l'effetto di avviare il modulo con il risultato delle ricerche già espanso, magari a seguito di una query sui Point Of Interest di rilievo effettuata a monte da un modulo terzo, o anche in seguito al fire di una notifica push da parte di Engagement.

1.7 Chiamate alle API Sii-Mobility

La caratteristica principale della app mobile è quella di interagire con il *core* del sistema Sii-Mobility tramite chiamate alle API predisposte. Queste sono documentate nel DE 3.16 e online a partire da http://www.disit.org/drupal/?q=node/6597. Lo strumento di riferimento per lo sviluppo delle chiamate alle API è Service Map, online su http://www.disit.org/ServiceMap/

Ogni modulo ha necessità di interrogare Sii-Mobility per ricavarne dati e informazioni con cui l'utente finale andrà ad interagire. In base alle sue azioni e a particolari eventi, inoltre, i moduli della app dovranno a loro volta inviare dati a Sii-Mobility. Questa interazione è realizzabile tramite le chiamate alle API Sii-Mobility secondo le logiche stabilite nel file Javascript di ciascun modulo.

Nel caso preso ad esempio di "Sapevi che nel…", una delle chiamate alle API Sii-Mobility è relativa alla funzione **search()**.

```
search: function ()
{
    ModuloSapeviCheNel.print('on Search');
    var CulturalActivityQuery = QueryManager.createCategoriesQuery(['CulturalActivity'], SearchManager.searchCenter, "user");
    APIClient.executeQuery(CulturalActivityQuery, ModuloSapeviCheNel.searchInformationForEachFeature, ModuloSapeviCheNel.errorQuery);
},
```

La chiamata dell'esempio qui sopra ha la finalità di recuperare l'elenco dei Point Of Interest categorizzati come "Cultural Activity" nell'intorno del marker posizionato sulla mappa. Più nel dettaglio:

- la query da effettuare viene definita istanziando un oggetto del Query Manager, filtrando per categoria "CulturalAcitivity" e posizionando l'utente al centro dell'insieme dei POI trovati. Ciò è possibile con l'istruzione **QueryManager.createCategoriesQuery(…)** con tutti i parametri appena descritti;
- la query così creata viene eseguita con il metodo **APIClient.executeQuery**, con parametri:
    - la query istanziata
    - funzione di callback on success
    - funzione di callback on failure

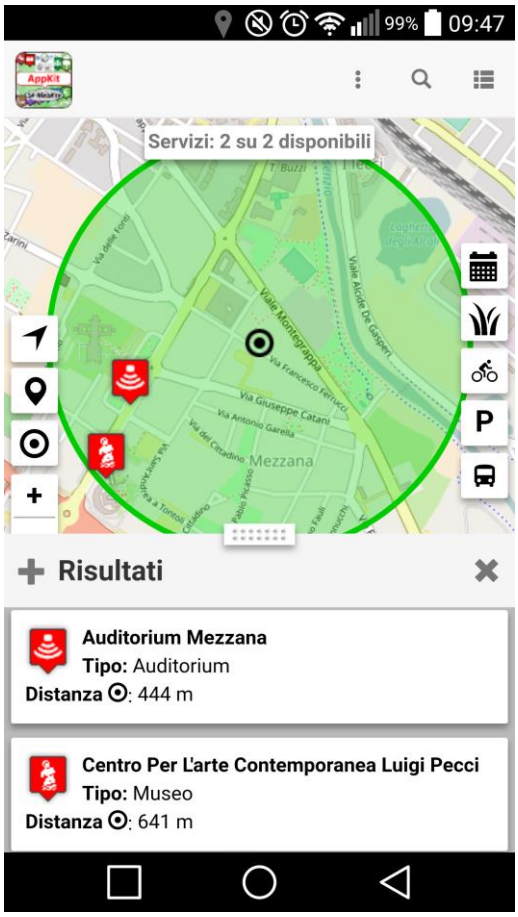
Nell'esempio appena citato, se la chiamata va a buon fine viene eseguita la funzione del modulo **searchInformationForEachFeature()**, che ciclerà sull'array JSON dei risultati contenuti nella *response* per poi renderizzarli nella view.

Nel caso finora seguito, la *response* conterrà una serie di Point Of Interest strutturati in un JSON simile a quello che segue:

```
{
  "Services":{
  "fullCount": 1,
  "type": "FeatureCollection",
  "features": [{
    "geometry": {"type": "Point","coordinates":[11.109862,43.863094]},
    "type": "Feature",
    "properties": {
    "name": "AUDITORIUM MEZZANA",
    "tipo": "Auditorium",
    "typeLabel": "Auditorium",
    "serviceType": "CulturalActivity_Auditorium",
    "distance": "0.4916674422843726",
    "serviceUri": "http://www.disit.org/km4city/resource/c7cc2e0affba9605bf4b43d04a8362f7",
       "photoThumbs": [],
    "multimedia": ""
    },
    "id": 1
    }]
  }
}
```


1.8 View con Mustache JS

L'effetto finale per l'utente della app sarà quindi simile allo screenshot qui sotto:



Per comporre una view come questa, è sufficiente definire un file HTML allo scopo. I moduli di Sii-Mobility utilizzano il frame work **Mustache JS** (https://mustache.github.io/) per il templating. Mustache aiuta lo sviluppatore a ciclare sugli array, compiere switch logici e stampare a video i valori dinamicizzati in variabile.

