Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Analisi di tecnologie per lo sviluppo di applicazioni web rivolte a dispositivi mobili

Laureando:
Filippo Baccaglini

Relatore:
Ch.mo Prof. Mauro Migliardi

14 Ottobre 2014

Anno Accademico 2013/2014

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati codice

Capitolo 1

Introduzione

Lo scopo di questa tesi è di analizzare tecnologie per lo sviluppo di applicazioni Web fruibili efficacemente da dispositivi mobili, ed è frutto di uno stage svolto presso l'azienda Altevie Technologies s.r.l. [1], facente parte del gruppo Ethica Consulting s.r.l..
L'idea nasce dall'interesse sempre crescente verso *web apps* ottimizzate per la fruizione su dispositivi mobili. Inoltre, grazie a tecnologie quali Adobe PhoneGap e Apache Cordova, l'ottimizzazione viene spinta su nuovi livelli, garantendo la possibilità di pacchettizzare tali applicazioni come native o ibride, permettendo l'utilizzo dell'hardware del dispositivo mobile.
Si è deciso quindi di testare pregi e difetti di varie tecnologie per lo sviluppo e, restando sempre in un'ottica di vantaggio per l'azienda, si sono considerate tecnologie note e compatibili con l'universo SAP.
Anche SAP dimostra interesse a questa tendenza, per cui ha rilasciato un framework, *SAP® NetWeaver Gateway*, che consente l'interfacciamento tra web app e sistemi preesistenti, suggerendo l'utilizzo del protocollo *RESTful* **OData**.
In questa tesi, quindi, viene svolta un'analisi dello stato dell'arte, consultando sia informazioni disponibili sul web che provenienti da fonti aziendali, scegliendo poi di studiare e valutare **NepTune Application Designer**, **Sencha Architect** e **SAPUI5**.
Questa analisi è stata effettuata sviluppando la medesima applicazione, frutto di un caso reale, per l'approvazione al pagamento di fatture memorizzate nel sistema: in tal modo vengono affrontati gli stessi problemi con tutte le tecnologie e valutate le loro potenzialità su un benchmark reale e non frutto di una sintesi artificiosa.
Le metriche utilizzate, definite e imposte dall'azienda presso cui e' stato svolto lo stage, sono:

- **Design dell'interfaccia**: valutazione del grado di difficoltà nella creazione dell'interfaccia grafica, unitamente alla resa visiva finale.
- **Sviluppo dei controlli**: valutazione della semplicità di creazione dei controlli nell'applicazione, compresi i meccanismi di comunicazione

[1] `http://www.altevie.com/`

- **Accesso all'hardware**: valutazione della facilità di utilizzo di funzioni e metodi che permettono l'accesso all'hardware del dispositivo
- **Performance**: valutazione delle tempistiche di funzionamento dell'applicazione
- **Tempo di acquisizione padronanza**: valutazione del tempo necessario per ottenere una comprensione ed una padronanza della tecnologia tali da poter velocemente replicare o sviluppare da zero una nuova applicazione
- **Manutenibilità del codice**: valutazione della facilità nella manutenzione e modifica del codice
- **Team working (versioning)**: valutazione della predisposizione della tecnologia all'utilizzo di sistemi di *versioning*
- **Offline**: valutazione dell'implementazione di un sistema per l'archiviazione offline dei dati
- **Trasportabilità dell'applicazione**: valutazione della facilità nel trasportare l'applicazione da un dispositivo/sistema ad un altro

Nel capitolo 2 è svolta un'introduzione all'architettura REST, per comprenderne i fondamenti, mentre nel capitolo 3 viene descritto il protocollo OData secondo la versione in utilizzo, l'*Open Data Protocol Specification v2.0*.
Il capitolo 4 consiste in un'analisi dello stato dell'arte, focalizzata sulle tecnologie preferite dall'azienda.
Il capitolo 5 invece affronta lo sviluppo dell'applicazione, esaltando le peculiarità delle varie modalità di sviluppo.
Infine nel capitolo 6 si traggono le conclusioni, valutando le tecnologie con le metriche sopra descritte.

Capitolo 2

L'architettura REST

Representational state transfer (**REST**) è un'astrazione dell'architettura del World Wide Web consistente in un insieme coordinato di vincoli architetturali applicati a componenti, connettori e dati in un sistema ipermediale distribuito. REST si focalizza sul ruolo dei componenti, sui vincoli nella loro interazione e la loro interpretazione di dati significativi.
Il termine Representational State Transfer è stato introdotto nel 2000 da Roy Fielding nella sua tesi di dottorato presso la UC Irvine. REST è stato utilizzato per descrivere architetture Web, identificare problemi esistenti, comparare soluzioni alternative e verificare che estensioni ai protocolli non violino i vincoli che stanno alla base del successo del Web. Fielding inoltre ha utilizzato REST per progettare HTTP 1.1 e gli URI (Uniform Resource Identifiers). Lo stile architetturale REST è applicato allo sviluppo di Web Services come alternativa alle altre specifiche per la computazione distribuita, come ad esempio SOAP.

2.1 Vincoli architetturali

Le proprietà architetturali di REST sono realizzate applicando specifici vincoli nell'interazione tra **componenti** (unità astratte di istruzioni software e stati interni che consentono la trasformazione dei dati tramite le loro interfacce), **connettori** (meccanismi astratti che mediano comunicazioni, coordinamento e cooperazioni tra i componenti) e **dati** (elementi di informazione trasferiti tra componenti attraverso connettori). Tali vincoli sono:

2.1.1 Client-Server

Client e Server sono separati da un'interfaccia uniforme. Il concetto di *separation of concerns* (letteralmente, separazione delle preoccupazioni, degli interessi) implica che, ad esempio, i client non devono preoccuparsi di salvare dati, che restano all'interno del server, che a sua volta non deve preoccuparsi dell'interfaccia utente. In questo modo si possono avere server più semplici e scalabili, ma soprattutto client e server possono evolversi indipendentemente, a patto che l'interfaccia tra i due non venga alterata.

2.1.2 Stateless

Le comunicazioni client-server sono vincolate dal fatto che il server non deve salvare informazioni riguardanti il client: ogni richiesta inviata al server contiene tutte le informazioni necessarie per essere servita, mentre lo stato della sessione è mantenuto dal client.

2.1.3 Cacheable

Le risposte del server devono definirsi, implicitamente o esplicitamente, salvabili in cache o meno, per impedire che i client riusino dati ricevuti in precedenza nelle future richieste. Un buon utilizzo della cache può portare ad un calo delle interazioni client-server, migliorando così scalabilità e performance.

2.1.4 Layered system

Un client non può sapere se è connesso al server finale o ad un intermediario (che può applicare politiche di gestione di carico della rete, di sicurezza, ecc.).

2.1.5 Code on demand (opzionale)

Unico vincolo opzionale dell'architettura, indica la possibilità dei server di inviare ai client codice eseguibile, per estenderne o personalizzare le funzionalità. Esempi di codice eseguibile sono componenti compilati, come *Java Applet*, e script lato client come *JavaScript*.

2.1.6 Uniform Interface

Vincolo fondamentale nella progettazione di un servizio REST, semplifica e disaccoppia l'architettura permettendo l'evoluzione indipendente delle sue parti. I principi di questa interfaccia sono:

Identificazione delle risorse

Nelle richieste sono identificate risorse individuali, le quali sono concettualmente separate dalla rappresentazione con cui sono servite al client; ad esempio un server può rispondere alle richieste utilizzando HTTP, XML o JSON, anche se nessuna di queste è la rappresentazione del dato utilizzata internamente nel server.

Manipolazione di risorse attraverso le rappresentazioni

Quando un client possiede una rappresentazione della risorsa, compresi i metadati, ha informazioni sufficienti per modificarla o cancellarla.

Messaggi autodescrittivi

Ogni messaggio include informazioni su come utilizzarlo, come ad esempio il proprio Internet media type (conosciuto anche come MIME type). Le risposte possono indicare esplicitamente se sono salvabili in cache.

Ipermedia come motore dello stato dell'applicazione

I client eseguono transizioni di stato solamente attraverso azioni dinamicamente identificate dal server tra gli ipermedia (come ad esempio i collegamenti negli ipertesti). Eccetto alcuni entry point fissi dell'applicazione, un client non prevede alcuna particolare azione al di fuori di quelle descritte nella rappresentazione ricevuta dal server.

Il rispetto di questi vincoli concede all'applicazione la qualifica *RESTful* che garantisce proprietà quali performance, scalabilità, semplicità, modificabilità, visibilità, portabilità ed affidabilità.

2.2 Applicazione ai Web Service

Le Application Platform Interface (API) di un Web Service che rispettano i vincoli sopra indicati sono dette RESTful. Tali interfacce, se basate su HTTP, sono definite tramite:

- **URI base**, come `http://example.com/resources/`
- **Internet media type dei dati**
- **metodi HTTP standard**, come GET, PUT, POST e DELETE
- **collegamenti ipertestuali** per referenziare stati
- **collegamenti ipertestuali** per referenziare risorse collegate.

La seguente tabella mostra l'utilizzo tipico dei metodi HTTP nell'implementazione di API RESTful.

Risorsa	**GET**	**PUT**	**POST**	**DELETE**
URI che identifica una **collezione** es.: `http://example.com/resources`	Ottenere una **lista** degli URI e altre informazioni sui membri della collezione	**Rimpiazzare** la collezione con un'altra	**Creare** una nuova entry nella collezione	**Cancellare** l'intera collezione
URI che identifica un **elemento** es.: `http://example.com/resources/item1`	**Ottenere** una rappresentazione dell'elemento	**Rimpiazzare** l'elemento con un altro o, se non esiste, **crearlo**	Non utilizzato	**Cancellare** l'elemento

Tabella 2.1: Esempio di utilizzo di API RESTful in HTTP

Capitolo 3

Il protocollo OData

L'Open Data Protocol (OData) permette la creazione di *HTTP-based data services*, i quali permettono la pubblicazione e la modifica di risorse da parte di client Web tramite semplici messaggi HTTP.
Tali risorse vengono identificate attraverso Uniform Resource Identifiers (**URI**) e definite in un modello di dati astratto.
Lo scopo di OData è fornire l'accesso ad informazioni da una varietà di fonti, tra cui database relazionali, file system, content management system e siti Web tradizionali.
SAP utilizza lo standard specificato nell'*Open Data Protocol Specification v2.0* [1] che sarà di seguito descritto.

3.1 Relazioni con altri protocolli

OData è costruito sulle convenzioni definite nell'Atom Publishing Protocol (**AtomPub**[2]) ed aggiunge tecnologie Web quali **HTTP**[3] e JavaScript Object Notation (**JSON**[4]) per creare un protocollo che permetta di accedere ad informazioni da una varietà di applicazioni, servizi e store.

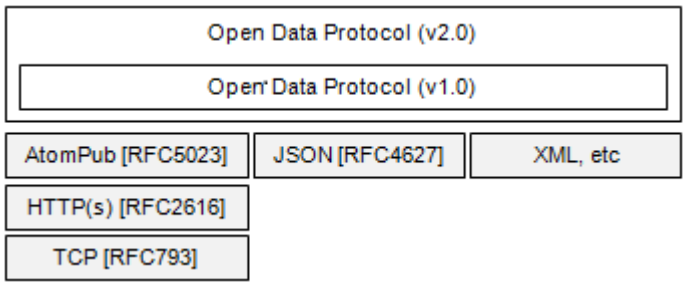

Figura 3.1: Schema relazioni del protocollo OData v2

[1] http://help.sap.com/fiori_bs2013/helpdata/en/ad/612bb3102e4f54a3019697fef65e5e/frameset.htm
[2] http://tools.ietf.org/html/rfc5023
[3] http://tools.ietf.org/html/rfc2616
[4] http://tools.ietf.org/html/rfc4627

3.2 Informazioni basilari su OData

Il cuore di OData sono i **feed**, che sono **Collections** di **Entries** tipizzate.
Ogni entry rappresenta un record strutturato con una chiave ed una lista di **Properties** di tipo primitivo o complesso (come ad esempio il tipo Indirizzo, che è costituito di tipi primitivi come via, città, ecc.).
Le entry possono far parte di una gerarchia di tipi e possono essere messi in relazione ad altre entry o ad altri feed attraverso i **Link**.

Esempio Il seguente URI rappresenta il feed di entry di tipo Product: `http://services.odata.org/OData/OData.svc/Products`.

Alcune entry sono "speciali", in quanto descrivono un elemento multimediale (solitamente un *BLOB*), diventando così due risorse collegate: la **Media Link Entry**, che contiene dati strutturati che descrivono il BLOB, e la **Media Resource** che è costituita dal BLOB stesso.
Semplici servizi OData possono essere costituiti da solamente un feed, mentre i più sofisticati possono avere più feed: in questi casi è utile esporre un documento di servizio che fornisca una lista di tutti i feed di livello più alto, in modo che i client possano scoprire la loro esistenza ed apprendere come raggiungerli.

Esempio `http://services.odata.org/OData/OData.svc` identifica il documento di servizio per un semplice servizio OData.

Oltre ai feed e le entry, i servizi OData possono esporre **Service Operations**, funzioni semplici e specifiche del servizio, che accettano parametri di input e ritornano entry o valori (complessi o primitivi).
Un servizio OData espone tutti questi costrutti (feed, entry, property nelle entry, link, documenti di servizio e documenti metadati) attraverso URI in uno o più formati di rappresentazione, sui quali i client possono agire (eseguendo query, update, insert, delete, ecc.) utilizzando richieste HTTP basilari.
Per aiutare i client a scoprire un servizio OData, la struttura delle sue risorse, i collegamenti tra esse e le Service Operations esposte, il servizio può esporre un **Service Metadata Document**.
Tale documento descrive l'Entity Data Model (**EDM**) di un dato servizio, ovvero il modello di dati astratto utilizzato per formalizzare la descrizione delle risorse che il servizio stesso espone.

Esempio L'URI `http://services.odata.org/OData/OData.svc/\$metadata` identifica il documento metadati per un esempio di servizio OData.

3.3 Metadati dei Servizi OData

Un servizio OData può esporre due tipi di documenti metadati per descriversi:

- un **Service Document** che fornisce una lista dei feed di alto livello, permettendo ai client di scoprirli e carpirne l'URI. Tale documento è disponibile alla **Service Root URI** e può essere formattato in Atom o in JSON.

- un **Service Metadata Document** che descrive il modello dei dati (ovvero la struttura e l'organizzazione di tutte le risorse) esposto mediante un endpoint HTTP dal servizio.

```
<service xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
    xml:base="http://services.odata.org/OData/OData.svc/">
  <workspace>
    <atom:title> Default </atom:title>
    <collection href="Products">
      <atom:title> Products </atom:title>
    </collection>
    <collection href="ProductDetails">
      <atom:title> ProductDetails </atom:title>
    </collection>
    <collection href="Categories">
      <atom:title> Categories </atom:title>
    </collection>
    <collection href="Suppliers">
      <atom:title> Suppliers </atom:title>
    </collection>
    <collection href="Persons">
      <atom:title> Persons </atom:title>
    </collection>
    <collection href="PersonDetails">
      <atom:title> PersonDetails </atom:title>
    </collection>
    <collection href="Advertisements">
      <atom:title> Advertisements </atom:title>
    </collection>
  </workspace>
</service>
```

Figura 3.2: Esempio di Service Document

3.4 Modello dei dati astratto

L'uso dell'**EDM** (Entity Data Model) come modello dati basilare per l'Open Data Protocol non impone l'utilizzo di una particolare implementazione o formato dati persistente da parte di un servizio OData. L'unico requisito è che l'interfaccia HTTP esposta sia consistente con il protocollo descritto nelle specifiche[5]. Un **Service Metadata Document** di un servizio OData descrive i propri dati in termini EDM usando un linguaggio XML per descrivere modelli, chiamato *Conceptual Schema Definition Language* (**CSDL**).

3.4.1 L'Entity Data Model (EDM)

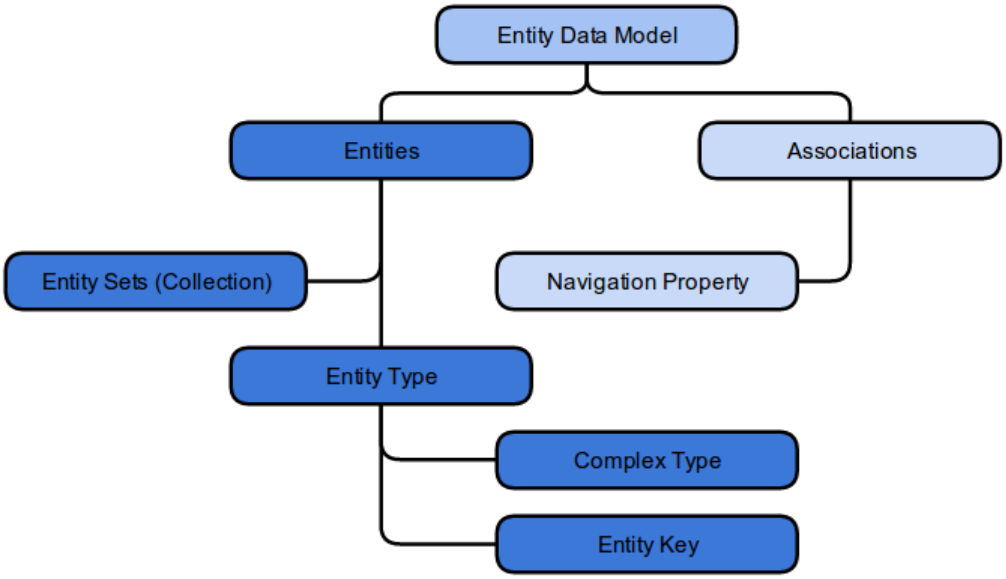

Figura 3.3: Schema sintetico dell'EDM

I concetti principali in un EDM sono **entità** ed **associazioni**.
Le **Entity** sono istanze di **Entity Type** (come ad esempio Cliente, Impiegato, ecc.) che sono record strutturati, consistenti in una chiave e proprietà aventi nome e tipo.
I **Complex Type** sono tipi strutturati consistenti anch'essi in una lista di proprietà ma **senza una chiave**, per cui possono solo esistere come proprietà di un'entità o come valore temporaneo.
Una **Entity Key** è costituita da un sottoinsieme di proprietà dell'Entity Type (ad esempio ID_Cliente o ID_Ordine) ed è un concetto fondamentale sia per identificare univocamente le istanze di Entity Type sia per permetterne la partecipazione alle relazioni.
Le Entity sono raggruppate in **Entity Set** (ad esempio, Clienti è un insieme di istanze di Entity Type Cliente).

[5]fare riferimento a `www.odata.org`

Le **Associations** definiscono la relazione tra due o più Entity Type (ad esempio, Impiegato *LavoraPer* Dipartimento); le istanze di associazioni sono raggruppate in **Associations Set**.
Le **Navigation Properties** sono proprietà speciali sugli Entity Type che sono legate a specifiche associazioni e possono essere usate per riferirsi alle associazioni di un'entità.
Infine, tutti i contenitori delle istanze (Entity Set ed Association Set) sono raggruppati in un **Entity Container**.
Riassumendo utilizzando i termini appena descritti, i feed esposti da un servizio OData sono rappresentati da **Entity Set** o da una Navigation Property su un Entity Type, che identifica una collezione di entità.

Esempio L'Entity Set identificato dall'URI `http://services.odata.org/OData/OData.svc/Products` o la collezione di entità identificate dalla navigation property "Products" in `http://services.odata.org/OData/OData.svc/Categories(1)/Products` identificano un feed di entry esposte dal servizio OData.

Ogni Entry di un feed OData è descritta nell'EDM da un **Entity Type** ed ogni collegamento tra le entità è descritto da una **Navigation Property**. Nella tabella è riportata una descrizione delle risorse OData.

Risorsa OData	**È descritta in un Entity Data Model da**
Collection	• Entity Set • Navigation property su un entity type che identifica una collezione di entità
Entry	• Entity Type, che possono essere parte di una gerarchia
Property di un'entry	• Property primitiva o complessa di un Entity Type
Complex Type	• Complex Type
Link	• Una Navigation Property definita su un Entity Type
Service Operation	• Function Import

Tabella 3.1: Descrizione delle risorse OData

3.4.2 Service metadata document

Un **Service Metadata Document** descrive il modello dei dati (cioè la struttura e l'organizzazione di tutte le risorse) ed è esposto come endpoint HTTP dal servizio utilizzando il Conceptual Schema Definition Language (CSDL).

Esempio Un Service Metadata Document che descrive tre Entity Type (Categories, Products e Suppliers), le relazioni tra di essi ed una Service Operation "ProductsByRating" è accessibile su `http://services.odata.org/OData/OData.svc/$metadata`

Figura 3.4: Documento ottenuto richiedendo il Service Metadata Document dell'esempio

```
<EntityContainer Name="DemoService" m:IsDefaultEntityContainer="true">
  <EntitySet Name="Products" EntityType="ODataDemo.Product" />
  <EntitySet Name="ProductDetails" EntityType="ODataDemo.ProductDetail" />
  <EntitySet Name="Categories" EntityType="ODataDemo.Category" />
  <EntitySet Name="Suppliers" EntityType="ODataDemo.Supplier" />
  <EntitySet Name="Persons" EntityType="ODataDemo.Person" />
  <EntitySet Name="PersonDetails" EntityType="ODataDemo.PersonDetail" />
  <EntitySet Name="Advertisements" EntityType="ODataDemo.Advertisement" />
```

Figura 3.5: Relazioni tra entity set ed entity type fornite dalla voce Entity Container

Le annotazioni CSDL, presenti come attributi nell'OData metadata namespace, sono usate per descrivere estensioni specifiche per OData, come riportato nella seguente tabella:

Annotazione	**Descrizione**
IsDefaultEntityContainer	Un documento CSDL può includere più Entity Container: questo attributo è utilizzato dai servizi per indicarne quello predefinito. Le entità nel container di default non necessitano che sia specificato nell'URI. Questo attributo può essere presente in qualunque elemento in un documento CSDL. Valori: *true* / *false*
DataServiceVersion	Indica la versione dell'annotazione OData usata nel Service Metadata Document. I consumatori di un endpoint metadata di un servizio dovrebbero leggere per primo questo valore per determinare se possono interpretare tranquillamente i costrutti nel documento. Questo attributo dovrebbe essere presente in tutti gli elementi.
HasStream	Questo attributo è utilizzato su un elemento per indicare che l'Entity Type descrive un Media Link Entry nel servizio OData associato. Valori: *true* / *false*
MimeType	Questo attributo è usato su un elemento per indicare il mime type del valore della property.
HttpMethod	Questo attributo è usato su un elemento che descrive una Service Operation esposta dal servizio OData. Il valore di questo attributo specifica il metodo HTTP da utilizzare per invocarla.

