



# FODD 2015 Mobile App

*FODD, Florence Open Data Day*  
*Salone de' Dugento, Palazzo Vecchio, Firenze*  
21/02/2015

Ing. Ph.D Ivan Bruno

Dipartimento di Ingegneria dell'Informazione, DINFO  
Università degli Studi di Firenze  
Via S. Marta 3, 50139, Firenze, Italy  
Tel: +39-055-2758511, fax: +39-055-2758570

**DISIT Lab**

<http://www.disit.org>  
[ivan.bruno@unifi.it](mailto:ivan.bruno@unifi.it)



# FODD 2015 App

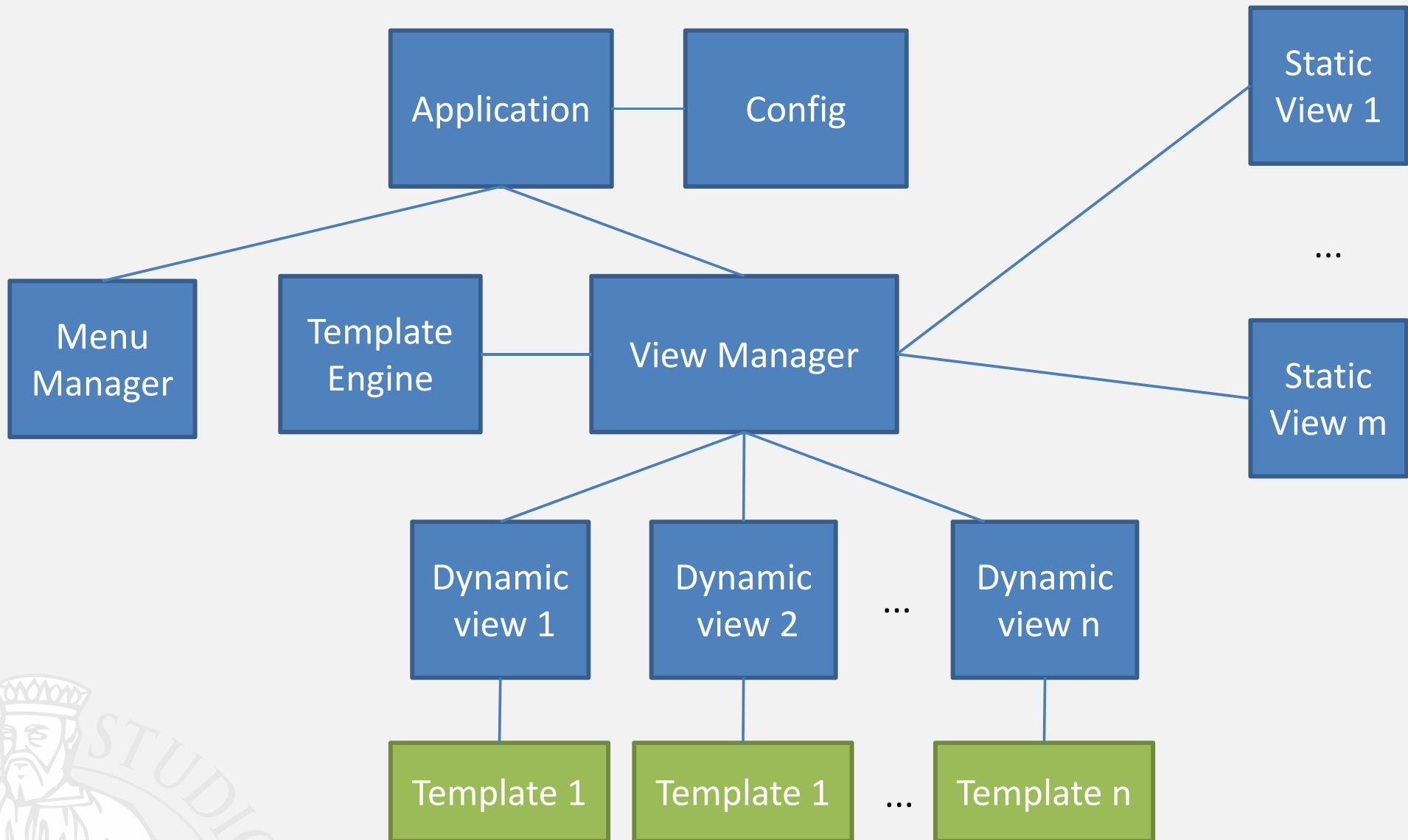
- **Obiettivo**
  - Utilizzare i servizi (API REST) esposti da [servicemap.disit.org](http://servicemap.disit.org)
  - Visualizzare informazioni tempo reale / dinamiche
  - Realizzare un app per l'evento
  - Non solo una demo ma un'app estendibile e modificabile
  - Semplificazione