Un esempio relativo allo screenshot sopra presente è il seguente:

```
<div class="panel panel-default" style="margin: 0px 5px 10px 5px; color: #000; text-decoration: none;" onclick="InfoManager.showInfoAboutOneMarker('{{#properties}}{{serviceUri}}{{/proper
    <div class="panel-body card-2" style="position:relative">
        {{#properties}}

            {{#imageThumb}}
                <img id="parkingMenuImage{{identifier}}" src="{{imageThumb}}" style="margin-right: 8px; float: left; max-height: 90px; max-width: 90px" class="card-3">
            {{/imageThumb}}

            <img src="{{imgsrc}}" style="padding-right: 8px; float: left">

            <b style="text-align: center">
                {{#unescapeHtml}}
                    {{name}}
                {{/unescapeHtml}}
            </b>

            {{#typeLabel}}
                <br>
                <b id="parkingMenuTypeLabel{{identifier}}">
                    <script>
                        $("#parkingMenuTypeLabel{{identifier}}").html(Globalization.labels.infoMenu.type)
                    </script>
                </b>
                {{typeLabel}}
            {{/typeLabel}}

            {{#distanceFromSearchCenter}}
                <br>
                <b id="parkingMenuTextSearchDistanceFromSearchCenter{{identifier}}">
                    <script>
                        $("#parkingMenuTextSearchDistanceFromSearchCenter{{identifier}}").html(Globalization.labels.textSearchMenu.distanceFromSearchCenter)
                    </script>
                </b>
                {{distanceFromSearchCenter}} m
            {{/distanceFromSearchCenter}}

            {{#distanceFromGPS}}
                <br>
                <b id="parkingMenuTextSearchDistanceFromGPS{{identifier}}">
                    <script>
                        $("#parkingMenuTextSearchDistanceFromGPS{{identifier}}").html(Globalization.labels.textSearchMenu.distanceFromGPS)
                    </script>
                </b>
                {{distanceFromGPS}} m
            {{/distanceFromGPS}}

        {{/properties}}
    </div>
</div>
```

Nell’esempio, è intuitivo identificare le varie sezioni, e come ciascuna sia popolata dinamicamente dagli elementi risultanti dalla query alle API Sii-Mobility, con le label tradotte. Ad esempio, il valore di ciascun Point Of Interest dal centro della mappa è recuperato stampando in Mustache la variabile **{{distanceFromSearchCenter}}**.

Il testo per ricavare la label da mostrare per questo dato – nel caso specifico, “Distanza” – è recuperato tramite l’elemento già illustrato **Globalization.labels.textSearchMenu.distanceFromSearchCenter**, che poi viene inserito nel div giusto tramite l’istruzione JQuery, che usa la variabile **{{identifier}}** per rendere univoci i singoli div:
$("#parkingMenuTextSearchDistanceFromSearchCenter{{identifier}}").html(Globalization.labels.textSearchMenu.distanceFromSearchCenter).

(3.5.3) Modulo profiling comportamento virtuoso

1.9 Descrizione

Modulo per web app e mobile per la richiesta di informazioni riguardo alle info personali e alle preferenze del City User, anche in relazione al suo comportamento passato (ad es., prevalenza bus, car, moto, bike, ecc.). Suggerisce comportamenti virtuosi, come l'uso di mezzi a minor impatto ambientale, l'uso di scambiatori, anche arrivando ad offrire bonus. se possibile prendendoli da un carnet di Bonus della PA o delle TPL. **Possibile necessità di caricare un modulo computazionale aggiuntivo in piattaforma Sii-Mobility.**

1.10 Funzionalità

1.10.1 Visualizzazione e salvataggio del percorso (Softec)

Questa funzionalità avrà lo scopo di tracciare, mediante richiesta esplicita inviata dall'utente, il percorso seguito da quest'ultimo, salvarlo ed inviarlo tramite API a Sii-Mobility.

All'interno del modulo sarà presente un pulsante tramite il quale l'utente decide di avviare la registrazione del percorso che si appresta ad intraprendere. Al click sul pulsante, un pop-up chiederà conferma all'utente per l'inizio del tracciamento del percorso che verrà effettuato fino alla richiesta di interruzione. Una volta data la conferma, l'applicazione traccerà il tracciato seguito dall'utente salvando progressivamente i punti del percorso seguito mediante le funzionalità di geolocalizzazione incluse nell'AppKit. La stessa icona per l'avvio del tracciamento, in modalità registrazione, si occuperà di interromperlo. Un altro pop-up chiederà conferma all'utente per l'interruzione della registrazione del percorso.

Dopo la conferma, l'applicazione invierà tramite API il percorso tracciato a Sii-Mobility, che si occuperà di salvarlo tra i percorsi abituali per quello specifico utente, e disegnerà sulla mappa il percorso seguito dall'utente per mostrare a quest'ultimo una panoramica del suo spostamento.

1.10.2 Suggerimenti alternativi e bonus (Effknow)

Questa funzionalità di occuperà di suggerire "comportamenti virtuosi" all'utente al fine di utilizzare mezzi a minor impatto ambientale (mezzi pubblici, bici, a piedi, ecc…) arrivando ad offrire possibili bonus, prendendoli da un carnet di Bonus della PA o delle TPL.

Sii-Mobility invia una notifica all'utente della app indicandogli un percorso alternativo a quello abituale (che Sii-Mobility si occuperà di prevedere in base ai percorsi abituali precedentemente tracciati dell'utente) al fine di agevolare il più possibile l'uso di mezzi alternativi all'automobile. L'utente apre l'applicazione tramite la notifica e un pop-up gli chiede conferma se vuole deviare dal percorso che sta seguendo in quel momento. In caso di rifiuto, non viene assegnato nessun bonus e l'utente prosegue col suo percorso abituale. In caso di accettazione da parte dell'utente, verrà assegnato un bonus per il suo "comportamento virtuoso" ed il sistema si occuperà di inviare il percorso modificato all'applicazione, che verrà disegnato sulla mappa. L'utente darà l'input per l'avvio della navigazione e gli verrà definitivamente assegnato il bonus.