Tabella 3.2: Annotazioni CSDL

3.5 Tipi di dato primitivi

In OData i tipi di dato primitivo utilizzabili sono:

- **Null**, per l'assenza di valore
- **Binary**, per dati binari
- **Boolean**, per dati logici
- **Byte**, per valori interi senza segno a 8 bit
- **DateTime**, per data e ora a partire dalle 00:00 del 01/01/1753 fino alle 23:59:59 del 31/12/9999
- **Decimal**, per valori numerici con precisione e scala fissi
- **Double**, per valori a virgola mobile con precisione di 15 cifre
- **Single**, per valori a virgola mobile con precisione di 7 cifre
- **Guid**, per gli identificatori universali
- **Int16/32/64**, per interi con segno rispettivamente a 16, 32 o 64 byte
- **SByte**, per valori con interi con segno a 8bit
- **String**, per i dati di tipo stringa
- **Time**, per dati riguardanti ore del giorno
- **DateTimeOffset**, per date e ore rappresentati come differenza dal fuso GMT.

Per una spiegazione più approfondita, vedi Appendice A.

3.6 Convenzioni URI

Per garantire l'estensibilità e la compatibilità tra ecosistemi, OData definisce un insieme di regole raccomandate per la costruzione di URI, con lo scopo di identificare dati e metadati esposti da un server OData, e un set di *query string operator* riservati, che se accettati dal server devono essere implementati come indicato.

3.6.1 Componenti URI

Un URI utilizzato in un servizio OData ha fino a tre parti significative: **service root URI**, **resource path** e **query string options**.

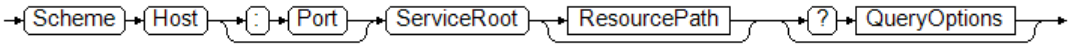

Figura 3.6: Schema di un URI OData

Esempio

`http://services.odata.org/OData/OData.svc`
Service root URI

`http://services.odata.org/OData/OData.svc/` Service root URI `Category(1)/Products` Resource path `?$top=2&$orderby=name` Query options

3.6.2 Service root URI

Il service root URI identifica la radice di un servizio OData. La risorsa identificata da questo URI deve essere un Service Document, per offrire ai client un semplice meccanismo per conoscere le collezioni di risorse disponibili nel servizio.

Esempio di URI richiesto	URI servizio OData
`http://services.odata.org:8080`	`http://services.odata.org:8080`
`http://services.odata.org/OData/OData.svc/Categories`	`http://services.odata.org/OData/OData.svc/`

Tabella 3.3: Esempi di service root URI

3.6.3 Resource path

La sezione resource path di un URI identifica la risorsa con la quale si desidera interagire, tra tutte quelle esposte. La costruzione dovrebbe avvenire come nell'immagine seguente:

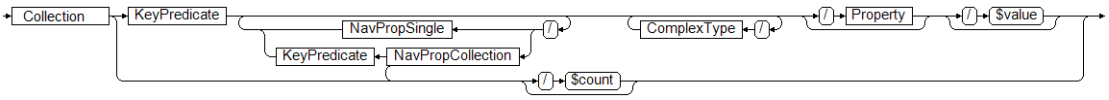

Figura 3.7: Schema di costruzione della *resource path*

- **Collection**: il nome della Collection o della Service Operation esposta dal servizio

- **KeyPredicate**: un predicato che identifichi il valore delle Key Properties di una Entry.
  Se la chiave è costituita di un unico valore si può indicare solo il valore della chiave, in caso contrario è necessario indicare la coppia nome/valore delle chiavi.

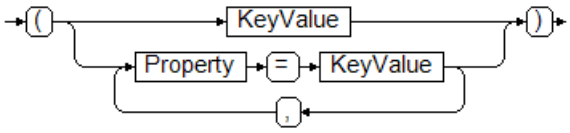

Figura 3.8: Schema di costruzione del *Key Predicate*

- **NavPropSingle**: il nome della Navigation Property definita nell'entry associata nel segmento precedente. Tale proprietà deve identificare una singola entità.

- **NavPropCollection**: come la precedente, con la differenza che deve essere identificata una collection di entry

- **ComplexType**: il nome di una Property dichiarata o dinamica della Entry o del Complex Type associata al segmento precedente.

- **Property**: il nome di una Property della Entry o del Complex Type associata al segmento precedente.

Esempio L'URI `http://services.odata.org/OData/OData.svc/Categories(1)/Products` identifica la collezione di prodotti associata all'entry Category con chiave 1. È descritta nel service metadata document dalla Navigation Property denominata Products nell'Entity Type Category.

