Alma Mater Studiorum-Università di Bologna - Scuola di Scienze

Corso di Laurea in Scienze di Internet

Applicazione web per la gestione del volontariato

Anno accademico 2013-14

Sessione II

Candidato

Diego Cimarosa

_____

Relatore

Prof. Antonio Messina

_____

Tre confetti rossi:

a mia madre Marina,

a mio padre Giovanni,

a mia sorella Antonietta

Due sorrisi infiniti:

a mio figlio Leonardo,

alla pazienza della mia compagna Patrizia

Sommario

Sommario

1 INTRODUZIONE

Un detto popolare molto noto in Italia è il seguente:

"**fare del bene agli asini si ricevono solo calci …**".

Evidentemente, la saggezza popolare, sintetizza in modo colorito e verace, il concetto che prodigarsi per aiutare il prossimo, non sia un'azione molto saggia.

Le esperienze vissute, mi hanno dimostrato, al contrario, che aiutare gli altri è fonte di grande soddisfazione personale; per questo modo di intendere la vita e per varie altre ragioni, sono entrato, diversi anni fa, in un'organizzazione di volontariato con sede a Bologna, denominata "*Corpo Provinciale Guardie Ecologiche Volontarie*".

Nel corso di circa 10 anni ho partecipato a svariate attività di formazione, intervento, educazione, controllo, vigilanza e solidarietà e, oltre a i profondi aspetti umani nei quali sono stato coinvolto, ho evidenziato alcune problematiche legate alla gestione di queste attività.

Per questo motivo ho cercato una *soluzione informatica* che si è concretizzata in questo mio lavoro di tesi di laurea.

1.1 Le ragioni di una scelta

Nell'associazione di cui faccio parte ci sono 450 soci; in quelle collegate a livello regionale, altri 2000 circa. Molte di queste persone hanno passato mesi ad aiutare i terremotati dell'Aquila, a spalare il fango delle alluvioni in Liguria, a spegnere incendi in Sicilia o nelle colline vicino casa; altri hanno sudato (e molto!) per montare le tende per i terremotati dell'Emilia o si sono svegliati presto per controllare il corretto conferimento dei rifiuti nel centro di Bologna e dei comuni limitrofi, per censire quanti cervi corrono liberi nelle nostre montagne, per sanzionare cacciatori e pescatori che non vogliono rispettare le regole, o chi pensa di usufruire di un parco pubblico come discarica. Non so esattamente perché queste migliaia di persone dedicano gran parte del loro tempo libero in simili attività. Forse, perché, molte di loro, essendo pensionate, hanno tempo da impiegare, o forse perché aiutare il prossimo è gratificante. Non si tratta di sicuro di politica o di religione, piuttosto di solidarietà ed empatia umana.

Il valore di un uomo dovrebbe essere misurato in base a quanto dà e non in base a quanto è in grado di ricevere.

"***Albert Einstein*** "

1.2 Il volontariato in Emilia Romagna

Si parla spesso riguardo al volontariato, di “galassia” poiché nessuno sa con esattezza quante siano le *associazioni* che ne fanno parte. Per evitare di sconfinare e perdermi in questo spazio cosmico, ho deciso di limitare il campo di applicazione alle associazioni con sede in Emilia Romagna e che abbiano i seguenti requisiti:

- *Iscrizione al registro del volontariato regionale*
- *Minimo numero di iscritti pari a 10*
- *Attività prevalenti nei settori ambientali e protezione civile*
- *Soci in possesso di decreto prefettizio come Guardia Giurata*

L’ultimo punto, in particolare, è lo sparti acque fra un volontario *normale* ed un volontario che collabora con le istituzioni pubbliche in modo professionale.

La regione Emilia Romagna, da sempre attenta al mondo del volontariato, promulgò nel 1989 una legge che conferisce ad alcune persone, adeguatamente addestrate, poteri e giurisdizione paragonabili a Pubblici Ufficiali con vincoli operativi ben precisati.

I termini usati in questo contesto sono quelli di “Guardia Ecologica Volontaria o Guardia Ambientale Volontaria”.

A seguito di questa iniziativa, altre 13 regioni italiane Abruzzo, Calabria, Campania, Lazio, Liguria, Lombardia, Marche, Molise, Piemonte, Puglia, Sicilia, Toscana ed Umbria hanno promulgato leggi simili riconoscendo a dei privati cittadini, in determinati contesti operativi, lo status giuridico di Pubblico Ufficiale.

Questo ha portato nel corso degli anni ad una stretta collaborazione fra le varie associazioni e gli Enti Pubblici, che impiegano, per mezzo di **convenzioni** che prevedono solo rimborsi spese, alcune migliaia di persone in svariati ambiti di sorveglianza e controllo del territorio.

Tengo a precisare che un **volontario**, per definizione, è una persona che non percepisce compensi “diretti” per l’opera che presta e che l’associazione di cui fa parte può concordare con le Amministrazioni Pubbliche dei rimborsi per le spese effettuate quali, ad esempio, il consumo di benzina per i servizi di vigilanza effettuati.

1.3 La Legge regionale 23 del 3/7/1989

Riporto per chiarezza il 1° articolo di questa legge:

ARTICOLO 1

Finalità

1. *La Regione Emilia - Romagna riconosce la funzione del volontariato per la salvaguardia dell'ambiente e ne favorisce lo sviluppo per le seguenti specifiche finalità:*

- *Diffondere la conoscenza ed il rispetto dei valori ambientali;*
- *Concorrere con le istituzioni pubbliche alla tutela del patrimonio naturale e dell'ambiente.*

2. *A tali fini la Regione promuove la formazione di guardie ecologiche volontarie.*

Seguono altri 12 articoli che dettagliano i compiti operativi, gli obblighi, limiti del mandato ed altro; brevemente:

ARTICOLO 2: organizzazione delle G.E.V.

ARTICOLO 3: compiti

ARTICOLO 4: corsi di formazione ed aggiornamento

ARTICOLO 5: guardie già in servizio

ARTICOLO 6: conferimento dell'incarico (approvazione della Provincia competente e del Prefetto)

ARTICOLO 7: sospensione e revoca dell'incarico

ARTICOLO 8: compiti delle provincie

ARTICOLO 9: attuazione dei programmi (convenzioni)

ARTICOLO 10: doveri delle G.E.V.

ARTICOLO 11: coordinamento regionale

ARTICOLO 12: norme finanziarie

ARTICOLO 13: modifica di norme

(Assemblea legislativa della Regione Emilia-Romagna - sito Demetra 1989)

L'**ambiente** è quindi il fulcro di questa legge: **va tutelato!** Non è un mero esercizio politico o filosofico. Tutti quanti dobbiamo concorrere alla salvaguardia dei luoghi nei quali abitiamo. Purtroppo, come le recenti cronache ci ricordano, questo semplice concetto non è ancora parte

integrante della nostra società. Fra i tanti non esaltanti primati, quello del numero di frane per abitanti è senz'altro il peggiore[1].

L'Italia è un paese meraviglioso ma fragile, troppo fragile. Troppi decenni di speculazioni, malaffare o solo miopia nella programmazione hanno portato l'intero territorio ad una criticità ambientale che non può più essere trascurata: la tabella seguente è una fotografia (purtroppo ampiamente superata dalla realtà) di questa situazione.

TABELLA 3.1. – I FENOMENI FRANOSI NELLE REGIONI

	Fenomeni Franosi			
	Totale	di cui con danni segnalati		**Indice franosità**
		v.a.	%	
Piemonte	35.023	3.443	9,8	9,1
Valle d'Aosta	4.359	3.050	70,0	16,0
Lombardia	130.538	4.100	3,1	13,9
Trentino	11.380	6.255	55,0	9,9
Veneto	9.476	6.833	72,1	1,2
Friuli Venezia Giulia	5.253	1.996	38,0	6,5
Liguria	7.515	195	2,6	7,9
Emilia Romagna	70.037	2.863	4,1	11,4
Toscana	29.208	3.919	13,4	4,5
Umbria	34.545	721	2,1	7,7
Marche	42.522	320	0,8	19,4
Lazio	10.548	729	6,9	2,0
Abruzzo	8.493	8.220	96,8	11,4
Molise	22.527	4.262	18,9	11,1
Campania	23.430	n.d.	n.d.	7,1
Puglia	843	609	72,2	0,4
Basilicata	9.004	n.d.	n.d.	3,0
Calabria	9.417	5.071	53,8	5,5
Sicilia	3.657	3.224	88,2	1,9
Sardegna	1.523	838	55,0	0,8
ITALIA	**469.298**	**56.648**	**12,1**	**6,6**

(Centro Studi Consiglio Nazionale Geologi 2007)

[1] (Corriere della Sera 2014)

1.4 Compiti delle Guardie Ecologiche

Come riportato all'articolo 10 comma 1 della L.R. 23/89, le G.E.V. devono seguire i programmi di lavoro stabiliti dalle Provincie nonché dalle convenzioni stipulate con Enti Pubblici. In sostanza, i compiti principali riguardano:

- Promuovere e diffondere informazioni in materia ambientale
- Accertare violazioni in materia di protezione del patrimonio naturale e dell'ambiente, di protezione della fauna selvatica, di esercizio della caccia e della pesca
- Collaborare con enti pubblici nella vigilanza in materia di inquinamento idrico, di smaltimento dei rifiuti, di escavazioni di materiali litoidi e di polizia idraulica, di tutela del patrimonio naturale e paesistico, di difesa dagli incendi boschivi
- Collaborare con le autorità in caso di pubbliche calamità e di emergenza di carattere ecologico

Le aree tematiche trattate comportano normative complesse anche per professionisti del settore e piuttosto ardue per dei semplici volontari. A scopo didattico e per facilitare questi compiti, la regione Emilia Romagna ha approntato alcuni documenti essenziali:

- Raccolta di leggi per i corsi di formazione delle Guardie Ecologiche Volontarie (Regione Emilia Romagna 2010)
- *PRONTUARIO delle violazioni accertabili dalle GUARDIE GIURATE ECOLOGICHE VOLONTARIE* (Regione Emilia Romagna 2011)

Da quanto finora esposto si inizia a capire che l'organizzazione di un'associazione o meglio di un raggruppamento di associazioni con le finalità ed i compiti fino a qui visti, non è esattamente banale.

Ovviamente, nel corso degli anni, ogni associazione ha cercato di automatizzare al meglio il flusso di informazioni necessarie alla propria attività istituzionale ma quello che è mancato fino ad oggi è uno strumento che faciliti lo scambio di informazioni fra le associazioni stesse, gli Enti Pubblici ed i privati cittadini interessati a questi argomenti.

Da queste riflessioni è nato **GesVol**.

1.5 Aspetti legali

Le Guardie Ecologiche Volontarie sono *viste* dalla legislazione nazionale come privati cittadini *incaricati di pubblico servizio* che svolgono la loro attività principalmente di concerto ad Enti Pubblici. La norma di riferimento è l'art. 358 del Codice Penale:

"Agli effetti della legge penale, sono incaricati di un pubblico servizio coloro i quali, a qualunque titolo, prestano un pubblico servizio.

Per pubblico servizio deve intendersi un'attività disciplinata nelle stesse forme della pubblica funzione, ma caratterizzata, dalla mancanza dei poteri tipici di quest'ultima, e con esclusione dello svolgimento di semplici mansioni di ordine e della prestazione di opera meramente materiale."

Alcune regioni, però, Emilia Romagna, Piemonte e Lombardia in primis, hanno paragonato le Guardie Ecologiche Volontarie a dei Pubblici Ufficiali a tutti gli effetti poiché hanno fatto riferimento nella loro legislazione regionale, al dispositivo dell'art. 357 del c.p.:

"Dispositivo dell'art. 357 Codice Penale

Fonti → Codice Penale → LIBRO SECONDO - Dei delitti in particolare → Titolo I - Dei delitti contro la personalità dello Stato (Artt. 241-313) → Capo III - Disposizioni comuni ai capi precedenti

Agli effetti della legge penale, sono pubblici ufficiali [2] coloro i quali esercitano una pubblica funzione legislativa, giudiziaria o amministrativa.

Agli stessi effetti è pubblica la funzione amministrativa disciplinata da norme di diritto pubblico e da atti autoritativi, e caratterizzata dalla formazione e dalla manifestazione della volontà della pubblica amministrazione o dal suo svolgersi per mezzo di poteri autoritativi o certificativi[3]. "

La differenza non è di poco conto: come Pubblici Ufficiali possiamo procedere all'identificazione dei trasgressori chiedendo l'esibizione di documenti e compiere vere e proprie attività di indagine **finalizzate** all'accertamento del/dei responsabili del dolo commesso.

Nelle varie associazioni e raggruppamenti questo aspetto è sempre al centro di dibattiti che definire "accesi" è un vero eufemismo. D'altronde se il Comune di Bologna firma una convenzione che coinvolge pesantemente le Guardie Ecologiche, prima dell'informazione, poi nell'educazione

[2] Esempi di pubblici ufficiali sono l'ufficiale giudiziario, il consulente tecnico, il testimone, l'assistente universitario, l'ispettore sanitario di un ospedale, i membri della commissione edilizia comunale, il portalettere, i carabinieri ed gli agenti di Pubblica Sicurezza, il geometra tecnico dell'ufficio comunale.

[3] A differenza delle funzioni legislativa e giurisdizionale che non danno luogo a problemi identificativi, la pubblica funzione amministrativa è stata la centro di numerosi dibattiti soprattutto al fine di costituirne una definizione in grado di specie per distinguerla con certezza dalle attività rientranti nel concetto di servizio pubblico. Tuttavia i parametri cui tale comma fa riferimento non sono scevri di problematiche, soprattutto per quanto riguarda i concetti di ordine pubblico e atto autoritativo, non pacifici in dottrina.

dei cittadini in merito al conferimento dei rifiuti urbani ed infine nelle sanzioni ai trasgressori con una "produzione" di circa 1500 verbali, diventa difficile, per l'Autorità Pubblica non riconoscerci questo ruolo. Altrimenti, come avremmo potuto elevare questi (ed altre migliaia) di verbali?

Un altro aspetto *spinoso* riguarda la *legge sulla privacy*[4] che è molto sentito nelle nostre associazioni in quanto operiamo spesso con e per i bambini. È normale, in queste attività chiedere ai genitori del minore di firmare una *liberatoria* per eventuali danni subiti e/o causati dal proprio figlio ed il *consenso al trattamento dei dati personali*.

Il D.lg. 30 giugno 2003 n. 196, è comporto da 183 articoli per un totale di 51 pagine a stampa; da notare che il termine "fotografia" non viene mai citato.

Eppure è essenzialmente l'uso delle foto e la possibilità di rendere pubblicarle immediatamente a generare discussioni interminabili fra i soci.

Queste brevi considerazioni sugli aspetti legali si concretizzano *informaticamente* nei **verbali di ispezione, nei verbali sanzionatori** e nella **modulistica** relativa al consenso per il trattamento dei dati personali e nella **liberatoria** di responsabilità durante lo svolgimento delle varie attività e che verranno esaminate più avanti.

[4] Vedi http://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/export/1311248

1.6 Il ruolo di Internet nel volontariato

A scopo ludico ho effettuato alcune semplici ricerche per provare a capire quanto sia diffuso in Internet il termine “volontariato”. A titolo di paragone ho incluso le ricerche di alcuni uomini politici ed altro.