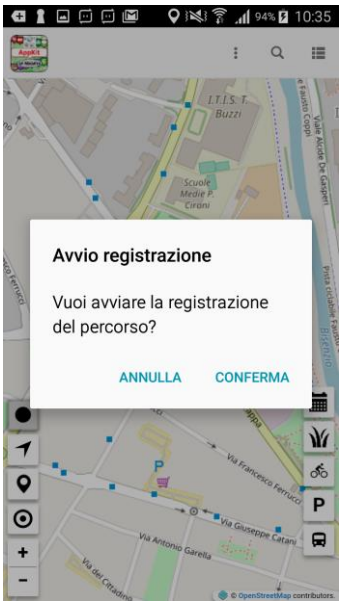
Sull'esempio di quanto già sviluppato per funzionalità analoghe di altri moduli dell'app mobile, è stata sviluppata una funzione con parametro in ingresso relativo al percorso alternativo per

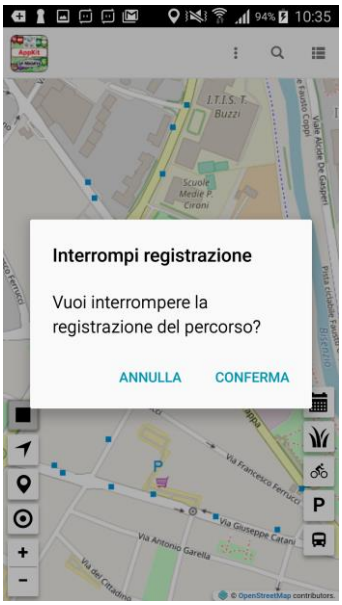
raggiungere il punto di destinazione (bus e soluzioni alternative ad utilizzo macchina). La funzione, una volta richiamata, fa visualizzare il percorso dalla geolocalizzazione dell'utente fino alla destinazione. La funzione è richiamata a seguito di conferma sul pop-up aperto dopo l'apertura dell'app dalla ricezione della notifica.

1.11 Attività svolte

Per questo modulo, "profiling comportamento virtuoso" abbiamo svolto le seguenti attività:

- Creato la cartella "comportamentoVirtuoso" dentro la path /www/js/modules che contiene il codice del modulo.
- Creato il branch su Git chiamato SoftecSiiMobility/softec-3.5.3-comportamento_virtuoso
- implementata la chiamata che tramite la posizione attuale e un punto GPS o POI (Point of interest) trova il percorso più breve (percorso a piedi più breve, come indicato di default per quell'endpoint).
- esposta una funzione "calculatePath" con un parametro in ingresso che deve essere il punto di destinazione (ID della risorsa, es. ad77f05cc308d96163c455e806b2694d, che verrà concatenato all'URL della risorsa per effettuare la query a Sii-Mobility). La funzione si occupa di eseguire la richiesta alle API di Sii-Mobility e successivamente, ottenuto il percorso da disegnare, mostrare sulla mappa tramite la funzione "drawWktIntoMap".
- E' stata sviluppata una funzione "drawWktIntoMap", nel modulo in oggetto, in quanto "**addGeometryWktElement**", presente nel core della app, non è pensata per disegnare un percorso.
- E' stato aggiunto un pulsante all'interfaccia in modo da consentire all'utente di avviare e terminare la registrazione del percorso, in modo da fargli successivamente scegliere se aggiungere il percorso a quelli delle preferenze utente.





1.12 Richieste

Sono stati richiesti a UNIFI alcuni sviluppi lato API e maggiori informazioni riguardo:

- **la API che permetta di ricavare informazioni personali e preferenze dell'utente**, oltre ad altre informazioni circa il suo comportamento nel passato (ad esempio prevalenza di autobus, automobile, bicicletta, ecc…);
- **la API per salvare il percorso compiuto** nel caso in cui l'utente intenda farlo;
- fire (**Notifica Push**) da Sii-Mobility (modulo *Engagement*) di notifiche atte ad attivare il modulo e consigliare all'utente il mezzo a minor impatto ambientale;
- aggiungere un metodo "createPathQuery" nel modulo ***QueryManager.js*** per permettere la creazione di query per l'endpoint shortestpath che prenda in ingresso i parametri descritti nella documentazione dell'API;
- aggiungere un metodo nel modulo ***MapManagerOL.js*** che permetta di disegnare il percorso della shortestpath tenendo di conto i mezzi di spostamento indicati nel percorso (automobile, autobus, bicicletta, ecc…) e che, secondo la documentazione ufficiale, sono ancora in via di sviluppo.
- API che invia dal sistema il percorso modificato a seguito della conferma dell'utente per il comportamento virtuoso, in modo tale da disegnarlo successivamente sulla mappa.

(3.5.5) Modulo “EasyPark” (QUESTIT)

1.13 Descrizione

Il sistema centrale Sii-Mobility colleziona ed integra dati provenienti dai gestori dei parcheggi che hanno aderito al sistema e per i quali sono stati realizzati appositi ETL di interfacciamento (Task 5.4). Tali dati permettono di avere nel sistema le informazioni in tempo reale sullo stato di occupazione dei parcheggi registrati (sia del tipo “in struttura” sia del tipo “lungo strada”) per i quali sia attivo un sistema di monitoraggio. I parcheggi in struttura si caratterizzano per la presenza di sistemi dedicati al controllo e la gestione degli ingressi e delle uscite. In tale categoria rientrano ovviamente i parcheggi realizzati in strutture (garage, box, silos etc.) ma anche i parcheggi all’aperto i cui varchi di accesso e uscita sono limitati e controllati da appositi sistemi (normalmente sbarre). Tale tipologia di parcheggi normalmente prevede nativamente l'uso di sistemi di monitoraggio che permettono di conoscere lo stato di occupazione dei posti. I parcheggi lungo strada, invece, mancano di tali caratteristiche e quindi il monitoraggio deve essere effettuato tramite l'installazione di appositi dispositivi hardware come ad esempio appositi sensori installati in ogni stallo oppure telecamere aeree con software di riconoscimento.