L'URI `http://services.odata.org/OData/OData.svc/Categories(1)/Products(1)/Supplier/Address/City` identifica la città del fornitore del prodotto 1 associato alla categoria 1. È descritta nel service metadata document dalla Property denominata City nel Complex Type Address.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xml:base="http://services.odata.org/OData/OData.svc/"
   xmlns="http://www.w3.org/2005/Atom"
   xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
   xmlns:georss="http://www.georss.org/georss"
xmlns:gml="http://www.opengis.net/gml">
   <id>http://services.odata.org/OData/OData.svc/Categories</id>
   <title type="text">Categories</title>
   <updated>2014-08-12T15:44:58Z</updated>
   <link rel="self" title="Categories" href="Categories" />
   <entry>
      <id>http://services.odata.org/OData/OData.svc/Categories(0)</id>
      <category term="ODataDemo.Category"
         scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
      <link rel="edit" title="Category" href="Categories(0)" />
      <link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Products"
         type="application/atom+xml;type=feed" title="Products"
      href="Categories(0)/Products" />
      <title type="text">Food</title>
      <updated>2014-08-12T15:44:58Z</updated>
      <author>
         <name />
      </author>
      <link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/relatedlinks/Products"
         type="application/xml" title="Products" href="Categories(0)/$links/Products" />
      <content type="application/xml">
         <m:properties><d:ID m:type="Edm.Int32">0</d:ID><d:Name>Food</d:Name></m:properties>
      </content>
   </entry>
...
```

Figura 3.9: Esempio di documento che fornisce una lista delle categorie

Utilizzo dei link tra entry

Come per le pagine Web, il modello di dati utilizzato nei servizi OData supporta le relazioni come costrutto di prima classe. Ad esempio, un servizio OData può esporre una Collection di Entry Products ciascuno dei quali è relazionato ad una Entry Category.
Tali associazioni sono raggiungibili in OData proprio come le Entry stesse, seguendo le regole in figura.

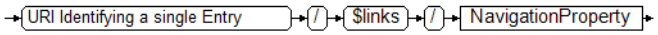

Figura 3.10: Schema di utilizzo delle relazioni

- **NavigationProperty**: Il nome di una Navigation Property dichiarata nell'Entry individuata dal segmento precedente alla porzione "*$links*".

Esempio L'URI `http://services.odata.org/OData/OData.svc/Categories(1)/$links/Products` Identifica l'insieme di prodotti in relazione alla Categoria 1, descritto dalla Navigation Property denominata "Products" dell'Entity Type Category nel service metadata document associato.

Utilizzo delle Service Operations

I servizi OData possono esporre delle Service Operation le quali, come le Entry, sono identificate utilizzando un URI. Le Service Operation sono semplici funzioni esposte da un servizio OData la cui semantica è definita dall'autore della funzione; esse possono accettare parametri di input di tipo primitivo e possono ritornare un singolo valore primitivo, un singolo tipo complesso, una collezione di primitivi, una collezione di tipi complessi, una singola entry, una collezione di entry o *void*. La regola base per costruire un URI che indirizzi ad una Service Operation e le passi dei paramentri è illustrata in figura:

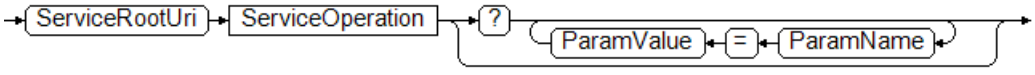

Figura 3.11: Schema di utilizzo di una *Service Operation*

- **ServiceRootUri**: Identifica la radice del servizio OData.
- **ServiceOperation**: Il nome della Service Operation esposta.
- **ParamName**: Il nome di un parametro accettato dalla funzione. Se la Service Operation accetta più parametri, l'ordine non è significativo.
- **ParamValue**: Il valore del parametro. Il formato è definito dalla colonna "literal form" della tabella dei tipi di dato primitivi.

3.6.4 Query string options

La sezione *query options* di un URI OData specifica tre tipi di informazione: **System Query Options**, **Custom Query Options** e **Service Operation Parameters**.

System Query Options

Le System Query Options sono stringhe di parametri di query che un client può specificare per controllare la quantità e l'ordine dei dati che il servizio ritorna per la risorsa identificata dall'URI. Tali stringhe sono precedute dal carattere di prefisso "**$**". Un servizio OData può supportare alcune o tutte le System Query Options definite. Le richieste che contengono opzioni non supportate devono essere rifiutate.

Query option	Descrizione	Esempio
$orderby	Questa opzione specifica quali valori vengono utilizzati per ordinare la collezione ottenuta, ed è valida solo nel caso in cui la resource path identifichi una Collection di Entry. Il servizio ordina di default per valori crescenti, nel caso si voglia un ordine decrescente è necessario specificare il suffisso *desc*.	`http://services.odata.org/OData/OData.svc/Products?$orderby=Rating,Category/Name desc`
$top	Questa opzione identifica un sottoinsieme delle entry nella collezione identificata dalla resource path, formato solo dai primi N oggetti nell'insieme, con N intero positivo specificato nella query option. Se non è specificata un'opzione *$orderby*, il servizio dovrà provvedere a fornire autonomamente un ordine, mantenendo sempre la stessa semantica per rendere ripetibili le operazioni.	`http://services.odata.org/OData/OData.svc/Products?$top=5`
$skip	Questa opzione identifica un sottoinsieme delle entry nella collezione identificata dalla resource path, formato solo dagli oggetti a partire dal $(N+1)$-esimo, con N intero positivo specificato nella query option. Se non è specificata un'opzione *$orderby*, il servizio dovrà provvedere a fornire autonomamente un ordine, mantenendo sempre la stessa semantica per rendere ripetibili le operazioni.	`http://services.odata.org/OData/OData.svc/Categories(1)/Products?$skip=2`

$filter	Questa opzione identifica un sottoinsieme delle entry nella collezione identificata dalla resource path, formato solo dagli oggetti che soddisfino il predicato indicato nella query option. Tale predicato può contenere riferimenti alle Property e a letterali, tra i quali vi sono stringhe (racchiuse tra apici), numeri, valori booleani e rappresentazioni indicate in Appendice A. Gli operatori utilizzabili verranno descritti nelle Tabelle B.1 e B.2 dell'Appendice B.	`http://services.odata.org/OData/OData.svc/Suppliers?$filter=Address/City eq Redmond`
$expand	Questa opzione permette di far restituire al servizio, oltre a quella selezionata, anche entry in relazione con essa. La sintassi è una lista di Navigation Properties separate da virgola	`http://services.odata.org/OData/OData.svc/Categories?$expand=Products`
$format	Questa opzione indica che la risposta deve essere fornita nel formato indicato Esempio valori: atom, xml, json	`http://services.odata.org/OData/OData.svc/Products?$format=atom`
$select	Questa opzione specifica che il servizio dovrà rispondere con il solo set di Property indicate nella query string. I valori vanno indicati come una lista separata da virgole di nomi di Property, nomi di Navigation Property o il simbolo *.	`http://services.odata.org/OData/OData.svc/Products?$select=Price,Name`
$inlinecount	Questa opzione indica che il servizio dovrà rispondere includendo il conteggio delle entry identificate, dopo l'eventuale applicazione di un filtro (se l'opzione $filter è utilizzata). Valori possibili: **allpages**(il servizio deve restituire il conteggio di tutte le entry selezionate), **none** (il servizio non include il conteggio, equivale a non specificare questa query option nell'URI)	`http://services.odata.org/OData/OData.svc/Products?$inlinecount=allpages`

Tabella 3.4: Tabella delle System Query Options

Custom Query Options

Le Custom Query Options offrono un estensione al servizio OData permettendo l'inserimento di informazioni specifiche nell'URI. Vengono definite come una o più coppie nome/valore dove il nome non inizia con il carattere speciale "$".

Esempio L'URI `http://services.odata.org/OData/OData.svc/Products?x=y` identifica tutte le entità Product e include una Custom Query Option "x" il cui significato è specifico del servizio.

Parametri delle Service Operations

Le Service Operation rappresentano funzioni esposte da un servizio OData e possono accettare zero o più parametri di tipo primitivo, che vengono passati come coppie nome/valore, ponendoli al termine dell'URI che identifica la Service Operation, preceduti dal carattere "**?**". I valori "**null**" vengono specificati omettendo il parametro nella query string.

Esempio L'URI `http://services.odata.org/OData/OData.svc/GetProductsByRating?rating=5` identifica la ServiceOperation "GetProductsByRating" e specifica che il parametro di input "rating" ha valore 5.

3.7 Considerazioni sulla sicurezza

L'Open Data Protocol è basato su HTTP, AtomPub e JSON, per cui è soggetto alle considerazioni sulla sicurezza applicabili a tali tecnologie.
Il protocollo **non definisce un nuovo schema per l'autenticazione o l'autorizzazione**, gli sviluppatori sono quindi incoraggiati ad utilizzare le tecnologie che meglio si adattano ai loro scopi. L'uso di meccanismi di autenticazione per prevenire l'aggiunta o la modifica di risorse esposte da un servizio OData da client sconosciuti o non autorizzati è raccomandato ma non richiesto.

Capitolo 4

Analisi delle tecnologie

Dato che si valuterà lo sviluppo di applicazioni i cui dati sono presi da sistemi SAP®, si è scelto di considerare tecnologie compatibili con questi sistemi che siano di interesse per l'azienda presso cui è stata svolta questa tesi, Altevie Technologies. La tecnologia che permette l'utilizzo delle soluzioni analizzate è *SAP® NetWeaver Gateway*, discussa di seguito, mentre per le soluzioni sono stati presi come termini di valutazione il tipo di **dispositivo** per cui sono state create, la possibilità di usufruire dell'applicazione in **offline**, i **pro** e i **contro** conosciuti ed il costo della **licenza**.

4.1 SAP® NetWeaver Gateway

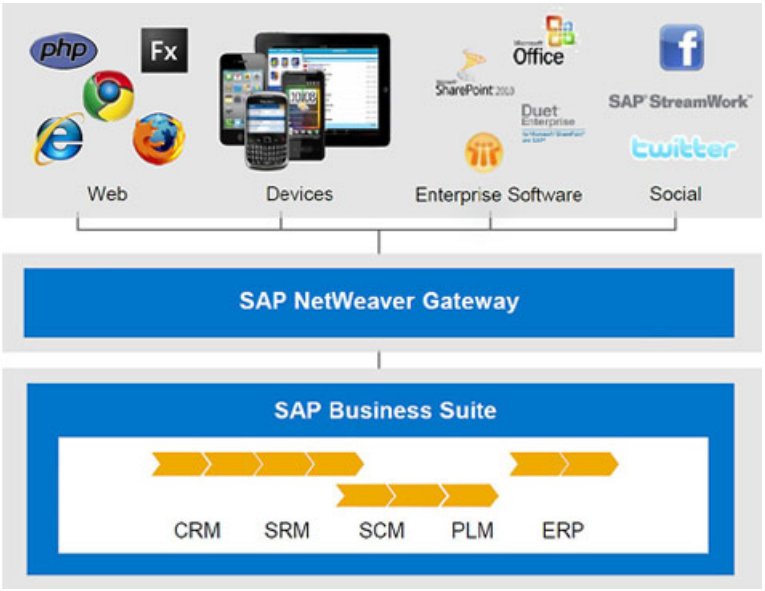

Figura 4.1: Posizionamento concettuale di SAP NetWeaver Gateway

Con l'avvento di SAP NetWeaver Gateway, nel 2011, si sono aperte moltissime possibilità per la connessione di dispositivi, ambienti e piattaforme a software

SAP. Si tratta di un framework che offre connettività ad applicazioni SAP utilizzando qualsiasi modello o linguaggio di programmazione senza rendere necessaria esperienza in SAP, sfruttando servizi REST ed i protocolli OData/ATOM.
Grazie alla potenza del neonato HTML5, è stato quindi possibile mettere a disposizione un interfaccia web che permettesse l'utilizzo delle applicazioni SAP direttamente dal web, rendendole di fatto accessibili da dispositivi di ogni tipo.
Data la preponderante tendenza ad abbandonare laptop e netbook in favore di smartphone e tablet, è sempre più sentita la necessità di migliorare l'esperienza su questi dispositivi, cercando di sfruttare al meglio anche l'hardware a disposizione.
Un importante passo avanti è stato fatto con il rilascio di Apache Cordova, che rende possibile la creazione di applicazioni "quasi-native" pur mantenendo un modello di programmazione web.
La tendenza di SAP a rilasciare le proprie API con licenza open source ha notevolmente agevolato la creazione di più soluzioni, sia da SAP stessa che da software house terze.

4.2 SAPUI5

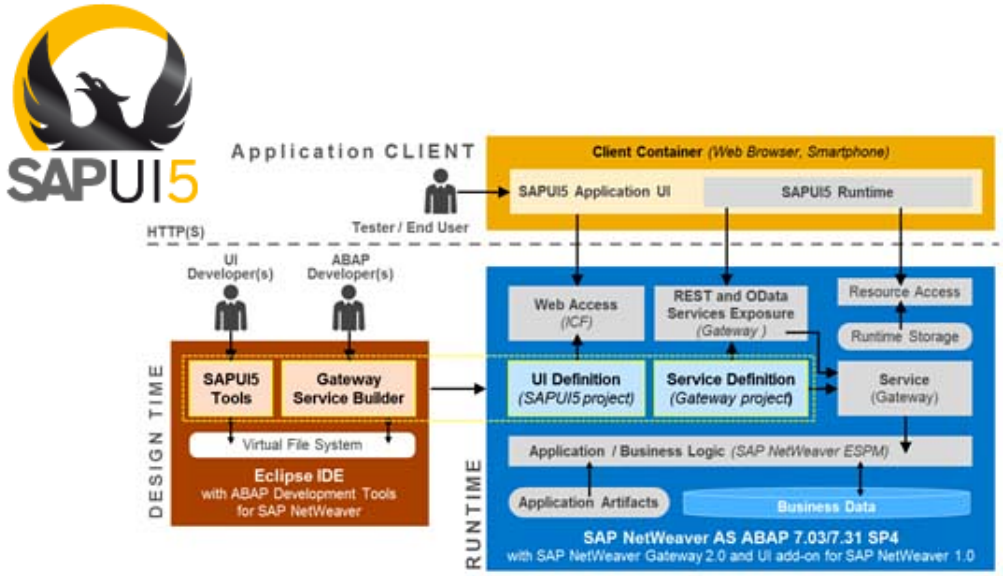

Figura 4.2: Schema concettuale utilizzo SAPUI5

SAPUI5 è una tecnologia rilasciata nel 2012 per la creazione di interfacce utente per applicazioni client utilizzando elementi di programmazione web (HTML, CSS3, JavaScript). La componente runtime di SAPUI5 è una libreria *client-side* per il rendering che offre un insieme di funzioni per lo sviluppo di applicazioni desktop e mobile. È disponibile un plugin per Eclipse per la programmazione in architettura MVC (Model View Controller), inoltre è possibile salvare ed avviare le proprie applicazioni in un *SAP NetWeaver Application Server* utilizzando il SAPUI5 ABAP Repository.

- **Dispositivo**: qualsiasi device che abbia un browser che supporti HTML5;
- **Offline**: supportato, sfruttando HTML5;
- **Pro**: sviluppo di applicazione web based con temi simili a quelli di altre applicazioni mobili SAP Fiori;
- **Contro**: non conformità con il tema del dispositivo mobile utilizzato per usufruire dell'applicazione;
- **Licenza**: SAP ha rilasciato le librerie con licenza Open Source (Apache2) con il nome di OpenUI5[1]

4.3 SMP 3.0

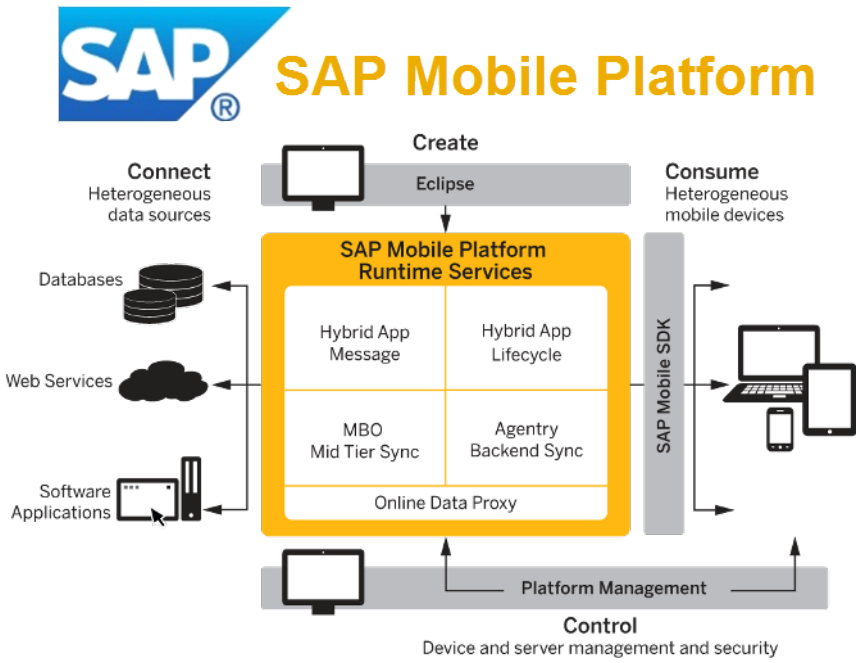

Figura 4.3: Schema concettuale SMP

SAP Mobile Platform 3.0, lanciata nel 2013[2], è una piattaforma per dispositivi mobili aperta che permette l'utilizzo offline delle applicazioni supportando OData, OSGi Spring, Apache Cordova, le API HTTP, REST e HTML5.
Tuttora non è pienamente supportata dai sistemi a disposizione e non sono disponibili grandi quantità di informazioni, se non quelle promozionali.

- **Dispositivo**: integrandosi con Apache Cordova, offre soluzioni per dispositivi *iOS*, *Android*, *BB10*, *BB7*, *W8*, *W8P*. Inoltre permette lo sviluppo di applicazioni native su *iOS* e *Android* con il rilascio di apposite librerie;

[1] `http://sap.github.io/openui5`

[2] `http://www.news-sap.com/new-version-of-sap-mobile-platform-to-provide-unmatched-productivity-flexibility-and-scalability/`

- **Offline**: supportato (probabilmente sfruttando HTML5);
- **Pro**: sviluppo di applicazioni native e cross-compilate;
- **Contro**: essendo disponibili solo informazioni promozionali, non si conoscono aspetti negativi nell'utilizzo di questa tecnologia;
- **Licenza**: è possibile effettuare un periodo di prova di 30 giorni attraverso la SAP Community Network ed è sisponibile gratuitamente un corso di sviluppo nella piattaforma openSAP.

4.4 Sencha Touch

Figura 4.4: Pubblicità di Sencha

Sencha è il produttore del framework JavaScript ExtJS e delle librerie Sencha Touch, molto simili a SAPUI5, che permettono la creazione di applicazioni web (ExtJS) anche per dispositivi mobili (Sencha Touch) sfruttando per le comunicazioni, tra gli altri, i protocolli RESTful in generale e oData in particolare. Vengono offerti controlli *GUI-based* ottimizzati per i dispositivi mobili. Per facilitare lo sviluppo, Sencha mette a disposizione un IDE molto strutturato, Sencha Architect, che, tra le altre funzionalità, permette di compilare la propria applicazione per renderla nativa sulle piattaforme Android e iOS.

- **Dispositivo**: qualsiasi device che abbia un browser che supporti HTML5;
- **Offline**: supportato, sia tramite le proprie librerie che con HTML5;
- **Pro**: sviluppo veloce di applicazioni web-based per dispositivi mobili dall'aspetto accattivante, con a disposizione librerie anche per l'interazione con l'hardware;

- **Contro**: non conformità con il tema del dispositivo mobile eventualmente utilizzato.

- **Licenza**: gratuita, con supporto a pagamento. L'utilizzo dell'IDE Sencha Architect è sottoposto a licenza di 399$. Le librerie Sencha Touch sono disponibili sotto licenza GPLv3.

4.5 Apache Cordova

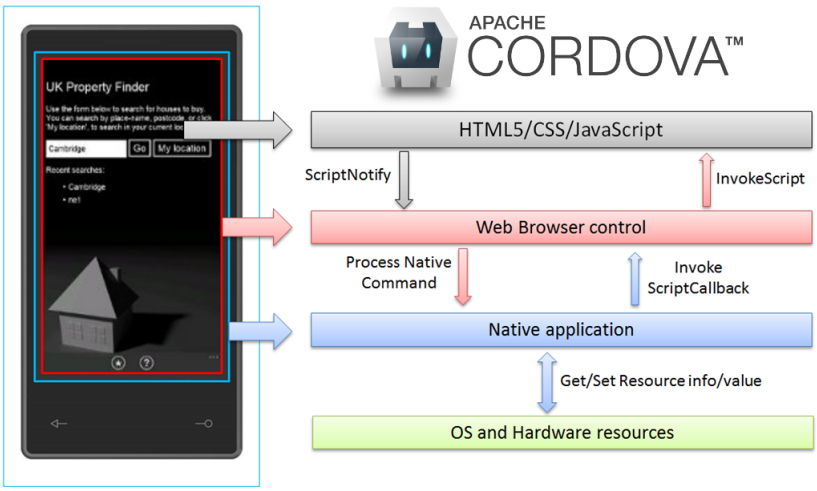

Figura 4.5: Schema concettuale del funzionamento di Apache Cordova

Derivata da Adobe PhoneGap e lanciata nel 2013, questa tecnologia, non legata come le altre allo sviluppo vero e proprio, permette la creazione di applicazioni native per una buona quantità di piattaforme conoscendo solo tecnologie per lo sviluppo web. È sufficiente compilare l'applicazione sviluppata con altre tecnologie (con accorgimenti particolari per permettere a Cordova di esprimere il proprio potenziale) per le piattaforme desiderate.
Nell'analisi ci si è focalizzati sullo sviluppo per piattaforma Android.

- **Dispositivo**: le piattaforme supportate sono iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, e Symbian;

- **Offline**: supportato, utilizzando alcune strutture native del dispositivo, come il DBMS SQLite;

- **Pro**: accesso ad hardware e funzionalità peculiari del dispositivo;

- **Contro**: ricompilazione dell'applicazione per ogni dispositivo, funzionalità supportate non omogeneamente tra i dispositivi;

- **Licenza**: Open Source Apache 2.

4.6 NepTune Application Designer

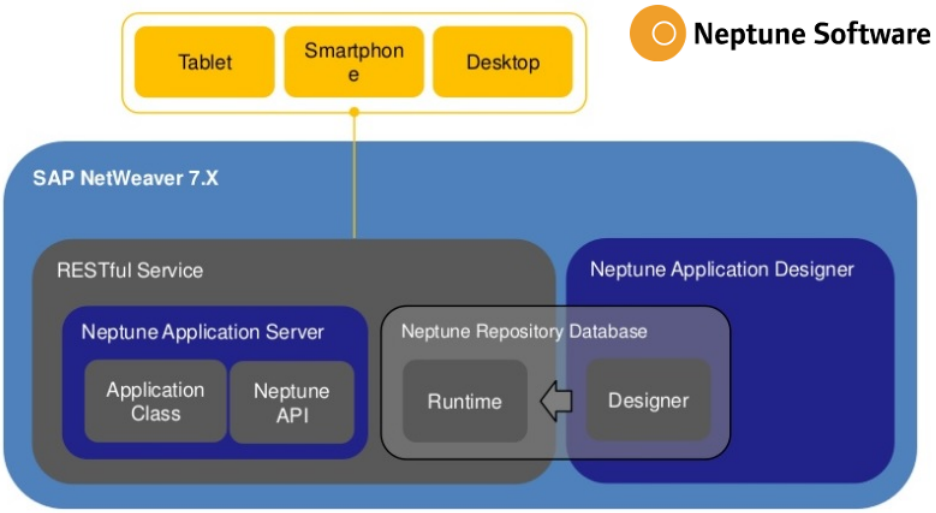

Figura 4.6: Schema concettuale della suite NepTune

Questa tecnologia è quella che meno si discosta dai sistemi SAP, perchè lo sviluppo avviene totalmente all'interno di essi: viene messo a disposizione un IDE grafico, le funzioni dell'applicazione sono create nel sistema stesso ed invocate tramite chiamate Ajax.
Dato che è necessaria una conoscenza limitatissima delle tecnologie di sviluppo Web, viene pubblicizzato come strumento per la creazione di applicazioni ideato per coloro che conoscono solo l'ABAP (il linguaggio di programmazione utilizzato in SAP).
Lo sviluppo avviene sfruttando (automaticamente) le librerie SAPUI5, mentre sono messe a disposizione le librerie JQuery, il servizio PhoneGap Build (compilazione online in applicazioni ibride) e la gestione della sicurezza con SAP Mobile Platform Cloud e SAP Afaria Cloud.