Termine	**# pagine (Google)**
volontariato basilicata	583.000
volontariato molise	601.000
volontariato marche	617.000
volontariato umbria	634.000
volontariato emilia romagna	662.000
volontariato abruzzo	675.000
volontariato puglia	688.000
volontariato friuli venezia giulia	713.000
volontariato valle d'aosta	716.000
volontariato campania	735.000
volontariato sicilia	771.000
volontariato piemonte	829.000
volontariato lazio	954.000
volontariato trentino alto adige	1.030.000
volontariato calabria	1.420.000
volontariato	1.520.000
volontariato veneto	1.660.000
volontariato sardegna	7.610.000
volontariato italia	***9.860.000***
elenco associazioni volontariato	*707.000*
applicazioni per volontariato	*385.000*
Silvio Berlusconi	*2.250.000*
Beppe Grillo	*15.600.000*
Matteo Renzi	*23.100.000*
Vladimir Putin	*99.300.000*
Barack Obama	*205.000.000*
Google	*128.000.000*
Volunteering	***178.000.000***
Microsoft	***1.290.000.000***

Se il termine “Microsoft” è indubbiamente il più referenziato in questa ricerca e se il presidente Barack Obama distanzia ampiamente tutti gli altri non mi stupisce particolarmente. La lingua franca di Internet è ovviamente l’inglese per cui, Microsoft e Obama, sono sicuramente avvantaggiati a prescindere dalla loro “importanza”. Meno evidente il fatto che una ricerca come

“volontariato italia” produca più risultati di “Silvio Berlusconi” ed ancora, se cerchiamo “***volunteering”*** otteniamo più risultati di “google”.

Evidentemente il tema del **volontariato** è di sicura importanza in tutto il mondo.

1.7 Siti web volontariato

In questa sezione elenco alcuni fra i principali siti Internet di livello nazionale, con una breve descrizione delle caratteristiche e tecnologie utilizzate:

- www.protezionecivile.gov.it

  Sicuramente il punto di riferimento per tutti i volontari di Protezione Civile. Grafica moderna, home page a 4 sezioni; selezione italiano/inglese; ricerca e **droplets** per ler news. CMS Wordpress con estensioni **motools** ed altro. Rank 125.400 (world) e 2566 (Italia) (Alexa).

  Non ha sezioni riservate e non sembra particolarmente ottimizzato.

- http://www.serviziocivile.gov.it/

  Gestito dall’Ufficio per il Servizio Civile Nazionale, raccoglie informazioni per l’accesso ai bandi ed alla banca dati Helios per la gestione dei volontari di Servizio Civile. Riporta una pagina con le statistiche Google Analytics su base mensile ed un elenco di siti tematici.

- http://www.assocarabinieri.it/

  Riferimento per gli appartenenti all’Arma e per gli interessati. Fra gli scopi dell’associazione, nata il 1° Marzo 1886 è da segnalare questo: "*promuovere e partecipare – anche costituendo appositi nuclei – ad attività di Volontariato per il conseguimento di finalità assistenziali, sociali e culturali.*". La home page risulta un po’ datata, con una grande foto tematica in intestazione. Le sezioni interne sono più tradizionali con una grafica migliore. Usa un CMS pressoché sconosciuto: *Serendip Portal*. Basso rank internazionale e nazionale.

- http://www.ana.it/

  Sito dell’Associazioni Nazionale Alpini che, sebbene gradevole, ha un menu fluttuante piuttosto fastidioso. Ricco di informazioni ha una certa tendenza alla pubblicità con banner laterali. Usa **jquery** e **dojo** per impaginazione ed animazione. Ha la particolarità

di integrare un calendario Google in home page. Discreto rank internazionale (680.631) e nazionale (12975).

- http://www.anvvfc.it/

  Associazione nazionale dei Vigili del Fuoco in congedo. Sito "classico" con header stilizzato e loghi, menu orizzontale semplice e chiaro. Statico, di grafica piacevole e navigazione veloce. Utilizza Google Drive per la condivisione di documenti riservati. Rank di 5.170.320

- http://www.anvvfv.it/

  Associazione Nazionale dei Vigili del Fuoco Volontari: pagine statiche, *linkate* da semplice menu. Area riservata e uso di **jquery** per la gestione *client*. Non risulta particolarmente accattivante. Rank di 11.781.426

- http://www.uniprotezionecivile.it/

  Sito web dell'associazione degli studenti laureati nei Corsi di Laurea in Coordinamento delle Attività di Protezione Civile. Sito moderno con grafica molto curata con tecnologia **WIX** ma con pochi contenuti. Nessun rank disponibile.

- http://www.anpasnazionale.org/

  Sito che riunisce le associazioni di Pubblica Assistenza fondato con l'unità d'Italia nel 1860. La home page ha un menu con 6 sezioni e varie sottosezioni. Webmail ed accesso riservato. Rank 1.602.123 (world) e 51.148 (Italia).

- http://www.agesci.org/home.php

  Probabilmente l'Associazione delle Guide e Scouts Cattolici Italiani, non rientra a pieno titolo nel campo del volontariato comunemente inteso, ma voglio citare questo sito, perché ho potuto verificare personalmente, in occasione del terremoto di Crevalcore del 2012, con quale serietà ed impegno quei ragazzi ci aiutavano nella gestione del campo. La home page del sito ha una grafica leggera e gradevole e spicca il menu a sezioni colorate "Lupetti-Coccinelle", "Esploratori-Guide", "Rover-Scolte" e "Capi". **PHP** lato server e **jquery** su client. Sito tradizione a 3 colonne con buon rank: 686.630 (world) e 14.561 (Italia).

2 ANALISI DEI REQUISITI

Dopo questa veloce introduzione, passo ad esaminare più nel dettaglio il problema della "**Gestione dei Volontari**" riportando le esigenze di segreteria ed operative.

Come riportato in "Compiti delle Guardie Ecologiche" a pag. 5

A. Promuovere e diffondere informazioni in materia ambientale

   1. Rientrano in questa categoria le informazioni fornite dalla Guardia in servizio nel territorio di competenza. Ad esempio: illustrare ai cittadini le norme di comportamento in un parco pubblico, la distribuzione di brochure tematiche, metodologie per la corretta potatura di un albero, norme di comportamento per il corretto conferimento nella raccolta differenziata dei rifiuti ecc.
   2. Attività didattiche nelle Scuole pubbliche o assimilate rivolte essenzialmente ai bambini delle elementari su temi quali "Il riciclo di rifiuti", "L'acqua bene prezioso" e simili. Queste attività richiedono un'attenta pianificazione e progettazione concordata con i dirigenti scolastici.
   3. Attività di laboratorio a partecipazione pubblica totalmente gratuite o con richiesta rimborsi per materiale ed altro finalizzate all'educazione ambientale dei bimbi in età scolare. Es.: "La zolla di terra", "Il Microscopio e la goccia d'acqua", "Le tracce degli animali", "Caccia alla foglia", "Costruiamo un acquedotto". Si tratta, in sostanza, di creare dei giochi di ruolo che riescano a coinvolgere il più possibile i bambini per trasmettere i concetti fondamentali di ambiente ed ecologia.
   4. Attività educative per adulti: passeggiate nei parchi, escursioni guidate a siti di interesse ambientale
   5. Partecipazione ad eventi/fiere/manifestazioni con stand e banchetti per distribuzione materiale informativo e raccolta adesioni

Possiamo evidenziare questi concetti: **associazione, norma, guardia, cittadino, attività, ente, sito, documento, evento, servizio, informazione** e quindi:

- Il **cittadino** *richiede* alla **guardia** una certa **informazione =>**
- <= La **guardia** *fornisce* **l'informazione** richiesta eventualmente consultando la **norma** e la **documentazione** particolareggiata che può essere fornita o meno dall'**ente** di riferimento per il **sito** in esame

- L'**associazione** *propone* un'**attività** ad un **ente**/altra **associazione** (e viceversa) ⇔
- <= La **guardia** svolge **l'attività** *proposta/richiesta* presso la sede dell'**ente** con fornitura di **documentazione** adeguata. L'**associazione** registra questa **attività** come **servizio** che, in base agli accordi con l'**ente** (terzo o primario) può comportare **rimborsi** o meno

- L'**associazione** *pubblica* l'**attività** di formazione/educazione ambientale e *raccoglie adesioni* dei **cittadini** =>
- <= Il **cittadino** *aderisce/partecipa* all'**attività** proposta

B. Accertare violazioni in materia di protezione del patrimonio naturale e dell'ambiente, di protezione della fauna selvatica, di esercizio della caccia e della pesca

   Si tratta di rilevare non conformità rispetto alla normativa, di individuare il/i responsabile/i e di sanzionare con le modalità definite dalla norma stessa. Questo è il verso servizio di "guardia": controllare, direttamente od indirettamente, un certo territorio verificando il comportamento dei cittadini ed in molti ambiti anche quello delle imprese. Rientrano in questo ambito:

   1. Servizi di vigilanza ambientale nei parchi pubblici
   2. Servizi finalizzati al controllo nei periodi di raccolta funghi e tartufi
   3. Come sopra per la caccia
   4. Come sopra per la pesca
   5. Come sopra per il conferimento rifiuti in ambito urbano
   6. Come sopra per spandimento materiali organici, e controllo zone agricole
   7. Sanzionare comportamenti non conformi previo accertamento dell'identità del colpevole che può comportare la richiesta di esibizione dei documenti o la ricerca presso enti pubblici.

Data la complessità della normativa vigente, per svolgere efficacemente i servizi elencati, occorrono volontari "specializzati". Mi sembra improbabile che una persona sola conosca in modo approfondito le delibere della giunta regionale, provinciale o comunale in ambito, ad esempio, di pesca sia i regolamenti in ambito di prevenzione incendi o di spandimenti agricoli. Per ovviare a queste problematiche, molte associazioni hanno realizzato dei *prontuari* normativi per settore ed un problema che le accomuna tutte è la completezza ed aggiornamento continuo e soprattutto la fruibilità "sul campo". Chi ad esempio, si ricorda di quanti millimetri devono essere le maglie di una bilancella per la pesca in acque classificate "B"? O quanto costa il tesserino per la raccolta funghi semestrale?

I concetti da evidenziare sono: **normativa, sanzione, guardia, controllo, territorio, cittadino, impresa, calendario, specie, verbale** dai quali:

- La **guardia** sulla base della **normativa** e del calendario predisposto dall'autorità pubblica effettua **servizi** per *controllare* **cittadini e le imprese** che agiscono sul

territorio e **sanziona** coloro che violano la **normativa** stessa *compilando* i **verbali di ispezione e sanzione**

C. Collaborare con enti pubblici nella vigilanza in materia di inquinamento idrico, di smaltimento dei rifiuti, di escavazioni di materiali litoidi e di polizia idraulica, di tutela del patrimonio naturale e paesistico, di difesa dagli incendi boschivi

Le associazioni stabiliscono dei legami con gli enti pubblici per il coordinamento delle attività segnalando agli stessi le anomalie riscontrate nei servizi di vigilanza. Questo coordinamento è particolarmente importante nei servizi di difesa dagli incendi boschivi per i quali sono richiesti attestati di abilitazione, idoneità fisica, attrezzature adeguate e continui aggiornamenti ed *esercitazioni*. Le regioni, e soprattutto le provincie, svolgono ruoli molto importanti, che possono fare la differenza fra territori martoriati da incendi (Liguria, Sicilia, Sardegna e Puglia) ed altri meno soggetti a disastri di natura dolosa o naturale (Emilia, Veneto, Friuli). Evidentemente, laddove il soggetto pubblico investe denaro, competenze e risorse nel coinvolgimento dei cittadini-volontari, i risultati non deludono.

Concetti da evidenziare: **associazioni, legami, enti, coordinamento, attività, segnalazioni, incendi, attrezzature, abilitazioni, attestati, esercitazioni.**

D. Collaborare con le autorità in caso di pubbliche calamità e di emergenza di carattere ecologico

In seguito al terremoto che ha colpito la nostra regione nel 2012, ho potuto constatare personalmente, la complessità di un Sistema di Protezione Civile e le difficoltà nella gestione delle strutture e delle persone coinvolte in simili eventi. La legge quadro nazionale della Protezione Civile n. 225/92[5] stabilisce quali siano le strutture operative del Servizio Nazionale di Protezione Civile e le competenze dei vari organi dello stato che a loro volta legiferano nel dettaglio. È proprio questa organizzazione, a mio avviso, a procurare attriti ed incomprensioni fra i molti soggetti coinvolti. Mi è capitato personalmente, nel campo terremotati di Crevalcore di assistere ad una discussione piuttosto vivace fra il Capo della Protezione Civile, il Dott. Gabrielli e l'allora sindaco, sig. Broglia riguardo alla recinzione dell'area sportiva nel quale era stato installato il campo terremotati. Se ricordo bene, il dott. Gabrielli richiedeva che l'intera zona dovesse essere recintata ma il sindaco si opponeva perché riteneva un peso aggiuntivo privare dell'uso di aree sportive buona parte della popolazione di Crevalcore che non era stata danneggiata dall'evento sismico.

Cito questo episodio solo per spiegare meglio la complessità della gestione di un evento di tale portata.

Di conseguenza, non ritengo fattibile, da una sola persona, anche solo un'analisi di massima propedeutica alla realizzazione di un software gestionale. Per meglio rendere l'idea riporto alcuni concetti che dovrebbero essere esaminati a fondo per una corretta analisi:

[5] Vedi http://www.protezionecivile.it/cms/attach/editor/225_1992.pdf

Fornitori, magazzino, attrezzature, impianti elettrici, tende, guardiola, tesserini riconoscimento, impianti fognari, impianti idraulici, montaggio e smontaggio, mensa, sorveglianza …

Un campo terremotati nel quale alloggiano 600/700 persone è a tutti gli effetti un piccolo paese.

C'è qualche software in grado di gestirlo?

Pertanto non tratterò in questo elaborato questo problema.

2.1 Soggetti partecipanti

Dalla macro analisi finora svolta evidenziamo questi soggetti:

- **associazione**
- **cittadino**
- **coordinamento**
- **ente**
- **guardia**
- **impresa**
- **specie**

Trattandosi di un'***applicazione web***, manca l'*entry point* nel sistema, ovvero, l'**utente** che deve essere differenziato fra un **utente web**, cioè un semplice visitatore del *sito* ed un **utente di sistema**, ovvero la persona fisica o giuridica che accede alle funzionalità *riservate* dopo essersi registrato (o essere stato registrato) ed aver fornito le proprie credenziali**.**

Molte attività umane sono strutturate secondo un modello gerarchico, nel quale, ad ogni livello sono associati particolari compiti e privilegi. Ad esempio: nell'antico Egitto, una moltitudine di schiavi, coordinati da un élite di funzionari, rese possibile ad una sola persona, il faraone, il riposo eterno in tombe passate alla storia come piramidi.

Il mondo del volontariato non è certo esente da questo paradigma. Anzi, tende ad estenderlo in modo vistoso nel senso che, per motivare sufficientemente un cittadino in compiti di pubblica utilità, serve, oltre ovviamente alla sua disponibilità, una qualche forma di incentivo che spesso si concretizza in *responsabilità* all'interno dell'organizzazione. Da qui, un organigramma che può risultare anche piuttosto complesso.

Nella pagina seguente riporto una gerarchia di *attori* reale, facente capo, cioè ad una particolare associazione di volontariato.