Questo modulo fornisce al cittadino un insieme di funzionalità basate sui dati di occupazione dei parcheggi monitorati presenti nel sistema centrale in modo da guidarlo nella fase di parcheggio del proprio autoveicolo.

Attraverso la mobile app il guidatore potrà così consultare il sistema per determinare la disponibilità di sosta in accordo alle sue preferenze. Se per esempio è già arrivato a destinazione, può visualizzare il numero e la posizione dei posti liberi intorno a sé che può raggiungere con ragionevole sicurezza oppure nel caso sia ancora distante dalla destinazione, può verificare la predizione di disponibilità di sosta in una zona in cui si sta recando col mezzo, e che raggiungerà nei successivi minuti.

1.14 Analisi funzionale

La funzionalità principale del modulo riguarda la ricerca e la visualizzazione dei posti disponibili più prossimi ad un punto di riferimento impostato dall'utente.

L'utente è chiamato ad inserire il luogo intorno al quale egli desidera ricercare un posteggio auto e, avviando la ricerca, il modulo restituirà una lista di parcheggi presenti nell'area. La scelta del luogo di ricerca potrà essere fatta selezionando:

1. la posizione attuale;
2. un punto sulla mappa.

Per ogni parcheggio restituito nella lista, saranno visualizzate le informazioni recuperate dal sistema centrale:

1. stato del parcheggio (aperto, chiuso, lavori in corso, …);
2. disponibilità di posti;
3. tasso di occupazione.

1.15 Implementazione

Versione 0.1

La prima versione del modulo (V 0.1) permette all'utente di visualizzare i dati relativi ai parcheggi in zona (basata su posizione GPS o su un punto manuale scelto sulla mappa) con evidenza dei dati dinamici dei parcheggi che espongono tali informazioni. Ecco i dati dinamici esposti:

- posti liberi del parcheggio;
- capacità del parcheggio;
- ultimo aggiornamento sui dati dinamici del parcheggio;
- marker sulla disponibilità di posti del parcheggio (semaforo rosso o verde).

Il file del modello Mustache implementa la struttura dinamica HTML della vista del modulo: *.\www\templates\EasyParkMenu.mst.html*. Aggiunti, rispetto al modulo di riferimento relativo ai parcheggi sviluppato da DISIT, i seguenti segmenti di codice relativi ai dati dinamici dei parcheggi stessi:

```
{{#freeParkingLots}}<b id="easyParkFreeParkingLotsLabel{{identifier}}">
<script>$("#easyParkFreeParkingLotsLabel{{identifier}}").html(Globalization.labels.easyParkLabels.freePar
kingLots)</script>
</b><b style="color: {{freeParkingLotsColor}}">{{freeParkingLots}}</b><br>{{/freeParkingLots}}

{{#capacity}}<b id="easyParkCapacityLabel{{identifier}}">
<script>$("#easyParkCapacityLabel{{identifier}}").html(Globalization.labels.easyParkLabels.capacity)</scri
pt>
</b>{{capacity}}<br>{{/capacity}}

{{#updating}}<b id="easyParkLastUpdateLabel{{identifier}}">
<script>$("#easyParkLastUpdateLabel{{identifier}}").html(Globalization.labels.easyParkLabels.updating)</
script>
</b>{{updating}}{{/updating}}

{{#marker}}<b style="position: absolute; top: 4px; right: 10px; font-size: 100px; color:
{{freeParkingLotsColor}}">{{marker}}</b>{{/marker}}
```

È stata poi aggiunta una cartella *EasyPark* nel path *.\www\js\modules*: essa conterrà il sorgente JSON e JavaScript del nuovo modulo.

Il file *.\www\js\modules\EasyPark\easyParkSearcher.principalMenu.json* contiene il JSON di configurazione del modulo:

```
[{          "callback":"PrincipalMenu.hide();MapManager.centerMapOnGps();SearchManager.search('EasyP
arkSearcher');",
  "iconId": "",
  "iconClass": "",
  "iconFontSize": "",
  "iconColor": "",
  "imgSrc": "",
  "imgHeight": "",
  "text": "ePark",
  "textFontSize": "38px",
  "textColor": "#CC0000",
  "captionId": "principalMenuEasyParkSearcher",
  "captionTextId": "moduleEasyParkSearcher",
  "step": "",
  "stepId": "",
  "ribbon": true,
  "ribbonId": "",
  "ribbonStyle": "background: #CC0000;background: linear-gradient(#FF6600 0%, #CC0000 100%);",
  "ribbonText":  "NEW",
  "removed": false,
  "index": 0
 }]
```


La nuova cartella contiene poi i 5 file di internazionalizzazione delle stringhe:

- tedesco: .*www\js\modules\EasyPark\easyParkSearcher.labels.deu.json*
- inglese: .*www\js\modules\EasyPark\easyParkSearcher.labels.eng.json*
- spagnolo: .*www\js\modules\EasyPark\easyParkSearcher.labels.esp.json*
- francese: .*www\js\modules\EasyPark\easyParkSearcher.labels.fra.json*
- italiano: .*www\js\modules\EasyPark\easyParkSearcher.labels.ita.json*

Ognuno dei file sopra descritti contiene le stringhe del nome del modulo, dei campi di ricerca, delle etichette utilizzate in interfaccia:


```
{
 "principalMenu": {
  "moduleEasyParkSearcher": "Lista Easy Park"
 },
 "easyParkMenu": {
  "title": "Easy Park",
  "gpsOrderText": "Più vicini <i class=\"glyphicon glyphicon-record\">",
  "searchCenterOrderText": "Più vicini <i class=\"glyphicon glyphicon-map-marker\">",
  "freeParkingOrderText":  "Posti liberi"
 },
 "easyParkLabels": {
  "freeParkingLots": "Posti liberi: ",
  "capacity": "Capacità: ",
  "updating": "Ultimo aggiornamento: "
 }
}
```

Infine il file *.\www\js\modules\EasyPark\EasyParkSearcher.js* implementa il codice JavaScript delle funzioni e della logica del nuovo modulo.