# Requisiti Principali

- Menu configurabile
- Gestione viste: una logica di gestione delle viste **statiche** e di quelle **dinamiche** da costruire a runtime sui dati JSON provenienti dalle chiamate REST via AJAX
- Semplificare la gestione delle viste costruite sui dati JSON utilizzando soluzioni **template-based**
- Rilevazione stato connessione internet del dispositivo
- Notifica di anomalie (connessione assente, errori di connessioni al server...)
- Portabilità su diversi dispositivi mobili

# Architettura



# Architettura: descrizione

- **Application:** il gestore dell'applicazione
- **Config:** variabili di configurazione
- **Menu Manager:** gestore del menu
- **View Manager:** gestore cambio vista e rendering
- **Template Engine:** gestore rendering dei template
- **Dynamic View:** gestore delle singole viste, specializzato nell'elaborazione dei dati richiesti via REST Call e nel rendering via *template engine*.



# TECNOLOGIE UTILIZZATE





# Mobile Apps

- Solitamente quando pensiamo ad applicazioni per dispositivi **mobile**, pensiamo ad applicazioni native, cioè scaricabili ed installabili su uno specifico dispositivo.
- In effetti lo sviluppo di applicazioni native resta ancora oggi il miglior modo per sfruttare pienamente le potenzialità di smartphone e tablet ed offrire un'esperienza utente integrata e coerente fra tutte le applicazioni.
- Il problema di questo approccio è che per supportare più di un sistema operativo è necessario sviluppare diverse applicazioni, moltiplicando così i tempi di realizzazione ed il codice da mantenere.
- La migliore risposta a questo problema arriva oggi con i **framework** per applicazioni web mobile, che hanno come primo obiettivo quello di fornire un'esperienza utente il più vicino possibile a quella nativa pur garantendo il supporto ad un largo numero di dispositivi e sistemi operativi diversi.



# Apache Cordova

- Apache Cordova è un insieme di API JavaScript che consentono allo sviluppatore di applicazioni per dispositivi di accedere alle funzioni native del dispositivo come la fotocamera o accelerometro, lo storage, la rete, il gps....
- Combinato con un framework di interfaccia utente, come jQuery Mobile o Dojo Mobile o Sencha Touch, permette lo sviluppo di applicazioni smartphone utilizzando solo HTML, CSS e JavaScript.
- Quando si utilizzano le API Cordova, un'applicazione può essere costruita senza alcun codice nativo (Java, Objective-C, ecc). Le tecnologie web utilizzate sono ospitate nella stessa applicazione a livello locale (in genere non su un server http remoto).
- Queste API JavaScript sono coerenti e valide per le differenti piattaforme di dispositivi mobili, in questo modo l'applicazione costruita sullo standard web, dovrebbe essere portabile con un minimo di cambiamenti.



# Apache Cordova

- Apps basate Cordova vengono realizzate come applicazioni package utilizzando le SDK della piattaforma, e possono essere resi disponibili per installazione dagli App Store di ciascun dispositivo.
- Cordova offre un set di librerie JavaScript che possono essere invocate in modo trasparente utilizzando il codice nativo della dispositivo.
- Cordova è disponibile per: iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, Symbian e.
- Documentazione e dowload sono disponibili sul sito ufficiale:  
<http://cordova.apache.org/>



# Cordova plugin

- cordova-plugin-battery-status@0.2.12
- cordova-plugin-camera@0.3.5
- cordova-plugin-console@0.2.13
- cordova-plugin-contacts@0.2.16
- cordova-plugin-device-motion@0.2.11
- cordova-plugin-device-orientation@0.3.11
- cordova-plugin-device@0.3.0
- cordova-plugin-dialogs@0.3.0
- cordova-plugin-file-transfer@0.5.0
- cordova-plugin-file@1.3.3
- cordova-plugin-geolocation@0.3.12
- cordova-plugin-globalization@0.3.4
- cordova-plugin-inappbrowser@0.6.0
- cordova-plugin-media@0.2.16
- cordova-plugin-media-capture@0.3.6
- cordova-plugin-network-information@0.2.15
- cordova-plugin-splashscreen@1.0.0
- cordova-plugin-vibration@0.3.13
- cordova-plugin-statusbar@0.1.10
- cordova-plugins@file-system-roots-0.1.0
- cordova-plugin-test-framework@0.0.1

...e altri reperibili in rete

E possibile realizzare propri plugins seguendo la guida per lo sviluppo.



# jQuery Mobile

- Libreria Javascript per mobile discendente della più nota JQuery
- La caratteristica peculiare di **jQuery Mobile** è che per realizzare un'applicazione base, basterà scrivere del codice **HTML5**.
- La libreria, infatti, fa leva sulla struttura semantica delle pagine HTML5 e sugli attributi data-\* per definire le varie parti dell'interfaccia.
- Una volta caricata la pagina, **jQuery Mobile** utilizzerà questa struttura per arricchirla con altri tag e *agganciare* gli eventi e le interazioni ai componenti dell'applicazione.
- Non richiede tecnologie particolari, per molti browser mobile JavaScript lo è, per rendere un contenuto navigabile.