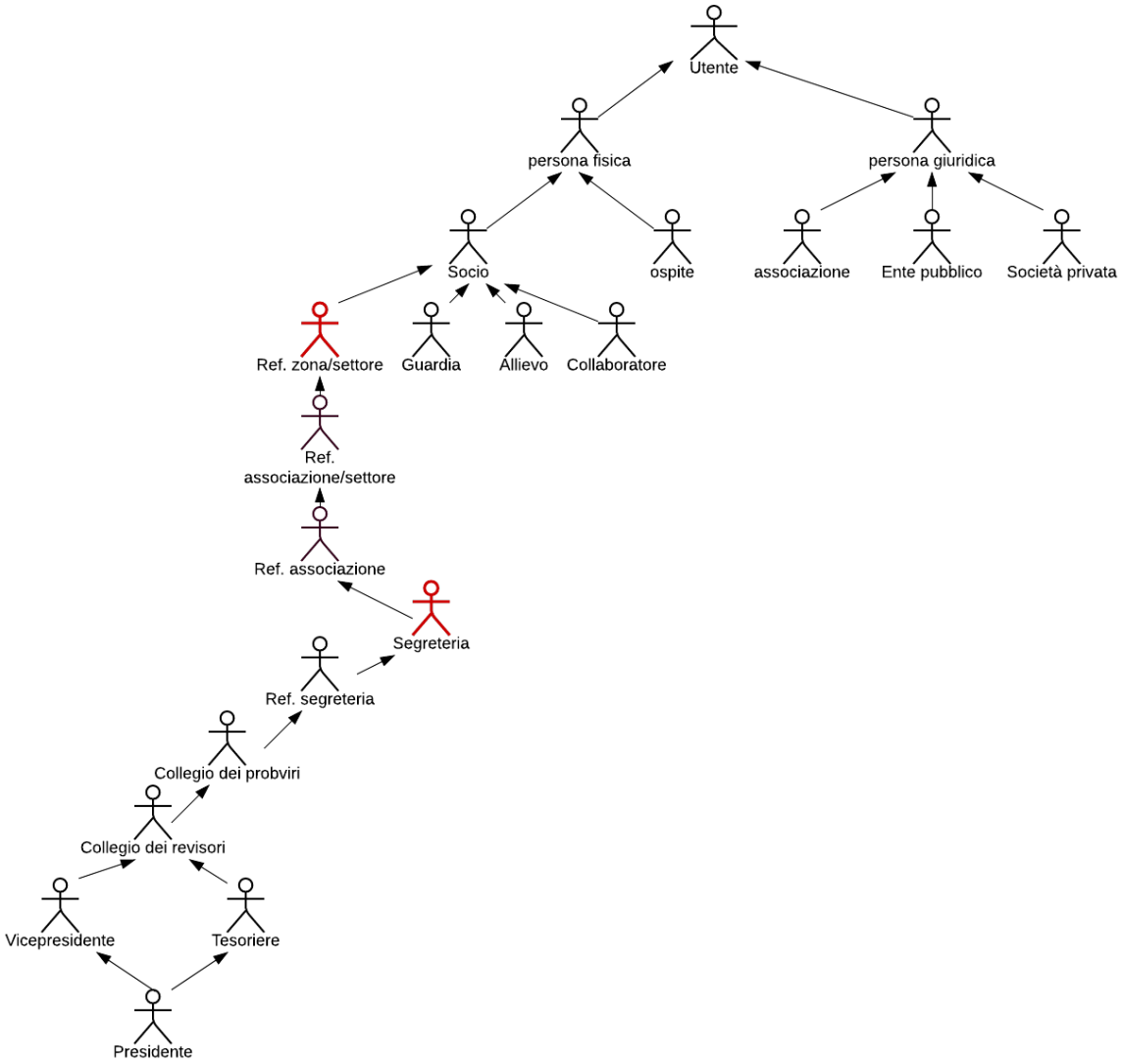


Figura 1 - Diagramma attori CPGEV

Ognuno di questi *soggetti*, o, più propriamente *attori*, interagisce col sistema con i cosiddetti *casi d'uso* (***use cases***). Nella figura sopra, gli elementi in rosso rappresentano un gruppo di *attori* che dovrebbe essere "esploso" per il dettaglio. Nella pagina successiva riporto un reticolo di competenze in forma testuale tabellare che descrive nel dettaglio l'organizzazione dell'associazione.

Tabella 1 – Matrice attività/competenze per zone

Attività	***Zone***							
	Apicale	**Ref. CPGEV**	**Bologna**	**San Lazzaro**	**Casalecchio**	**Imola**	**San Giovanni In Persiceto**	**Montagna**
Animali D'affezione								
ARPA								
Caccia								
Censimenti								
Educazione Ambientale			**X**					
Fauna Minore								
Flora								
Funghi E Sottobosco								
Liquami E Spandimenti						**X**		
Parchi, SIC, ZPS								
Pesca								
Taglio Boschi								
Potature								
Protezione Civile								
Progetto Istrice								
Progetto Pellegrino								
Progetto Lupo								
Rifiuti								
Tartufi								
Automezzi								
Biblioteca								
Convenzioni								
Internet/Gufo								
Inventario								
Segrreteria								
Corsi/Formazione								
Fiere, Manifestazioni	**X**							

Nella realtà, ovviamente, non tutte le caselle fanno riferimento ad un volontario: in diversi casi una persona occupa più ruoli ed in altri il ruolo è ricoperto da più persone. Più avanti riporto il dettaglio di alcune applicazioni e come sono state realizzate in software; in questa introduzione riporto alcuni schemi esemplificativi.

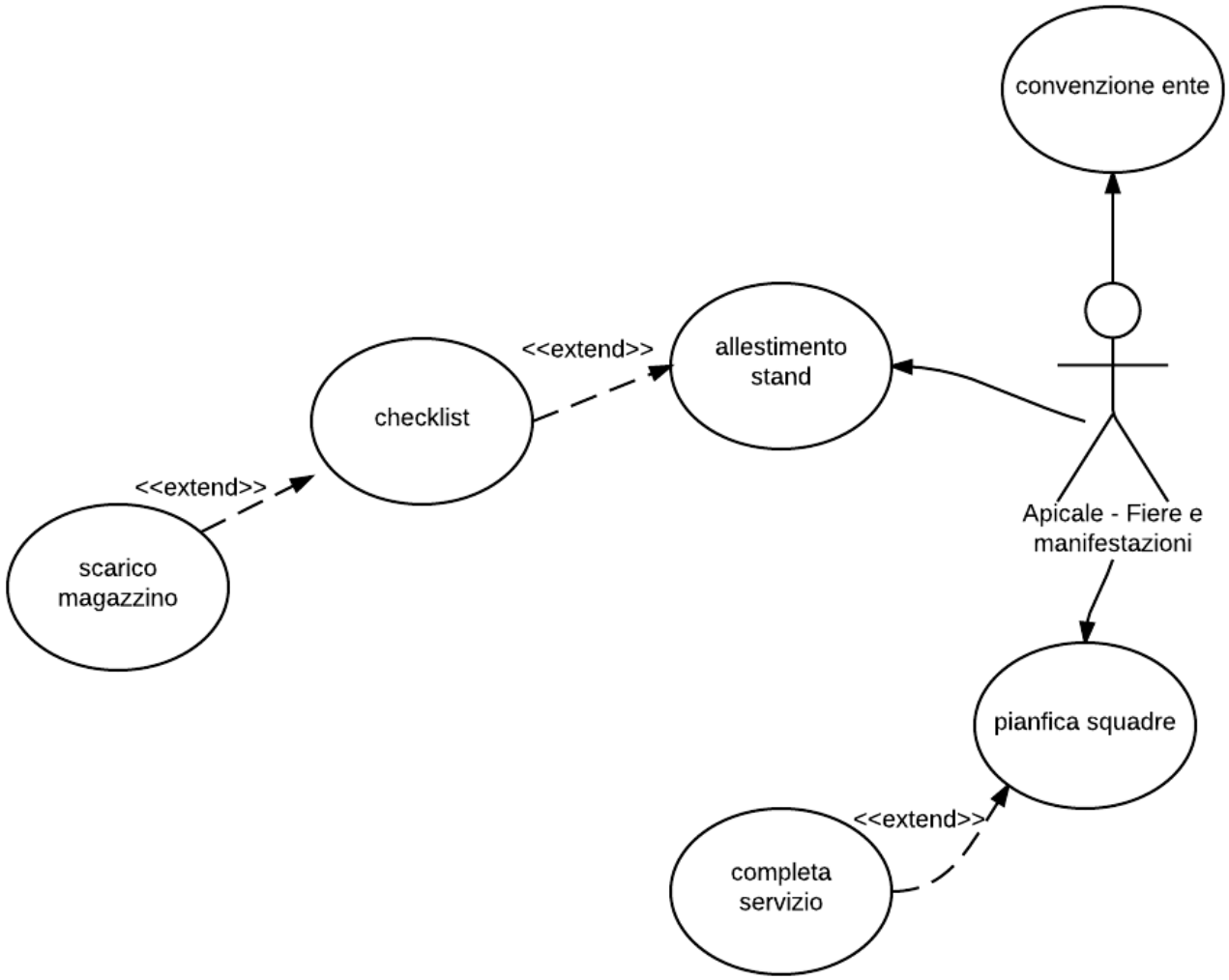


Figura 2 - Use case: Apicale, Fiere e manifestazioni

Il responsabile dell'attività "Fiere e manifestazioni" stipula convenzione con l'Ente di riferimento; pianifica l'attività della squadra/squadre ed allestisce lo stand che richiede una checklist del materiale e l'appropriata gestione del magazzino. A fine attività completa e verifica i servizi dei volontari.

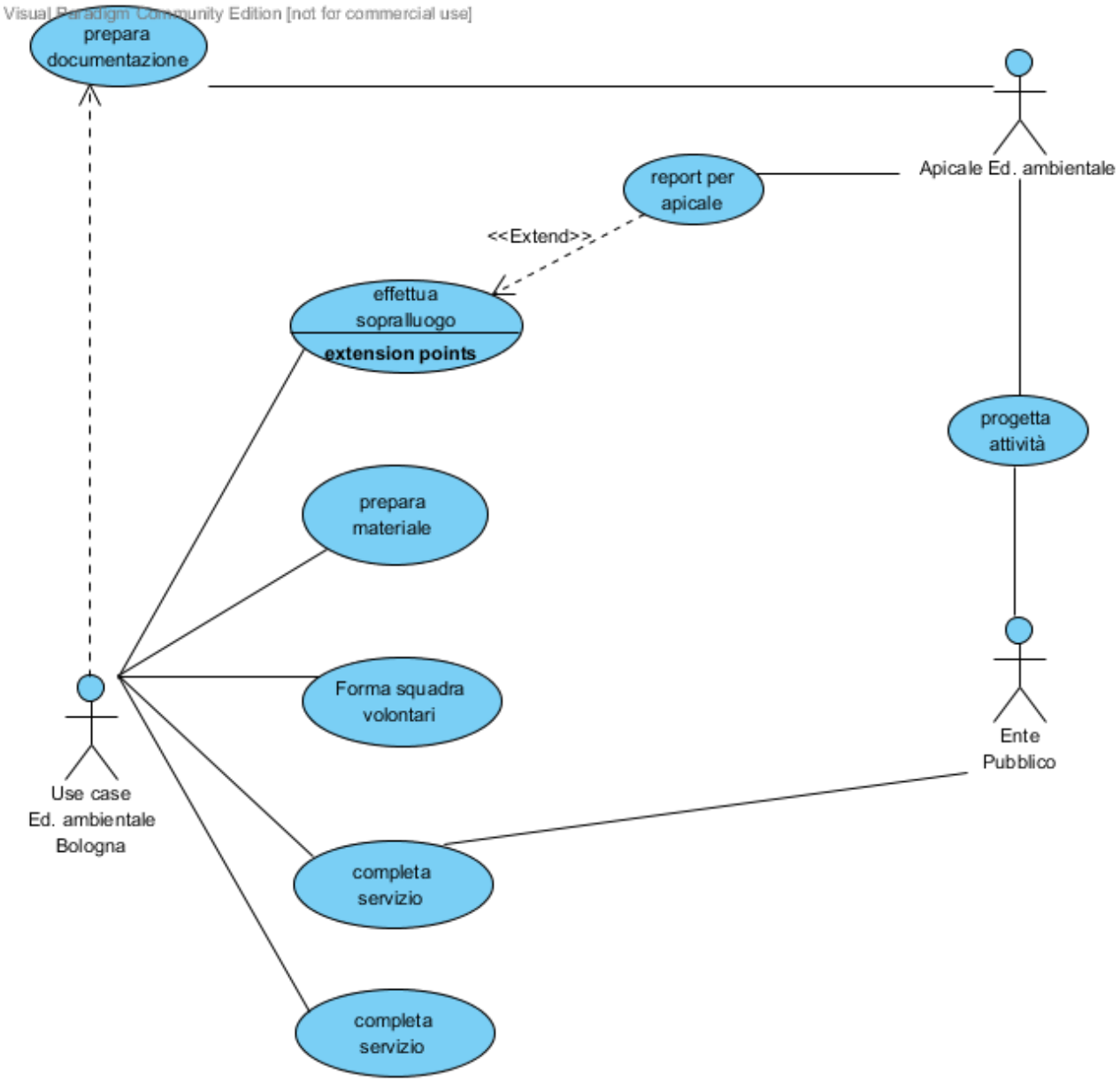


Figura 3 - Use case: Educazione ambientale zona Bologna

Il responsabile dell'attività "Educazione ambientale" propone ad un Ente/viene contattato per un progetto tematico. Il responsabile prepara la documentazione necessaria che viene recepita dal referente della zona Bologna che effettua un sopralluogo di verifica, prepara il materiale necessario, forma una squadra di volontari in base alle competenze richieste, completa/effettua il servizio concordato con l'Ente e completa il servizio a fini interni redigendo un rapporto.

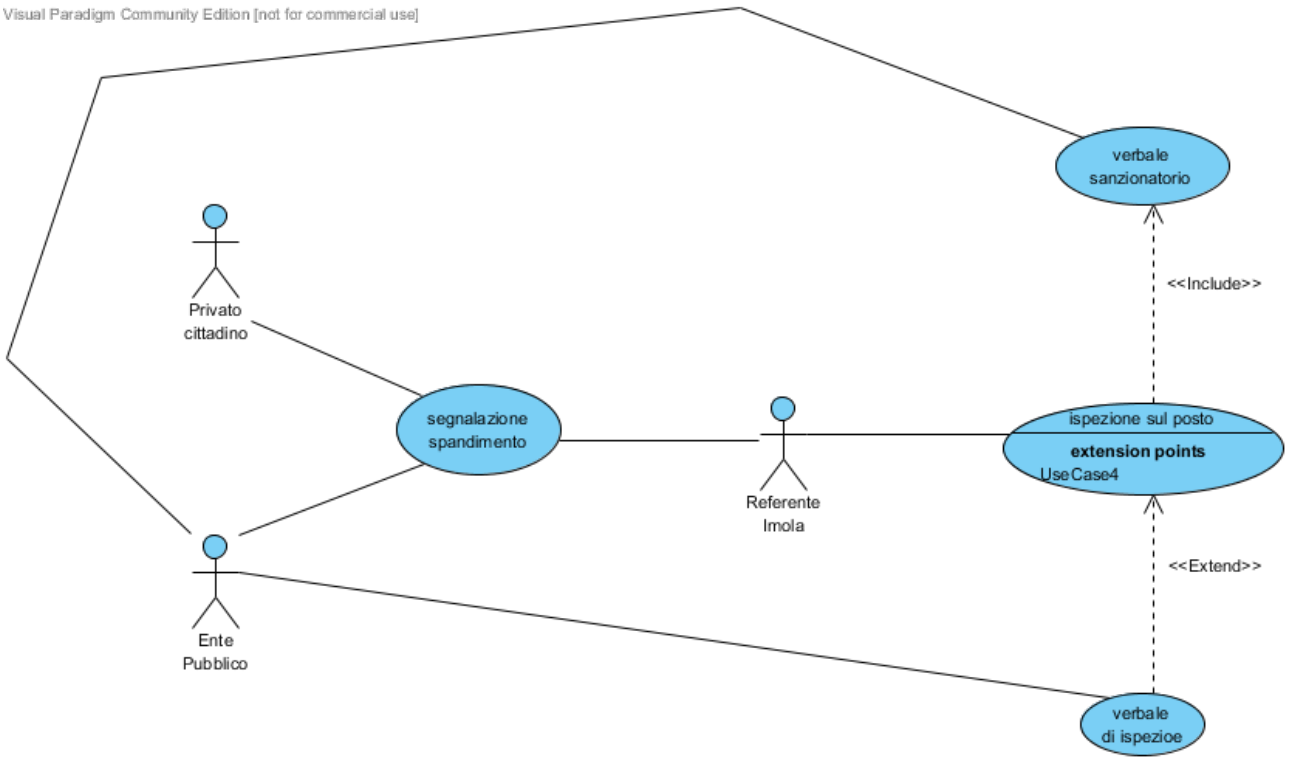


Figura 4 - Use case: Spandimenti di liquami - Zona Imola

Al referente viene recapita una segnalazione da parte di un privato cittadino o di un ente pubblico in merito a spandimenti liquami. Il referente, dopo aver accertato la gravità dell'inquinamento, redige un verbale di ispezione che viene inviato all'Ente di competenza (generalmente il Comune). A seconda della gravità del fatto il referente procede alla compilazione del verbale che viene inoltrato all'Ente per la notifica al trasgressore.