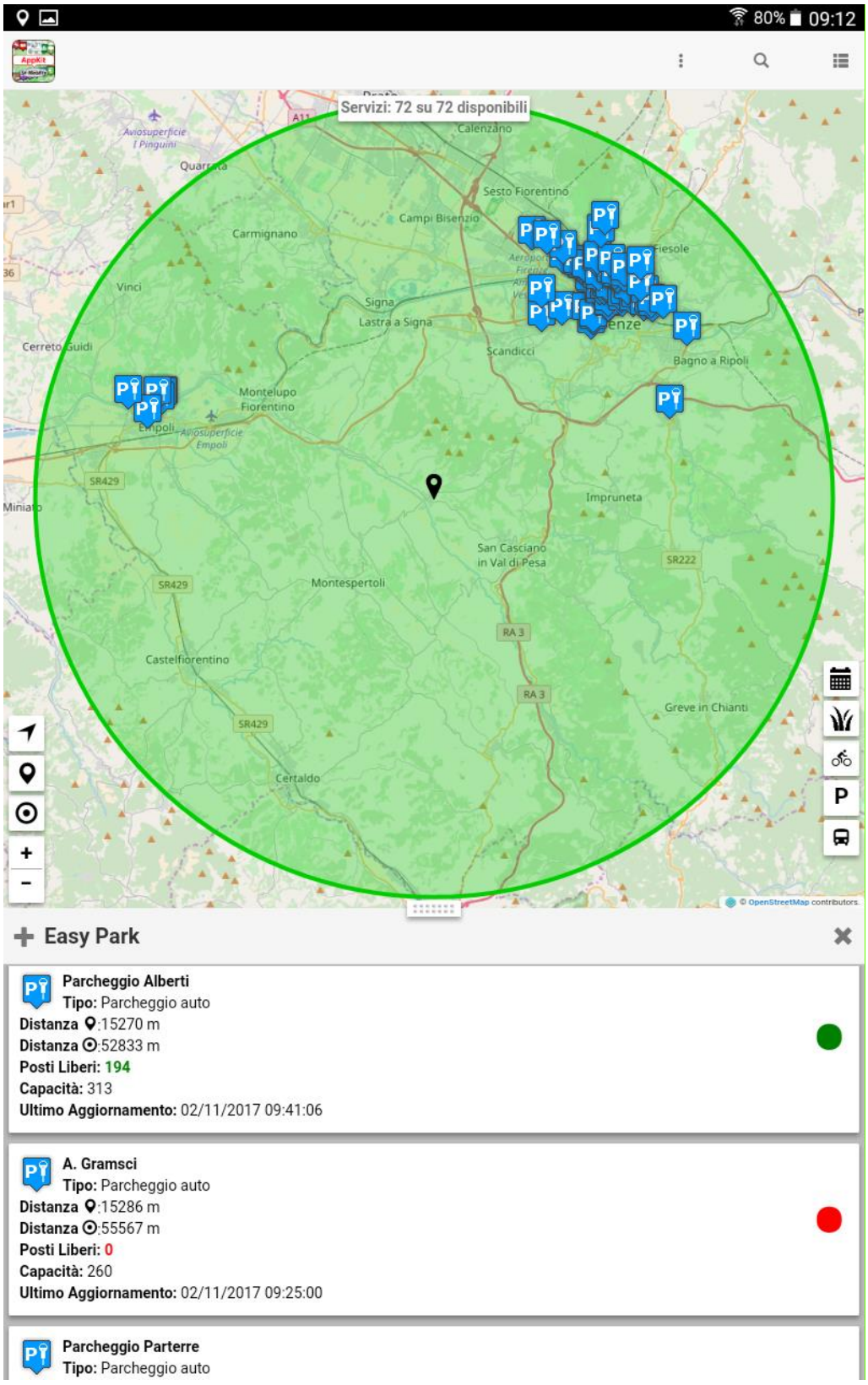
Viene riportato qui lo snippet del nuovo codice per la logica dei dati dinamici relativi al parcheggio:

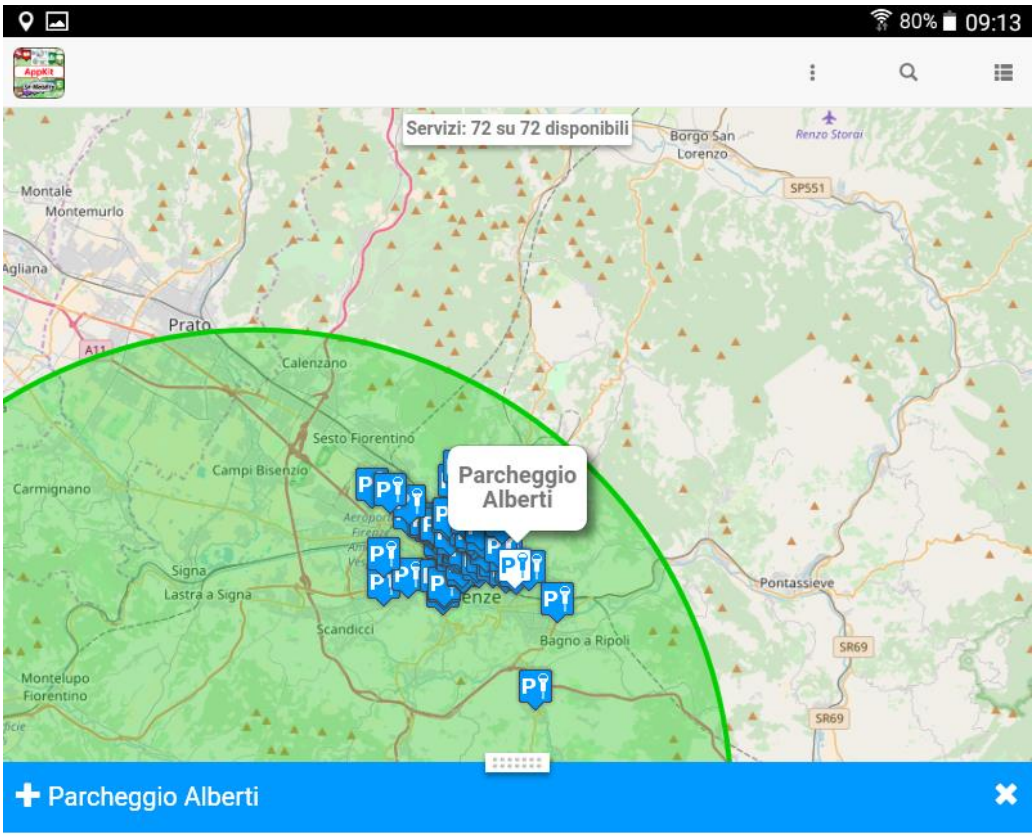
```
response[category].features[0].properties.freeParkingLots                                    =
response.realtime.results.bindings[0].freeParkingLots.value;
if (response[category].features[0].properties.freeParkingLots > 20) {
   response[category].features[0].properties.freeParkingLotsColor = "green";
} else if (response[category].features[0].properties.freeParkingLots > 0) {
   response[category].features[0].properties.freeParkingLotsColor = "orange";
} else {
   response[category].features[0].properties.freeParkingLotsColor = "red";
}

response[category].features[0].properties.capacity = response.realtime.results.bindings[0].capacity.value;

var date = new Date(response.realtime.results.bindings[0].updating.value);
var day = date.getDate() < 10 ? ("0" + date.getDate()) : date.getDate();
var monthIndex = (date.getMonth() + 1) < 10 ? ("0" + (date.getMonth() + 1)) : (date.getMonth() + 1);
var year = date.getFullYear();
var hours = date.getHours() < 10 ? ("0" + date.getHours()) : date.getHours();
var minutes = date.getMinutes() < 10 ? ("0" + date.getMinutes()) : date.getMinutes();
var seconds = date.getSeconds() < 10 ? ("0" + date.getSeconds()) : date.getSeconds();
response[category].features[0].properties.updating = day + "/" + monthIndex + "/" + year + " " + hours + ":"
+ minutes + ":" + seconds;

response[category].features[0].properties.marker = "•";
```

Di seguito due screenshot per esplicitare il comportamento del nuovo modulo EasyPark: una volta entrati nel modulo dell'app è possibile scegliere, attraverso un *AlertDialog*, se cercare i parcheggi attorno al corrente punto GPS oppure se scegliere manualmente un punto sulla mappa attorno al quale effettuare la ricerca.

Avviata la ricerca, viene chiamata la API verso ServiceMap che restituisce i dati statici (nome del parcheggio, indirizzo, posizione GPS del parcheggio, ecc.) e dinamici dei parcheggi trovati.

Sotto la mappa compare la lista dei parcheggi, arricchita dei dati dinamici dei parcheggi che li rendono disponibili. Cliccando su una riga dell'elenco dei parcheggi si espande le informazioni relative al parcheggio selezionato.


80% 09:12
Servizi: 72 su 72 disponibili
Calenzano
Sesto Fiorentino
Campi Bisenzio
Carmignano
Vinci
Signa
Lastra a Signa
Scandicci
Bagno a Ripoli
Montelupo Fiorentino
Empoli
Impruneta
San Casciano in Val di Pesa
Montespertoli
Castelfiorentino
Greve in Chianti
Certaldo
SR429
SR222
RA 3
© OpenStreetMap contributors
Easy Park
Parcheggio Alberti
Tipo: Parcheggio auto
Distanza :15270 m
Distanza :52833 m
Posti Liberi: 194
Capacità: 313
Ultimo Aggiornamento: 02/11/2017 09:41:06
A. Gramsci
Tipo: Parcheggio auto
Distanza :15286 m
Distanza :55567 m
Posti Liberi: 0
Capacità: 260
Ultimo Aggiornamento: 02/11/2017 09:25:00
Parcheggio Parterre
Tipo: Parcheggio auto