# jQuery Mobile

- Per usare jQuery Mobile è necessario utilizzare:
  - un file css con lo stile usato da jQuery Mobile
  - un file javascript contenente il codice di jQuery
  - un file javascript con il codice sorgente di jQuery Mobile
- Sito ufficiale: <http://jquerymobile.com/>



```
<!DOCTYPE html>
<html>
<head>
  <title>CoseNonJaviste</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css" />
  <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.js"></script>
</head>
<body>

<div data-role="page">
  <div data-role="header">
    <h1>CoseNonJaviste</h1>
  </div>
  <div data-role="content">
    <p>Hello world!</p>

  </div>
  <div data-role="footer" data-position="fixed">
    <h4>Footer</h4>

  </div>

</div>
</body>
</html>
```



Fonte:

<http://www.cosenonjaviste.it/tutorial-jquery-mobile-come-creare-una-semplce-web-app-per-smartphone/>

# jQuery Mobile

- Ovviamente il risultato non è ai livelli delle app native per Android e iOS, per esempio nelle transazioni fra una pagina e l'altra soprattutto su dispositivi Android si nota un po' di ritardo.
- Comunque con il passare del tempo il gap si sta assottigliando sempre di più grazie al miglioramento dei browser e dell'hardware dei dispositivi mobile.
- La curva di apprendimento per imparare a usare questo framework è quasi piatta, se siete sviluppatori web anche senza troppa esperienza vi sentirete subito a vostro agio.



# Mustache JS

- La libreria è indipendente da framework specifici ma esistono plugin per l'integrazione con jQuery, Dojo e YUI.
- Consente di manipolare gli oggetti javascript e quindi di sfruttare la comunicazione dei dati in formato JSON provenienti da una chiamata REST via AJAX.
- I template per Mustache possono essere assegnati o caricati come stringa ad una variabile e i placeholder sono identificati da due parentesi graffe, ad esempio: `{{miopplaceholder}}`.
- Una delle feature più interessanti della libreria è il **supporto a valori enumerabili (array) e condizionali (boolean)** che permettono di realizzare strutture HTML complesse.





```
var data = {  
  risultato: true,  
  titolo: Città italiane,  
  descrizione: Lista delle città italiane,  
  citta: [  
    {nome: Milano, sigla: MI},  
    {nome: Roma, sigla: RM}  
  ]  
};
```

```
var data2 = {  
  risultato: false,  
  titolo: Città italiane 2,  
  descrizione: Lista delle città italiane 2,  
  citta: []  
};
```

```
<h1>{{titolo}}</h1>  
<p>{{descrizione}}</p>  
{{#risultato}} //solo se risultato è true  
  <ul>{{#citta}}  
    <li>{{nome}} ({{sigla}})</li>  
  {{/citta}}</ul>  
{{/risultato}}  
{{^risultato}} //altrimenti...  
  <p><em>Nessuna città trovata!</em></p>  
{{/risultato}};
```

## Città italiane

Lista delle città italiane

- Milano (MI)
- Roma (RM)