- **Dispositivo**: qualsiasi device che abbia un browser che supporti HTML5. Per lo sviluppo è necessario un sistema SAP in cui sia installato NepTune;
- **Offline**: supportato, sfruttando HTML5;
- **Pro**: profonda integrazione con ABAP
- **Contro**: rigidità nello sviluppo, possibile solo attraverso l'IDE.
- **Licenza**: 8.000€/50 licenze

4.7 SAP Fiori

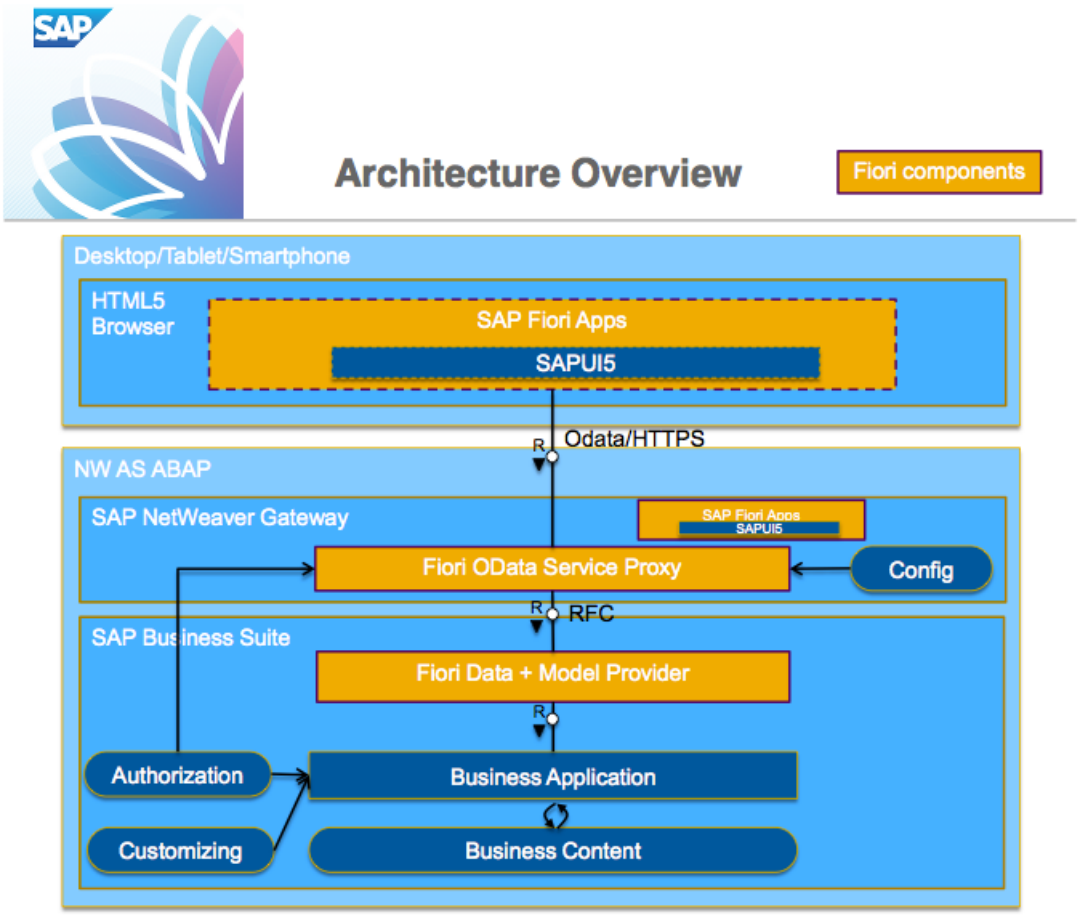

Figura 4.7: Schema concettuale dell'architettura di SAP Fiori

Raccolta di applicazioni già sviluppate con SAPUI5 che dovrebbero coprire una larga parte delle necessità dei clienti. È comunque possibile personalizzarle, essendo disponibili i sorgenti.

- **Dispositivo**: qualsiasi device che abbia un browser che supporti HTML5;
- **Offline**: supportato, sfruttando HTML5;
- **Pro**: applicazioni già sviluppate. Con l'avvento di SAP Fiori client (per Android e iOS) vengono offerte anche le potenzialità di Apache Cordova (con cui è costruita);
- **Contro**: difficoltà nell'estensione e personalizzazione dell'applicazione;
- **Licenza**: 150€/utente

Capitolo 5

Sviluppo dell'applicazione

SAP offre una soluzione strutturata per il ciclo di pagamento delle fatture, partendo dalla scansione della fattura cartacea alla contabilizzazione. L'approvazione al pagamento solitamente avviene in maniera non strutturata (benestare tramite mail, telefonata, ecc.): l'applicazione sviluppata si propone proprio di offrire una soluzione strutturata e disponibile sia da postazioni fisse che da dispositivi mobili.

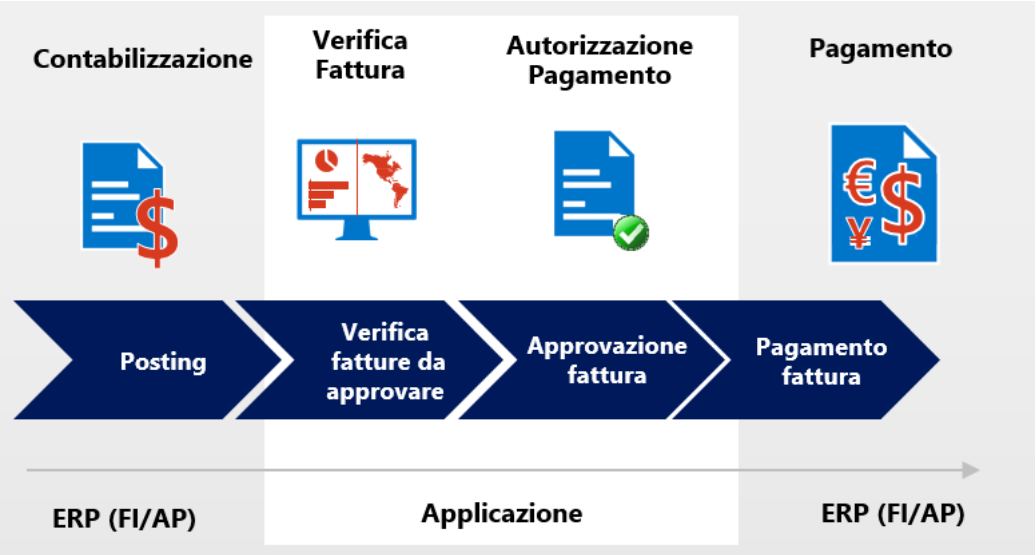

Figura 5.1: Collocazione dell'applicazione nel workflow di approvazione

Interfacciandosi con il sistema SAP ERP, l'applicazione consente di visualizzare e verificare le fatture pendenti per l'utente ed autorizzarne il pagamento o rifiutarlo, lasciando un commento. L'interfaccia è quindi strutturata seguendo il modello master/detail: una vista primaria con la lista delle fatture pendenti o rifiutate, una vista secondaria con il dettaglio della fattura ed i relativi controlli per l'approvazione/rifiuto.

Si è scelto di sviluppare l'applicazione utilizzando le tecnologie **NepTune**, **Sencha** e **SAPUI5**.

5.1 Implementazione dell'applicazione

In questa sezione si discuterà delle problematiche generiche da affrontare nello sviluppo dell'applicazione con le varie tecnologie.

5.1.1 Scopo dell'applicazione

Come già detto nel paragrafo precedente, lo scopo di questa applicazione è quello di consentire la visualizzazione degli estremi delle fatture da pagare ed eventualmente accettarne il pagamento (stato *accepted*) o rifiutarlo (stato *rejected*), obbligando all'inserimento della motivazione del rifiuto.

5.1.2 Pattern di programmazione utilizzati

Si è deciso di sviluppare l'applicazione adottando il pattern architetturale *MVC* (Model-View-Controller), le cui componenti sono:

- **Model**: componente centrale dell'architettura, gestisce dati, logica e regole dell'applicazione, indipendentemente dall'interfaccia utilizzata. Quando avviene un cambiamento di stato, lo notifica a *view* e *controller* associati, permettendo un cambiamento della visualizzazione (*view*) e dei comandi disponibili (*controller*). È possibile l'utilizzo di un comportamento "passivo" del modello, nel quale sono gli altri componenti ad interrogare il *model* per pervenire eventuali cambiamenti di stato.
- **View**: rappresenta e visualizza le informazioni; sono permesse più *view* per le stesse informazioni.
- **Controller**: accetta input e li converte in comandi a *view*, per modificare la visualizzazione, e *model*, per aggiornarne lo stato.

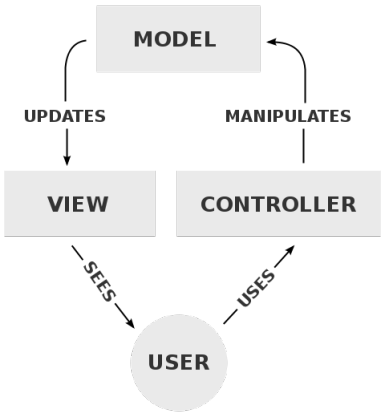

Figura 5.2: Interazione tra i componenti MVC

Data la natura dell'applicazione, si è deciso di strutturare l'interfaccia grafica secondo il modello *master-detail*, che consiste in una tabella *master*, contenente le informazioni essenziali per l'identificazione del record, da cui si può visualizzare una tabella *detail* dell'oggetto interessato.

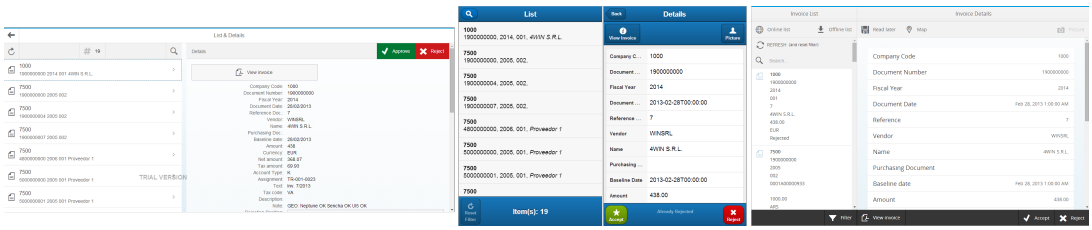

Figura 5.3: Esempi delle *GUI* in stile master-detail create

5.1.3 Dati, servizi e metodi disponibili sul sistema

Premessa

In tutti gli ambienti SAP utilizzati sono in uso le convenzioni sui nomi, per cui gli elementi il cui nome inizia con la lettera "Z" sono oggetti creati da utenti e non già disponibili nel sistema (o generati automaticamente). Verranno illustrate solo le componenti necessarie allo sviluppo, le altre verranno tralasciate perché, essendo un sistema condiviso, sono necessarie ad altri scopi.

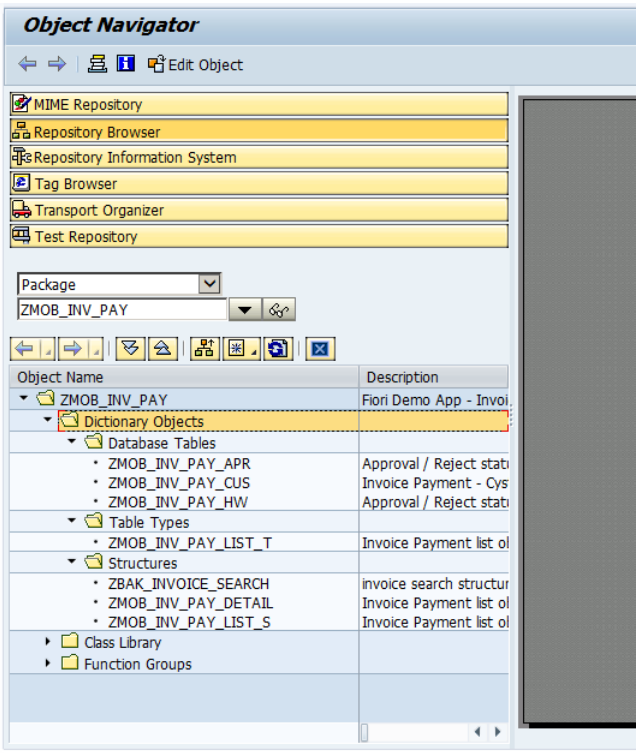

Figura 5.4: Visuale del package nella transazione SE80 con il nodo *Dictionary objects* completamente espanso

Tutto ciò che riguarda l'applicazione è stato incluso in un unico package, **ZMOB_INV_PAY**.

Per poterne consultare il contenuto si utilizza la transazione **SE80**, dal nome *Object navigator*: qui infatti è possibile accedere a tutti gli oggetti del package e verificarne forma e contenuti.
Come si può vedere in figura 5.4, il package nell'albero ha tre nodi figli:

- Dictionary objects
- Class Library
- Function groups

Dictionary Objects

Questo nodo è padre di tre altri nodi, ciascuno dei quali contiene elementi fondamentali per l'applicazione.

- **Database tables**: in questo nodo sono contenute delle tabelle ausiliarie in cui andranno inseriti dei dati direttamente dall'applicazione:
  - *ZMOB_INV_PAY_APR* è la tabella che contiene i valori chiave per individuare le fatture approvate o rifiutate: in questo ultimo caso viene salvata anche la nota con la motivazione del rifiuto
  - *ZMOB_INV_PAY_HW* è la tabella che contiene i valori chiave per individuare le fatture, unitamente a dati relativi ad informazioni aggiuntive quali coordinate per la geolocalizzazione ed un'immagine acquisita all'interno dell'applicazione
- **Table types**: in questo nodo sono contenute le tabelle che verranno popolate al momento della chiamata, per ciascuna viene definita la struttura da utilizzare come modello per le entry:
  - *ZMOB_INV_PAY_LIST_T* è la tabella che verrà utilizzata come "master", la cui struttura per le entry è *ZMOB_INV_PAY_LIST_S*
- **Structures**: in questo nodo sono contenute le strutture dati utilizzate:
  - *ZBAK_INVOICE_SEARCH*, struttura per la ricerca creata per l'applicazione NepTune
  - *ZMOB_INV_PAY_LIST_S*, struttura dati con tutti i campi da visualizzare in una entry della tabella "master"
  - *ZMOB_INV_PAY_DETAIL*, struttura dati con i campi da visualizzare nella tabella "detail" dell'elemento "master" selezionato

Class Library

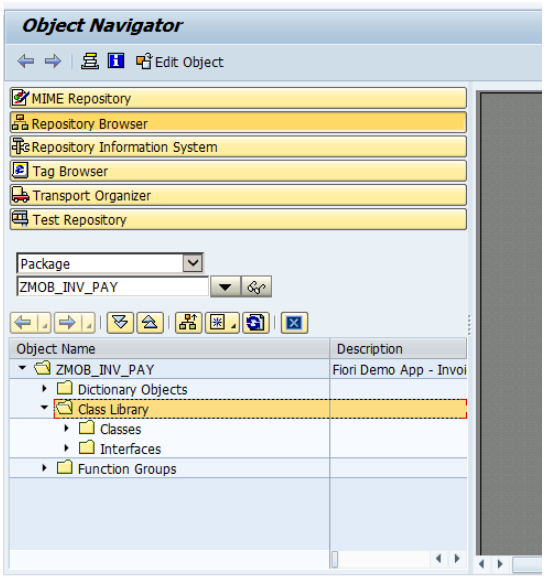

Figura 5.5: Visuale del package nella transazione SE80 con il nodo *Class Library* completamente espanso

In questo nodo sono contenuti classi ed interfacce, generalmente autogenerate. L'unico elemento di interesse è la classe *ZCL_BAK_INVOICELIST_CLASS*, creata per supportare l'applicazione NepTune, consiste sostanzialmente nel suo "controller"

Function Groups

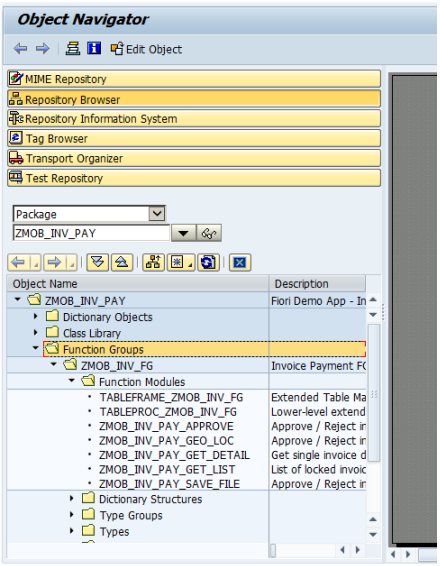

Figura 5.6: Visuale del package nella transazione SE80 con il nodo *Function Groups* completamente espanso

In questo nodo sono contenuti funzioni, tipi di dato, dizionari, subroutine, moduli e inclusioni.
In questo caso l'unico sottonodo è *ZMOB_INV_FG*, padre dei nodi sopraelencati. L'unico nodo figlio di interesse è *Function Modules*, contenente le funzioni che potranno essere chiamate dall'applicazione:

- *ZMOB_INV_PAY_APPROVE*, funzione che permette l'approvazione/rifiuto della fattura, con l'invio della nota nell'ultimo caso. Andrà ad inserire dati nella *database table ZMOB_INV_PAY_APR*

- *ZMOB_INV_PAY_GEO_LOC*, funzione che permette l'inserimento delle coordinate per la geolocalizzazione. Andrà ad inserire dati nella *database table ZMOB_INV_PAY_HW*

- *ZMOB_INV_PAY_GET_DETAIL*, funzione che permette di ottenere i dati per la tabella "detail" in visualizzazione. Nello specifico restituisce la *structure ZMOB_INV_PAY_DETAIL*

- *ZMOB_INV_PAY_GET_LIST*, funzione che permette di ottenere i dati per la tabella "master" in visualizzazione. nello specifico restituisce il *table type ZMOB_INV_PAY_LIST_T*

- *ZMOB_INV_PAY_SAVE_FILE*, funzione che permette l'inserimento di un'immagine. Andrà ad inserire dati nella *database table ZMOB_INV_PAY_HW*

5.1.4 Servizio OData esposto

Nel caso studiato, i dati sono presenti in un sistema non direttamente accessibile dall'esterno dell'azienda. È stato quindi predisposto un sistema apposito per l'accesso via web, il quale espone il servizio OData fornendo dati presi dal sistema interno.
Per visualizzare il servizio OData esposto, è necessario accedere alla transazione **/IWFND/MAINT_SERVICE**.

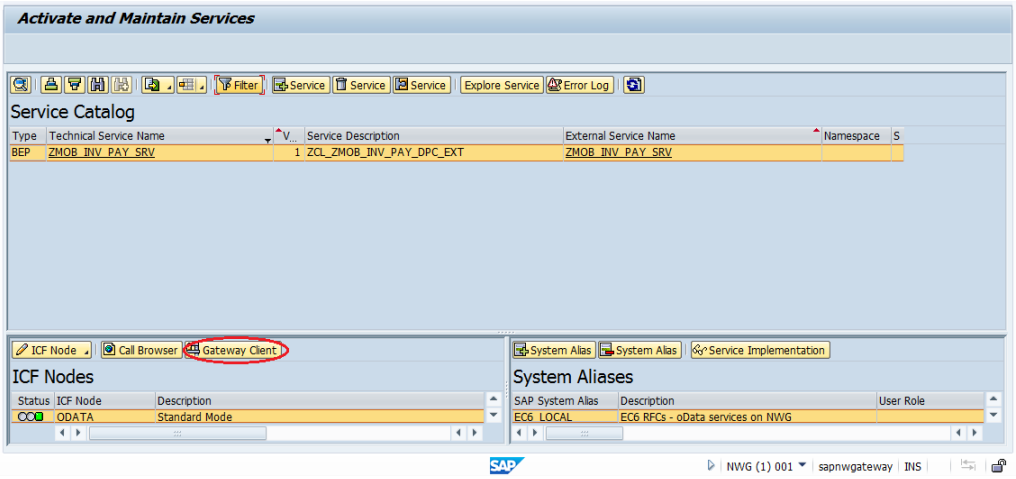

Figura 5.7: Visualizzazione del servizio nella transazione

Dalla figura 5.7 è possibile evincere il nome del servizio nella colonna *External Service Name*, mentre cliccando sul tasto *Gateway Client* (nel cerchio rosso), è possibile sia carpire la parte finale dell'URI del servizio (che andrà messo a suffisso dell'URI del sistema), che utilizzare il client per interrogare il servizio, ad esempio richiedendo il **Service metadata document** (disponibile all'appendice C).

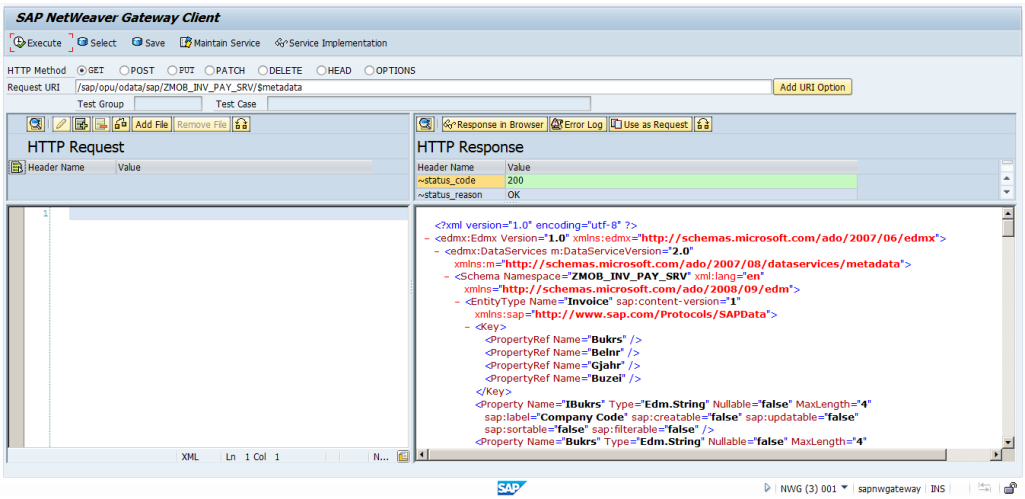

Figura 5.8: Gateway Client del servizio

Per consultare in maniera più efficace l'offerta del servizio, è necessario accedere alla transazione **SEGW**.

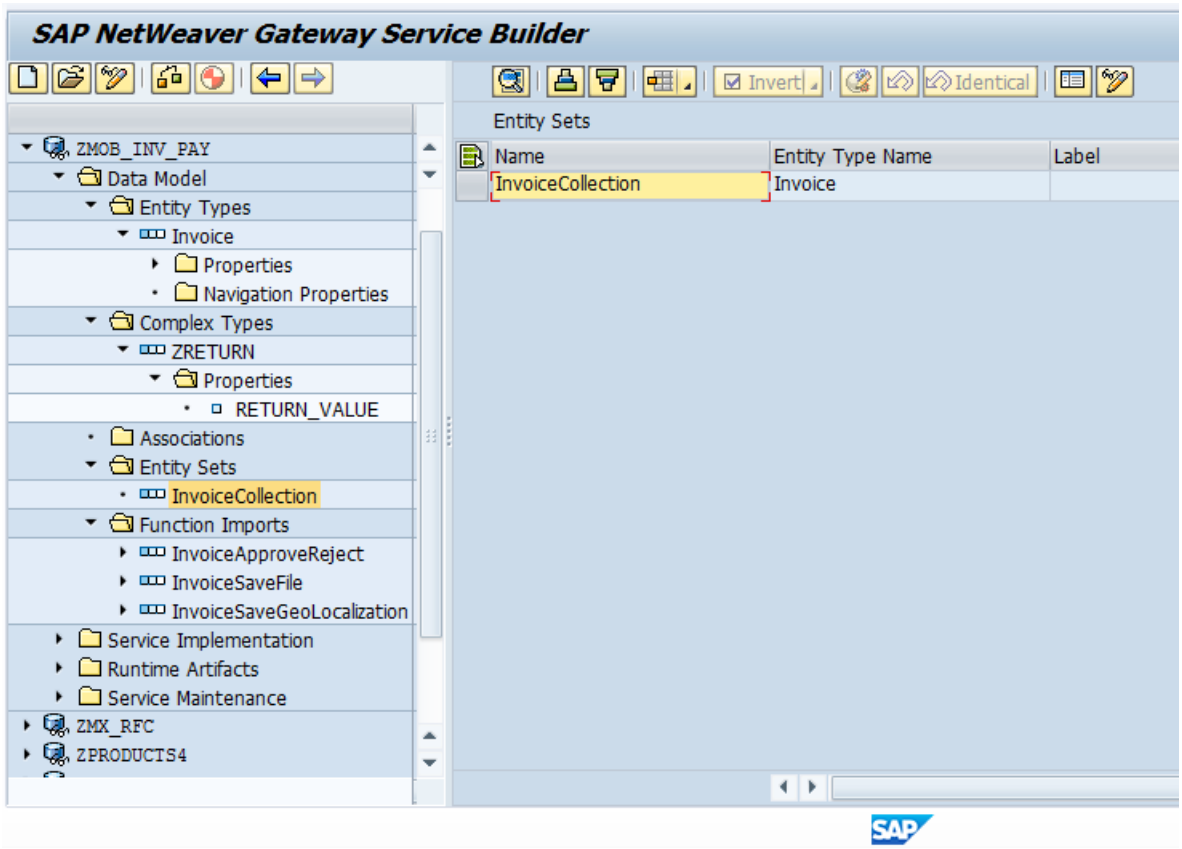

Figura 5.9: Visualizzazione della transazione *SEGW*

Come si può notare in figura 5.9, il servizio contiene quattro sottonodi:

- **Data Model**, contenente le informazioni sul servizio, molto vicine a quelle messe a disposizione nel *Service Metadata Document*.
- **Service Implementation**, contenente i collegamenti ai metodi autogenerati per effettuare le operazioni CRUD.
- **Runtime Artifacts**, contenente componenti autogenerate per l'esecuzione nel sistema del servizio.
- **Service Maintenance**, contenente collegamenti a metodi per la manutenzione del servizio.

Il nodo di maggior interesse per comprendere il funzionamento del servizio è il primo, **Data Model**, il quale comprende i nodi:

- **Entity types**, come indica il nome, comprende gli entity type raggiungibili dal servizio. In questo caso l'unico entity type esposto è *Invoice*, le cui *Properties* sono i campi che lo compongono, mentre non è definita alcuna *Navigation Property*.
- **Complex Types**, come indica il nome, comprende i complex type raggiungibili dal servizio. In questo caso l'unico complex type è **ZRETURN**, dato fittizio inviato in risposta alle operazioni di update delle tabelle ausiliarie in cui l'applicazione può scrivere.
- **Associations**, come indica il nome, comprende le associations raggiungibili dal servizio. In questo caso non ce ne sono.
- **Entity Sets**, come indica il nome, comprende gli entity set raggiungibili dal servizio. In questo caso l'unico entity set esposto è *InvoiceCollection*, raccolta di istanze di tipo *Invoice*.
- **Function Imports**, come indica il nome, comprende i function import raggiungibili dal servizio, equivalenti a service operation. In questo caso sono:
  - *InvoiceApproveReject*, per l'approvazione/rifiuto della fattura;
  - *InvoiceSaveFile*, per il salvataggio di un'immagine relativa alla fattura;
  - *InvoiceSaveGeoLocalization*, per il salvataggio di coordinate per la geolocalizzazione.

5.1.5 Sicurezza

In tutte le applicazioni si è scelto di utilizzare l'autenticazione prevista nei sistemi SAP al momento della richiesta dei dati. Non è stata perciò implementata alcuna misura aggiuntiva.

5.2 Sviluppo dell'applicazione con NepTune Application Designer

Come accennato nel capitolo precedente, questa tecnologia è quella che meno si discosta dalla normale programmazione ABAP in SAP.
La macchina contenente l'ambiente di sviluppo NepTune è la stessa contenente i dati, non accessibile direttamente all'esterno della rete aziendale.
Per accedere a NepTune Application Designer è necessario entrare nella transazione **/N/NEPTUNE/DESIGNER** e da qui accedere all'applicazione, da me denominata *Z_BAK_INVOICELIST*.

5.2.1 Interfaccia grafica

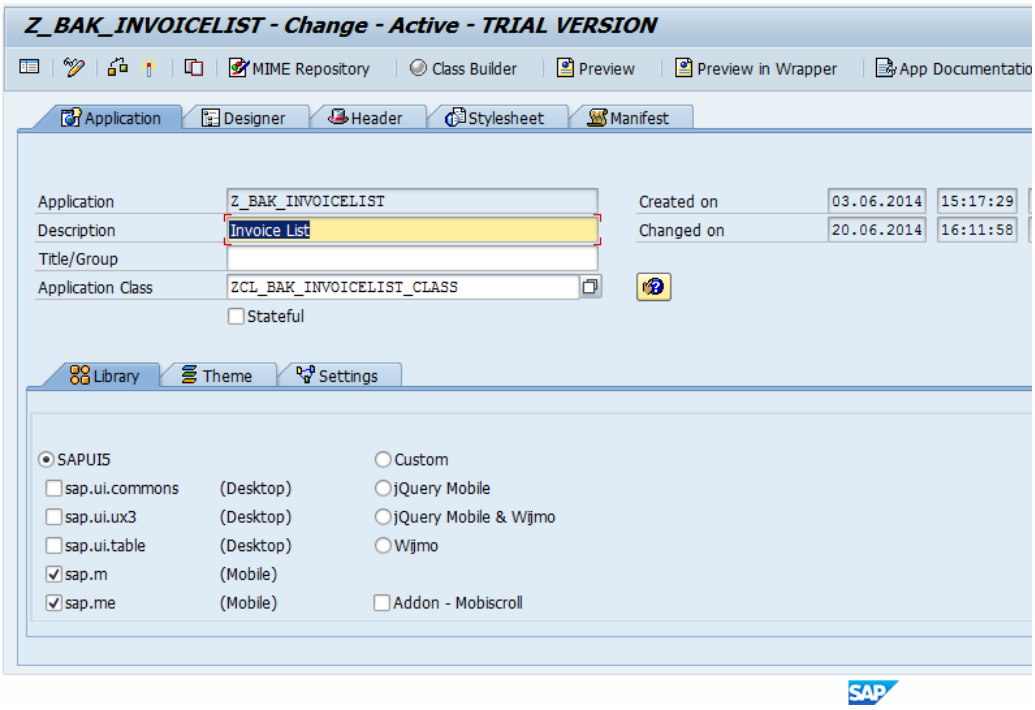

Figura 5.10: Schermata di impostazioni dell'applicazione in NepTune

Si aprirà l'applicazione nella sezione **Application** (figura 5.10), in cui è necessario dare le impostazioni di base.
Nella parte superiore viene visualizzato il nome dell'applicazione, inserito nella fase precedente, ed è possibile inserire una descrizione dell'applicazione, un titolo, e l'**Application Class**, che è una classe di supporto all'applicazione stessa.
Nel tab **Library** vanno indicate le librerie che verranno utilizzate: in questo caso si tratta di un'applicazione **SAPUI5** e le librerie di cui necessiteremo sono quelle per dispositivi mobili (contrassegnate dalla dicitura *Mobile*), **sap.m** e **sap.me**.
Le impostazioni negli altri due tab, *Theme* e *Settings*, sono già corrette con i

valori di default: tema *Bluecrystal* di SAPUI5, Application Type: *Application* e Content Type: *HTML5*.
Accedendo poi alla sezione **Designer**, si raggiungono tutti gli strumenti per lo sviluppo vero e proprio.

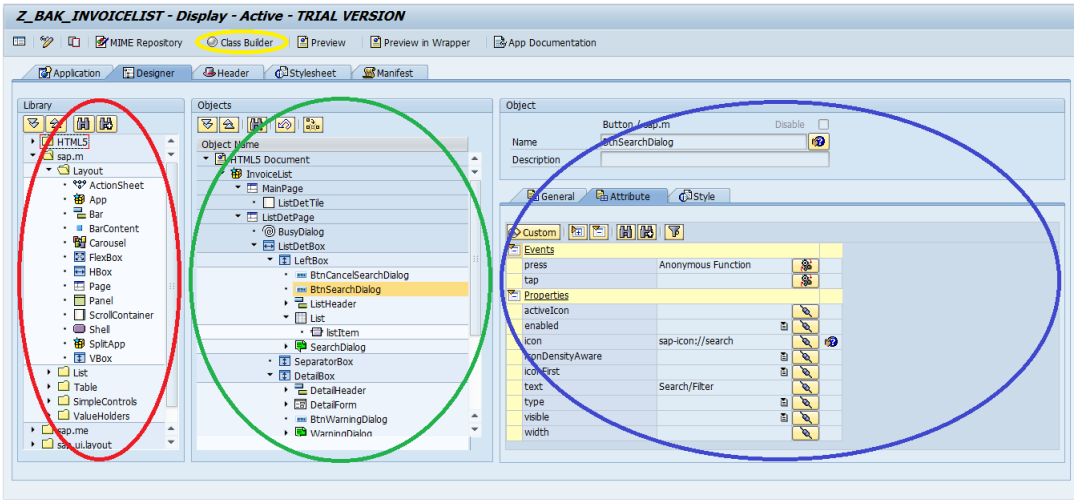

Figura 5.11: Schermata di sviluppo nella sezione *Designer*

Troviamo qui tre aree principali (fare riferimento alla figura 5.11):

- **Library**, contrassegnata in rosso, contiene in una struttura ad albero tutte le componenti supportate dall'IDE; si può notare come siano incluse le librerie che erano state selezionate nella fase precedente.

- **Objects**, contrassegnata in verde, contiene in una struttura ad albero tutti gli elementi utilizzati nell'applicazione. L'aggiunta di un nuovo elemento avviene per trascinamento di un componente dall'area precedente.
  È importante notare che l'elemento radice è **HTML5**, il cui unico figlio è l'applicazione sviluppata.

- **Object**, contrassegnata in blu, è l'area dedicata alle impostazioni peculiari dell'oggetto.
  Oltre al nome, che dev'essere unico per ogni elemento, si hanno tre sezioni:

  – *General*, per il collegamento di un oggetto ad un modello (struttura dati) e ad una funzione (Ajax ID)
  – *Attribute*, per le impostazioni relative ad eventi JavaScript sull'elemento e attributi di stile
  – *Style*, per l'aggiunta di personalizzazioni sullo stile dell'elemento

Infine, un'ultima porzione importante dell'interfaccia di sviluppo è il pulsante **Class Builder**, contrassegnato in giallo, che consente di accedere alla classe di supporto all'applicazione (l'*Application Class* indicata nella fase precedente), contenente tutte le strutture dati e le funzioni per l'interfacciamento dell'applicazione con il sistema.

5.2.2 Application Class

La creazione della classe avviene accedendo alla transazione **SE24**, denominata *Class Builder*.
Una volta effettuato l'accesso alla classe, è necessario impostare nella sezione **Interfaces** l'interfaccia che verrà adottata per accedere a tutte le potenzialità messe a disposizione da NepTune, il cui nome è **/NEPTUNE/IF_NAD_SERVER**: in questo modo verranno ereditati i metodi già implementati per le comunicazioni e sarà necessario solo personalizzare le parti di interesse.

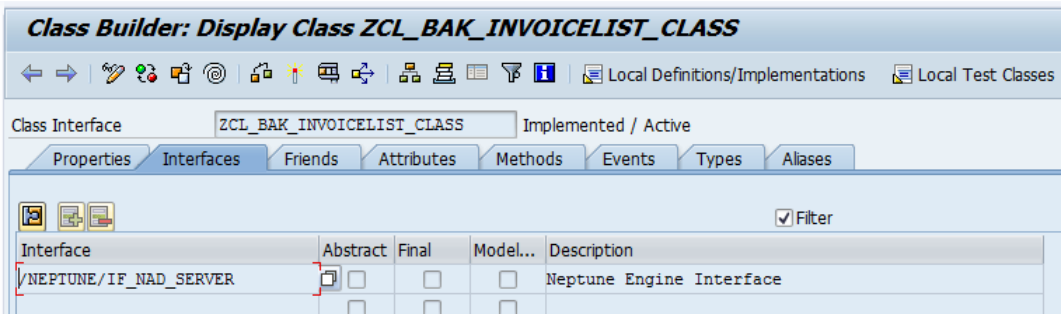

Figura 5.12: Interfacce della classe di supporto

Successivamente si accede alla sezione **Attributes**: qui si impostano le strutture dati che verranno utilizzate.

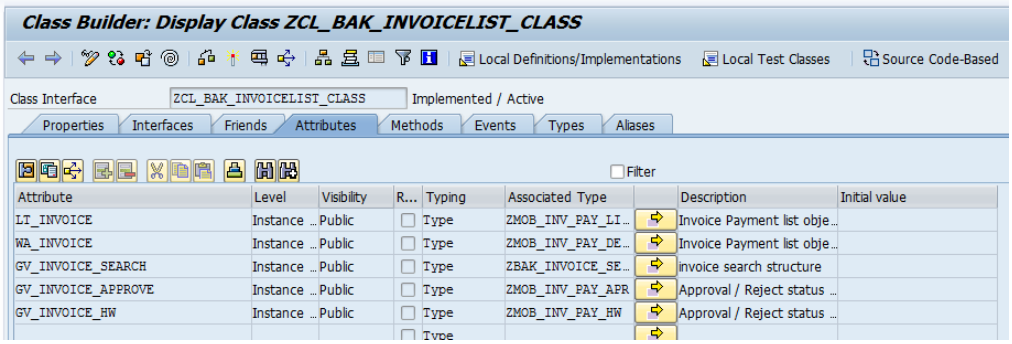

Figura 5.13: Attributi della classe di supporto

Con riferimento alla figura 5.13, gli attributi sono:

- **LT_INVOICE**, di tipo *ZMOB_INV_PAY_LIST_T*: nominato secondo la convenzione (LT sta per **Local Type**), è un riferimento alla tabella che verrà utilizzata come "master" in visualizzazione
- **WA_INVOICE**, di tipo *ZMOB_INV_PAY_DETAIL*: nominato secondo la convenzione (WA sta per **Work Area**), è un riferimento alla struttura dati del dettaglio dell'elemento selezionato nella tabella master
- **GV_INVOICE_SEARCH**, di tipo *ZBAK_INVOICE_SEARCH*: nominato secondo la convenzione (GV sta per **Global Variable**), è un riferimento alla struttura dati creata per la ricerca

- **GV_INVOICE_APPROVE**, di tipo *ZMOB_INV_PAY_APR*: nominato secondo la convenzione, è un riferimento alla *database table* per il tracciamento delle fatture approvate e rifiutate
- **GV_INVOICE_HW**, di tipo *ZMOB_INV_PAY_HW*: nominato secondo la convenzione, è un riferimento alla *database table* per il salvataggio dei dati acquisiti dal dispositivo (posizione ed immagine)

L'ultima parte da modificare, contenente codice ABAP, è la sezione relativa ai metodi, **Methods**.

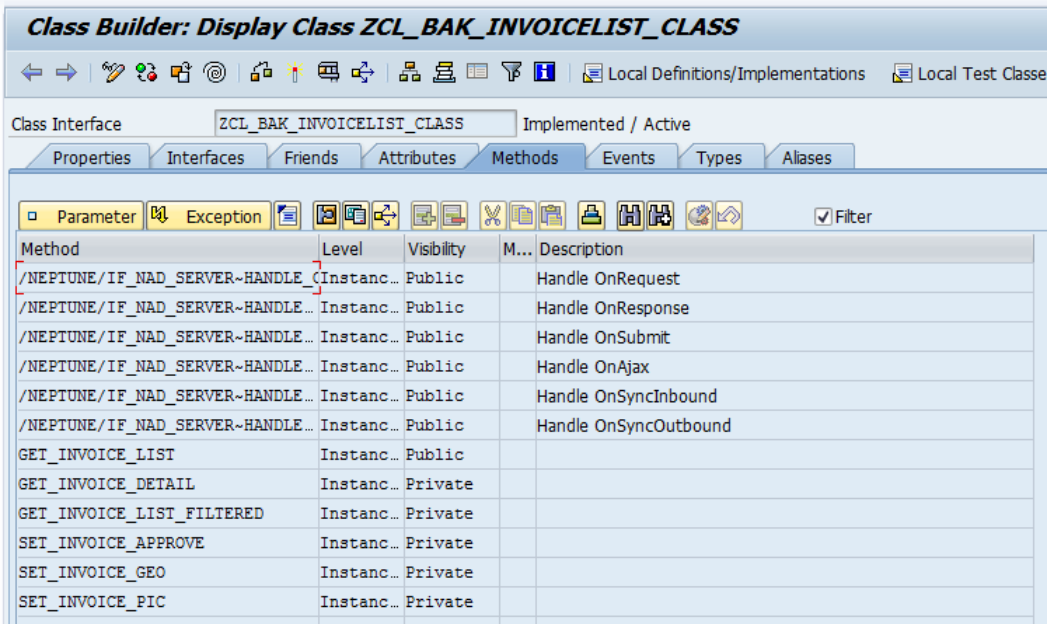

Figura 5.14: Metodi della classe di supporto

Con riferimento alla figura 5.14, si notano una serie di metodi ereditati dall'interfaccia impostata, il cui nome inizia con **/NEPTUNE/**: l'unico che va modificato è il quarto, ovvero **/NEPTUNE/IF_NAD_SERVER~HANDLE_ON_AJAX** che, ricevuta la chiamata Ajax, seleziona il metodo che andrà effettivamente a gestire la chiamata, in base all'ID fornito.