Tipo: Parcheggio auto

Descrizione: Abbonamenti: N/A----Abbonamenti_autovetture:- Abbonamento settimanale 24h: ? 70,00* - Abbonamento mensile 24h ordinario: ? 120,00* - POSTI DISPONIBILI - Abbonamento mensile 24h riservato ai soli residenti: (? 80,00) PROMOZIONALE ? 60,00* - POSTI DISPONIBILI - Abbonamento mensile diurno valido esclusivamente nella fascia oraria dalle ore 8,00 alle ore 20,00 dal lunedì al sabato, eccetto domeniche e festivi: ? 75,00* - Tessera a scalare dal valore di ? 120,00 Euro al costo di ? 100,00* *salvo disponibilità, tariffa attivabile esclusivamente con pagamento anticipato. Validità - mese solare, ovvero dal primo all?ultimo giorno del mese solare di riferimento indipendentemente dalla data di acquisto. L? abbonamento mensile potrà essere utilizzato anche in un mese successivo a quello dell?acquisto.----Abbonamenti_moto: - Abbonamento mensile 24h ordinario: ? 50,00* *salvo disponibilità, tariffa attivabile esclusivamente con pagamento anticipato. Validità - mese solare, ovvero dal primo all?ultimo giorno del mese solare di riferimento indipendentemente dalla data di acquisto. L?abbonamento mensile potrà essere utilizzato anche in un mese successivo a quello dell?acquisto.---- Tariffa_giornaliera: N/A----Tariffa_oraria_autovetture: N/A----Tariffa_oraria_ordinaria_lun_sab: N/A---- Tariffa_oraria_speciale_lun_sab: N/A----Tariffe: - Tariffa oraria: dalle ore 08:00 alle ore 20:00: ? 1,60 ogni ora o frazione di ora - Tariffa oraria: dalle ore 20:00 alle ore 08:00: ? 1,00 tariffa notturna a forfait - Tariffa giornaliera: ? 20,00 ---- Tariffe_dom_festivi: N/A----Tariffe_moto: N/A

Indirizzo: VIA DEL CAMPOFIORE, 50136 Firenze FI

Telefono: 05550302209

Fax: 05550302219

Posti Liberi	Posti Occupati
194	119
Capacità	**Occupazione (%)**
313	

2017-11-02T09:41:06+01:00

1.16 Sviluppi futuri

Durante la fase di specifica sono state individuate una serie di caratteristiche aggiuntive che sono in fase di valutazione per essere inserite nelle versioni successive del modulo. La scelta e la possibilità di realizzazione dipenderà principalmente dalla disponibilità di relative API nel sistema centrale SIIMobility che implementino alcune funzionalità necessarie.

L'inserimento da parte dell'utente del luogo intorno a cui avviare la ricerca del posto auto potrebbe essere potenziato permettendo (oltre alle due modalità già presenti):

1. di fornire un indirizzo conosciuto (es. “via De' Calzaiuoli n. 3, Firenze”);
2. di selezionare un POI tra un insieme di preferiti (es. “Casa”,”Ufficio”, ecc...);
3. di individuare un POI tramite ricerca per nome (es. “Galleria degli Uffizi, Firenze”).

Per permettere tali modalità di inserimento, è necessaria l'esistenza sul sistema centrale di un servizio di “GeoCoding” che risolva le richieste del modulo.

Per ogni parcheggio restituito dalla ricerca, si può individuare e restituire all'utente una serie di informazioni aggiuntive “**derivate**” ovvero informazioni ottenute dall'elaborazione dei dati presenti nel sistema (attuali o storici):

- distanza del parcheggio dal luogo inserito nella ricerca;
- trend di occupazione calcolato negli ultimi 5 min (si sta riempiendo, si sta svuotando, ocupazione costante, ecc...);
- stime di occupazione (a breve, ad esempio nei prossimi 3-5 min, o in data/orario da inserire).

Anche in questo caso, sarà necessario che sul sistema centrale SIIMobility siano disponibili servizi (API o WS) che forniscano i dati aggregati e le predizioni da visualizzare parcheggio per parcheggio.

1.17 Richieste

Nessuna richiesta.

(3.5.10) Sarebbe opportuno che...

1.18 Descrizione

In modalità navigazione, il City User, che percorre abitualmente gli stessi itinerari ma che è comunque geolocalizzato dalla app, può ricevere da Sii-Mobility delle notifiche che gli consigliano di deviare dal percorso abituale.

1.19 Funzionalità

1.19.1 Suggerimenti opportuni di aiuto (Softec)

Questa funzionalità ha come fine il fornire all'utente indicazioni in tempo reale sulle strade o vie alternative che potrebbe prendere per raggiungere le sue destinazioni abituali, tenendo conto di eventuali problematiche, come incidenti o traffico intenso, per fargli ridurre il carico su certe direttrici.

L'applicazione, con i servizi di geolocalizzazione attivi, se si troverà in modalità navigazione, riceverà direttamente dal sistema le indicazioni di eventuali itinerari alternativi. Se l'applicazione non sarà in modalità navigazione, il sistema si occuperà di prevedere la possibile destinazione dell'utente basandosi sui suoi percorsi abituali e di notificargli in tempo reale eventuali alternative al percorso abituale nel caso di presenza di problematiche su quest'ultimo (incidente, traffico, lavori in corso, ecc…). L'utente, una volta aperta l'applicazione tramite la notifica inviata dal sistema, visualizza nel modulo un pop-up con spiegata l'eventuale problematica a cui andrebbe incontro sul suo possibile percorso, dandogli la possibilità di scegliere se avviare la navigazione verso la sua possibile destinazione seguendo una strada alternativa oppure no.

Se data la conferma verrà richiamata una funzione nel modulo che disegnerà sulla mappa il percorso alternativo inviato dal sistema all'app, chiedendo conferma all'utente per l'avvio della navigazione

1.19.2 Condivisione suggerimenti utenti in real time (Softec)

Gli utenti avranno facoltà di poter inviare e visualizzare col resto del network le proprie segnalazioni sulle criticità riscontrate durante i loro tragitti. Selezionando un punto sulla mappa, potranno usare l'opzione *Invia segnalazione* per segnalare il problema riscontrato (traffico, incidenti, lavori in corso, ecc…) incontrato in quel determinato punto. Una funzione si occuperà di inviare all'API del la segnalazione effettuata dall'utente per poter essere successivamente processata da Sii-Mobility.