## Città italiane 2

Lista delle città italiane 2

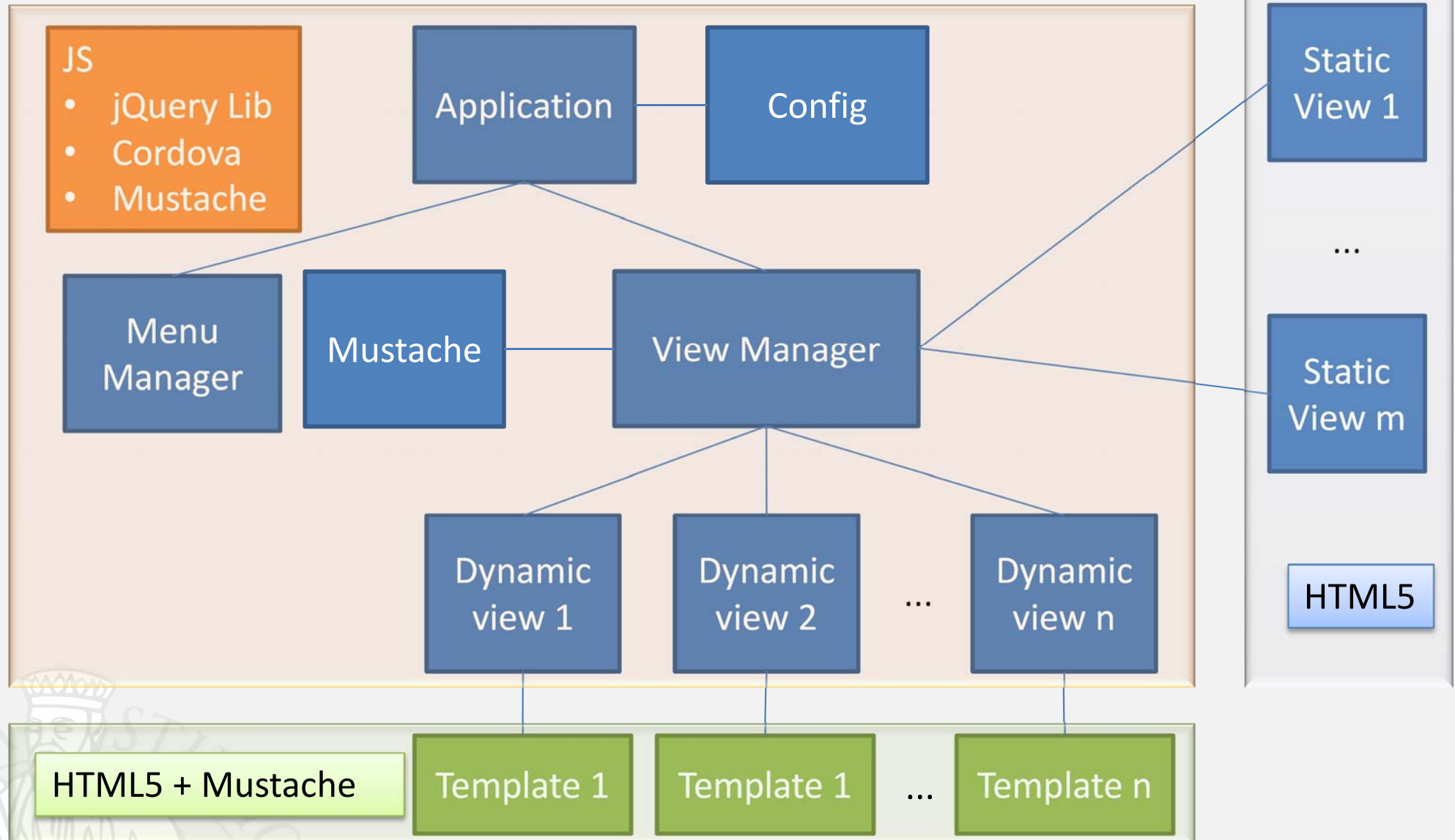
*Nessuna città trovata!*



# SPECIFICHE PROGETTO APP



# Architettura & Tecnologie



# Configurazione App

- **Timeout** – tempo di attesa sulle connessioni internet
- **Server** – url del server per le chiamate REST

```
var config={  
    "server":"http://servicemap.disit.org/WebAppGrafo/api?",  
    "timeout":30000,  
};
```



# Menu: struttura

- **Programma**
  - Agenda dell'evento FODD
- **Servizi Vicini**
  - Visualizzazione mappa con dislocazione di Sensori traffico, fermate bus e Servizi
- **Ponte Vecchio (DL)**
  - Informazioni sul Ponte Vecchio (Digital Location)
- **Previsione Meteo**
  - Previsione del tempo su Firenze
- **Stato alla Pensilina**
  - Orario di transito autobus alla Fermata Pensilina in P.zza Stazione a Firenze
- **Parcheggio Stazione**
  - Stato parcheggio della stazione
- **Parcheggio Empoli**
  - Stato parcheggio Via Buozzi
- **Sensore Empoli**
  - Rilevazione tempo reale di un sensore di traffico
- **Leggimi**
  - Pagina informativa sull'app
- **Exit**
  - Uscita dall'applicazione

# Viste, template e chiamate REST (1)

- Le viste dinamiche coinvolte nelle chiamate REST sono:
  - Servizi Vicini
    - **Titolo:** Servizi Vicini
    - **Vista (view):** servizi
    - **Template:** servizi.mst.html
    - **url:** REST Call URL
  - Previsione Meteo
    - **Titolo :** Previsione Meteo
    - **Vista (view):** meteo
    - **Template:** meteo.mst.html
    - **url:** REST Call URL
  - Stato alla Pensilina
    - **Titolo:** Stato alla Pensilina
    - **Vista (view):** bus
    - **Template:** palina.mst.html
    - **url:** REST Call URL



# Viste, template e chiamate REST (2)

- Ponte Vecchio
  - **Titolo:** Ponte Vecchio (DL)
  - **Vista (view):** digitalLocation
  - **Template:** location.mst.html
  - **url:** REST Call URL
  
- Parcheggio Stazione
  - **Titolo:** Parcheggio Stazione
  - **Vista (view):** parking
  - **Template:** parking.mst.html
  - **url:** REST Call URL
  
- Parcheggio Empoli
  - **Titolo:** Parcheggio Empoli
  - **Vista (view):** parking
  - **Template:** parking.mst.html
  - **url:** REST Call URL
  
- Sensore Empoli
  - **Titolo:** Sensore Empoli
  - **Vista (view):** sensori
  - **Template:** sensori.mst.html
  - **url:** REST Call URL