```
method /NEPTUNE/IF_NAD_SERVER~HANDLE_ON_AJAX.
        case ajax_id.

        when 'GET_INVOICE_LIST'.
        call method get_invoice_list().
        when 'GET_INVOICE_DETAIL'.
        call method get_invoice_detail(ajax_value).
        when 'GET_INVOICE_LIST_FILTERED'.
        call method get_invoice_list_filtered(ajax_value).
        when 'SET_INVOICE_APPROVE'.
```

```
            call method set_invoice_approve(ajax_value).
           when 'SET_INVOICE_GEO'.
            call method set_invoice_geo(ajax_value).
           when 'SET_INVOICE_PIC'.
            call method set_invoice_pic(ajax_value).

            endcase.
endmethod.
```

Codice 5.1: NepTune: Metodo per la gestione delle chiamate Ajax

Come si nota nel listato 5.1, la gestione avviene su due livelli: impostando nell'oggetto chiamante un ID che identifica il tipo di operazione che si vuole effettuare, viene selezionato il metodo con l'ID corrispondente, al quale viene inoltrato il parametro *ajax_value*. È importante notare che è possibile utilizzare un solo parametro di tipo stringa, per cui è stato deciso un pattern di serializzazione nel caso debbano essere inviati più parametri.

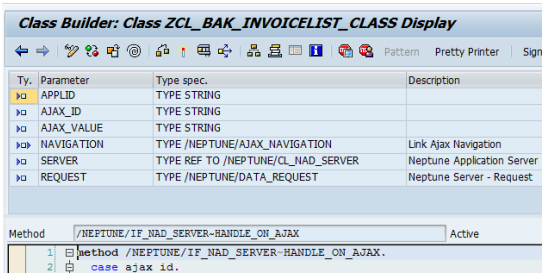

Figura 5.15: Parametri del metodo che gestisce le chiamate Ajax

Il pattern scelto è del tipo `<valore>**<valore>`, con `**` separatore.
La decodifica avviene utilizzando la funzione **Split**[1], che accetta un stringa ed un separatore come input e una serie di strutture dati come output, che provvederemo poi a convertire nel tipo di dati appropriato. Per fare ciò, ci si avvale dell'utilizzo dei **field-symbol**, che in ABAP sono molto simili ai puntatori in C.
Come si vedrà nel listato 5.2, il procedimento avviene dividendo la stringa nelle variabili desiderate, la cui lunghezza deve essere uguale a quella del tipo di dato che andremo ad assegnare; successivamente vengono creati i *field-symbol* corrispondenti e a questi viene assegnato il valore di ciascuna variabile, con il tipo adatto.

[1] `http://help.sap.com/abapdocu_702/en/abapsplit.htm`

```
method GET_INVOICE_LIST_FILTERED.
DATA: ABUKRS(4), AKOSTL(10), APROJK(8), DEL(2) VALUE '**'.

SPLIT AJAX_VALUE AT DEL INTO ABUKRS AKOSTL APROJK.

FIELD-SYMBOLS: <fs1> TYPE ANY, <fs2> TYPE ANY,
               <fs3> TYPE ANY.

ASSIGN ABUKRS TO <fs1> CASTING TYPE BUKRS.

ASSIGN AKOSTL TO <fs2> CASTING TYPE KOSTL.

ASSIGN APROJK TO <fs3> CASTING TYPE PS_PSP_PNR.

CALL FUNCTION 'ZMOB_INV_PAY_GET_LIST'
EXPORTING
I_USER_NAME          = 'GMARCON'
I_BUKRS              = <fs1>
I_KOSTL              = <fs2>
I_PROJK              = <fs3>
IMPORTING
E_INVOICE_LIST       = LT_INVOICE
.

endmethod.
```

Codice 5.2: NepTune: Metodo GET_INVOICE_LIST_FILTERED

I metodi creati consistono in una chiamata a funzione, utilizzando il costrutto **Call function**, che consente di chiamare una funzione già presente nel sistema, passando nella sezione **Exporting** le variabili in input e nella sezione **Importing** una struttura dati adeguata a ricevere le variabili in output.
Dato che i metodi sono costituiti per lo più da deserializzazioni e da chiamate a funzione, si riporta il codice del solo metodo GET_INVOICE_LIST_FILTERED nel listato 5.2.
I metodi creati ed il loro scopo sono:

- **GET_INVOICE_LIST**: ottenere la lista delle fatture per popolare in visualizzazione la tabella "master"
- **GET_INVOICE_DETAIL**: ottenere il dettaglio dell'elemento selezionato nella tabella "master"
- **GET_INVOICE_LIST_FILTERED**: ottenere una lista "master" filtrata con valori immessi dall'utente

- **SET_INVOICE_APPROVE**: impostare come approvata/rifiutata una fattura
- **SET_INVOICE_GEO**: impostare le coordinate per la geolocalizzazione in una fattura
- **SET_INVOICE_PIC**: impostare un'immagine in una fattura

5.2.3 Utilizzo delle chiamate Ajax

Ogni componente per cui si prevede dovrà esserci un'interazione con il sistema deve essere impostato affinché possa soddisfare le condizioni dei metodi appena definiti.
In particolare, è necessario specificare un *Modello* dei dati utilizzati e l'*Ajax ID* della chiamata che permette l'ottenimento/invio degli stessi.

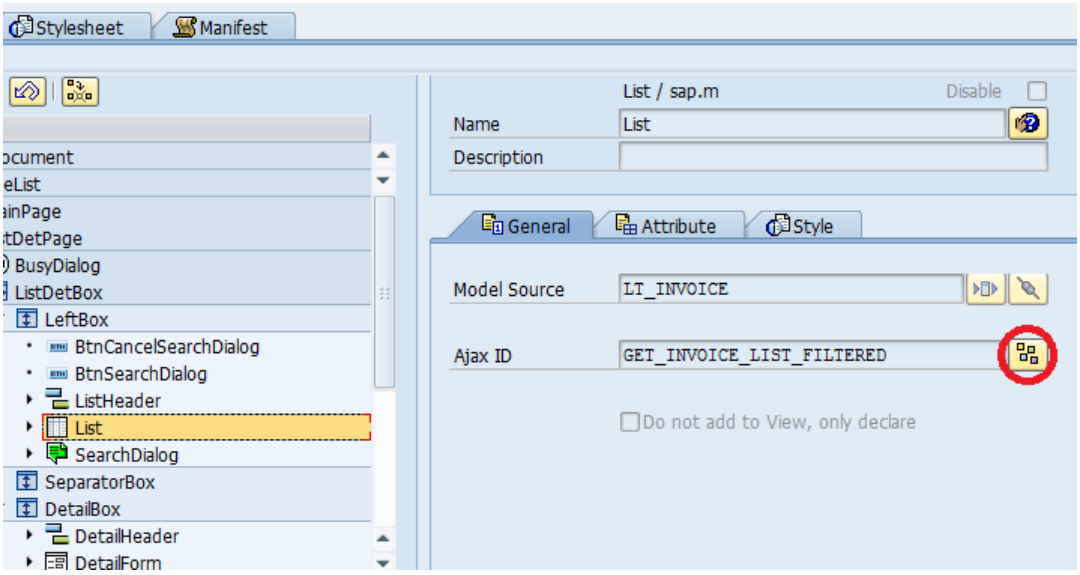

Figura 5.16: Impostazioni globali dell'oggetto List

Seguendo la figura 5.16, i dati ottenuti dalla chiamata Ajax di ID *GET_INVOICE_LIST_FILTERED* popolano il modello *LT_INVOICE*, che verrà visualizzato correttamente.

Nel caso si desideri che la chiamata interagisca anche con altri modelli, o preveda anche l'invio di dati, si deve cliccare sul pulsante *Additional Model Send/Receive* (cerchiato in rosso nella figura 5.16) e spuntare le caselle relative ai modelli di interesse.

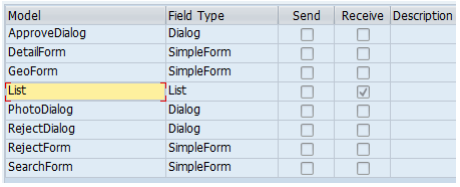

Figura 5.17: *Additional Model Send/Receive* della chiamata Ajax dell'oggetto List

Un'ultima importante peculiarità di questa tecnologia è che, per i componenti che prevedono una chiamata Ajax, viene automaticamente reso disponibile il metodo JavaScript `getOnlineNomeOggetto(ajax_value)`, che consente l'invio della chiamata al sistema.

5.2.4 Metodi di accesso all'hardware del dispositivo

Il seguente listato 5.3, mostra come accedere alle funzionalità di geolocalizzazione del dispositivo.

```
navigator.geolocation.getCurrentPosition(onSuccess,
 onError, {enableHighAccuracy:true});
function onSuccess(position) {
 $('#inGeoFormGEO-inner').attr('value',
  position.coords.latitude + ',' +
   position.coords.longitude);
}
function onError(){
 console.log('GEO error');
}
```

Codice 5.3: NepTune: Metodo per l'acquisizione delle coordinate per la geolocalizzazione

Viene sfruttato il metodo per la geolocalizzazione dell'oggetto HTML5 **navigator**, che consente l'accesso all'hardware.
Il seguente listato 5.4 mostra invece come acquisire un immagine.

```
photoChooser.addEventListener('change', function() {
 var files = photoChooser.files;
 if (files.length === 1) {
  Photo.setSrc(URL.createObjectURL(photoChooser.files[0] ));
  var canvas=document.getElementById('tempCanvas');
  var img = new Image;
  img.src = Photo.getSrc();
  img.onload = function() {
   var ctx = canvas.getContext('2d');
   var dialog = $('#PhotoDialog-scrollCont');
   resizeCanvasImage(img, canvas, dialog.width()-20,
    dialog.height()-20);
   var bukrs = modelDetailForm.getData().BUKRS;
   var belnr = modelDetailForm.getData().BELNR;
   var gjahr = modelDetailForm.getData().GJAHR;
   var buzei = modelDetailForm.getData().BUZEI;
   getOnlinePhotoDialog(bukrs+"**"+belnr+"**"+gjahr+
    "**"+canvas.toDataURL());
   getOnlineDetailForm(bukrs+"**"+belnr+"**"+gjahr+
```

```
    "**"+buzei);
    getOnlineList();
    PhotoDialog.close();
  }
 }
 else console.log('-----pic error');
});
$('#photoChooser').click();
```

Codice 5.4: NepTune: Metodo per l'acquisizione di un'immagine su NepTune

La funzione **resizeCanvasImage** è disponibile all'appendice D.
L'acquisizione avviene sfruttando solamente JavaScript; al termine vengono effettuate tre chiamate Ajax, **getOnlinePhotoDialog**, **getOnlineDetailForm** e **getOnlineList** rispettivamente per inviare l'immagine, aggiornare il dettaglio ed aggiornare la lista "master".

5.2.5 Test dell'applicazione

Ogniqualvolta si desideri controllare il corretto funzionamento dell'applicazione sviluppata, è necessario avviare una sorta di compilazione, cliccando sul pulsante *Activate*, cerchiato in verde in figura 5.18: dopo aver confermato le porzioni di applicazione alle quali si desidera vengano applicate le modifiche, il sistema provvederà ad attivarle.

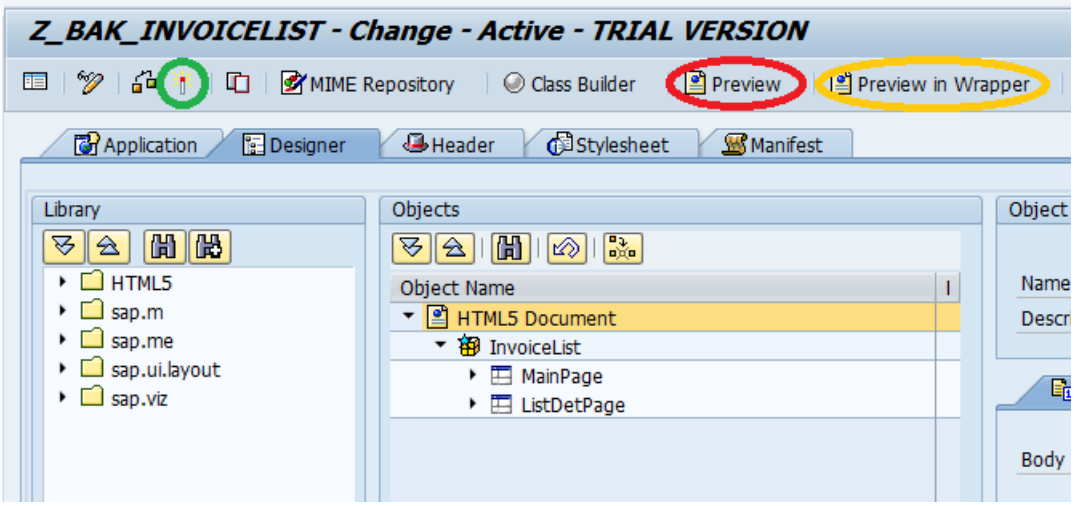

Figura 5.18: Pulsanti per attivazione e preview

Essendo l'applicazione sviluppata direttamente nei sistemi, è raggiungibile nella rete aziendale. È sufficiente cliccare sul pulsante *Preview*, cerchiato in rosso nella figura 5.18, per aprire il browser. In alternativa, per avere un'idea di come si possa presentare in un dispositivo mobile, cliccare sul pulsante *Preview in Wrapper*, cerchiato in giallo.

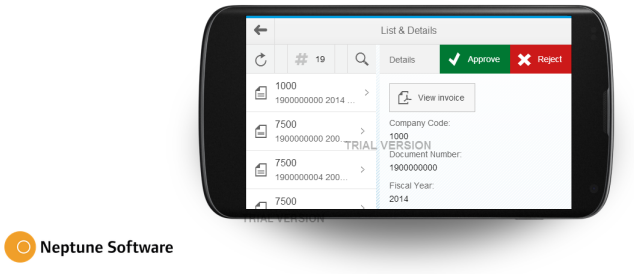

Figura 5.19: Visualizzazione della *Preview in Wrapper*

5.2.6 Considerazioni

Lo sviluppo con questa tecnologia è molto veloce anche se, non essendo un programmatore ABAP, non posso apprezzarne la vicinanza, ma la comunità in rete ne è molto entusiasta. La programmazione trascinando gli elementi è abbastanza veloce: come per tutti gli IDE di questo tipo, tale vantaggio è compensato dalla difficoltà ad uscire dai binari imposti. Nel complesso, lo sviluppo di una semplice applicazione, per lo più votata alla visualizzazione di dati, è estremamente vantaggioso; appena però si desidera creare qualcosa di più complesso si finisce con il dover intraprendere strade molto complicate, seppur composte di elementi semplici: ad esempio, l'applicazione di proprietà CSS può avvenire creando un oggetto personalizzato, le cui proprietà devono essere riscritte, o utilizzando JQuery, il che comporta lunghi tempi di debug per individuare le classi e gli ID autogenerati da NepTune.

5.3 Sviluppo dell'applicazione con Sencha Architect

Tra gli IDE analizzati, questo è il più completo dal punto di vista dello sviluppo, perché, oltre alla possibilità di programmare trascinando elementi da un insieme di componenti preimpostate, consente di gestire l'architettura MVC più direttamente (già nell'interfaccia le componenti sono separate tra *Model*, *View* e *Controller*, vedi figura 5.20) e la pacchettizzazione in applicazione ibrida senza lasciare l'ambiente di sviluppo, con l'aggiunta di poche impostazioni.

5.3.1 Interfaccia grafica

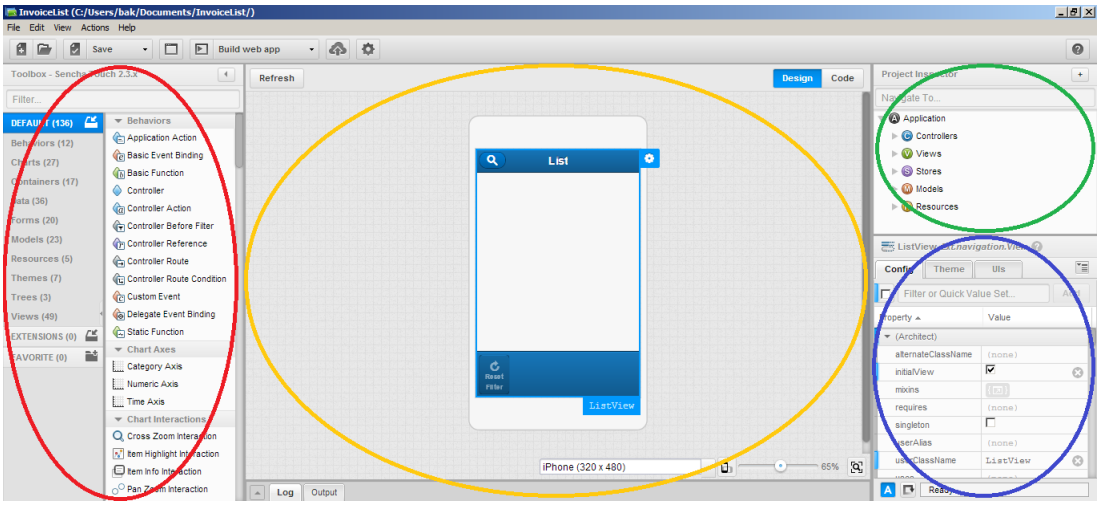

Figura 5.20: Interfaccia grafica dell'IDE Sencha Architect

Come per la tecnologia precedente, facendo riferimento alla figura 5.20, lo sviluppo avviene trascinando componenti dall'area *Toolbox*, contrassegnata in rosso, all'area *Project Inspector*, segnalata in verde. Quest'ultima è organizzata ad albero, la cui radice è **Application**, l'applicazione che si sta sviluppando, i cui sottonodi sono:

- **Controllers**, dove sono raccolti tutti i controller contenenti funzioni e riferimenti ad oggetti
- **Views**, dove sono raccolte tutte le view dell'applicazione, con i loro vari componenti
- **Stores**, dove sono raccolti riferimenti alle varie strutture di archiviazione, ognuna delle quali deve essere connessa da un modello
- **Models**, dove sono raccolti i modelli dell'applicazione
- **Resources**, dove sono raccolte le altre componenti e le risorse esterne a quelle messe a disposizione da Sencha Architect

L'area contrassegnata in blu consente di cambiare le impostazioni (grafiche e non) dell'oggetto selezionato nel *Project Inspector*.
Infine l'area gialla consente sia di visualizzare una (molto carente) anteprima delle view create, sia di scrivere codice JavaScript, dove consentito.

5.3.2 Manipolazione dei dati e interazione con i sistemi

La peculiarità di questa tecnologia è l'utilizzo, qualora si desideri consultare o manipolare dati, della terna *Model-Store-Proxy*.

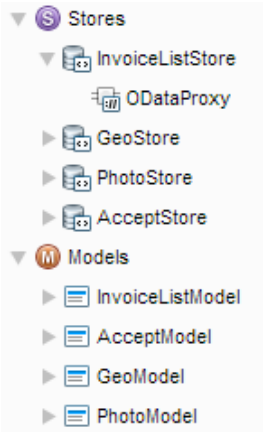

Figura 5.21: Store, proxy e modelli creati

È stato creato un modello per ogni entity set e service operation esposte, in particolare:

- **InvoiceListModel**, per l'entity set *InvoiceCollection*
- **AcceptModel**, per la service operation *InvoiceApproveReject*
- **GeoModel**, per la service operation *InvoiceSaveGeoLocalization*
- **PhotoModel**, per la service operation *InvoiceSaveFile*

Per ognuno di questi è stato creato e associato uno store, ciascuno dei quali è connesso ad un proxy. La funzione del proxy è di fornire metodi per la comunicazione, mantenendo tra le proprie impostazioni l'URI completo dei relativi entity set service operation.
Non essendo direttamente disponibile un proxy OData in Sencha Architect, si è importato un proxy disponibile su Sencha Market[2], di difficile utilizzo perché, essendo scritto per lo sviluppo senza l'ausilio dell'IDE, va riadattato alle nuove esigenze e non garantisce il corretto funzionamento di tutti i metodi esposti, tranne naturalmente la connessione all'URI impostato.
Per effettuare le varie connessioni si è quindi aggirato il problema ricostruendo di volta in volta l'URI dell'operazione desiderata, come mostrato nel listato 5.5.

[2] `https://market.sencha.com/extensions/sencha-touch-odata-connector-for-sap`