Dopo questi esempi di casi d'uso passo ad esaminare nel capitolo successivo le metodologie di sviluppo.

3 Metodologie di sviluppo

Nei prossimi paragrafi riporto le principali metodologie di sviluppo usate nell'industria del software che, è bene ricordarlo, muoverà il prossimo anno, 2015, a livello mondiale circa **330Mld** di dollari come questa immagine evidenzia:

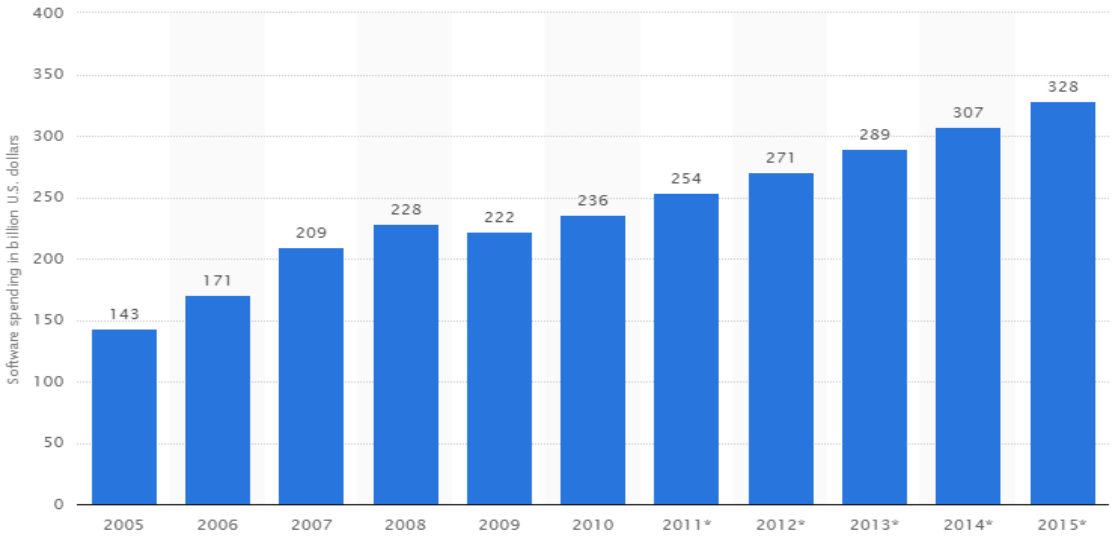


Figura 5 - Trend 2005-2015 spese mondiali software (Statista s.d.)

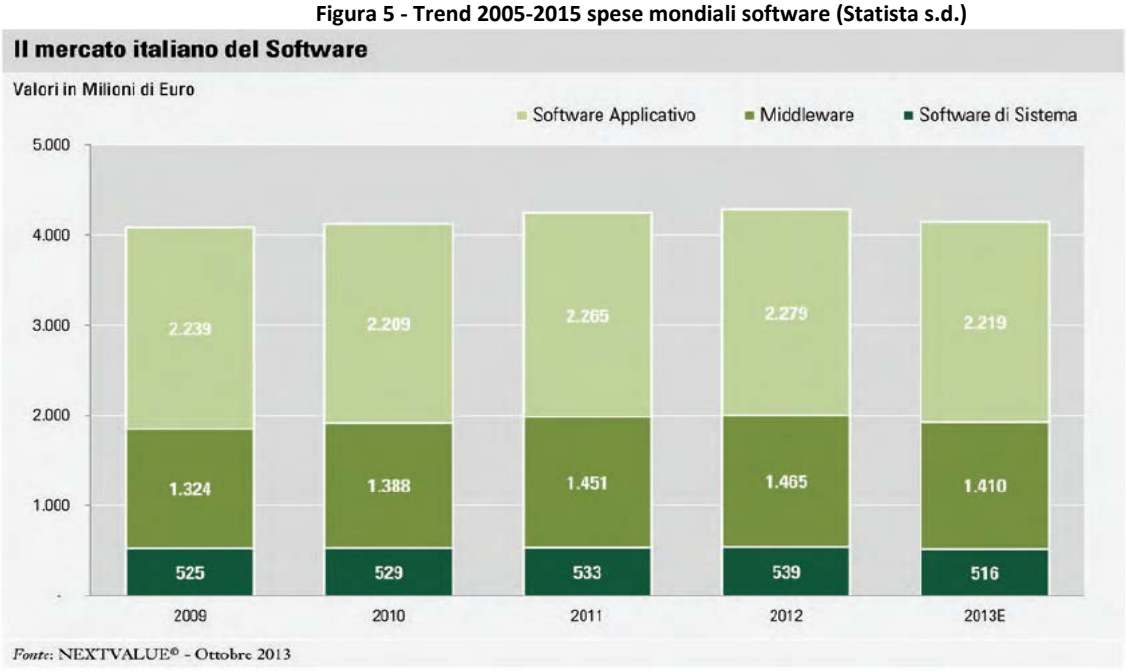


Figura 6 - Mercato del software in Italia – 2013 (Assintel 2013)

Per la realtà italiana, dove il mercato vale circa l'1.26% del globale, si ha comunque una spesa di circa 4Mld di euro, poche nel raffronto, non trascurabile per i nostri standard.

3.1 Processo di sviluppo del software

È l'insieme delle attività che trasformano un'idea per la risoluzione di un problema in un software eseguibile da un computer. Può essere l'estensione di un software già esistente col *riuso* di parti di codice o scritto totalmente da zero.

Nell'ingegneria del software si distinguono tre tipologie:

- *programming in the small* (**PitS**)[6]

  Un solo programmatore che segue il ciclo:

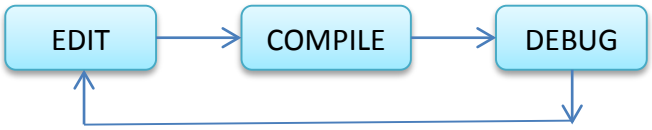


Figura 7 - Ciclo *programming in the small*

  Il più semplice ciclo di costruzione di un software. Normalmente tende ad escludere o a ridurre al minimo la documentazione col risultato di rendere ingestibile la manutenzione

- programming in the large (**PitL**)

  Coinvolge molte persone con compiti differenti. Usa differenti modelli di sviluppo ma è sempre contraddistinto da abbondante produzione di documentazione. Termini importanti in questo contesto sono:

  loc, sloc, kloc, lloc, rispettivamente: *lines of code*, *source lines of code*, Kilo *lines of code*, *logical lines of code*

  La misurazione del numero di line di codice, in modo formalizzato, è stato senza dubbio un valido strumento per misurare la complessità del software, quindi del tempo necessario allo sviluppo ed alfine dei costi. Al crescere delle tecnologie e delle competenze informatiche del team di sviluppo, può essere però fuorviante. Ad esempio: in software che anno dopo anno vengono affinati e migliorati aggiungendo funzionalità riducendo anche il numero di linee di codice[7].

[6] Frank DeRemer; Hans Kron,"*Programming-in-the large versus programming-in-the-small*". "Proceedings of the international conference on Reliable software", 1974, Los Angeles, California: ACM. pp. 114–121. doi:10.1145/800027.808431.

[7] Il software Microsoft sembra non aderire a quanto esposto: Windows, ad esempio, è passato da 5Mln di linee di codice di NT 3.1 a circa 50Mln di Windows 7: il pacchetto Office è passato dalle 60Kloc del 1995 a circa 250Mln di linee della versione 2010.

baseline: specifica (o modulo) che è stato formalmente approvato e che può essere modificato solo con procedure rigorose e formalizzate

release: una *baseline* visibile anche al di fuori del team di sviluppo

- programming in the many (**PitM**)[8]

  Con l'esplosione della diffusione dei terminali portatili, dei tablets, forse degli *smart watches*, *smart tv, fitness trackers* e futuri dispositivi, l'ondata di *Internet Of Things* sta per travolgerci. Solo chi saprà *surfare* e **programmare** (per questi) **molti** dispositivi potrà sopravvivere ed evolvere "la specie". Il *paper* presentato dagli autori citati in nota, focalizza l'attenzione proprio nella sfida di produrre *tools* e metodologie che possano affrontare la crescente varietà di dispositivi connessi che invaderanno il mercato nel prossimo futuro. Non è difficile immaginare automobili che "spontaneamente" trasmettono un'auto-diagnostica all'officina autorizzata, frigoriferi che compilino in automatico la lista della spesa sul telefonino, sensori nei tacchi delle scarpe che avvisino il tuo medico di un possibile problema all'adduttore lungo delle coscia. Droni che ti consegnino al semaforo, mentre sei in coda, il mazzo di rose per tue moglie, che, ovviamente ti sei ricordato solo all'ultimo minuto …

  Qualcuno sorriderà pensando alla fantascienza (un collare che ti porta a spasso da solo il cane …) ma l'Internet delle Cose ci cambierà sul serio la vita.

  A patto che siano disponibili strumenti software per la creazione dei nuovi software.

Nelle pagine seguenti riporto alcuni fra i *modelli di processo* più utilizzati nell'industria del software a titolo di raffronto con questo mio lavoro.

[8] Nenad Medvidovic, Marija Mikic-Rakic, *Programming-in-the-Many: A Software Engineering Paradigm for the 21st Century*, Computer Science Department, L.A.

3.1.1 Modello Waterfall

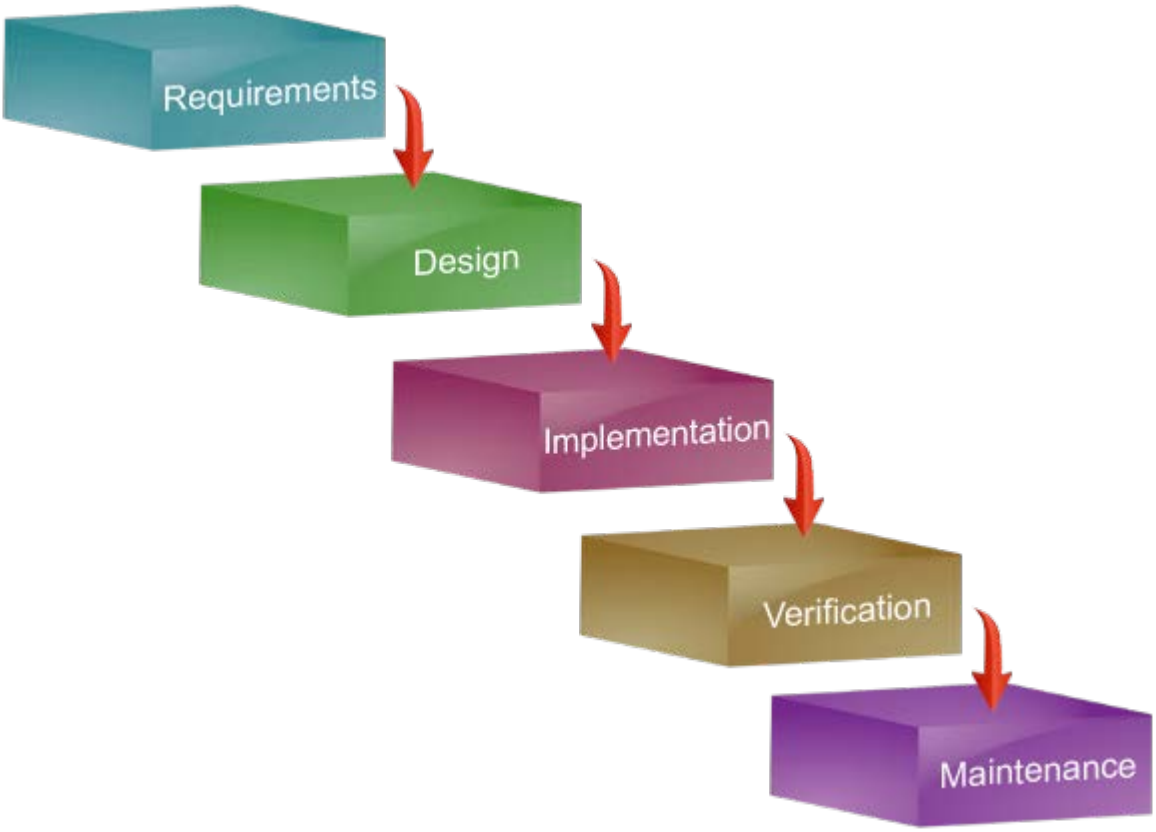


Figura 8 - Il modello di sviluppo "a cascata"

Tra le prime metodologie formalizzate ed impiegate nello sviluppo del software, trae le sue origini dalle metodologie utilizzate nell'industria manifatturiera e delle costruzioni edili dove, *ripensamenti* dopo la conclusione del progetto od in corso d'opera erano praticamente impossibili. Appare per la prima volta nel 1956, presentato ad un simposio su organizzato dalla marina militare USA su "Metodi di programmazione avanzata per computer digitali" (Benington s.d.) anche se la prima descrizione formale spesso citata è del 1970.

Come evidenziato dalla figura, le fasi logiche sono ben distinte ma, come in una cascata l'acqua non può tornare su, un errore o mancanza nelle fasi di analisi dei requisiti, *requirements*, e soprattutto nella progettazione, *design*, può portare al fallimento del progetto.

È piuttosto difficile dire: "Oops! Abbiamo dimenticato il garage!" e costruirlo quando il grattacielo è all'80esimo piano …

Il punto critico di questa metodologia è proprio nel tempo che occorre dedicare alle prime due fasi prima di rendersi conto se il software prodotto sia o meno realmente utilizzabile. Nella realtà, i cambiamenti sono spesso molto più veloci di quello che si possa pensare.

3.1.2 Modello RUP

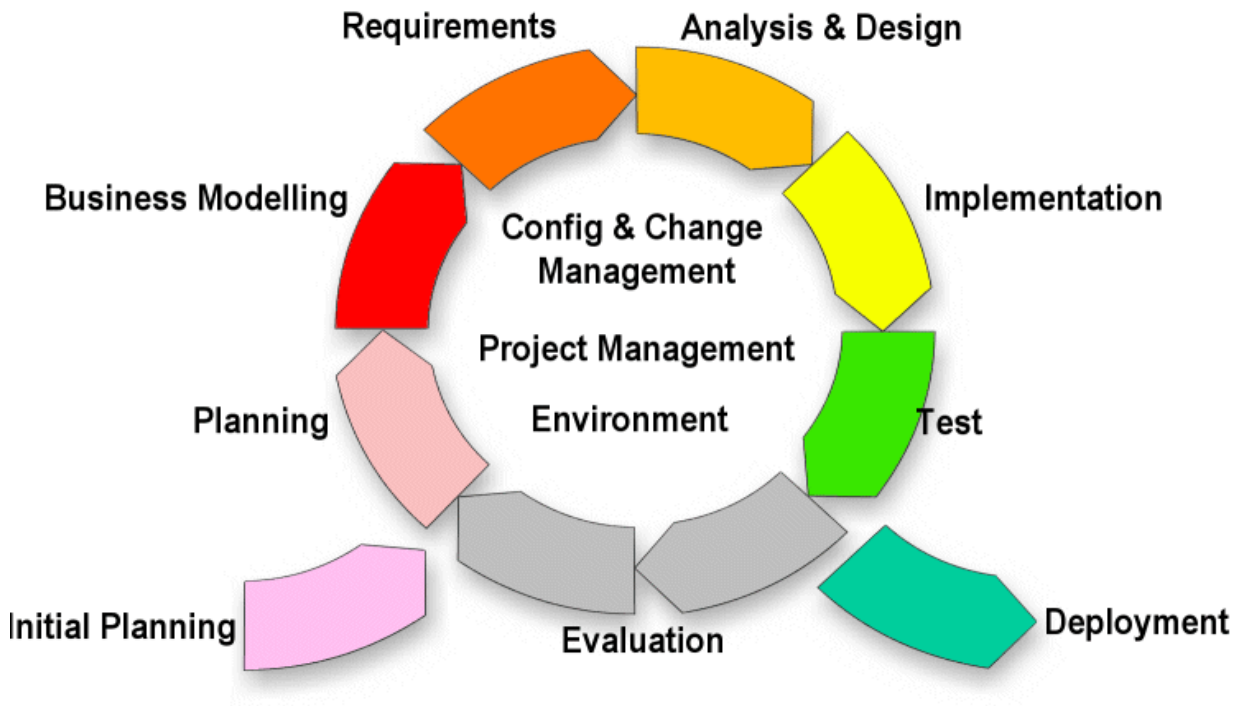


Figura 9 - Il modello di sviluppo Rational Unified Process (RUP)

Inizialmente sviluppato dalla Rational Software Corporation nel 1985, come *framework* di sviluppo integrato per il linguaggio Ada, che, fra i punti di forza prevede la forte tipizzazione delle variabili, si trasformò con la seconda generazione del 1995, per il lavoro di Rumbaugh, Jacobson and Booch, scherzosamente identificata dalla comunità internet come “The Three Amigos" in un vero e proprio “linguaggio di modellazione unificato”, meglio noto come UML, “Unified Modeling Language”.