# REST CALL

- `config.server = "http://servicemap.disit.org/WebAppGrafo/api?"`
- Servizi Vicini
  - `config.server+"selection=COMUNE%20di%20FIRENZE&categorie=Accommodation%3Bboarding_house%3Bagritourism%3Bhotel%3Bbed_and_breakfast%3Bcamping%3Brest_home%3Breligios_guest_house%3Bsummer_residence%3Bday_care_center%3Bhostel%3Bvacation_resort%3Bfarm_house%3Bhistoric_residence%3Bmountain_dew%3Bbeach_resort%3Bholiday_village%3BRoadSensor%3BNearBusStops&risultati=100&raggio=100&format=json«`
- Ponte Vecchio (DL)
  - `config.server+"serviceUri=http://www.disit.org/km4city/resource/4573d318065f48a2214066f7b32d94c4&format=json"`
- Previsioni Meteo
  - `config.server+"serviceUri=http://www.disit.org/km4city/resource/Firenze1423642032000&format=json"`

# REST CALL

- Stato alla Pensilina
  - config.server+"serviceUri=<http://www.disit.org/km4city/resource/FM0022>&format=json"
- Parcheggio Stazione
  - config.server+"serviceUri=<http://www.disit.org/km4city/resource/RT04801702315PO>&format=json"
- Parcheggio Empoli
  - config.server+"serviceUri=[http://www.disit.org/km4city/resource/RT048014PK003EM\\_EMPOLI](http://www.disit.org/km4city/resource/RT048014PK003EM_EMPOLI)&format=json"
- Sensore Empoli
  - config.server+"serviceUri=<http://www.disit.org/km4city/resource/EM0100102>&format=json"

# Viste statiche

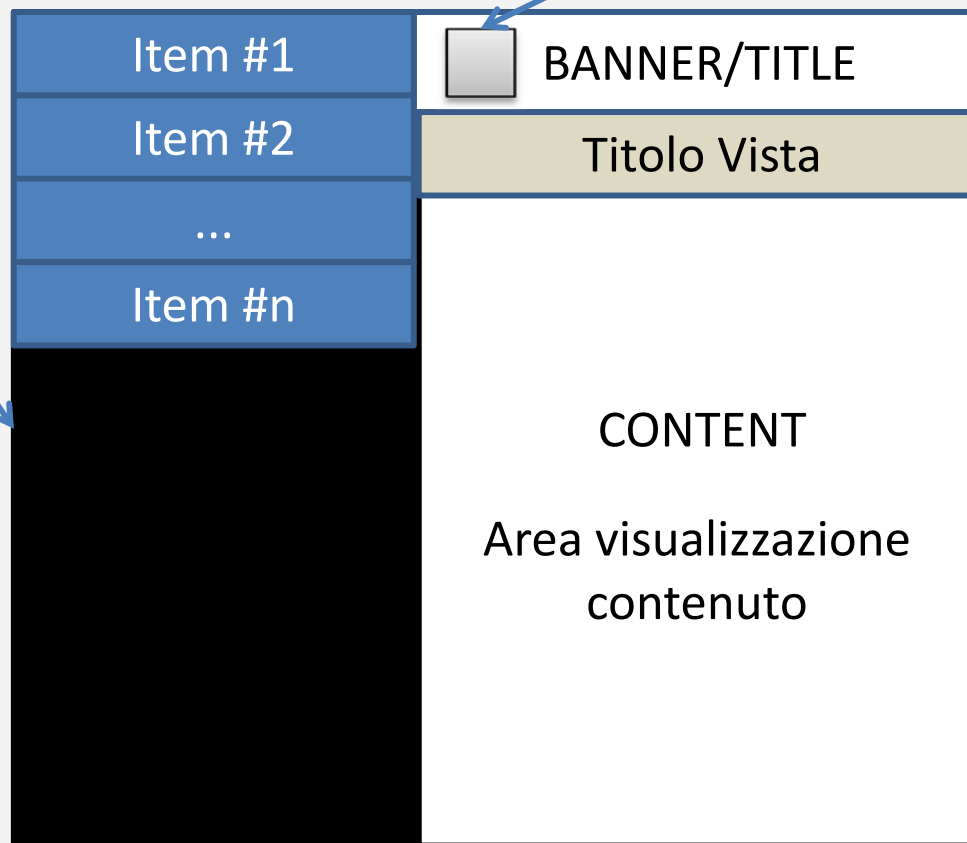
- Le viste statiche del menu sono:
  - Programma
    - **Titolo:** Programma
    - **url:** agenda.html
  - Leggimi
    - **Titolo:** Leggimi
    - **url:** about.html



# Vista Principale App

Menu Laterale a  
Scomparsa

Pusante attivazione Menu



Intestazione Fissa  
ospita il titolo o un  
banner. Allo scroll non  
si muove



# Vista Principale

- L'idea di base è che tutte le viste sia statiche che dinamiche vengano inserite nell'area «CONTENT» andando a rimpiazzare la precedente
- All'avvio l'app mostrerà la vista corrispondente alla prima voce del menu
- Il pannello a scomparsa...perchè lo fanno tutte le app!!