```
var store = Ext.StoreManager.get('InvoiceListStore');
   //oggetto store
var proxy = store.getProxy();
   //oggetto proxy
var url = proxy._url;
   //salvataggio URI "di base"
var ep = "(Bukrs='"+bukrs+"',Belnr='"+belnr+
         "',Gjahr='"+gjahr+"',Buzei='"+buzei+"')";
   //costruzione stringa con i parametri
var base = proxy._url.replace
           ("?$filter=IUserName eq 'ALTEVIE'", '');
   //cancellazione porzione di URI inutile all'operazione
proxy._url=base+ep;
   //utilizzo del nuovo URI costruito
store.load(function(){
  //avvio della comunicazione
  [...] //operazioni da effettuare al termine
        //della comunicazione
proxy._url=url;
   //ripristino URI originale
}
```

Codice 5.5: Sencha: Esempio di ciclo di modifica dell'URI per un'operazione di filtro sulla lista delle fatture

Il ciclo delle operazioni da effettuare consiste quindi in:

- **ottenimento** degli oggetti *Store* e *Proxy* interessati
- **salvataggio** dell'URI originale in una variabile temporanea
- **costruzione** dell'URI dell'operazione
- **rimpiazzo** dell'URI originale con il nuovo URI
- **avvio** della comunicazione, con il metodo *load* dello store
- **ripristino** dell'URI originale al termine della comunicazione

Gli URI originali dei proxy sono visibili in tabella 5.1.

Store (proxy)	**URI**
InvoiceListStore	`http://sapnwgateway.altevielab.com:8000/sap/opu/odata/sap/ZMOB_INV_PAY_SRV/InvoiceCollection?$filter=IUserName eq 'ALTEVIE'`
GeoStore	`http://sapnwgateway.altevielab.com:8000/sap/opu/odata/sap/ZMOB_INV_PAY_SRV/InvoiceSaveGeoLocalization?`
PhotoStore	`http://sapnwgateway.altevielab.com:8000/sap/opu/odata/sap/ZMOB_INV_PAY_SRV/InvoiceSaveFile?`
AcceptStore	`http://sapnwgateway.altevielab.com:8000/sap/opu/odata/sap/ZMOB_INV_PAY_SRV/InvoiceApproveReject?`

Tabella 5.1: URI utilizzati nei proxy

5.3.3 Metodi di accesso all'hardware del dispositivo

Il seguente listato 5.6, mostra come accedere alle funzionalità di geolocalizzazione del dispositivo.

```
try{
 navigator.geolocation.getCurrentPosition({
  allowHighAccuracy: true,
  success: function(position) {
  Ext.getCmp('MapsLocalization')
   .setValue(position.replace(' ', ','));
  },
  failure: function() {
   Ext.Msg.alert('Geolocation error',
    'something went wrong!');
  }
 });
}
catch(e){
 var geo = Ext.create('Ext.util.Geolocation', {
  autoUpdate: false,
  allowHighAccuracy: true,
```

```
  listeners: {
   locationupdate: function(geo) {
         Ext.getCmp('MapsLocalization')
         .setValue(geo.getLatitude()+','
          +geo.getLongitude());
   },
   locationerror: function(geo, bTimeout,
    bPermissionDenied, bLocationUnavailable, message) {
     if(bTimeout)
      Ext.Msg.alert('Timeout occurred',
       "Could not get current position");
         else
     alert('Error occurred.');
   }
  }
 });
 geo.updateLocation();
}
```

Codice 5.6: Sencha: Metodo per l'acquisizione delle coordinate per la geolocalizzazione

Per permettere la fruizione delle funzionalità di geolocalizzazione sia da dispositivo desktop che da dispositivo mobile, si è implementata la funzione in due modi: il primo, all'interno del costrutto *try*, come per l'applicazione sviluppata con NepTune, permette l'acquisizione delle coordinate sfruttando i metodi messi a disposizione da HTML5, mentre il secondo, all'interno del costrutto *catch*, utilizza le librerie Sencha per dispositivi desktop. Il motivo della doppia implementazione è che utilizzando le librerie Sencha su dispositivi desktop il metodo HTML5 non funziona.
L'acquisizione delle immagini avviene invece come mostrato nel listato 5.7.

```
try{
 Ext.device.Camera.capture({
  source: 'camera',
  destination: 'data',
  quality: 75,
  success: function(image) {
   Ext.getCmp('showPhoto')
    .setSrc(''+image);
   Ext.getCmp('showPhoto').setHidden(false);
   Ext.getCmp('Photo').setValue(Ext.getCmp('showPhoto')
    .getSrc());
  },
  failure: function() {
   Ext.Msg.alert('Error',
```

```
    'There was an error when acquiring the picture.');
  },
  scope: this
 });
}
catch(e){
 Ext.getCmp('photoContainer').setHtml('<canvas
  style="display: none;" width="300" height="300"
  id="tempCanvas"></canvas><input type="file"
  capture="camera" id="photoChooser" accept="image/*" />');
 var photoChooser=document.getElementById('photoChooser');
 photoChooser.addEventListener('change', function() {
  var files = photoChooser.files;
  if (files.length === 1) {
   Ext.getCmp('showPhoto').setSrc(URL.
    createObjectURL(photoChooser.files[0]));
   Ext.getCmp('showPhoto').setHidden(false);
   var canvas=document.getElementById('tempCanvas');
   var img = new Image();
   img.src = Ext.getCmp('showPhoto').getSrc();
   img.onload = function() {
    var ctx = canvas.getContext('2d');
    resizeCanvasImage(img, canvas, 1024, 768);
    var photoField = Ext.getCmp('Photo');
    photoField.setValue(canvas.toDataURL());
   };
   Ext.getCmp('photoContainer').setHtml("");
  }
  else {
   Ext.Msg.alert('No image', 'No image selected');
  }
 });
}
```

Codice 5.7: Sencha: Funzione per l'acquisizione delle immagini

La funzione **resizeCanvasImage** è disponibile all'appendice D.
Come si può notare, anche in questo caso si è implementato la funzione in due modi: il primo, nel costrutto *try*, utilizza le librerie Sencha per l'acquisizione di immagini dalla fotocamera di dispositivi mobili, mentre il secondo, nel costrutto *catch*, prevede l'acquisizione da dispositivo desktop dove non è possibile accedere alla fotocamera. In questo secondo caso la tecnica utilizzata è molto simile a quella adottata su NepTune, tranne che per il fatto che si è scritto del codice HTML per utilizzare elementi come *canvas* e campi di *input*, che altrimenti sarebbero stati difficilmente sfruttabili all'interno dell'IDE.

5.3.4 Test dell'applicazione

Lo sviluppo con questa tecnologia è possibile anche in locale, a patto che si disponga di un server HTTP.

Facendo riferimento alla figura 5.22, cliccando sul pulsante *Project Settings*

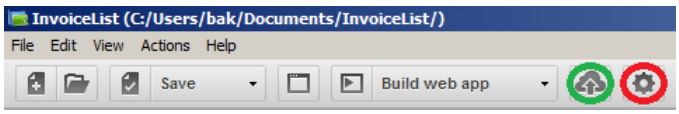

Figura 5.22: Pulsanti per la pubblicazione

(cerchiato in rosso), è possibile impostare la cartella di destinazione della pubblicazione, mentre cliccando sul pulsante *Publish* (cerchiato in verde), è possibile salvare e pubblicare l'applicazione.

CORS

Durante i test nello sviluppo in locale, ci si imbatte nella politica di sicurezza dei browser **same-origin security policy**, che permette l'esecuzione di script e l'utilizzo di **XMLHttpRequest** (XHR) solo all'interno dello stesso sito (*same-origin*). Per ovviare a questo fatto si ricorre al **cross-origin resource sharing**[3] (CORS), che permette l'utilizzo di XHR che vadano anche al di fuori del dominio di origine. L'utilizzo di questa modalità è possibile nella maggior parte dei server e dei browser moderni. Dato che questo problema si riscontra solo nei test in locale, si è configurato solamente il browser.

Si è scelto di utilizzare Google Chrome, sia per l'immediata disponibilità di buoni strumenti per lo sviluppo, sia perché è tra i browser in cui è più facile configurare il CORS, semplicemente aggiungendo al comando che avvia il programma l'opzione "**–disable-web-security**".

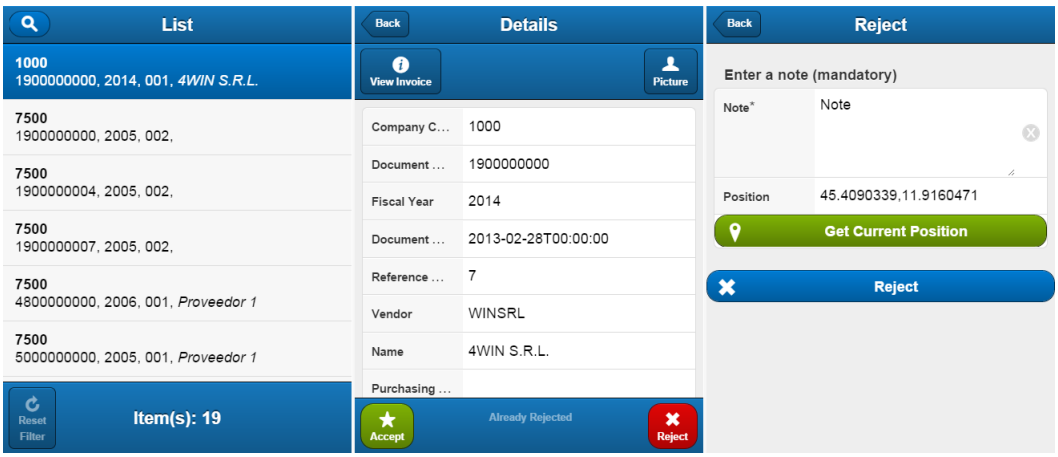

Figura 5.23: Alcune view dell'applicazione sviluppata

[3] `http://enable-cors.org/`

5.3.5 Creazione applicazione ibrida

Uno dei più grandi pregi di questo IDE è la possibilità di pacchettizzare l'applicazione sviluppata con l'inserimento di pochissime impostazioni.
Viene analizzato il caso della pacchettizzazione per piattaforma Android.

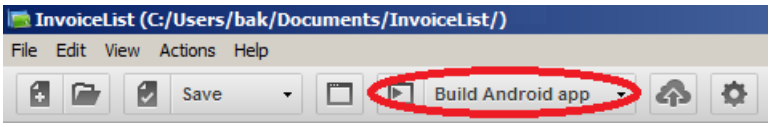

Figura 5.24: Pulsante e menù per la compilazione

Scegliendo dal menu cerchiato in figura 5.24 la voce **Build Android App**, si aprirà una finestra che consente l'inserimento dei settaggi in tre sezioni, come da figura 5.25:

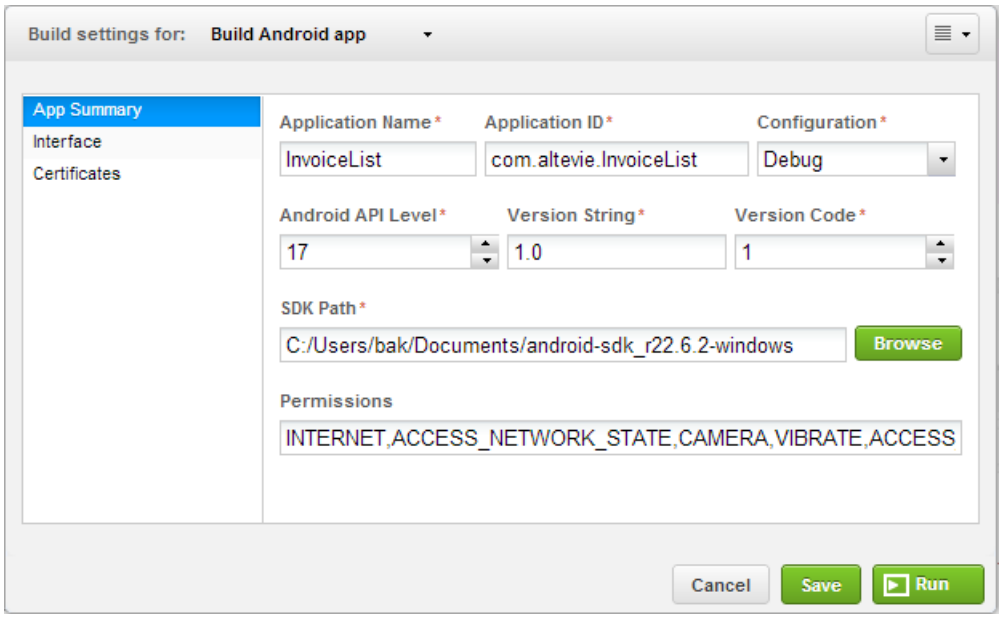

Figura 5.25: Impostazioni per la pacchettizzazione per Android

- **App Summary**: in questa sezione vanno indicati nome, ID, configurazione (a scelta tra *Debug* e *Release*), varie impostazioni Android (API level e versione), la posizione dell'Android SDK nella macchina e i permessi di utilizzo delle funzionalità del dispositivo

- **Interface**: in questa sezione si indicano quali orientazioni del dispositivo sono supportate e le icone dell'applicazione, una per ogni dimensione (36×36, 48×48 e 72×72)

- **Certificates**: in questa sezione si indicano password e posizione del certificato generato come da Step 1 della guida[4] nella documentazione Sencha Touch 2, per la pacchettizzazione in modalità *Release*

Un volta terminate le impostazioni, è sufficiente cliccare il pulsante *Run* per avviare il procedimento.

5.3.6 Considerazioni

Come per la tecnologia precedente, Sencha Architect consente lo sviluppo di applicazioni semplici in tempi brevissimi. Per applicazioni più complesse, che richiederebbero di uscire dai "binari" predisposti, si richiede una conoscenza sempre maggiore del prodotto: raramente si può modificare interamente l'oggetto che si è inserito o si possono controllarne i metodi, spesso è solamente predisposto l'inserimento di snippet in posizioni precisate. È prevista comunque la creazione di *override*, anche se viene sconsigliato dall'IDE stesso.
I punti di forza di questa tecnologia sono la migliore resa grafica e la possibilità di creare applicazioni ibride con pochi click.

[4] `http://docs.sencha.com/touch/2.0.2/#!/guide/native_android`

5.4 Sviluppo dell'applicazione con SAPUI5

Il più grande pregio e il più grande difetto di questa tecnologia stanno nel fatto che non è fornito un IDE, ma ci si appoggia ad Eclipse, per cui SAP ha sviluppato molti plugin: non esistono binari prestabiliti e anche ciò che viene generato automaticamente attraverso vari *wizard* è modificabile.
Un altro vantaggio di questa tecnologia è che una volta padroneggiata, è possibile modificare ed estendere anche applicazioni disponibili su SAP Fiori, che ultimamente sta avendo molto successo.

5.4.1 Plugin utilizzati

Dato che si sfrutta per questa tecnologia l'IDE Eclipse, il cui utilizzo è largo e ampiamente documentato, in questa sezione si discuteranno i plugin utili allo sviluppo dell'applicazione.
I plugin sviluppati da SAP sono per la versione *Juno* di Eclipse e sono raggiungibili all'url `https://tools.hana.ondemand.com/juno`.
Di questo pacchetto sono da installare:

- **ABAP Development Tools for SAP NetWeaver**, per la connessione del progetto ai sistemi ABAP e la creazione di CR (Change Request, simile ad un sistema di versioning centralizzato)
- **SAP Netweaver Gateway Productivity Accelerator (GWPA)**, per la connessione al servizio OData e la creazione automatica della base del progetto con un "*wizard*"
- **UI Development Toolkit for HTML5**, per le librerie SAPUI5

5.4.2 Creazione del progetto

Una volta installati i plugin, si procede alla creazione del progetto, selezionando, sotto la voce **OData Development**, **Starter Application Project**.

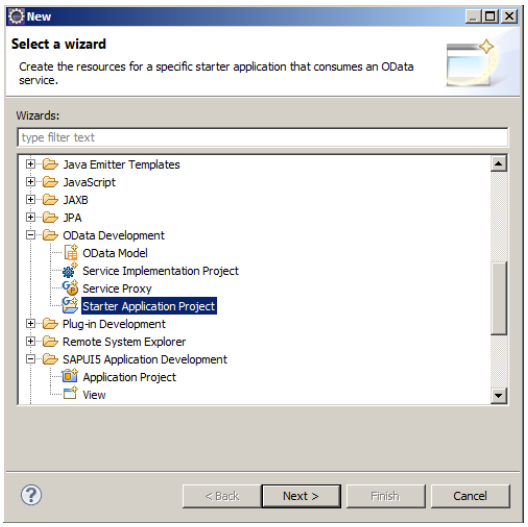

Figura 5.26: Fase iniziale della creazione del progetto

Nella schermata successiva si inserisce il nome del progetto e per l'impostazione *Create a new project for*, nel menu a tendina, si seleziona **HTML5**.

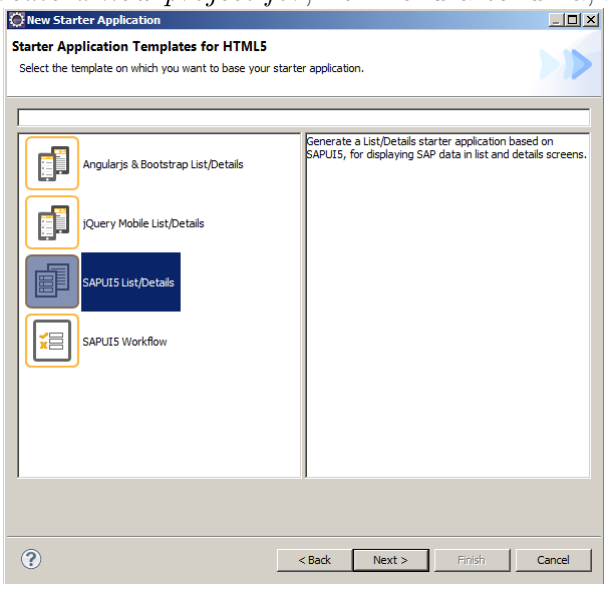

Figura 5.27: Selezione del template dell'applicazione

Successivamente si sceglie come template **SAPUI5 List/-Details**, per avere una visualizzazione in stile "*master-detail*".

Come **Remote location** si inserisce l'URI del servizio, `http://sapnwgateway.altevielab.com:8000/sap/opu/odata/sap/ZMOB_INV_PAY_SRV` e, dopo aver effettuato il login, nella parte inferiore della finestra appaiono i dettagli del servizio in forma di diagramma ad albero.

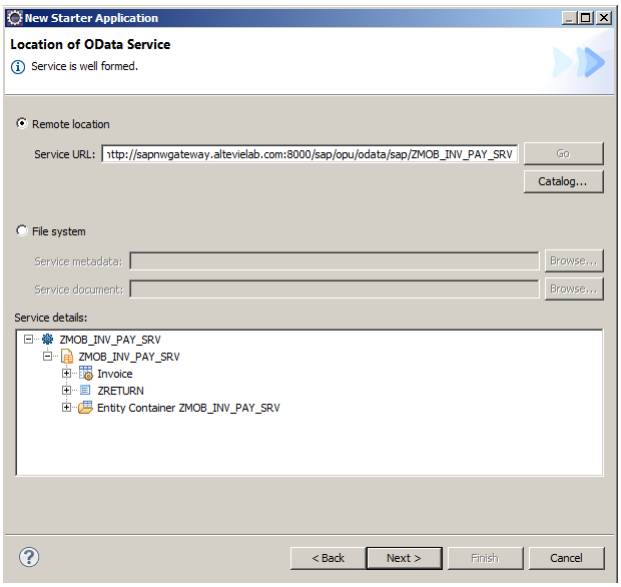

Figura 5.28: Connessione al servizio OData

Nella schermata successiva si impostano le view "master" e "detail", specificando i campi che devono essere visualizzati.
A questo punto si dispone di una semplice applicazione che visualizza i dati connettendosi al sistema.

5.4.3 Modifiche e aggiunte al progetto generato automaticamente

Si rende necessario fare in modo che l'applicazione visualizzi solo le fatture sotto la responsabilità dell'utente: dato che ci si è comunque dovuti autenticare all'avvio dell'applicazione(vedi 5.1.5), si possono riutilizzare tali dati, come indicato nel listato 5.8

```
loadContent : function() {
 var view = this.getView();
 var userName = "";
 oUser = sap.ui2.shell.getUser();
 oUser.load({
  depthAtRoot : 0,
  }, function() {
   userName = oUser.getId();
   var filters = new Array();
   var userFilter = new sap.ui.model.Filter("IUserName",
        sap.ui.model.FilterOperator.EQ, userName);
   filters.push(userFilter);
   view.oList.bindItems("/InvoiceCollection",
    view.itemTemplate, null, filters);
   }, function() {
   alert("Error retrieving logged in user information");
 });
}
```

Codice 5.8: SAPUI5: Metodo *loadContent* per l'acquisizione dell'utente nel controller della view "master"

L'oggetto **oUser** ottenuto dal metodo **sap.ui2.shell.getUser()** permette l'accesso a tali informazioni per applicare un filtro ai dati, grazie al metodo **bindItems** dell'oggetto **oList**, consistente nella lista "master".

Dato che si desidera che vengano visualizzate più informazioni nel dettaglio dell'elemento selezionato, per poterle manipolare è necessario creare un modello separato da popolare con i dati ricevuti dalla richiesta.