Il continuo sviluppo di *frameworks* per i moderni e complessi sistemi software portato avanti dalla Rational software Corp. si concretizzò alla fine nel modello **RUP** evidenziato in figura.

Questo modello *appoggia* su tre concetti fondamentali:

1. Un processo *su misura* guidato dallo sviluppo
2. La presenza e l’uso di *tools* che ne renda automatica l’applicazione
3. *Servizi* che accelerino l’adozione sia del processo che dei relativi *tools*

Il software prodotto, giunse ad un tale livello che nel 2003 IBM rilevò la Rational Software Corp. per circa 2.3Mld di dollari inglobandola come nuova divisione. Usato principalmente in progetti di grosse dimensioni ha una sua utilità, almeno a livello concettuale, per qualunque attività complessa, sia in ambito software che in svariati ambiti ingegneristici. Di per sé non è soltanto uno schema in quanto prevede l’uso, come sopra ben evidenziato, di software dedicato e per il costo, riservato a grandi organizzazioni.

3.1.3 Modello AGILE

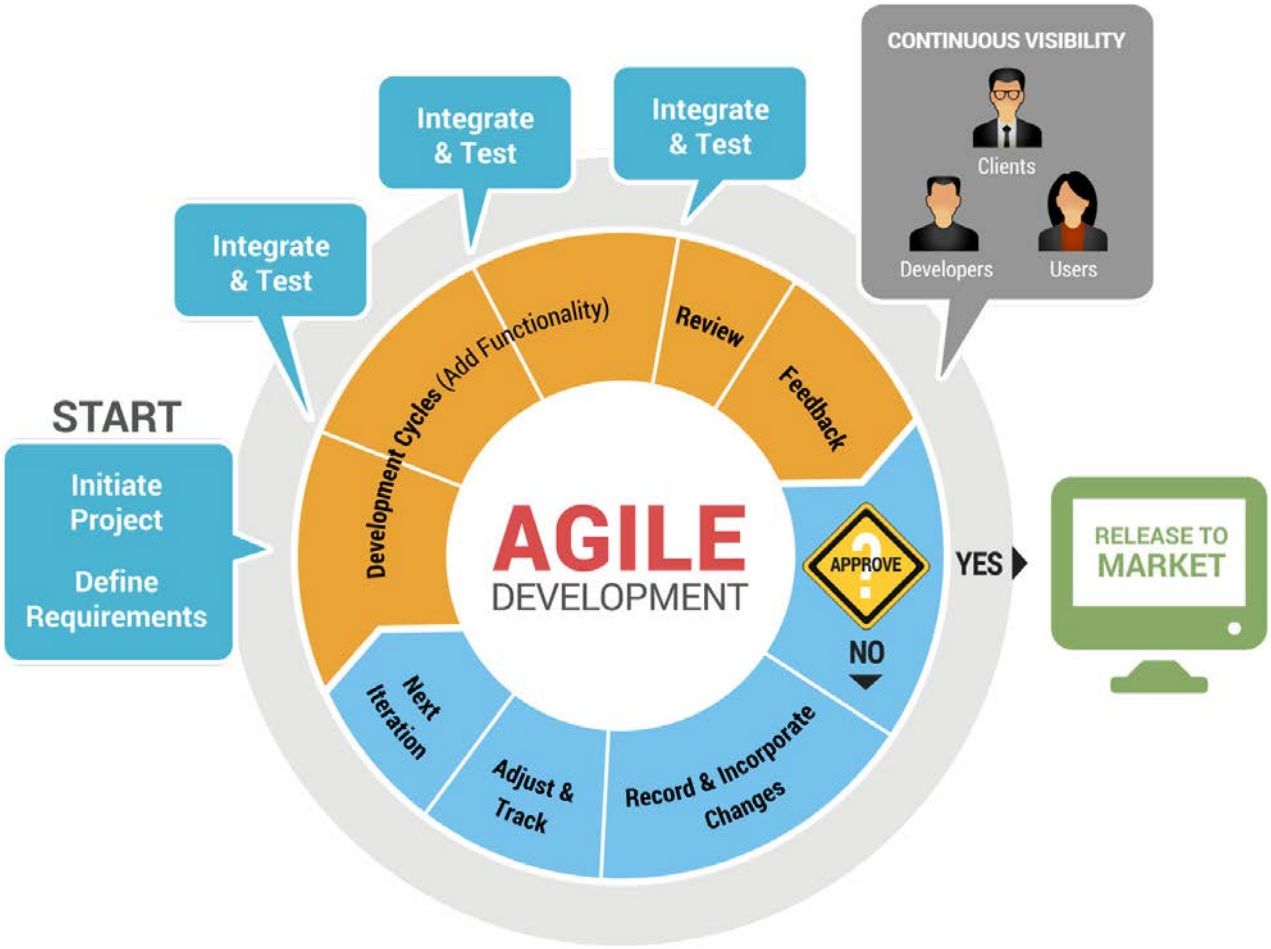


Figura 10 - Il modello di progettazione Agile

La parola chiave *AGILE* è da intendersi con lo stesso significato in italiano ed in inglese.

In pratica ribalta completamente la visione rigida, strutturata e puntigliosa adottata fino al 2001 concretizzando nell' *Agile Manifesto* le idee proposte da 17 *top programmers*[9] in questi concetti chiave:

- **Individui e interazioni** *piuttosto che* **processi e tools**
- **Software funzionante** *piuttosto che* **documentazione dettagliata**
- **Collaborazione con il cliente** *piuttosto che* **contratti dettagliati**
- **Risposta ai cambiamenti** *piuttosto che* **seguire un piano**

Sembra quasi un rivisitazione delle idee *hippie* degli anni '60: "software libero" …

[9] Tratto da: http://en.wikipedia.org/wiki/Agile_software_development

Ovviamente, non è così. Leggendo per intero il manifesto ci si rende presto conto di una banalità che fa la differenza: se il mio fornitore di software mi fa aspettare sei mesi per quella miglioria che gli ho chiesto, di sicuro, alla scadenza del contratto di manutenzione non lo richiamo più[10].

Questa è l'*agilità* insita nel metodo:

1. Soddisfazione del cliente con una rapida consegna del software
2. La modifica dei requisiti è ben accetta, anche quando si è avanti con lo sviluppo
3. Software funzionante è rilasciato con tempistiche strette, settimane anziché mesi
4. Collaborazione stretta, continua, fra i clienti e gli sviluppatori
5. I progetti si realizzano attorno a persone motivate, alle quali va data fiducia
6. La conversazione faccia a faccia è la migliore
7. Il software funzionante è la principale misura del progresso (di un progetto)
8. Sviluppo sostenibile ad un ritmo costante
9. Attenzione continua all'eccellenza tecnica ed alla migliore progettazione
10. Semplicità – l'arte di massimizzare l'ammontare d lavoro non fatto – è essenziale
11. Team di sviluppo autonomi
12. Adattamento regolare al mutare delle circostanze

Questo è umanesimo informatico! È il programmatore di Vitruvio!

Una piccola considerazione: questo modo di approcciare lo sviluppo software è tipico nella realtà delle minuscole *software-houses* italiane. Ma qui stiamo parlando di quelle grandi americane, con migliaia di sviluppatori. Porre al centro di questo mondo iper-tecnologico, l'uomo è la chiave del successo.

[10] Una nota società di software di Casalecchio di Reno dovrebbe leggere con attenzione questo manifesto

4 SCELTA DEI TOOLS

Esaminata la parte concettuale dello sviluppo software voglio concentrare l'attenzione sui *tools* (attrezzi) che rendono possibile lo sviluppo del software moderno.

Nell'epoca informatica attuale, totalmente incentrata nel **web**, termini informatici quali *compilatore, linker, assemblatore* denotano immediatamente l'età di chi li usa. Negli anni '80 il PC era uno strumento e*soterico*, oltre ad avere il costo di un'utilitaria dell'epoca. Chi spenderebbe oggi 10.000 € per l'acquisto di un Personal Computer? Il modem a 2400 baud[11] che ogni tanto tiro fuori dal cassetto ricorda tanto le bambole di ceramica di fine '800. Peccato che mi era costato l'equivalente di un IPhone 6 di oggi.

In quell'epoca non c'era molto da scegliere e quella esistente era una funzione della classe del computer: Basic sui PC-IBM, Fortran sui sistemi Digital, COBOL Assembler/360 e PL/1 sui mainframe.

Altri linguaggi, quali REXX, SQL erano appannaggio di pochi ed altri, come il Pascal, Simula e Prolog, grosso modo confinati nelle aule di ricerca universitarie.

Eppure, la rivoluzione era iniziata alcuni anni prima, accesa da un trio di ricercatori dei Bell Labs, Brian Kernighan, Ken Thompson e Dennie Ritchie che "scoprirono" il sistema operativo Unix ed il linguaggio di programmazione "C". Ogni tesi che riguarda l'informatica dovrebbe includere un riferimento a questi tre scienziati visionari poiché è gran parte merito loro lo sviluppo incredibile ed esplosivo che è seguito.

Brian Kernighan, oltre ad aver inventato il "C", si *dilettava* (nel tempo perso, immagino), in sciocchezzuole quali **awk** e **troff**[12].

Chi ha usato od usa questo linguaggio in progetti di una certa complessità, è indubbiamente d'accordo con la sua *legge*:

"**Il debugging è due volte più difficile rispetto alla stesura del codice. Di conseguenza, chi scrive il codice nella maniera più intelligente possibile non è, per definizione, abbastanza intelligente per eseguirne il debugging.**" (Kernighan s.d.)

Detta così *sembra* una battuta arguta e spiritosa, ma racchiude in sé una semplice quanto assoluta verità *informatica*: programmare è *difficile*, programmare senza errori, molto di più!

[11] Sono sicuro che si è persa memoria del significato di questo termine: il baud è il numero di simboli in 1 secondo che è diverso dal numero di bit/s. Nello standard V.32 bis in uso all'epoca, la modulazione impaccava 6 bits in un simbolo per cui la velocità di 2400 baud (baud rate) trasformata in bit/s era: 2400x6 = 14.400 bit/s ovvero 14.4 Kb/s

[12] Il loro libro **il** "**K&R**" è tutt'oggi la *bibbia* ufficiale del linguaggio C

Il linguaggio C, a mio avviso, non ammette mezze misure: o lo si ama, o lo si odia. Eppure, generazioni di programmatori hanno realizzato con questo *tool* cose incredibili: grafica, sistemi di calcolo parallelo, riconoscimento del parlato umano e moltissimo software di alto livello **scientifico**.

Molto, molto meno, in campo "**commerciale**".

Un linguaggio di questo tipo non è stato pensato per gestire grandi quantità di dati, ma per risolvere il crescente traffico telefonico delle città americane negli anni'70. La chiave di volta di questo progetto era la possibilità di realizzare il *context-switching* nel minor tempo possibile. Da queste esigenze nacque Unix e tutta o quasi la visione *scientifica* dell'informatica.

L'altra faccia della Luna era occupata da un simbolo blu con marchio IBM che probabilmente non dice nulla ai ragazzi nati dopo il 2000.

Era, ed è, una visione "**commerciale**" del mondo IT: il software si produce per *business*.

Big Blue è stata per molti anni l'immaginario collettivo del grande "cervellone" quasi fosse un sinonimo di "America". Ancora oggi, molte fra i più grandi istituti finanziari del mondo, usano sistemi IBM.

La terza "visione" del mondo informatico si deve al signor "Fattura Cancelli" altrimenti noto come Bill Gates che riuscì a creare un nuovo software che permise a tutti (o quasi …) di utilizzare un computer.

La vera innovazione introdotta con Windows, fu quella di creare un oggetto divertente oltre che utile. Migliaia di piccole imprese poterono dotarsi di questi nuovi strumenti per automatizzare moltissime attività svolte manualmente fino ad allora. Le ***finestre*** riuscirono nell'intendo in cui Apple fallì: quello di portare un computer in tutte le case. Al crescere della potenza di calcolo delle CPU crebbero parimenti le possibilità di utilizzo del PC: audio, video, lavoro e divertimento erano tutti racchiuse in una piccola scatola di metallo.

Mi ricordo dell'acquisto nel 1995 da parte di IBM della software house Lotus Development Corporation, creatrice del primo magico programma di calcolo numerico, il lotus 123, per l'incredibile cifra di 4.000.000.000 di lire dell'epoca (circa 3.5 miliardi di dollari del 1995. L'intento di IBM era chiaramente quello di contrastare Microsoft ed il suo pacchetto Office che stava spopolando.

Quello che IBM non capì, fu, a parer mio, il discorso "semantico" introdotto da Gates.

L'aggettivo ***professional*** per un software di computer ti obbligava in pratica all'acquisto. Pura strategia di marketing ma, ovviamente, vincente.

Negli anni '80, un'altra piccola software, Ashton-Tate Corporation riuscì a trasformare un garage in una miniera d'oro perché crearono il primo programma che permetteva una gestione dei dati con la stessa facilità allora disponibile su macchine di costo 10 volte superiore, il DBIII. Finì la sua corsa nel 1991 con l'acquisizione da parte di Borland creatrice del famoso "Turbo C" e Delphi.

Quindi, col passare del tempo, crescevano anche i ***tools*** di sviluppo ed i tecnici informatici furono costretti a specializzarsi. Diventava sempre più difficile definirsi *poliglotti*.

IBM perse la sua battaglia anche nella tecnologia di rete: ***token ring, SNA***, chi furono costoro?

Il protocollo IP tenne fede al suo nome: diventò sul serio il protocollo di comunicazione della rete delle reti.

Impossibile quindi non citare di conseguenza, Sir Tim Berners-Lee e la sua invenzione: il protocollo **http**.

Un minuto di raccoglimento per i *defunti* protocolli quali **gohper** e *moribondi* quali **ftp** e **telnet** mi sembra d'obbligo. Amen. Strano caso è quello di **netbios** che sopravvive ancora come *zombie* nei sistemi Windows. Con grande gioia per gli hackers di tutto il mondo.

4.1 Panoramica delle tecnologie web

Dopo la piccola divagazione sul "*Palio dei protocolli morti*", voglio ritornare ad *http* ed al suo sinonimo *www* e a ciò che ha comportato nell'ambito delle tecnologie di programmazione.

Per le persone comuni, cioè quelle che non passano almeno 5 ore al giorno davanti un computer, Internet, WWW e PC sono in buona sostanza la stessa cosa. Credo che molti tecnici sorridano di fronte a questa mia affermazione, ma non dimentichiamo: *Vox populi, vox Dei*.

Guarda caso è la stessa identica visione del futuro di *piccole* software houses che rispondono al nome di Google e Microsoft. Ultimamente, Microsoft, infatti, dopo l'allontanamento di Ballmer, sta spingendo moltissimo nel cloud, nel mobile e *SASS*[13].