# Modello JQM/HTML Vista Principale

```
<div data-role="page" id="index" role="main">
  <div data-role="header" class="page-header" data-position="fixed">
    <a href="#menu-panel" id=menu-btn><i class=icon-menu></i></a>
    
  </div>
  <div id=menu-panel data-role="panel" data-position="left" data-display="push"
data-theme="b">
    <ul id=main-menu data-role=listview>
      <li id="menu-title" data-icon="">
        <b>Open Data Day App Menu</b>
      </li>
    </ul>
  </div>
  <div id="subheader" data-role="header">
    <h1 id="titlebar">View Title</h1>
  </div>
  <div id=content data-role="main" class="ui-content" data-enhance="true">
  </div>
</div>
```

# JS Menu & Pagine

```
var menu={
  "pages": [
    {
      "id":"agenda",
      "title": "Programma",
      "url":"agenda.html",
      "icon":"icon-list",
      "name":"Agenda"
    },
    {
      "id":"meteo",
      "title": "Previsioni Meteo",
      "url":config.server+"serviceUri=http://www.disit.org/km4city/resource/Firenze14236420
32000&format=json",
      "icon":"icon-cloud",
      "view":"meteo",
      "template":"templates/meteo.mst.html",
    },
    ...
  ]};
```

Vista statica



Vista Dinamica con Url REST Call e template

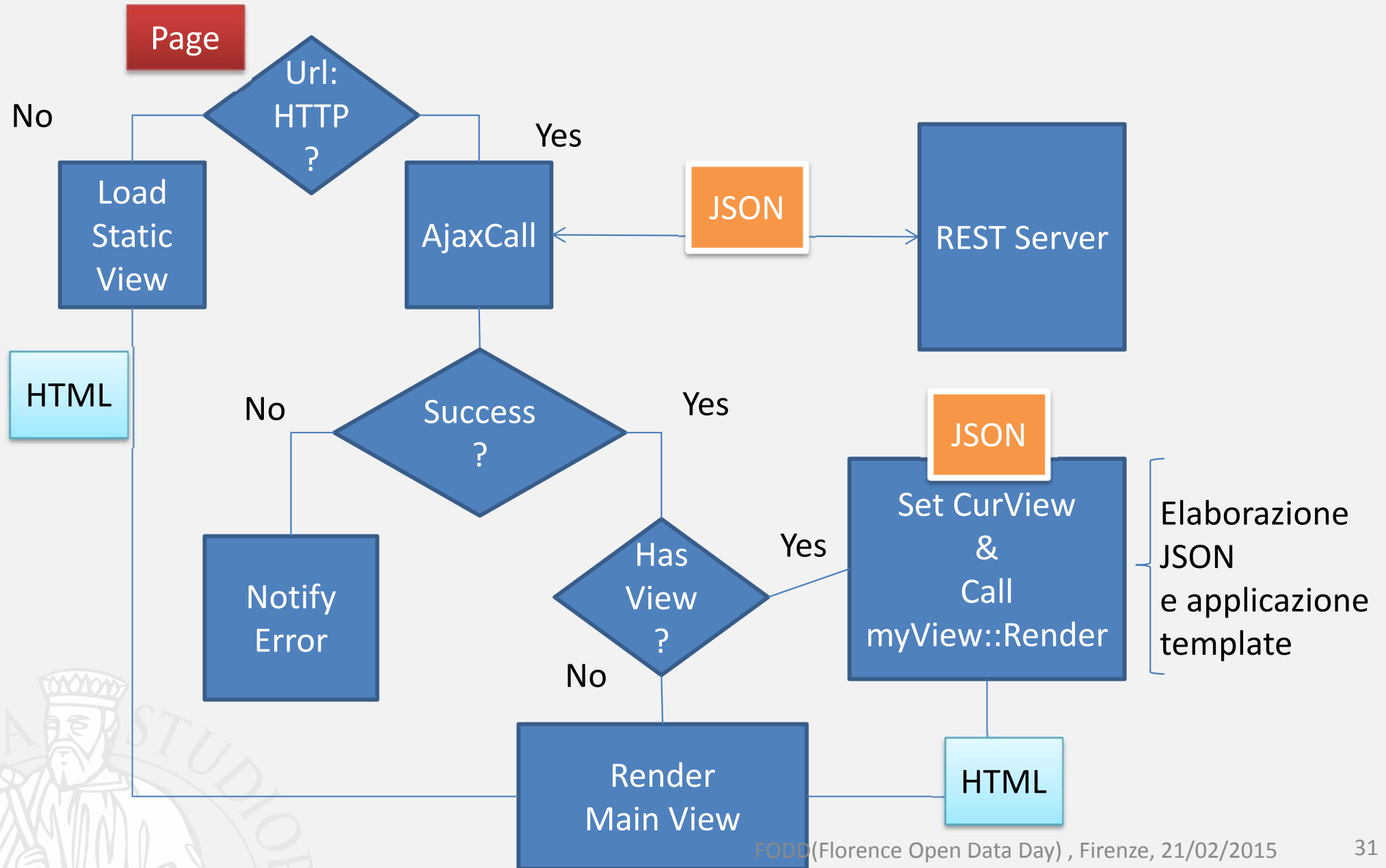




# View Manager

- Gestisce (handle) la richiesta di visualizzazione di una vista selezionata dal menu e la imposta come vista corrente
- Sulla base del tipo di vista esegue la generazione, caricamento e inserimento del codice html nell'area «CONTENT» della vista principale (main)
- La discriminante è il valore del parametro url:
  - Se contiene un url locale (no `http://....`) la vista è interna all'app
  - Se contiene un url esterna (sì `http://...`) la vista deve essere gestita dalla specifica vista che elabora i dati (JSON) ottenuti e li applica al template associato