```
onListItemTap : function(oEvent) {
 //creazione e popolamento del modello del dettaglio
 var oBindingContext = oEvent.oSource.getBindingContext();
 var oModelDetail = new sap.ui.model.odata.ODataModel(
  getServiceURL(),false, "", "", null, null, null, true);
 oModelDetail.read(oBindingContext.getPath(), null,
  null, false, function(oData, oResponse) {
   var oODataJSONModel = new sap.ui.model.json.JSONModel();
```

```
    oODataJSONModel.setData(oData);
    sap.ui.getCore().setModel(oODataJSONModel,
     "DetailModel");
    }, function() {
          alert("Error retrieving Invoice Details");
 });
 //Visualizzazione dettagli con i dati appena ricevuti
 sap.ui.getCore().getEventBus().publish("nav", "to", {
  viewId : "app.details.Invoice2",
  data : { bindingContext : oBindingContext }
 });
}
```

Codice 5.9: SAPUI5: Metodo *onListItemTap* per la selezione della fattura nel controller della view "master"

Nella prima parte del listato 5.9 avviene la creazione del modello **oModelDetail**, mantenendo il *binding context* (in questo caso l'entity set *InvoiceCollection*). Successivamente, con il metodo **read** del nuovo modello creato, avviene il poloplamento.
La seconda parte invece esemplifica il passaggio della visualizzazione al dettaglio, fornendo come parametro i dati appena ricevuti.
Entrambi i listati sono da sostituire ai metodi autogenerati nel controller della view "master".

5.4.4 Metodi di accesso all'hardware

Dato che l'accesso all'hardware, per come è stata progettata l'applicazione, avviene con controlli presenti nella view "detail", le seguenti porzioni di codice sono da inserire nel controller della view summenzionata.

```
var geoBtn = new sap.m.Button( {
 icon : "sap-icon://map",
 text : "Get current position",
 tap : function() {
  navigator.geolocation.getCurrentPosition(
   function(position) {
         geoField.setValue(position.coords.latitude + ','
          + position.coords.longitude);
   },
   function() {
    alert('Geolocalization Error');
```

```
    },
    { enableHighAccuracy : true});
 },
});
```

Codice 5.10: SAPUI5: Acquisizione delle coordinate per la geolocalizzazione nel controller della view "detail"

Nel listato 5.10 viene mostrata l'acquisizione delle coordinate per la geolocalizzazione all'interno della definizione di un bottone per l'interfaccia: come per le altre tecnologie, anche qui si utilizza il metodo offerto da HTML5.

```
var photoContainer = new sap.ui.core.HTML;
photoContainer.setContent('<canvas style="display: none;"
 width="300" height="300" id="tempCanvas"></canvas>
 <input type="file" capture="camera" id="photoChooser"
 accept="image/*" />');
photoChooser.addEventListener( 'change',
 function() {
  var files = photoChooser.files;
  if (files.length === 1) {
   var canvas = document.getElementById('tempCanvas');
   var img = new Image();
   img.src = URL.createObjectURL(photoChooser.files[0]);
   img.onload = function() {
        var ctx = canvas.getContext('2d');
        resizeCanvasImage(img,canvas, 1024, 768);
        pic.setSrc(canvas.toDataURL());
   };
  } else {
   alert('no image selected');
  }
 }
);

[...]

onUpdatePicBtn : function() {
 try {
  navigator.camera.getPicture(
   function(imageData) {
        pic.setSrc = ""+ imageData;
   },
   function() { alert("Camera error");
   },
   {quality : 75,
```

```
            destinationType : destinationType.DATA_URL
        }
      );
    }
    catch (e) {
            console.log('not on mobile device');
    }
```

Codice 5.11: SAPUI5: Acquisizione dell'immagine nel controller della view "detail"

La funzione **resizeCanvasImage** è disponibile all'appendice D.
Anche per questa tecnologia si utilizza lo stesso approccio adottato con lo sviluppo su Sencha Architect: se l'applicazione viene eseguita su dispositivo mobile si usufruisce del metodo HTML5 per l'acquisizione da fotocamera, altrimenti viene adottato l'approccio per dispositivi desktop. A differenza delle altre implementazioni, non è possibile istanziare il controllo per l'applicazione desktop all'interno del costrutto *catch*, per l'impossibilità di aggiungere l'*event listener* all'oggetto **photoChooser** creato in codice HTML5, dato che con questa tecnologia verrebbe istanziato solo nel caso in cui il costrutto *try* fallisse l'esecuzione.

5.4.5 Test dell'applicazione

Grazie all'utilizzo dei plugin sopra citati, si ha a disposizione l'accesso ai sistemi SAP ed al relativo sistema di versioning, come illustrato in figura 5.29.

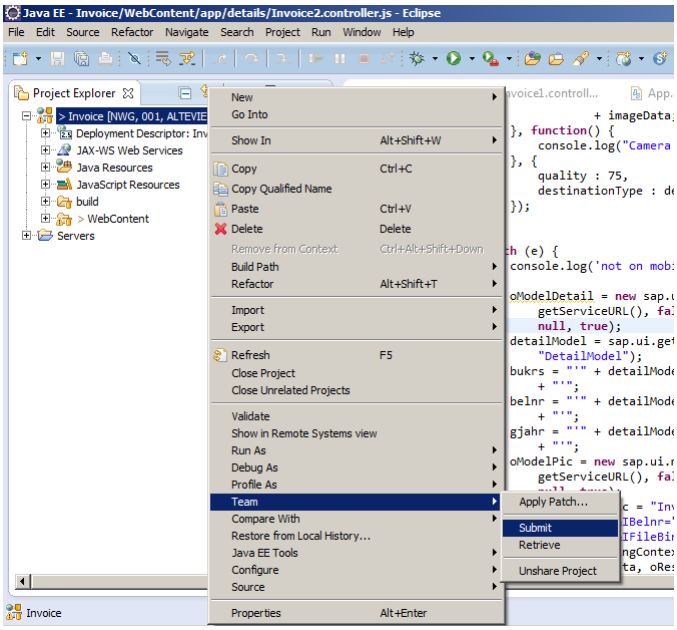

Figura 5.29: Procedura per l'upload dell'applicazione

Si aprirà quindi una finestra dove è possibile selezionare quali file modificati inviare al server.

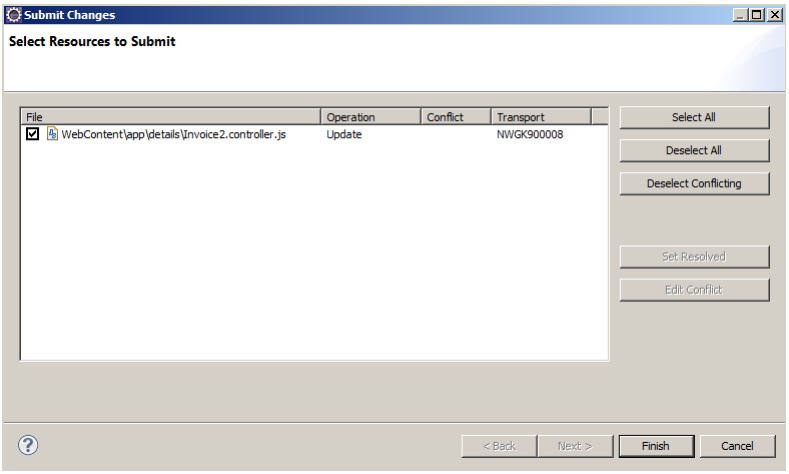

Figura 5.30: Selezione dei file da inviare

I test avvengono quindi eseguendo di volta in volta l'upload dell'applicazione e verificando il corretto funzionamento, connettendosi ad essa con un browser web. Il risultato è mostrato in figura 5.31.

5.4.6 Considerazioni

Tra le tecnologie sperimentate questa è quella che lascia più libertà nello sviluppo, dato che le componenti autogenerate sono poche e comunque sempre modificabili: è necessaria quindi una maggior competenza nella programmazione.
L'importanza ed il numero di plugin sviluppati da SAP per Eclipse denota una certa attenzione per questa metodologia di sviluppo, che si rende fondamentale per comprendere ed utilizzare efficacemente le nuove tecnologie messe a disposizione da SAP: Fiori e la promettente SAP Mobile Platform 3.0.
Inoltre il tema proposto conferisce all'applicazione lo stesso aspetto di quelle disponibili su Fiori, infondendo così un senso di continuità nel cliente che la utilizza.

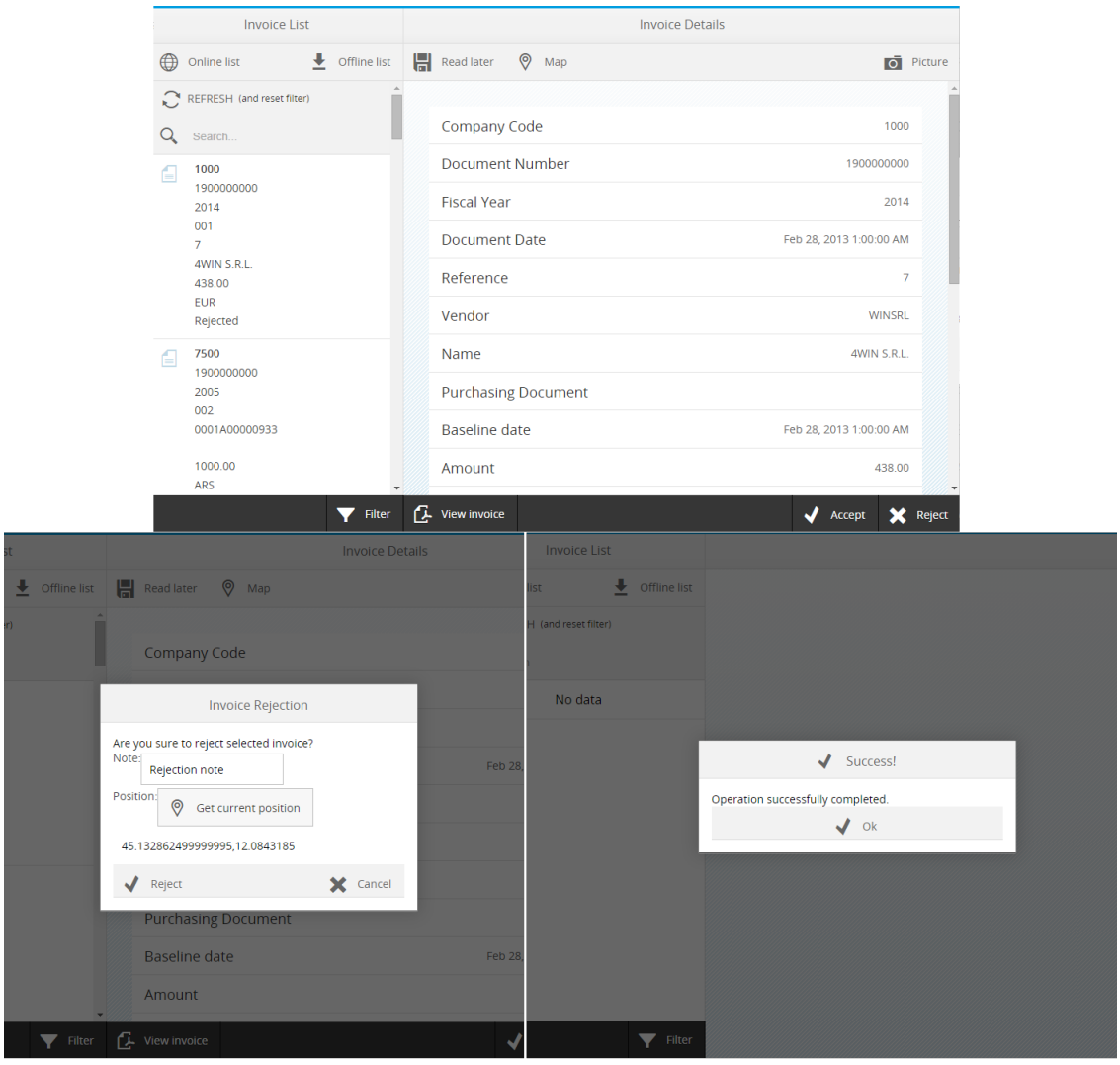

Figura 5.31: Alcune view dell'applicazione sviluppata

Capitolo 6

Conclusioni

Utilizzando le metriche proposte, è possibile eseguire un confronto tra le metodologie di sviluppo utilizzate.
Oltre ad una breve descrizione del risultato ottenuto, si dà una valutazione sintetica da 1 (scarso/insufficiente) a 5 (ottimo), che viene mostrata nella tabella riassuntiva 6.1.

6.1 Design dell'interfaccia

Con questa metrica si valuta il grado di difficoltà nella creazione dell'interfaccia grafica, unitamente alla resa visiva finale.
Il voto 1 corrisponde alla valutazione "molto impegnativo/implementazione manuale", mentre il voto 5 corrisponde a "molto semplice/implementazione *WYSIWYG*[1]".

6.1.1 NepTune

Grazie allo sviluppo per trascinamento di componenti da un repository al diagramma ad albero dell'applicazione, la creazione dell'interfaccia grafica è molto veloce; tuttavia è difficile applicare proprietà personalizzate ai componenti.
Voto: 3.

6.1.2 Sencha

Come per la tecnologia precedente, lo sviluppo avviene per trascinamento dei componenti, garantendo velocità e semplicità. La personalizzazione delle singole componenti è semplice ed integrata nell'IDE e la resa grafica è molto accattivante anche con l'utilizzo delle componenti standard offerte dall'ambiente di sviluppo.
Voto: 4.

[1] WYSIWIG: What You See Is What You Get

6.1.3 SAPUI5

Questa tecnologia, pur garantendo la massima flessibilità, non permette la creazione rapida dell'interfaccia, al di fuori di quella minimale ottenuta partendo dal "wizard": si rendono quindi necessarie più conoscenze di programmazione JavaScript.
Voto: 2.

6.2 Sviluppo dei controlli

Questa metrica valuta la semplicità di creazione dei controlli nell'applicazione, compresi i meccanismi di comunicazione.
Il voto 1 corrisponde alla valutazione "molto impegnativo/nessuno o pochi controlli/comunicazioni implementabili", mentre il voto 5 corrisponde a "molto semplice/ampia gamma di controlli/comunicazioni implementabili".

6.2.1 NepTune

Con questa tecnologia è possibile la gestione dei componenti scrivendo poco codice JavaScript, per lo più per attivare le comunicazioni con il sistema. Queste ultime sono create in una classe a parte e scritte in ABAP ad oggetti: si rende però necessaria una riformattazione dei dati per consentirne l'utilizzo.
Voto: 3.

6.2.2 Sencha

La creazione e la gestione dei controlli con questa tecnologia avviene in modo molto strutturato ed organizzato. Tuttavia l'inadeguatezza del proxy OData utilizzato nell'IDE rende difficile l'implementazione delle comunicazioni.
Voto: 2.

6.2.3 SAPUI5

La libertà dai "binari" imposti dalle altre tecnologie fa trionfare SAPUI5 in questa metrica, una volta comprese le librerie.
Voto: 5.

6.3 Accesso all'hardware

Con questa metrica si valuta la facilità di utilizzo di funzioni e metodi che permettono l'accesso all'hardware del dispositivo. Particolare rilievo viene dato alla possibilità di effettuare impostazioni nell'acquisizione.
N.B.: In tutte le tecnologie sono necessari accorgimenti per diversificare l'esperienza dell'utente a seconda che si trovi su dispositivo mobile o desktop.

Il voto 1 corrisponde alla valutazione “molto impegnativo/nessun accesso all’hardware”, mentre il voto 5 corrisponde a “molto semplice/accesso completo e configurabile all’hardware”.

6.3.1 NepTune

Questa tecnologia permette l’utilizzo dell’hardware solo tramite metodi HTML5, ma non consente l’acquisizione di immagini da fotocamera.
Voto: 2.

6.3.2 Sencha

Questa tecnologia, grazie alla presenza di librerie apposite, è quella che meglio si adatta all’utilizzo dell’hardware.
Voto: 4.

6.3.3 SAPUI5

Questa tecnologia, come per la prima, sfrutta solo HTML5, permette però anche l’utilizzo della fotocamera.
Voto: 3.

6.4 Performance

Questa metrica valuta le tempistiche di funzionamento dell’applicazione.
Il voto 1 corrisponde alla valutazione “inefficiente”, mentre il voto 5 corrisponde a “efficienza comparabile ad un’applicazione nativa”.
Tale valutazione è risultata ininfluente, dato che il caricamento dell’applicazione in tutti i casi avviene in poche centinaia di millisecondi, contro un tempo di attesa per i dati provenienti dai sistemi mediamente tra i 20 e i 30 secondi, sia che si utilizzi la web application che l’applicazione ibrida.

6.5 Tempo di acquisizione padronanza

Con questa metrica si valuta il tempo necessario per ottenere una comprensione ed una padronanza della tecnologia tali da poter velocemente replicare o sviluppare da zero una nuova applicazione. Particolare rilievo viene dato a disponibilità di documentazione, esempi e comunità di sviluppatori. Il voto 1 corrisponde alla valutazione “molto lungo”, mentre il voto 5 corrisponde a “molto breve”.

6.5.1 NepTune

Essendo lo sviluppo con NepTune Application Designer molto strutturato, le casistiche più frequenti sono affrontate quasi totalmente con la creazione dell’ap-

plicazione "di base" proposta nella documentazione. Documentazione ed esempi sono all'interno della Sap Community Network (SCN[2]).
Voto: 4.

6.5.2 Sencha

Lo sviluppo con questa tecnologia è il più strutturato in assoluto, anche per il fatto che l'utilizzo del pattern architetturale *MVC* è esplicito. Creando l'applicazione "di base" e consultando i progetti di esempio forniti con l'IDE, completi e funzionanti, è possibile acquisire confidenza in quasi tutte le situazioni. È presente un'ottima documentazione per le librerie Sencha Touch e una fervente comunità, anche se la maggioranza degli esempi e delle discussioni sono incentrate su Sencha Touch, non direttamente sull'IDE, rendendo necessario un lavoro di riadattamento.
Voto: 5.

6.5.3 SAPUI5

Nonostante siano presenti un'attivissima comunità (SCN), molti esempi ed una documentazione completa, la mancanza di un apposito IDE strutturato tende a far smarrire il programmatore, che deve spesso consultare la documentazione.
Voto: 3.

6.6 Manutenibilità del codice

Questa metrica valuta la facilità nella manutenzione e modifica del codice. Particolare rilevanza è data alla facilità di individuazione degli elementi di interesse e all'assenza di ripercussioni che hanno tali modifiche sul resto dell'applicazione. Il voto 1 corrisponde alla valutazione "molto impegnativo", mentre il voto 5 corrisponde a "molto semplice".

6.6.1 NepTune

La natura strutturata dell'IDE consente una forte modularità, permettendo di effettuare manutenzioni e cambiamenti del codice mirati, le cui conseguenze non si propagano al resto del codice.
Voto: 5.

6.6.2 Sencha

Come nella tecnologia precedente, l'IDE utilizzato è molto strutturato e modulare, per cui manutenzione e cambiamenti coinvolgono solo la porzione di codice interessata.
Voto: 5.

[2] `www.scn.sap.com`

6.6.3 SAPUI5

Non essendo utilizzato un IDE strutturato, la modularità del codice deve essere prevista sin da subito dallo sviluppatore: la manutenibilità e l'assenza di ripercussioni sono difficili da ottenere.
Voto: 2.

6.7 Team working

Con questa metrica si valuta la predisposizione della tecnologia all'utilizzo di sistemi di *versioning*, dando rilevanza alla possibilità di utilizzare un sistema distribuito, come ad esempio Git.
Il voto 1 corrisponde alla valutazione "non predisposto", mentre il voto 5 corrisponde a "predisposizione all'utilizzo a più sistemi di *versioning*".

6.7.1 NepTune

Questa tecnologia non prevede l'utilizzo di alcun sistema di versioning, se non quello utilizzato obbligatoriamente all'interno di SAP, che è centralizzato e consente quindi l'accesso ad uno sviluppatore alla volta.
Voto: 2.

6.7.2 Sencha

Questa tecnologia prevede, all'interno dell'IDE, l'utilizzo di molti sistemi di versioning, tra cui Git, il cui utilizzo è incoraggiato da parte di Sencha stessa.
Voto: 5.

6.7.3 SAPUI5

Questa tecnologia prevede l'utilizzo dei sistemi di versioning compatibili con Eclipse, tuttavia l'uso dei plugin SAP conduce preferenzialmente allo sfruttamento del sistema centralizzato integrato.
Voto: 3.

6.8 Offline

Questa metrica valuta la possibilità di implementare nell'applicazione un sistema per l'archiviazione offline dei dati, permettendo la sincronizzazione appena si ha la possibilità di tornare online.
N.B.: L'utilizzo di tale funzionalità è poco affrontato nelle community, per cui si sono incontrate alcune difficoltà nella sua implementazione.
Il voto 1 corrisponde alla valutazione "funzionalità assente/non implementabile", mentre il voto 5 corrisponde a "implementazione rapida e semplice, predisposta nativamente nella tecnologia".

6.8.1 NepTune

Questa tecnologia prevede l'implementazione di tali funzionalità appoggiandosi solamente a funzioni HTML5. L'utilizzo di tali metodi è però ostacolato dalla natura dell'IDE, rendendo difficile il corretto posizionamento di tale codice.
Voto: 2.

6.8.2 Sencha

Sencha prevede nelle proprie librerie l'utilizzo dell'archiviazione sia offline che online, l'unica differenza è nelle impostazioni dello *Store*.
Voto: 4.

6.8.3 SAPUI5

Come per NepTune, l'implementazione di tali funzionalità avviene appoggiandosi solamente a metodi HTML5; data la natura libera di questa tecnologia, il posizionamento del codice è più semplice.
Voto: 3.

6.9 Trasportabilità dell'applicazione

Con questa metrica si valuta la facilità nel trasportare l'applicazione da un dispositivo/sistema ad un altro.
N.B.: Nel caso si effettui la pacchettizzazione in applicazione ibrida, è sufficiente distribuire il pacchetto; in caso di migrazione del sistema SAP contenente i dati, l'applicazione dovrà essere corretta e ridistribuita. L'accesso come Web Application è possibile da qualunque browser.
Il voto 1 corrisponde alla valutazione "molto impegnativo/non trasportabile", mentre il voto 5 corrisponde a "molto semplice/trasporto assistito da *wizard*".

6.9.1 NepTune

Dato che lo sviluppo avviene all'interno del sistema SAP, sono già previste funzionalità per la migrazione dell'applicazione.
Voto: 5.

6.9.2 Sencha

La migrazione dell'applicazione avviene copiandola nel nuovo sistema.
Voto: 5.

6.9.3 SAPUI5

Eseguendo l'upload nel nuovo sistema con modalità simili alla creazione di un nuovo progetto, è possibile migrare l'applicazione.
Voto: 5.

Metrica	**NepTune**	**Sencha**	**SAPUI5**
Design dell'interfaccia	3	4	2
Sviluppo dei controlli	3	2	5
Accesso all'hardware	2	4	3
Performance	-	-	-
Tempo di acquisizione padronanza	4	5	3
Manutenibilità del codice	5	5	2
Team working (versioning)	2	5	3
Offline	2	4	3
Trasportabilità dell'applicazione	5	5	5

Tabella 6.1: Tabella riassuntiva delle valutazioni

Appendice A

Tipi di dato primitivi in OData

La seguente tabella, presa direttamente dalle specifiche, riassume l'insieme di tipi primitivi supportati e come devono essere rappresentati quando utilizzati in un URI o in un header HTTP OData.

Primitive Types	**Literal Form**	**Example**
Null Represents the absence of a value	null	Example 1: *null*
Edm.Binary Represent fixed -or variable-length binary data	$binary'[A-Fa-f0-9][A-Fa-f0-9]*'$ $ORX`[A-Fa-f0-9][A-Fa-f0-9]*'$ NOTE: X and binary are case sensitive. Spaces are not allowed between binary and the quoted portion. Spaces are not allowed between X and the quoted portion. Odd pairs of hex digits are not allowed.	Example 1: $X'23AB'$ Example 2: $binary'23ABFF'$
Edm.Boolean Represents the mathematical concept of binary-valued logic	true / false	Example 1: *true* Example 2: $false$
Edm.Byte Unsigned 8-bit integer value	$[A-Fa-f0-9]+$	Example 1: FF

Edm.DateTime Represents date and time with values ranging from 12:00:00 midnight, January 1, 1753 A.D. through 11:59:59 P.M, December 9999 A.D.	$datetime'yyyy - mm - ddThh : mm[: ss[.fffffff]]'$ NOTE: Spaces are not allowed between datetime and quoted portion. datetime is case-insensitive	Example 1: $datetime'2000 - 12 - 12T12 : 00'$
Edm.Decimal Represents numeric values with fixed precision and scale. This type can describe a numeric value ranging from negative $10^255 + 1$ to positive $10^255 - 1$	$[0 - 9] + .[0 - 9] + M\|m$	Example 1: $2.345M$
Edm.Double Represents a floating point number with 15 digits precision that can represent values with approximate range of $\pm 2.23e - 308$ through $\pm 1.79e + 308$	$[0 - 9] + ((.[0 - 9]+)\|[E[+\|-][0 - 9]+])d$	Example 1: $1E + 10d$ Example 2: $2.029d$ Example 3: $2.0d$

Edm.Single Represents a floating point number with 7 digits precision that can represent values with approximate range of $\pm 1.18e-38$ through $\pm 3.40e+38$	$[0-9]+.[0-9]+f$	Example 1: $2.0f$
Edm.Guid Represents a 16-byte (128-bit) unique identifier value	$guid'dddddddd-dddd-dddd-dddd-dddddddddddd'$ where each d represents $[A-Fa-f0-9]$	Example 1: $guid'12345678-aaaa-bbbb-cccc-ddddeeeeffff'$
Edm.Int16 Represents a signed 16-bit integer value	$[-][0-9]+$	Example 1: 16 Example 2: -16
Edm.Int32 Represents a signed 32-bit integer value	$[-][0-9]+$	Example 1: 32 Example 2: -32
Edm.Int64 Represents a signed 64-bit integer value	$[-][0-9]+L$	Example 1: $64L$ Example 2: $-64L$
Edm.SByte Represents a signed 8-bit integer value	$[-][0-9]+$	Example 1: 8 Example 2: -8
Edm.String Represents fixed -or variable- length character data	$'<anyUTF-8character>'$ Note: See definition of UTF8-char in [RFC3629]	Example 1: $'HelloOData'$

Edm.Time Represents the time of day with values ranging from 0:00:00.x to 23:59:59.y, where x and y depend upon the precision	$time' < timeLiteral >'$ timeLiteral = Defined by the lexical representation for time at `http://www.w3.org/TR/xmlschema-2`	Example 1: $13:20:00$
Edm. DateTimeOffset Represents date and time as an Offset in minutes from GMT, with values ranging from 12:00:00 midnight, January 1, 1753 A.D. through 11:59:59 P.M, December 9999 A.D	$datetimeoffset' < dateTimeOffsetLiteral >'$ dateTimeOffsetLiteral = Defined by the lexical representation for datetime (including timezone offset) at `http://www.w3.org/TR/xmlschema-2`	Example 1: $2002-10-10T17:00:00Z$

Tabella A.1: Tabella dei tipi di dato primitivo

Appendice B

Operatori e funzioni permesse per l'opzione $filter in OData

Operatore	Descrizione	Esempio
Operatori logici		
Eq	Uguale	`/Suppliers?$filter=Address/City eq Redmond`
Ne	Diverso	`/Suppliers?$filter=Address/City ne London`
Gt	Maggiore	`/Products?$filter=Price gt 20`
Ge	Maggiore o uguale	`/Products?$filter=Price ge 10`
Lt	Minore	`/Products?$filter=Price lt 20`
Le	Minore o uguale	`/Products?$filter=Price le 100`
And	AND logico	`/Products?$filter=Price le 200 and Price gt 3.5`
Or	OR logico	`/Products?$filter=Price le 3.5 or Price gt 200`
Not	NOT logico	`/Products?$filter=not endswith(Description,milk)`
Operatori aritmetici		
Add	Addizione	`/Products?$filter=Price add 5 gt 10`
Sub	Sottrazione	`/Products?$filter=Price sub 5 gt 10`
Mul	Moltiplicazione	`/Products?$filter=Price mul 2 gt 2000`
Div	Divisione	`/Products?$filter=Price div 2 gt 4`
Mod	Modulo	`/Products?$filter=Price mod 2 eq 0`
Operatori di raggruppamento		
( )	Raggruppatore di precedenza	`/Products?$filter=(Price sub 5) gt 10`

Tabella B.1: Operatori permessi per l'opzione $filter

Funzioni stringa	
Funzione	**Esempio**
bool substringof(string po, string p1)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=substringof(Alfreds, CompanyName) eq true`
bool endswith(string p0, string p1)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=endswith(CompanyName, Futterkiste) eq true`
bool startswith(string p0, string p1)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=startswith(CompanyName, Alfr) eq true`
int length(string p0)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=length(CompanyName) eq 19`
int indexof(string p0, string p1)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=indexof(CompanyName, lfreds) eq 1`
string replace(string p0, string find, string replace)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=replace(CompanyName, , ) eq 'AlfredsFutterkiste'`
string substring(string p0, int pos)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=substring(CompanyName, 1) eq lfredsFutterkiste`
string substring(string p0, int pos, int length)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=substring(CompanyName, 1, 2) eq lf`

string tolower(string p0)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=tolower(CompanyName) eq alfredsfutterkiste`
string toupper(string p0)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=toupper(CompanyName) eq ALFREDSFUTTERKISTE`
string trim(string p0)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=trim(CompanyName) eq AlfredsFutterkiste`
string concat(string p0, string p1)	`http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=concat(concat(City, , ), Country) eq Berlin, Germany`
Funzioni con date	
int day(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=day(BirthDate) eq 8`
int hour(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=hour(BirthDate) eq 0`
int minute(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=minute(BirthDate) eq 0`
int month(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=month(BirthDate) eq 12`

int second(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=second(BirthDate) eq 0`
int year(DateTime p0)	`http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=year(BirthDate) eq 1948`
Funzioni matematiche	
double round(double p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=round(Freight) eq 32d`
decimal round(decimal p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=round(Freight) eq 32`
double floor(double p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=round(Freight) eq 32d`
decimal floor(decimal p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=floor(Freight) eq 32`
double ceiling(double p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=ceiling(Freight) eq 33d`
decimal ceiling(decimal p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=floor(Freight) eq 33`
Funzioni con tipi	
bool IsOf(type p0)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=isof(NorthwindModel.Order)`

bool IsOf(expression p0, type p1)	`http://services.odata.org/Northwind/Northwind.svc/Orders?$filter=isof(ShipCountry,Edm.String)`

Tabella B.2: Funzioni permesse per l'opzione $filter

Appendice C

Service Metadata Document del servizio esposto

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="1.0"
 xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
 <edmx:DataServices m:DataServiceVersion="2.0"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/
   metadata">
  <Schema Namespace="ZMOB_INV_PAY_SRV" xml:lang="en"
   xmlns="http://schemas.microsoft.com/ado/2008/09/edm">
   <EntityType Name="Invoice" sap:content-version="1"
    xmlns:sap="http://www.sap.com/Protocols/SAPData">
    <Key>
     <PropertyRef Name="Bukrs"/>
     <PropertyRef Name="Belnr"/>
     <PropertyRef Name="Gjahr"/>
     <PropertyRef Name="Buzei"/>
    </Key>
    <Property Name="IBukrs" Type="Edm.String" Nullable="false"
     MaxLength="4" sap:label="Company Code" sap:creatable="false"
     sap:updatable="false" sap:sortable="false"
     sap:filterable="false"/>
    <Property Name="Bukrs" Type="Edm.String" Nullable="false"
     MaxLength="4" sap:label="Company Code" sap:creatable="false"
     sap:updatable="false" sap:sortable="false"
     sap:filterable="false"/>
    <Property Name="Belnr" Type="Edm.String" Nullable="false"
     MaxLength="10" sap:label="Document Number"
     sap:creatable="false" sap:updatable="false"
     sap:sortable="false" sap:filterable="false"/>
    <Property Name="Gjahr" Type="Edm.String" Nullable="false"
     MaxLength="4" sap:label="Fiscal Year" sap:creatable="false"
     sap:updatable="false" sap:sortable="false"
     sap:filterable="false"/>
    <Property Name="Buzei" Type="Edm.String" Nullable="false"
     MaxLength="3" sap:label="Line item" sap:creatable="false"
     sap:updatable="false" sap:sortable="false"
     sap:filterable="false"/>
```

```
<Property Name="Lifnr" Type="Edm.String" Nullable="false"
 MaxLength="10" sap:label="Vendor" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Name1" Type="Edm.String" Nullable="false"
 MaxLength="35" sap:label="Name" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Name2" Type="Edm.String" Nullable="false"
 MaxLength="35" sap:label="Name 2" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Wrbtr" Type="Edm.Decimal" Nullable="false"
 Precision="13" Scale="2" sap:label="Amount"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Waers" Type="Edm.String" Nullable="false"
 MaxLength="5" sap:label="Currency" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Bschl" Type="Edm.String" Nullable="false"
 MaxLength="2" sap:label="Posting Key" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="BschlText" Type="Edm.String" Nullable="false"
 MaxLength="20" sap:label="Name of posting key"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Koart" Type="Edm.String" Nullable="false"
 MaxLength="1" sap:label="Account Type" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="KoartText" Type="Edm.String" Nullable="false"
 MaxLength="60" sap:label="Short Descript."
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Mwskz" Type="Edm.String" Nullable="false"
 MaxLength="2" sap:label="Tax code" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="MwskzText" Type="Edm.String" Nullable="false"
 MaxLength="50" sap:label="Description" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Zuonr" Type="Edm.String" Nullable="false"
 MaxLength="18" sap:label="Assignment" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Sgtxt" Type="Edm.String" Nullable="false"
 MaxLength="50" sap:label="Text" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Matnr" Type="Edm.String" Nullable="false"
 MaxLength="18" sap:label="Material" sap:creatable="false"
```

```
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Ebeln" Type="Edm.String" Nullable="false"
 MaxLength="10" sap:label="Purchasing Document"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Fdwbt" Type="Edm.Decimal" Nullable="false"
 Precision="13" Scale="2" sap:label="Planned amount"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Fdtag" Type="Edm.DateTime"
 sap:label="Planning date" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Bldat" Type="Edm.DateTime"
 sap:label="Document Date" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Note" Type="Edm.String" Nullable="false"
 sap:label="Notes for Approval, Reject or Return"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="IUserName" Type="Edm.String" Nullable="false"
 MaxLength="12" sap:label="User Name" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="IKostl" Type="Edm.String" Nullable="false"
 MaxLength="10" sap:label="Centro di Costo"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Xblnr" Type="Edm.String" Nullable="false"
 MaxLength="16" sap:label="Reference" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="Mwsts" Type="Edm.Decimal" Nullable="false"
 Precision="13" Scale="2" sap:label="Tax Amount"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Nebtr" Type="Edm.Decimal" Nullable="false"
 Precision="13" Scale="2" sap:label="Net Amount"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="Rejected" Type="Edm.String" Nullable="false"
 MaxLength="8" sap:label="Rejected?" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
<Property Name="MapsLocalization" Type="Edm.String"
 Nullable="false" sap:label="Maps Localization"
 sap:creatable="false" sap:updatable="false"
 sap:sortable="false" sap:filterable="false"/>
<Property Name="FileBinary" Type="Edm.Binary"
 sap:label="File Binary" sap:creatable="false"
 sap:updatable="false" sap:sortable="false"
 sap:filterable="false"/>
```

```
 <Property Name="Url" Type="Edm.String" Nullable="false"
  sap:label="Invoice URL" sap:creatable="false"
  sap:updatable="false" sap:sortable="false"
  sap:filterable="false"/>
</EntityType>
<ComplexType Name="ZRETURN">
 <Property Name="RETURN_VALUE" Type="Edm.String"
  Nullable="false" MaxLength="1" sap:label="Return value"
  sap:creatable="false" sap:updatable="false"
  sap:sortable="false" sap:filterable="false"
  xmlns:sap="http://www.sap.com/Protocols/SAPData"/>
</ComplexType>
<EntityContainer Name="ZMOB_INV_PAY_SRV"
 m:IsDefaultEntityContainer="true">
 <EntitySet Name="InvoiceCollection"
  EntityType="ZMOB_INV_PAY_SRV.Invoice" sap:creatable="false"
  sap:updatable="false" sap:deletable="false"
  sap:pageable="false" sap:content-version="1"
  xmlns:sap="http://www.sap.com/Protocols/SAPData"/>
 <FunctionImport Name="InvoiceApproveReject"
  ReturnType="ZMOB_INV_PAY_SRV.ZRETURN" m:HttpMethod="GET"
  sap:label="Approve or Reject"
  xmlns:sap="http://www.sap.com/Protocols/SAPData">
  <Parameter Name="IBukrs" Type="Edm.String" Mode="In"
   MaxLength="4">
   <Documentation>
    <Summary>Company Code</Summary>
    <LongDescription/>
   </Documentation>
  </Parameter>
  <Parameter Name="IGjahr" Type="Edm.String" Mode="In"
   MaxLength="4">
   <Documentation>
    <Summary>Fiscal Year</Summary>
    <LongDescription/>
   </Documentation>
  </Parameter>
  <Parameter Name="IBelnr" Type="Edm.String" Mode="In"
   MaxLength="10">
   <Documentation>
    <Summary>Document Number</Summary>
    <LongDescription/>
   </Documentation>
  </Parameter>
  <Parameter Name="IApprove" Type="Edm.String" Mode="In"
   MaxLength="1">
   <Documentation>
    <Summary>Approved?</Summary>
    <LongDescription/>
   </Documentation>
  </Parameter>
  <Parameter Name="IReject" Type="Edm.String" Mode="In"
   MaxLength="1">
   <Documentation>
```

```
      <Summary>Rejected?</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
    <Parameter Name="INote" Type="Edm.String" Mode="In">
     <Documentation>
      <Summary>Note</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
   </FunctionImport>
   <FunctionImport Name="InvoiceSaveGeoLocalization"
    ReturnType="ZMOB_INV_PAY_SRV.ZRETURN" m:HttpMethod="GET"
    sap:label="Save Geo Localization"
    xmlns:sap="http://www.sap.com/Protocols/SAPData">
    <Parameter Name="IBelnr" Type="Edm.String" Mode="In"
     MaxLength="10">
     <Documentation>
      <Summary>Document Number</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
    <Parameter Name="IBukrs" Type="Edm.String" Mode="In"
     MaxLength="4">
     <Documentation>
      <Summary>Company Code</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
    <Parameter Name="IGjahr" Type="Edm.String" Mode="In"
     MaxLength="4">
     <Documentation>
      <Summary>Fiscal year</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
    <Parameter Name="IGeoLocalization" Type="Edm.String"
     Mode="In">
     <Documentation>
      <Summary>Geo Localization</Summary>
      <LongDescription/>
     </Documentation>
    </Parameter>
   </FunctionImport>
   <FunctionImport Name="InvoiceSaveFile"
    ReturnType="ZMOB_INV_PAY_SRV.ZRETURN" m:HttpMethod="POST"
    sap:label="Save File"
    xmlns:sap="http://www.sap.com/Protocols/SAPData">
    <Parameter Name="IBelnr" Type="Edm.String" Mode="In"
     MaxLength="10">
     <Documentation>
      <Summary>Document Number</Summary>
      <LongDescription/>
     </Documentation>
```

```
      </Parameter>
      <Parameter Name="IBukrs" Type="Edm.String" Mode="In"
       MaxLength="4">
       <Documentation>
        <Summary>Company Code</Summary>
        <LongDescription/>
       </Documentation>
      </Parameter>
      <Parameter Name="IFIleBinary" Type="Edm.String" Mode="In">
       <Documentation>
        <Summary>FIle Binary</Summary>
        <LongDescription/>
       </Documentation>
      </Parameter>
     <Parameter Name="IGjahr" Type="Edm.String" Mode="In"
      MaxLength="4">
      <Documentation>
       <Summary>Fiscla Year</Summary>
       <LongDescription/>
      </Documentation>
     </Parameter>
    </FunctionImport>
   </EntityContainer>
   <atom:link rel="self"
    href="http://sapnwgateway.altevielab.com:8000/sap/opu/odata/
     sap/ZMOB_INV_PAY_SRV/$metadata"
    xmlns:atom="http://www.w3.org/2005/Atom"/>
   <atom:link rel="latest-version"
    href="http://sapnwgateway.altevielab.com:8000/sap/opu/odata/
     sap/ZMOB_INV_PAY_SRV/$metadata"
    xmlns:atom="http://www.w3.org/2005/Atom"/>
   </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

Appendice D

funzione resizeCanvasImage