PC, WWW ed INTERNET sono (stanno diventando) la stessa cosa.

Non è difficile immaginare quale rivoluzione si sta attuando al crescere della velocità di trasmissione dati. Assisteremo al crollo delle TV "generaliste", delle grandi catene di negozi *fisici*.

Non dimentichiamo che Microsoft, da sola, ha una capitalizzazione azionaria di circa 330 miliardi di dollari.

Chi oserebbe immaginare che questo gigante abbia ancora pochi anni di vita?

Io.

E per queste ragioni: le rivoluzioni, quelle vere, non sono imposte, guidate, pensate dall'alto ma sono invece come la marea. Sono milioni di persone che credono in un'idea, in un motto, in un pezzo di plastica con dentro un microchip che permette loro di condividere la foto della gamba rotta con gli amici sparsi in tutto il mondo a fare le vere *rivoluzioni*.

Ciò che ha reso grande IBM negli anni 60, 70 ed 80 era "l'odore di America" che diffondeva in tutto il mondo. Era la forza economica e scientifica di quella grande nazione. Il testimone passò per le *finestre* e milioni di giovani entusiasti nei primi anni '80 le vollero per se quando passarono, in giacca e cravatta, nel mondo *professional*.

Quanti sistemi operativi Windows sono "originali" nel mondo? Negli ultimi 4/5 anni direi la maggioranza perché preinstallati su portatili di ogni genere. Ma prima, dal 1985 al 2005, quante copie "pirata" sono state installate sui PC fissi? Pensare che una software house come Microsoft che per 20 anni ha attirato le migliori menti informatiche del pianeta non fosse in grado di proteggere in modo adeguato la sua forza trainante mi sembra una barzelletta. Che non fa ridere. Bill Gates, ha dichiarato molte volte di avere una visione del futuro nel medio-lungo periodo, di 10 anni almeno. Il suo successore, Ballmer, era più concentrato sul business immediato. Con questo voglio dire che Microsoft ha sempre avuto l'interesse nel diffondere quanto più possibile il suo software per creare una rete di milioni di programmatori che avrebbero creato alla fine un effetto *lock-in* per imprese e singoli.

Ai tempi di IBM il direttore finanziario parlando al direttore dei servizi informatici diceva che non si sbagliava mai a scegliere IBM.

Sento ripetere da un po' di temo gli stessi concetti riguardo Microsoft.

Perché crollerà? Perché la maggioranza degli abitanti della terra dotati di un device per accesso ad Internet, non utilizza un prodotto Microsoft, ma bensì Android.

[13] Software as a service: software come servizio

Soltanto in Cina ci sono circa 650.000.000 di persone che quotidianamente accedono alla Rete con smartphones sempre più potenti e con installato Android.

E' vero: ci sono esperimenti quali FireOs ed il sistema di AliBaba, leader mondiale del commercio elettronico ma li considero quello che sono: esperimenti, per l'appunto.

Quando milioni di ragazzini, saranno abbastanza grandi e capaci di utilizzare appieno gli strumenti informatici che si troveranno fra le mani fra pochi anni, mi sembra inevitabile pensare alla fine di Windows.

E di tutti i ***toolchaines*** a contorno. Forse fra 10 anni, un programmatore esperto, alla domanda: "Conosci DevStudio?" ti guarderà stupito pensando che vieni da un altro pianeta.

Questo è ciò che vedo collegato alla scelta di una tecnologia piuttosto che un'altra: un business colossale che può spostare l'asse (economico!) della Terra.

4.2 HTML

Al contrario di molte tecnologie Internet, lo sviluppo di HTML sembra procedere molto lentamente. È notizia di fine Ottobre il rilascio definitivo delle specifiche delle versione 5.

L'ultimo aggiornamento. V. 4.01 risale infatti al 24 Dicembre1999.

Un aggiornamento dopo quasi 15 anni non è normale nel mondo I.T. dove tutto si muove in modo incredibilmente veloce. Sorge il sospetto che il consorzio W3C sia tale solo sulla carta e che dei 402 membri[14], solo pochi siano quelli che veramente dettano le regole.

Comunque sia, la versione 5 è arrivata in porto. La tendenza che si sta affermando è quella di creare una piattaforma unica per lo sviluppo centrata su questa tecnologia.

In ambito mobile le cose sembrano procedere più speditamente come la pagina: **STANDARDS FOR WEB APPLICATIONS ON MOBILE: CURRENT STATE AND ROADMAP[15]** ci informa.

Il progetto HTML5Apps[16] si vuole spingere oltre e creare uno *store* pubblico che superi i limiti ed i vincoli di quelli che conosciamo oggi.

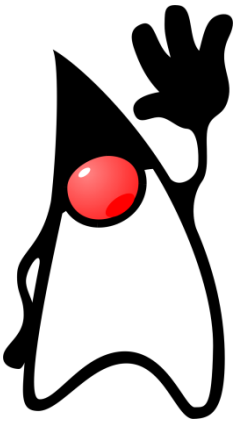
In quanto tempo ciò accadrà e se accadrà realmente è più un esercizio per indovini che per informatici, ma la tendenza mondiale verso lo sviluppo di applicazioni HTML/CSS/JavaScript sia ormai un treno in corsa mi sembra palese.

Che moltissime società di software vedano in pericolo il loro business basato su tools proprietari e che tentino in tutti i modi di frenare questa nuova ondata mi sembra normale come immagino normale fra qualche anno sentire parlare di *legacy java software* come oggi si parla di *legacy Cobol systems*.

[14] Vedi http://www.w3.org/Consortium/Member/List
[15] Vedi http://www.w3.org/Mobile/mobile-web-app-state/
[16] Vedi http://html5apps-project.eu/

4.3 Java vs JavaScript

Questo è stato per me un dilemma che mi ha preso circa 6 mesi di tempo per la valutazione e che non ho ancora risolto del tutto.



Nel 1995 vedere ***Duke*** che ti faceva "ciao ciao" con la sua manina stilizzata da dentro una pagina web sembrava il futuro del futuro. Ci si appassionò subito a quel linguaggio: pulito, potente, molto ben strutturato, con debugger razionale, un'ottima documentazione, la promessa del "*write once, run anywhere*"[17] e, con il nuovo mondo degli oggetti[18] sembrava la panacea per tutti i mali dell'informatica. Una software house di piccole dimensioni, prima focalizzata e concentrata su di unica piattaforma vedeva aprirsi nuove possibilità e mercati potenziali: Unix, Windows, Digital ed anche IBM. Sembrava fosse veramente possibile "scrivere una volta … e vendere a diversi clienti".

Il web degli inizi abbracciò questa tecnologia, che fu però originariamente pensata da Sun Microsystem per sistemi embedded, quando Netscape Corporation decise di incorporare la JVM[19] all'interno del suo *browser*. Da quel momento il *web* vide nascere l'interattività per mezzo delle applets e del contenzioso fra la comunità dei programmatori C e quelli Java.

Due visioni della tecnologia diverse che si rispecchiano bene nella sintassi del programma più famoso di sempre: "HelloWorld":

[17] Promessa nota con il suo acronimo: ***WORA*** oppure ***WORE*** dove la E finale sta per *everythere* anziché *anythere*
[18] Che, ovviamente non era. Già nel 1967, il linguaggio Simula faceva uso di classi
[19] Java Virtual Machine: macchina virtuale java

Java:

```
class HelloWorld
 {
     public static void main(String[] args)
     {
         System.out.println("Hello World");
     }
 }
```

```
C:
int main()
{
    printf("Hello, World\n");
}
```

Col passare degli anni la tecnologia Java si arricchì in modo a mio giudizio, eccessivo, come la figura che riporto sotto evidenzia.

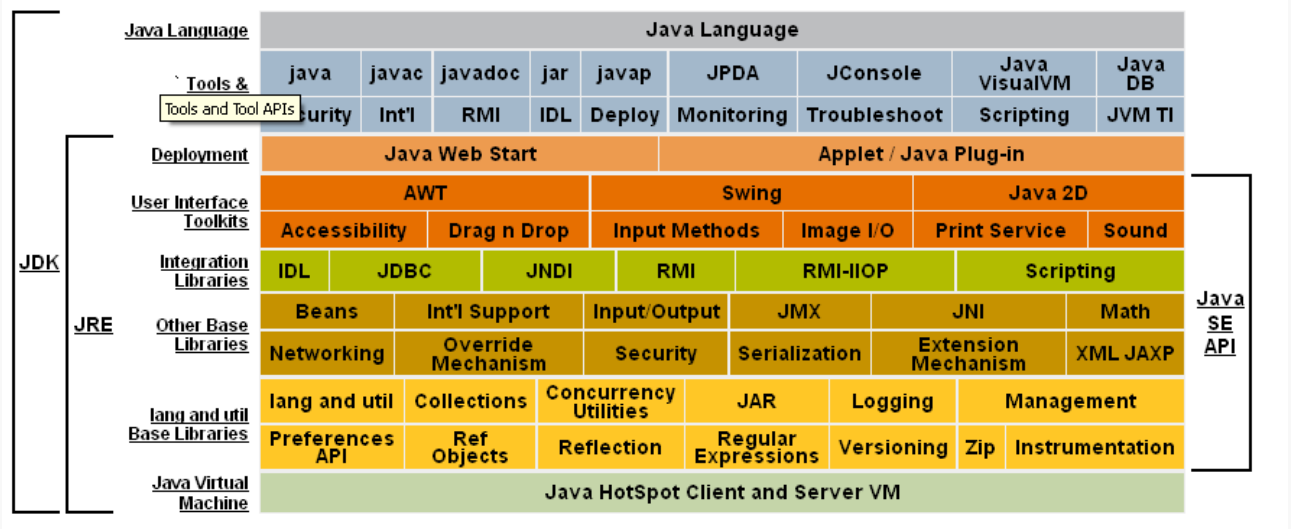


Figura 11 - Linguaggio java

Duke aveva perso il suo sorriso ed iniziava a fare un po' paura …

Java era diventato un business di per sé, complicandosi sempre di più e complicando la vita a chi lo usava per lavoro.

Di certo è ancora tra i linguaggi più popolari e molto apprezzato *lato server*. La complessità generale lo pone *naturalmente* in ambienti professionali fortemente strutturati. Non lo ritengo adatto per progetti di piccole/medie dimensioni. La vastissima scelta di librerie per ogni possibile esigenze ne fanno però il candidato ideale per un'intranet aziendale dove l'architettura *client/server* può essere controllata a piacimento.

Per Internet, il discorso cambia: quante “pagine” propongono ancora applets Java? Duke ed il suo laghetto riflettente[20] sembrano essere scomparsi dal web.

Due diverse librerie per l’interfaccia grafica, sono state, a mio giudizio, controproducenti per l’utilizzo di java nel web; oltre, ovviamente, la difficile interazione col modello a oggetti del browser, DOM, e dei fogli di stile, i *Cascade Style Sheet*.

Inoltre, la carenza per molto tempo, di *IDE*[21] con supporto a *WYSIWYG*[22] non ha di certo facilitato l’utilizzo del linguaggio Java nelle pagine html.

Come detto in precedenza, il linguaggio Java non è stato pensato e progettato per il web, per il semplice fatto che non esisteva ancora. Doveva far funzionare le TV via cavo ed altri dispositivi elettronici embedded, non di certo manipolare il DOM o *flippare una raw image nel canvas ad un frame rate di almeno 18FPS*[23].

[20] Era una famosa applet che simulava il movimento delle onde in un laghetto rispecchiando l’immagine della parte superiore

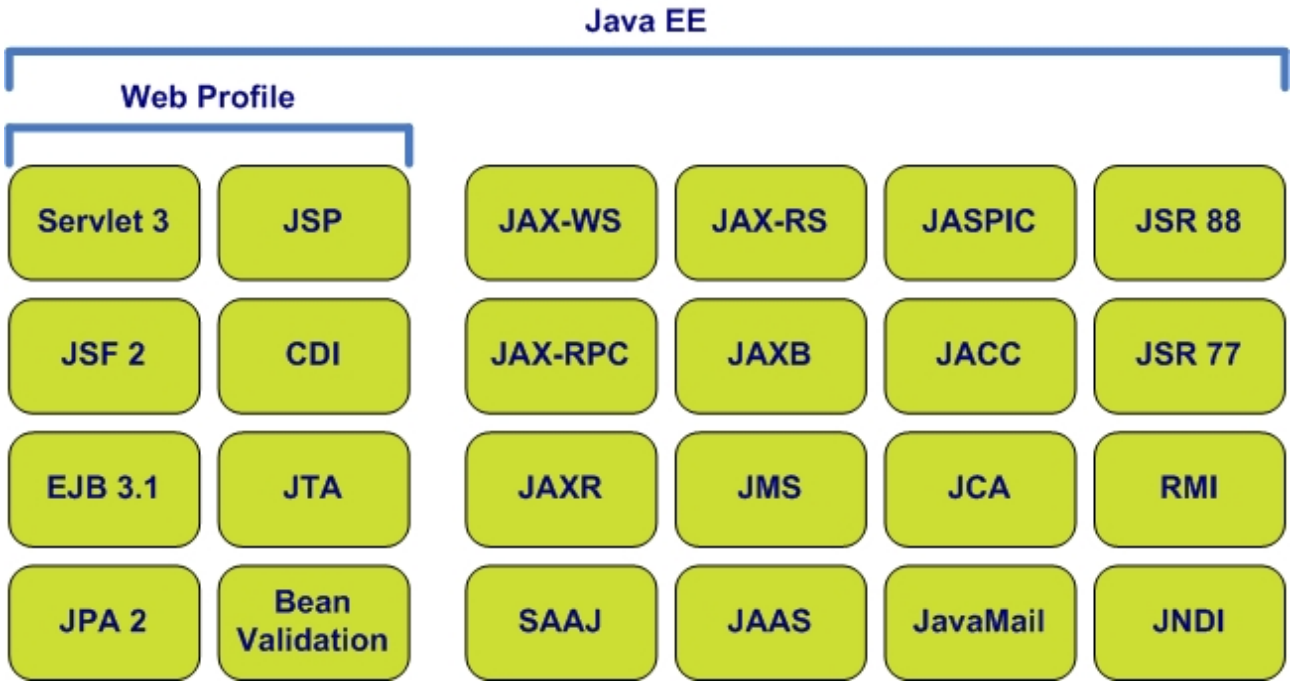
[21] Non è un refuso tipografico in quanto significa *Integrated Development Environment*

[22] Non è colpa mia se l’America ha vinto la II Guerra Mondiale ed ha imposto al mondo intero la lingua inglese, specialmente in ambito economico ed informatico. D’altronde, come avrebbero dovuti scriverli i compilatori? In lingua bantu?

[23] Lessico e nuvole in Messico: è questo il *tipico* linguaggio dei programmatori. È un codice che accomuna chi ne ha la chiave e che esclude *i non addetti ai lavori*.

4.4 Java EE

La piattaforma ***Java Platform Enterprise Edition*** è ampiamente usata nella programmazione Web, specialmente per grossi progetti e società. L'immagine sotto fa capire perché:



4.5 I frameworks

4.5.1 Javabased

Figura 12 - Alcune icone che identificano Java

'na tazzuriella'caffè, dicono a Napoli.

OAK! WebSpinner! Silk! DNA! Java. Dissero I responsabili di Sun Microsystem nel gennaio del 1995 quando si riunirono a Santa Clara, California, per stabilire il nome del nuovo linguaggio. Il brand (marca) è spesso più importante del prodotto stesso; non importa se identifica una bevanda, un orologio o un paio di scarpe, o come in questo caso, un software. Verrà recepito, condiviso, utilizzato da milioni di persone, e la sua definizione è cruciale per il suo successo.

E fu proprio Java, ad esser definito da Time Magazine, uno dei 10 migliori prodotti del 1995, in quanto il nome Oak era già un marchio registrato.

Chissà se avessero scelto il nome *Silk* (seta) come si sarebbe evoluta la tecnologia Silk: ho dei dubbi sulle "facce di seta del server".