# View Manager: gestione viste statiche e dinamiche



# Struttura JS di un gestore della vista

- Si tratta di una «classe» definita in JS. La sua definizione prevede:

```
function myView()  
{  
  this.render=function(page,data)  
  {  
    /* Codice per la gestione della vista */  
    /* Elaborazione dati JSON memorizzati in data */  
    var template = view.loadTemplate(page.template);  
    if(template!="")  
    {  
      /* Si ritorna l'html renderizzato dal template engine */  
      page.html = Mustache.render(template,data);  
    }  
  }  
}  
/* Definizione di metodi utili per elaborazione o altro */  
}
```

# Vista Servizi

- Richiesta di una collezione di 100 informazioni per Sensori di traffico, Fermate Autobus e Servizi nel Comune di Firenze
- Visualizzare la mappa centrata sul centro geografico richiesto (Comune Firenze)
- Visualizzare i segnalini (Pin) relativi ai Sensori di traffico, Fermate Autobus e Servizi
- Popup con informazione di dettaglio per ciascuno dei risultati forniti

# Vista Servizi (servizi.js)

- Per la realizzazione della Mappa
  - Open Street Map (leaflet.js)
- Per la gestione dei dati JSON e del disegno della mappa
  - **buildIcon**: selezione icona da associare al tipo di informazione (pin sulla mappa)
  - **createMap**: generazione Mappa, popolamento pin e definizione dei popup



# Servizi: Template & Mustache

```
<link rel="stylesheet" href="css/leaflet.css" />
<link rel="stylesheet" href="css/leaflet.awesome-markers.css" />
<script src="js/leaflet.js"></script>
<script src="js/leaflet.awesome-markers.min.js"></script>

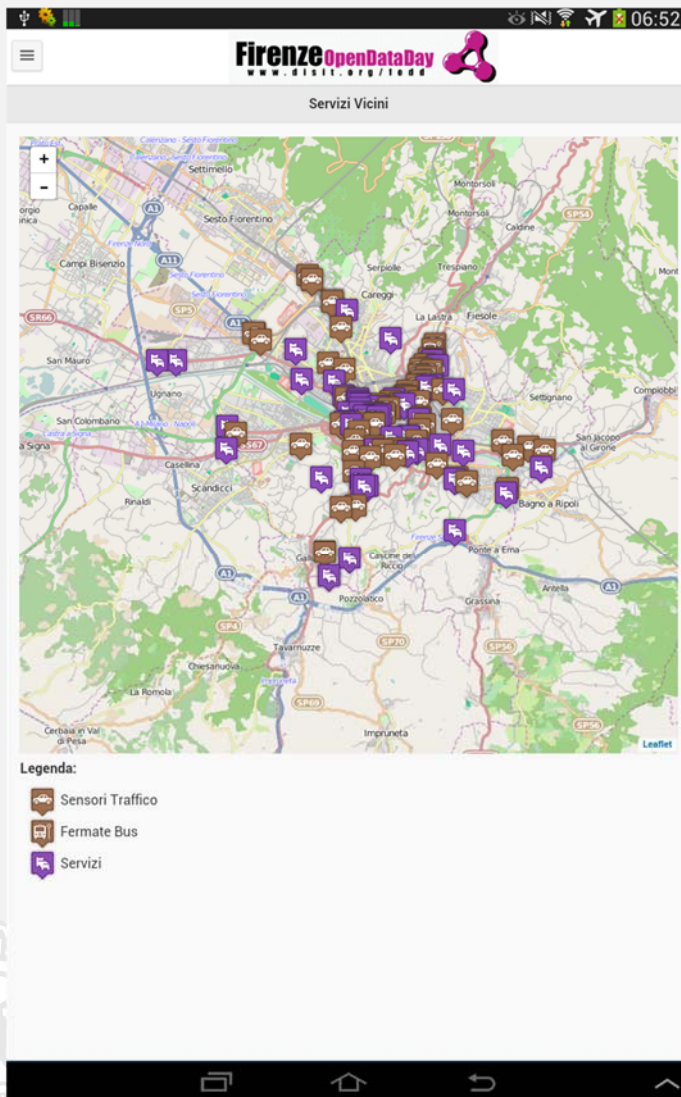
<div id="map">

</div>
<div id=map-legend>
  <b>Legenda:</b>
  <ul>
    {{#legend}}<li>{{tipo}}</li>
  </ul>
</div>

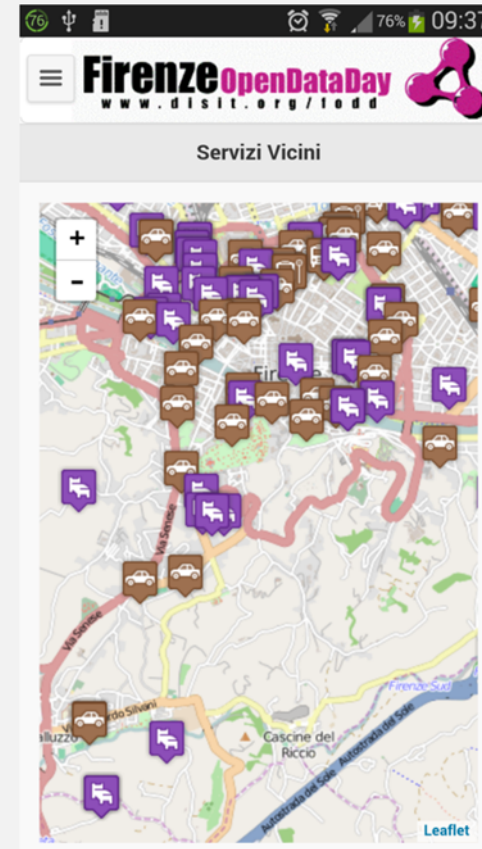
<script>
  $(document).ready(function(){
    if(view.curView.constructor.name=="servizi"){
      view.curView.createMap();
    }
  });
</script>
```