L'utente riceverà in risposta un pop-up con un messaggio che lo ringrazierà per la segnalazione.

1.20 Attività svolte

Analisi del contesto di utilizzo del modulo all'interno dell'applicazione.

- Creato la cartella "SarebbeOpportunoChe" dentro la path /www/js/modules che contiene il codice del modulo.
- Creato il branch su Git chiamato SoftecSiiMobility/softec-3.5.10-sarebbe_opportuno_che

- Aggiunta possibilità di inserimento di una segnalazione (es. incidente, traffico congestionato). **Manca API per il salvataggio delle segnalazioni**.

1.21 Richieste

Sono stati richiesti a UNIFI degli sviluppi lato API e maggiori informazioni riguardo:

- fire (**Notifica Push**) da parte di Sii-Mobility che richiami il modulo nel momento in cui il sistema riconosca che c'è del traffico - o comunque altri impedimenti al normale itinerario - per informare l'utente della problematica sul percorso e proporgli dei percorsi alternativi. *La gestione di questo flusso è prioritaria anche per lo sviluppo di altri moduli analoghi, in cui le scelte dell'utente vengono "innescate" da una notifica push proveniente da Sii-Mobility, in concomitanza con gli eventi individuati.*
- **API per il salvataggio delle segnalazioni**, anche con parametro coordinate GPS (ossia segnalazioni non legate a POI).
- API che invia dal sistema, a seguito della richiesta dell'utente, il percorso modificato per disegnarlo successivamente sulla mappa.

(3.5.11) Modulo sapevi che nel…

1.22 Descrizione

Questo modulo sfrutta gli strumenti di geolocalizzazione dell'app per individuare dove si trova l'utente e se si trova in prossimità di un certo punto di interesse che magari può rientrare nelle sue preferenze culturali sulla base del suo profilo. Il sistema gli invia una notifica in push (il telefono chiede ed il server invia) dove si informa che riguardo a quel punto si possono raccontare aspetti presi da informazioni in rete (Wikipedia o simili) o da altra sorgente. Possibile necessità di caricare un modulo computazionale aggiuntivo in piattaforma Sii-Mobility (per elaborare e fare cache delle info estratte da dbpedia o da altri) oppure una base di dati.

1.23 Funzionalità

1.23.1 Suggerimenti di prossimità culturali (Softec)

Questa funzionalità individua la posizione dell'utente mediante gli strumenti di geolocalizzazione dell'app. Se in prossimità dell'utente è presente un punto di interesse compatibile col suo profilo, il sistema invierà una notifica all'app che informerà l'utente della presenza di quel dato punto di interesse che potrebbe interessargli. Il sistema invia la notifica, l'utente apre l'app e un messaggio lo informa della presenza di questo POI di suo probabile interesse, con spiegati aspetti del punto di interesse presi tramite API. L'utente può ignorare il messaggio o dare la conferma all'applicazione di avviare la navigazione verso quello specifico punto di interesse. In caso di conferma di navigazione, l'app avvierà la modalità navigazione, disegnerà sulla mappa il percorso e guiderà l'utente fino al punto di interesse.

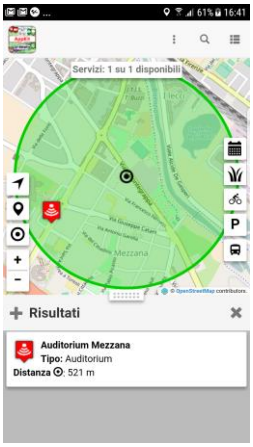
1.23.2 Gestione notifiche push (Ewings)

In corrispondenza della notifica di messaggi proveniente da Sii-Mobility verso la app attiva del City User, questi potrà visualizzare sulla mappa il POI segnalato, e aprire il dettaglio per consultare ulteriori informazioni da fonti esterne.

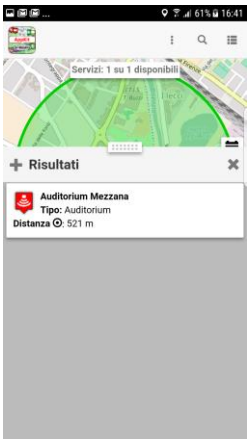
1.24 Attività svolte

Per questo modulo, "Sapevi che nel…" abbiamo svolto le seguenti attività:

- Creato la cartella "sapeviCheNel"dentro la path /www/js/modules che contiene il codice del modulo.
- Creato il branch su Git chiamato oftecSiiMobility/softec-3.5.11-sapevi_che_nel
- Mostrare sulla mappa tutti i punti culturali nei pressi della geolocalizzazione dell'utente



- Gestire il menu in basso con la lista dei vai punti di interesse e la loro distanza



- Click sulla lista o sulla mappa e apre il dettaglio del punto culturale

- Pushato e committato sul repository Git
- Fatta la Pull Request sul mast/origin del repository principale disit/siiMobilityAppKit che è stata accettata e quindi mergiata nel master/origini principale nella commit "afa71217b8c2d897813133509621d10e6d8c202b"

1.25 Richieste

Sono stati richiesti a UNIFI degli sviluppi lato API e maggiori informazioni riguardo:

- se nella mappa veniva scelto un punto preciso e successivamente veniva cliccato **"Qui intorno"** prima mostrava tutti i punti (anche quelli non culturali).
- fire (**Notifica Push**) da parte di Sii-Mobility nel caso il sistema individui un punto di interesse compatibile col profilo dell'utente e invio tramite API dati ed informazioni su quel determinato punto di interesse.