4.5.1.1 JSF

Fu però soltanto nel 2001 che apparsero[24] le **API**[25] Java Server Face, nome in codice: "Moonwalk", un *framework* applicativo per la creazione di interfacce utente sul web.

Sun Microsystems, IBM, Oracle Corporation e Siemens si associarono per definirne le specifiche, sulla base dell'innovativo lavoro di un ingegnere Sun che ne effettuò la prima stesura con rilascio nella versione JSF 1.0 il 3/11/2004.

Tecnicamente, il termine JSF identifica solo le specifiche ma comunemente si associa all'intero framework. Essendo queste specifiche di dominio pubblico, l'*Apache Software Foundation*, ha potuto creare una propria *release* denominata **Apache MyFaces[26].**

Una valida alternativa a quella rilasciata a suo tempo da Sun, oggi in capo ad Oracle Corporation[27].

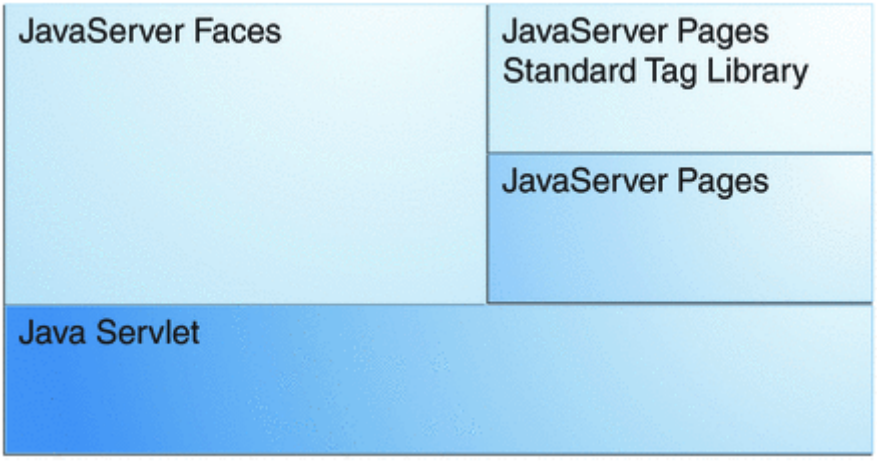


Figura 13 - JSF a blocchi

In sostanza, la tecnologia JSF permette a programmi scritti in java in esecuzione su un *server java-based* di creare codice HTML che verrà inviato al client sotto forma di pagina web.

Le API permettono una gestione completa degli eventi associati agli oggetti che creano la **UI**[28] generando codice opportuno. Possono, nelle ultime versioni, generare anche grafici in formati diversi, es.:

---

[24] Word si ostina a segnalare errore. Come faccio a fargli capire: "..ché, rotti i lacci a l'elmo suo, d'un salto/ (mirabil colpo!) ei le balzò di testa; / e le chiome dorate al vento sparse, / giovane donna in mezzo 'l campo apparse." [Torquato Tasso (*Gerusalemme liberata*, Canto 3.21)]

[25] Application Programming Interface: Interfaccia di Programmazione di un'Applicazione

[26] Vedere http://myfaces.apache.org/ per maggiori informazioni

[27] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

```
<h:graphicImage value="resources/images/sofa.png" />[29]
```

Che viene *tradotta* in HTML come:

```
<img src="resources/images/sofa.png;" alt="" />
```

Quindi, JSF, che è basato su XML, tramite *dichiarative* chiamate *tag(s)*[30] viene "tradotto" dal server nel linguaggio standard del Web che è l'HTML.

L'intero framework sposa il *modello* **MVC**[31] col supporto di **Ajax**[32] lato client. Se la definizione delle interfacce grafiche è piuttosto semplice, non lo è l'integrazione con le varie componenti sul server.

Per illustrare meglio il concetto riporto un esempio di *Login Form* o Maschera/Pannello/Modulo di autenticazione basato su JSF[33].

Dichiarazione del form di login:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns=http://www.w3.org/1999/xhtml xmlns:h="http://java.sun.com/jsf/html">
        <h:body>
                <h:form>
                   <h:panelGrid columns="2" rendered="#{!login.loggedIn}">
                   <h:outputLabel for="username">Username:</h:outputLabel>
                   <h:inputText id="username" value="#{credentials.username}"/>
                   <h:outputLabel for="password">Password:</h:outputLabel>
                  <h:inputText id="password" value="#{credentials.password}"/>
                </h:panelGrid>
                <h:commandButton value="Login" action="#{login.login}" rendered="#{!login.loggedIn}"/>
                <h:commandButton value="Logout" acion="#{login.logout}" rendered="#{login.loggedIn}"/>
                </h:form>
        </h:body>
</html>
```

Il *tag* **h** raggruppa tutti i componenti UI definiti dalle specifiche[34]; in questo caso, la riga

<h:panelGrid columns="2" rendered="**#{!login.loggedIn}**">

Produrrà una griglia di due colonne sul client, se il valore della variabile mantenuta sel server, **login.loggedIn** è 0.

---

[28] User Interface: Interfaccia Grafica. L'insieme dei componenti visuali che permettono l'interazione uomo-macchina
[29] Da http://www.mkyong.com/jsf2/jsf-2-graphicimage-example/
[30] Letteralmente: contrassegno, etichetta, cartellino. Da *to tag* = contrassegnare
[31] Model View Controller: traducibile con "Modello Vista Controllo" descritto più avanti
[32] Asynchronous JavaScript and XML, descritto più avanti
[33] Tratto da: http://docs.jboss.org/webbeans/reference/1.0.0.BETA1/en-US/html/example.html

[34] Vedi https://docs.oracle.com/javaee/6/javaserverfaces/2.0/docs/pdldocs/facelets/

Le due colonne conterranno le etichette ed i campi di input da ritrasmettere al server.

Il contenuto fra apici "#{variabile}" è il **bind** (collegamento) col valore inviato dal server o trasmesso ad esso tramite:

```
@Named @RequestScoped

public class Credentials {
   private String username;
   private String password;
   public String getUsername() { return username; }
   public void setUsername(String username) { this.username = username; }
   public String getPassword() { return password; }
   public void setPassword(String password) { this.password = password; }
}
```

Queste righe di codice rappresentano un cosiddetto **java bean** (letteralmente *chicco di java*, e, visto che Java è l'isola tailandese famosa per il caffè, si intende *chicco di caffè*.).

Il *java bean* è quindi una classe scritta in java che aderisce a particolari convenzioni sulle definizione dei nomi e dei metodi e che viene utilizzata per poter incapsulare più oggetti in un unico oggetto. I nomi che iniziano con il simbolo '@' sono, le cosiddette *annotation*[35] che, dalla versione JSF 2.0 rimpiazzano molte direttive che dovevano essere configurate nel file *faces-config.xml*. Nella Figura 13 - JSF a blocchi a pag. 37 vengono evidenziati altri componenti che completano **JSF** che brevemente spiego qui:

4.5.1.2 JSP - Standard Tag Library (JSTL)

É una libreria che racchiude come semplici tags, funzionalità comuni a molte applicazioni Web quali cicli, salti condizionali, logica di manipolazione dei documenti XML e del linguaggio SQL facilitando l'integrazione di *tags* utente. Di seguito riporto un esempio[36] di una progetto completo per visualizzare una pagina web che visualizza un elenco di impiegati.

[35] Per esteso "Faces Managed Bean Annotation Specification", vedi https://docs.oracle.com/cd/E17802_01/j2ee/javaee/javaserverfaces/2.0/docs/managed-bean-api/index.html?overview-summary.html

[36] Tratto da: http://www.journaldev.com/2090/jstl-tutorial-with-examples-jstl-core-tags

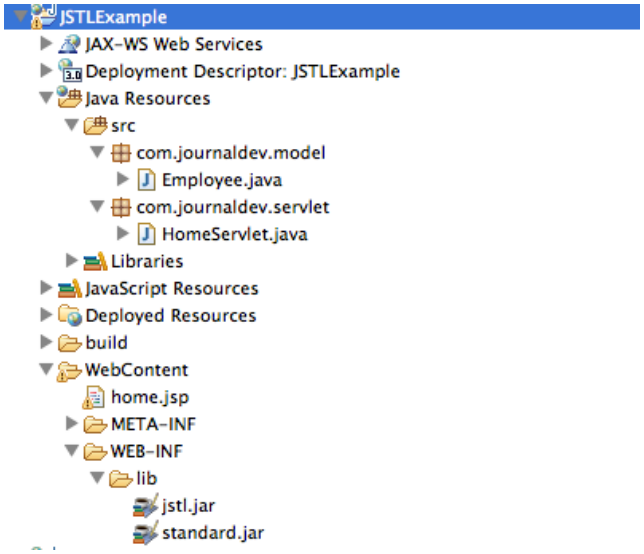


Figura 14 - Esempio di progetto Java EE che include JSTL

Snapshot di un progetto Eclipse di una pagina web-dinamica con la struttura delle directory creata dall'IDE.

Java Bean Class
Employee.java

```
1
2       package com.journaldev.model;
3
4       public class Employee {
5
6           private int id;
7           private String name;
8           private String role;
9           public Employee() {}
10          public int getId() { return id; }
11          public void setId(int id) { this.id = id; }
12          public String getName() { return name; }
13          public void setName(String name) { this.name = name; }
14          public String getRole() { return role; }
15          public void setRole(String role) {this.role = role; }
16
17      }
```

Classe *Java Bean* per la definizione della struttura dati dell'impiegato con i *getters* e *setters* opportuni.

Da notare la riga 8 col costruttore vuoto della classe.

HomeServlet.java

```
1       package com.journaldev.servlet;
2
3       import java.io.IOException;
4       import java.util.ArrayList;
5       import java.util.List;
6
7       import javax.servlet.RequestDispatcher;
8       import javax.servlet.ServletException;
9       import javax.servlet.annotation.WebServlet;
10      import javax.servlet.http.HttpServlet;
11      import javax.servlet.http.HttpServletRequest;
12      import javax.servlet.http.HttpServletResponse;
13
14      import com.journaldev.model.Employee;
15
16      @WebServlet("/HomeServlet")
17      public class HomeServlet extends HttpServlet {
18          private static final long serialVersionUID = 1L;
19
```

```
20          protected void doGet(HttpServletRequest request, HttpServletResponse response)
21              throws ServletException, IOException {
22              List<Employee> empList = new ArrayList<Employee>();
23              Employee emp1 = new Employee();
24              emp1.setId(1); emp1.setName("Pankaj");emp1.setRole("Developer");
25              Employee emp2 = new Employee();
26              emp2.setId(2); emp2.setName("Meghna");emp2.setRole("Manager");
27              empList.add(emp1);empList.add(emp2);
28              request.setAttribute("empList", empList);
29
30              request.setAttribute("htmlTagData", "<br/> creates a new line.");
31              request.setAttribute("url", "http://www.journaldev.com");
32              RequestDispatcher rd = getServletContext().getRequestDispatcher("/home.jsp");
33              rd.forward(request, response);
34          }
35
        }
```

La *servlet* che prepara ed invia la lista al *client*: nel dettaglio:

```
riga      1: definizione del package
riga      3: inclusione della classe per la gestione delle eccezioni
righe   4-5: inclusione delle classi per la gestione delle liste e degli array
riga      7: inclusione della classe per ridirezionare le richieste del client verso altre risorse
riga      8: inclusione della classe per la gestione delle eccezioni servlet
riga      9: inclusione della classe per l'annotazione usata per creare un servlet
righe 10-12: inclusione delle classi per la gestione delle richieste/risposte HTTP
riga     14: inclusione della classe per la definizione dei dati dell'impiegato
riga      16: questa annotazione permette di specificare direttamente nel file .java a quale entry
              point risponderà la servlet, nell'esempio: http://<nome sito>/HomeServlet. Evita di
              specificare nel file di configurazione web.xml tutti gli entry points dell'applicazione
              (pagine web richieste dal client)
riga     17: definizione della classe HomeServlet come estensione di HttpServlet
riga       18: codice aggiunto automaticamente dall'IDE Eclipse per facilitare e controllare la
              serializzazione/deserializzazione dei dati della classe. Se, nel corso del tempo, la
              struttura della classe cambia, si può fare riferimento ad un vecchio UID per la
              struttura corrispondente
riga      20: override del metodo doGet di HttpServlet con la logica per soddisfare la richiesta GET
              del client
riga      21: eccezioni sollevate in caso di errori di I/O o se la richiesta GET è malformata od
              altro
riga     22: definisce una lista di impiegati
riga     23: definisce un nuovo impiegato
riga     24: l'impiegato Panka ha codice 1 e ricopre il ruolo di Developer
riga     25: definisce un nuovo impiegato
riga     26: l'impiegato Meghna ha codice 2 e ricopre il ruolo di Manager
riga     27: aggiunge i due impiegati alla lista
riga     28: imposta l'attributo della richiesta empList con i valori della lista
righe 30-31: impostazione dei valori degli attributi
riga     32: ottiene un riferimento alla jsp home.jsp
riga     33: esegue l'inoltro alla pagina home passandogli la struttura dati relativa alla richiesta
              ricevuta e quella della risposta
```

La pagina home.jsp:

I **tags jsp** sono riportati in **rosso** e quelli **jstl** in **verde**.

La sintassi di una *direttiva jsp* è la seguente:

```
<%@ directive name [attribute name="value" attribute name="value" ....]%>
```

Notare la sequenza <%@…%> che identifica le direttive e la sequenza <%--…--%> che racchiude un commento.

In **jsp** esistono solo tre tipi di direttive:

1. Direttiva ***page***
2. Direttiva ***include***
3. Direttiva ***taglib***

Ognuna definisce degli attributi e dei parametri come vediamo qui:

home.jsp