# Servizi Rendering Finale



Tablet (Samsung Galaxy Tab)



Smartphone (Samsung S3)



# Osservazioni

- Questo modello di architettura consente di svincolarsi dalla gestione della logica di paginazione
- Lascia al programmatore il compito di:
  - Definire una vista come voce del menu
  - Lavorare con un Template della vista
  - Sviluppare un gestore associato alla vista





# Osservazioni

- L'uso delle tecnologie individuate consente di realizzare i template e il rendering anche utilizzando un normale browser, magari nella configurazione mobile così da manipolare immediatamente CSS e HTML così come verrebbe visualizzato dal dispositivo.





UNIVERSITÀ  
DEGLI STUDI  
FIRENZE



**Firenze**OpenDataDay  
www.disit.org/fodd



DISIT Lab, Distributed Data Intelligence and Technologies  
Distributed Systems and Internet Technologies  
Department of Information Engineering (DINFO)  
<http://www.disit.dinfo.unifi.it>

# Domande ?





# AMBIENTE DI SVILUPPO: ECLIPSE + CORDOVA



# ECLIPSE

- L'ambiente di sviluppo consigliato è ECLIPSE ADT
- Il pacchetto preconfezionato (bundle) per lo sviluppo di applicazioni per Android
- Prerequisiti:
  - Android SDK



# Creare un progetto Cordova

- **Creare l'App in una cartella dei progetti**
  - cordova create FODD2015 org.disit.FODD2015 FODD2015
    - FODD2015 → nome della cartella
    - org.disit.fodd → Package Name
    - FODD2015 → Nome del progetto
- **Aggiungere Android Plattform, nella cartella generata**
  - cordova platform add android
- **Aggiungere i plugins da utilizzare**

# Plugin Cordova Richiesti

- Basic device information (Device API)
  - cordova plugin add org.apache.cordova.device
- Access files on device or network (File API):
  - cordova plugin add org.apache.cordova.file
  - cordova plugin add org.apache.cordova.file-transfer
- Notification via dialog box
  - cordova plugin add org.apache.cordova.dialogs
- Splashscreen
  - cordova plugin add org.apache.cordova.splashscreen
- Network Connection
  - cordova plugin add org.apache.cordova.network-information
- Open new browser windows
  - cordova plugin add org.apache.cordova.inappbrowser



# Cordova e Eclipse

- Al termine delle operazione descritte nella cartella è stato generato il file di progetto per ECLIPSE
- E' sufficiente importarlo in ECLIPSE oppure creando un progetto per ANDROID «from existing resource», ovvero nello wizard si localizza la cartella dell'app
- Per maggiori dettagli la guida di CORDOVA è esaustiva

# Build dell'APP

- L'applicazione deve essere prima generata attraverso il comando in linea nella console di WINDOWS
  - cordova build android
- Al termine del build da Eclipse è possibile lanciare l'app su un emulatore o direttamente sul dispositivo opportunamente impostato per il DEBUG (Impostazioni => Altro => Opzioni sviluppatore)



# FODD: PROJECT & CODE