```
function resizeCanvasImage(img, canvas, maxWidth,
 maxHeight) {
 var imgWidth = img.width;
 var imgHeight = img.height;
 var ratio = 1, ratio1 = 1, ratio2 = 1;
 ratio1 = maxWidth / imgWidth;
 ratio2 = maxHeight / imgHeight;
 if (ratio1 < ratio2) ratio = ratio1;
 else ratio = ratio2;
 var canvasContext = canvas.getContext("2d");
 var canvasCopy = document.createElement("canvas");
 var copyContext = canvasCopy.getContext("2d");
 var canvasCopy2 = document.createElement("canvas");
 var copyContext2 = canvasCopy2.getContext("2d");
 canvasCopy.width = imgWidth;
 canvasCopy.height = imgHeight;
 copyContext.drawImage(img, 0, 0);
 canvasCopy2.width = imgWidth;
 canvasCopy2.height = imgHeight;
 copyContext2.drawImage(canvasCopy, 0, 0,
  canvasCopy.width, canvasCopy.height, 0, 0,
  canvasCopy2.width, canvasCopy2.height);
 var rounds = 2;
 var roundRatio = ratio * rounds;
 for (var i = 1; i <= rounds; i++) {
  canvasCopy.width = imgWidth * roundRatio / i;
  canvasCopy.height = imgHeight * roundRatio / i;
  copyContext.drawImage(canvasCopy2, 0, 0,
   canvasCopy2.width, canvasCopy2.height, 0, 0,
   canvasCopy.width, canvasCopy.height);
  canvasCopy2.width = imgWidth * roundRatio / i;
  canvasCopy2.height = imgHeight * roundRatio / i;
  copyContext2.drawImage(canvasCopy, 0, 0,
   canvasCopy.width, canvasCopy.height, 0, 0,
   canvasCopy2.width, canvasCopy2.height);
  }
 canvas.width = imgWidth * roundRatio / rounds;
 canvas.height = imgHeight * roundRatio / rounds;
 canvasContext.drawImage(canvasCopy2, 0, 0,
```

```
  canvasCopy2.width, canvasCopy2.height, 0, 0,
  canvas.width, canvas.height);
}
```

Lo scopo di questa funzione è quello di ridimensionare un'immagine alll'interno di un canvas (elemento HTML che permette la manipolazione di immagini).

Bibliografia

[1] Dissertazione di Roy Thomas Fielding
Capitolo 5, "Representational State Transfer (REST)"
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[2] Open Data Protocol
http://www.odata.org

[3] Sap Community Network
Shabarish Vijayakumar, "Let's Talk OData, Shall We? And keep it simple please!"
http://scn.sap.com/community/mobile/blog/2013/10/03/lets-talk-odata-shall-we

[4] Microsoft Developer Network
David Chappell, "Introducing OData"
http://msdn.microsoft.com/en-us/data/hh237663.aspx

[5] Microsoft Developer Network
Chris Sells, Junlin Zheng, "Open Data Protocol by Example"
http://msdn.microsoft.com/en-us/library/ff478141.aspx

[6] Zhihui Yang, Michael Jiang, "Using Eclipse as a Tool-Integration Platform for Software Development", IEEE Software (Volume:24, Issue: 2), ISSN 0740-7459, INSPEC 9356285
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4118656

[7] Sap Community Network
http://scn.sap.com/

[8] OpenUI5 SDK
https://openui5.hana.ondemand.com

[9] NepTune Software
http://neptune-software.com/

[10] Sencha
http://www.sencha.com/

[11] Software Sustainability Institute
Mike Jackson, Steve Crouch, Rob Baxter, "Software Evaluation: Criteria-based Assessment"
http://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf

[12] Software Sustainability Institute
Mike Jackson, Steve Crouch, Rob Baxter, "Software Evaluation: Tutorial-based Assessment"
http://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationTutorial.pdf

[13] Carnegie Mellon University, Software Engineering Institute
Michael S. Bandor, "Quantitative Methods for Software Selection and Evaluation"
http://www.sei.cmu.edu/reports/06tn026.pdf

[14] Politecnico di Torino
M. Morisio, A. Tsoukiàs, "IusWare: a methodology for the evaluation and selection of software products", IEE PROCEEDINGS. SOFTWARE ENGINEERING, Vol.undefined, pp.undefined, ISSN:1364-5080
http://softeng.polito.it/morisio/papers/iusware.pdf