```
01 <%@ page language="java" contentType="text/html; charset=US-ASCII" pageEncoding="US-ASCII"%>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
03 <html>
04 	<head>
05 		<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
06 		<title>Home Page</title>
07 		<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
08 		<style>
09 			table,th,td { border:1px solid black; }
10 		</style>
11 	</head>
12 	<body>
13 		<%-- Using JSTL forEach and out to loop a list and display items in table --%>
14 		<table>
15 			<tbody>
16 				<tr><th>ID</th><th>Name</th><th>Role</th></tr>
17 				<c:forEach items="${requestScope.empList}" var="emp">
18 					<tr><td><c:out value="${emp.id}"></c:out></td>
19 					<td><c:out value="${emp.name}"></c:out></td>
20 					<td><c:out value="${emp.role}"></c:out></td></tr>
21 				</c:forEach>
22 			</tbody>
23 		</table>
24 		<br><br>
25 		<%-- simple c:if and c:out example with HTML escaping --%>
26 		<c:if test="${requestScope.htmlTagData ne null }">
27 			<c:out value="${requestScope.htmlTagData}" escapeXml="true"></c:out>
28 		</c:if>
29 		<br><br>
30 		<%-- c:set example to set variable value --%>
31 		<c:set var="id" value="5" scope="request"></c:set>
32 		<c:out value="${requestScope.id }" ></c:out>
33 		<br><br>
34 		<%-- c:catch example --%>
35 		<c:catch var ="exception">
36 		  <% int x = 5/0;%>
37 		</c:catch>
38 		<c:if test = "${exception ne null}">
39 		  <p>Exception is : ${exception} <br />
40 		  Exception Message: ${exception.message}</p>
41 		</c:if>
42 		<br><br>
43 		<%-- c:url example --%>
44 		<a href="<c:url value="${requestScope.url }"></c:url>">JournalDev</a>
45 	</body>
46 </html>
```

Nel dettaglio (escludo le righe con puro codice html)

Riga 01: *page* definisce una pagina *java* di tipo testuale con set di caratteri US-ASCII

Riga 07: *taglib* dichiara che la pagina *jsp* utilizza un set di *tags* personalizzati, ne indica il percorso *uri*[37] e dichiara con *prefix* che il namespace associato è "c"

Righe 16-21: conl e dichiarative *jstl* si realizza il concetto di web dinamico, in quanto nel ciclio *foreach*, l'attributo *items* riceverà i dati provenienti dal server e resi disponibili nel campo **requestScope.empList**

Righe 26-28: si controlla che **requestScope.htmlTagData** non sia nullo ed in tal caso ne riporta il valore nella in output rimuovendo le tags XML se presenti. Se andiamo a rileggere la riga 39 del file HomeServlet.java noteremo che inserirà una nuova riga e stamperà il messaggio "`creates a new line.`"

Righe 31-32: esempio di impostazione di una variabile. Viene definita una variabile con id "id", valore "5" nello spazio contestuale della richiesta

Righe 35-41: esempio di cattura di un'eccezione. Viene eseguita una divisione per zero che viene intercettata e stampata subito dopo.

Tutto questo si integra con il ciclo di vita di una pagina JSP sotto riportato.

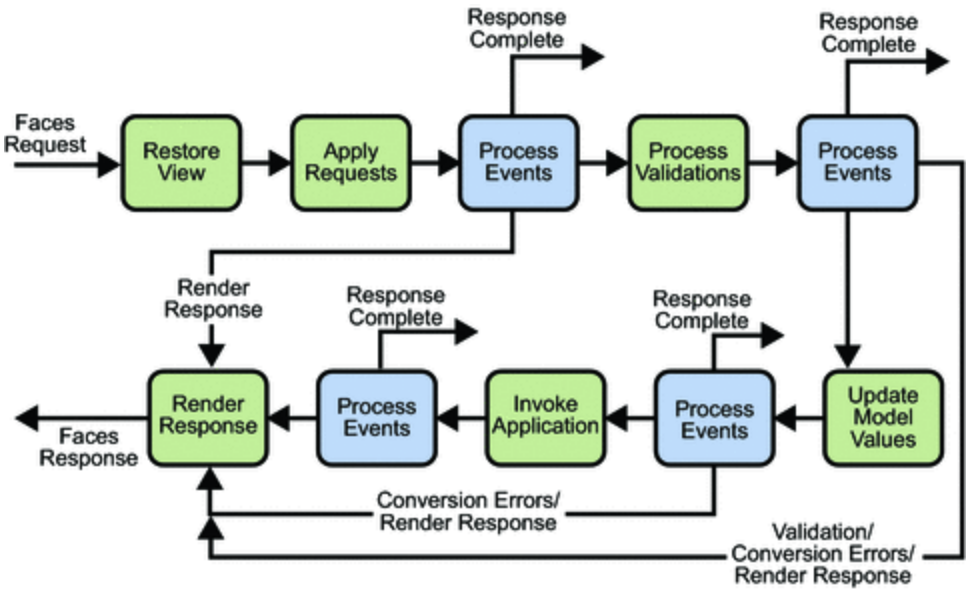


Figura 15 - Ciclo di vita di una pagina JSP

Ho provato per circa 3 mesi queste tecnologie e mi sono convinto che siano fondamentalmente da non utilizzare poiché nascondono completamente i *linguaggi nativi* di Internet, ovvero: **HTML, CSS** e **JavaScript**. Non ho dubbi sulla solidità, sicurezza e completezza di Java EE come piattaforma di sviluppo Web, anche se ne ho sperimentato solo una piccola parte, ma, in tutta sincerità, mi sono convinto che non sia il *tool* ideale per lo sviluppo di un'applicazione quale GesVol.

D'altronde, il termine *Enterprise* in Java EE non lasciava dubbi in merito: i vantaggi si vedono solo in progetti di larga scala, ammettendo che ce ne siano.

[37] Uniform Resource Identifier: Identificatore Uniforme di Risorsa. Può essere una pagina web, un servizio web o un qualsiasi tipo di documento digitale

4.5.1.3 Rich Internet Application

Riassumo nella lista seguente I principali frameworks per lo sviluppo delle cosiddette "Applicazioni Internet Ricche[38]" che ho sperimentato o dei quali in qualche modo ne ho approfondito la conoscenza.

Framework	Software license	Windows	Mac OS X	Linux	Built formats
AngularJS	MIT License	Yes	Yes	Yes	JavaScript
Apache Flex formerly Adobe Flex	Apache 2	Yes	Yes	Yes	Adobe AIR, SWF
Appcelerator Titanium	Apache 2	Yes	Yes	Yes	Android, iOS (iPad, iPhone)
Ember.js	MIT License	Yes	Yes	Yes	JavaScript
ExtJS	GPLv3 or commercial	Yes	Yes	Yes	JavaScript
Google Web Toolkit	Apache 2	Yes	Yes	Yes	JavaScript
Dojo	modified BSD license	Yes	Yes	Yes	JavaScript
Meteor	MIT	Yes	Yes	Yes	JavaScript
MotherApp	Proprietary	Yes	Yes	Yes	Android, BlackBerry, iOS, Symbian, Windows Mobile
OpenLaszlo	Open source	Yes	Yes	Yes	DHTML, SWF
PhoneGap	Open source	Yes	Yes	Yes	iPhone, Android, BlackBerry OS, WebOS, Windows Phone 7, Symbian, Bada
qooxdoo	LGPL, EPL	Yes	Yes	Yes	JavaScript
.NET Framework	Proprietary	Yes	Yes	No	Microsoft Silverlight
Smart GWT & SmartClient	LGPL, commercial	Yes	Yes	Yes	Java server, JavaScript client
Vaadin	Apache 2	Yes	Yes	Yes	Java server, JavaScript client
JavaFX	Open source	Yes	Yes	Yes	jar, applet
Kendo UI	Apache 2 or commercial	Yes	Yes	Yes	

Tabella 2 - Lista dei frameworks RIA

Le righe con sfondo azzurrino identificano i frameworks che ho sperimentato per almeno una settimana, quelle in giallo sono i frameworks che ho scelto per lo sviluppo, e le altre, in grigio i frameworks di cui mi sono documentato cercando informazioni su Internet.

[38] È inutile provarci con le traduzioni: la nostra meravigliosa lingua non si adatta alla tecnologia ed, in generale al mondo anglofono. Chi comprerebbe un whisky (non traducibile, per fortuna!) che si chiama "Giovanni camminatore speciale?"

Questa è stata sicuramente la fase più difficile e lunga del progetto ma se ci catapultiamo, ad esempio, nel mondo dei sommelier, su quali basi uno può dire che un vino è migliore di un altro? O un critico cinematografico come può affermare che vale la pena vedere un film o un altro?

Se l'onestà intellettuale prevale, il primo dovrà assaggiare molti calici di vino ed il secondo passare molte notti in sala di proiezione.

Nel *migliore dei mondi possibili* succede questo.

Nel paragrafo "Scelta del framework" a pag 72, spiego perché ho evidenziato quelle righe.

La lista completa è consultabile all'indirizzo:

http://en.wikipedia.org/wiki/List_of_rich_Internet_application_frameworks

4.5.1.4 GWT

L'evoluzione verso un web più interattivo è passata di sicuro attraverso questo framework presentato da Google il 17 Maggio del 2006. L'idea di Google, innovativa per l'epoca, era quella di sposare la tecnologia Java con il linguaggio JavaScript. All'epoca il browser Chrome non esisteva ancora ed il *market-share* dei principali browsers era questo[39]:

2006	Mozilla	Internet Explorer	Firefox	Netscape	Opera
November	2.5 %	60.6 %	29.9 %	0.2 %	1.5 %
September	2.3 %	62.1 %	27.3 %	0.4 %	1.6 %
July	2.3 %	62.4 %	25.5 %	0.4 %	1.4 %
May	2.3 %	63.0 %	25.7 %	0.3 %	1.5 %
March	2.4 %	64.7 %	24.5 %	0.5 %	1.5 %
January	3.1 %	66.0 %	25.0 %	0.5 %	1.6 %

Tabella 3 - Market share dei browsers nel 2006

La guerra dei browsers iniziata 10 anni prima col famoso memorandum di Bill Gates ai top manager Microsoft, vedeva ancora dominare Internet Explorer pur con una forte riduzione di share. Nel 2003 aveva infatti il 95% e se aveva sconfitto Netscape, Firefox e Mozilla avevano eroso quote significative di mercato.

Questa guerra, perché di questo si tratta, costò a Microsoft cifre spaventose: si parla di circa 100Mln di dollari per anno dal 1996. Il team di sviluppo raggiunse un picco di 1000 persone nel 1999.

Google Inc. nasce nell'Agosto del 1998 grazie ad un finanziamento di 100.000$ da parte del co-fondatore della Sun, e di certo non poteva competer col gigante Microsoft su quel terreno di battaglia. Per espandersi, doveva attirare nuovi utenti, ed elaborò un piano per produrre nuovi applicazioni web molto innovative. La visione di Schimdt che si rifiutò a lungo di entrare nel campo di battaglia dei browser fu a dir poco profetica. Le nuove applicazioni, GMail, Wallet, Flights, Cars erano tutte basate sul nuovo paradigma di sviluppo web, l'**Ajax**. Ma l'incompatibilità di Internet Explorer, Mozilla e degli altri browsers con gli standard Ajax era troppo marcata.

Così partì il progetto di un compilatore scritto in java, denominato Google Web Toolkit o GWT, che generava codice JavaScript tenendo conto di tutte le differenze nella gestione fra i browsers per essere poi eseguito nel client.

[39] Da http://www.w3schools.com/browsers/browsers_stats.asp

La capacità del compilatore di trasformare classi scritte in java in files JavaScript da inviare per esecuzione nel browser sembra rafforzare il concetto WORA esposto a pag. **Errore. Il segnalibro non è definito.**.

Un punto di forza di GWT è senz'altro la capacità di debugging integrata da Google nel *plugin*[40] per la nota *ide* Eclipse.

Uno degli aspetti *tediosi* dello sviluppo web in ambiente Java EE è il ciclo di *compiling-packaging-deploying* dell'intera applicazione nel webserver *java-enabled* quale Tomcat, Glassfish od altro per effettuare il test; l'intero processo può durare parecchi minuti che moltiplicati per il numero di tests per sviluppatore per giorno può portare ad un rallentamento significativo nello sviluppo di un applicazione.

Per ovviare a questo problema, GWT introduce due modi di esecuzione:

1. Web Mode

   In questo modo l'applicazione viene eseguita come JavaScript+html, risultato della compilazione delle classi java, e del packaging. È l'output finale, pronto per la *produzione*

2. Hosted Mode

   In questo modo il *bytecode* java delle classi compilate, viene eseguito direttamente nella JVM e di conseguenza può essere lanciato direttamente dall'IDE per sessioni di debug con il browser web incorporato nel framework.

   Il ciclo di sviluppo si riduce notevolmente, in quanto l'ambiente Eclipse, contiene tutti i *tools* necessari (editor, debugger, compiler, browser) per cui, oltre a facilitare enormemente il *debugging* rende molto semplice o annulla la complessità del *setup* e gestione dei server *java-enabled*.

   Nella pagina seguente riporto uno *screen-shot* di una sessione di *debugging* in Eclipse.

[40] Dal verbo inglese *to plu* = infilare la spina. Si intende in informatica, un modulo software che si *inserisce* in un altro software per ampliarne le capacità in modo semplice e trasparente per lo sviluppatore

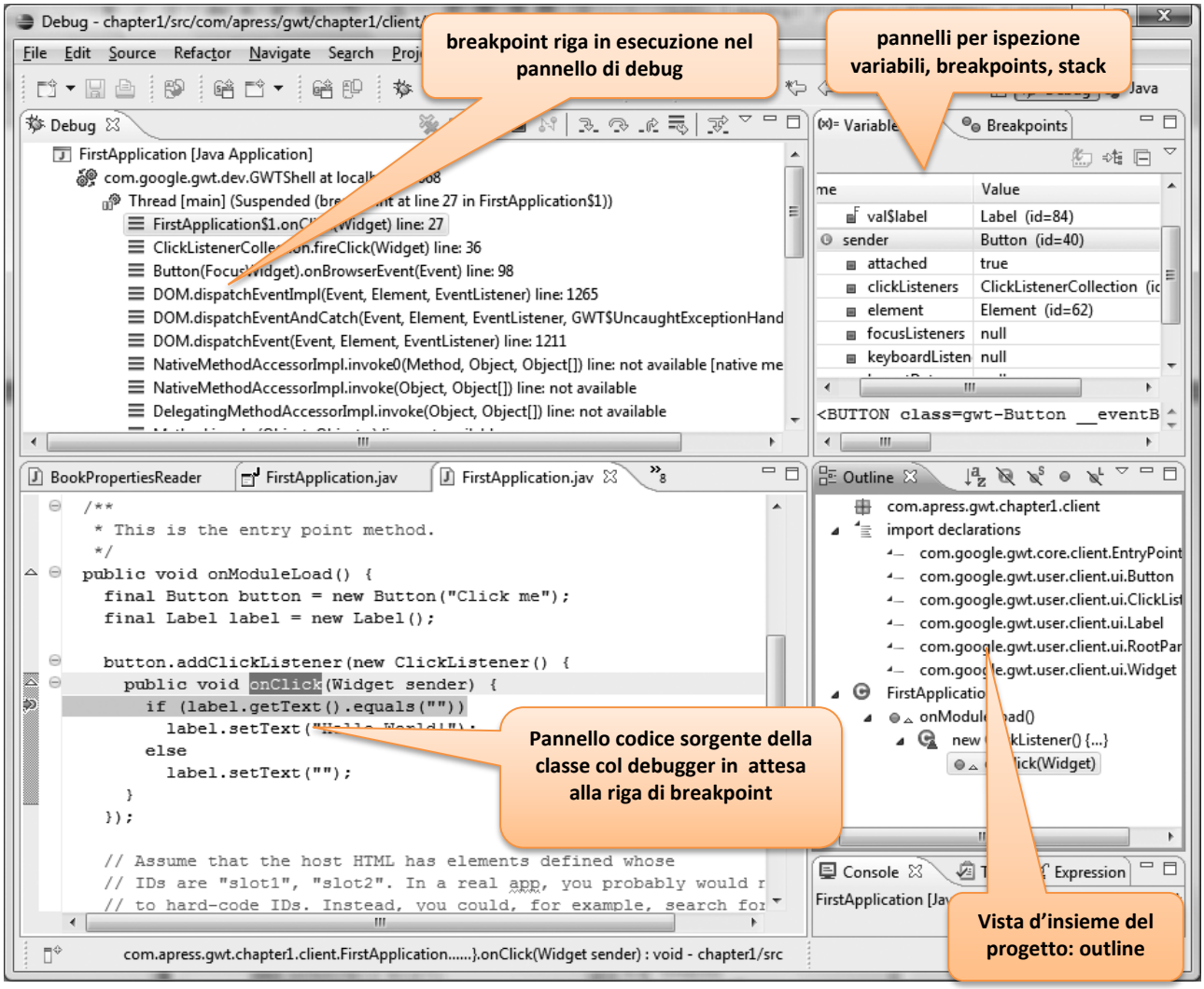


Figura 16 - Debugging di una applicazione GWT in Eclipse in modo *hosted*[41]

Questa figura, inoltre, rappresenta una tipica IDE (vedi pag. 35) *moderna* per quanto può considerarsi *moderno* un software del 2006. Oggi, ovviamente, l'IDE Eclipse ha progredito notevolmente con molte funzioni aggiuntive, ma lo schema di base è lo stesso.

La vera novità dell'approccio di Google nello sviluppo di *web-application* sta tutta nella lettera "A" dell'acronimo ***AJAX*** (vedi più avanti a pag. 58).

Google riuscì, con GWT, ad *appianare* le differenze nella gestione delle chiamate RPC asincrone, con l'*api XMLHttpRequest* non standardizzata nei vari browsers.

Ho sperimentato per qualche settimana lo sviluppo con GWT che non mi convinse del tutto. Pur apprezzando la semplificazione rispetto a Java EE, l'ho trovato ancora "sovradimensionato" rispetto alle mie esigenze, e, a parte questo, non mi convinse molto la gradevolezza dell'interfaccia grafica o *User Interface* (UI).

[41] Vedi Vipul Gupta, New York, Apress, 2008, p. 27

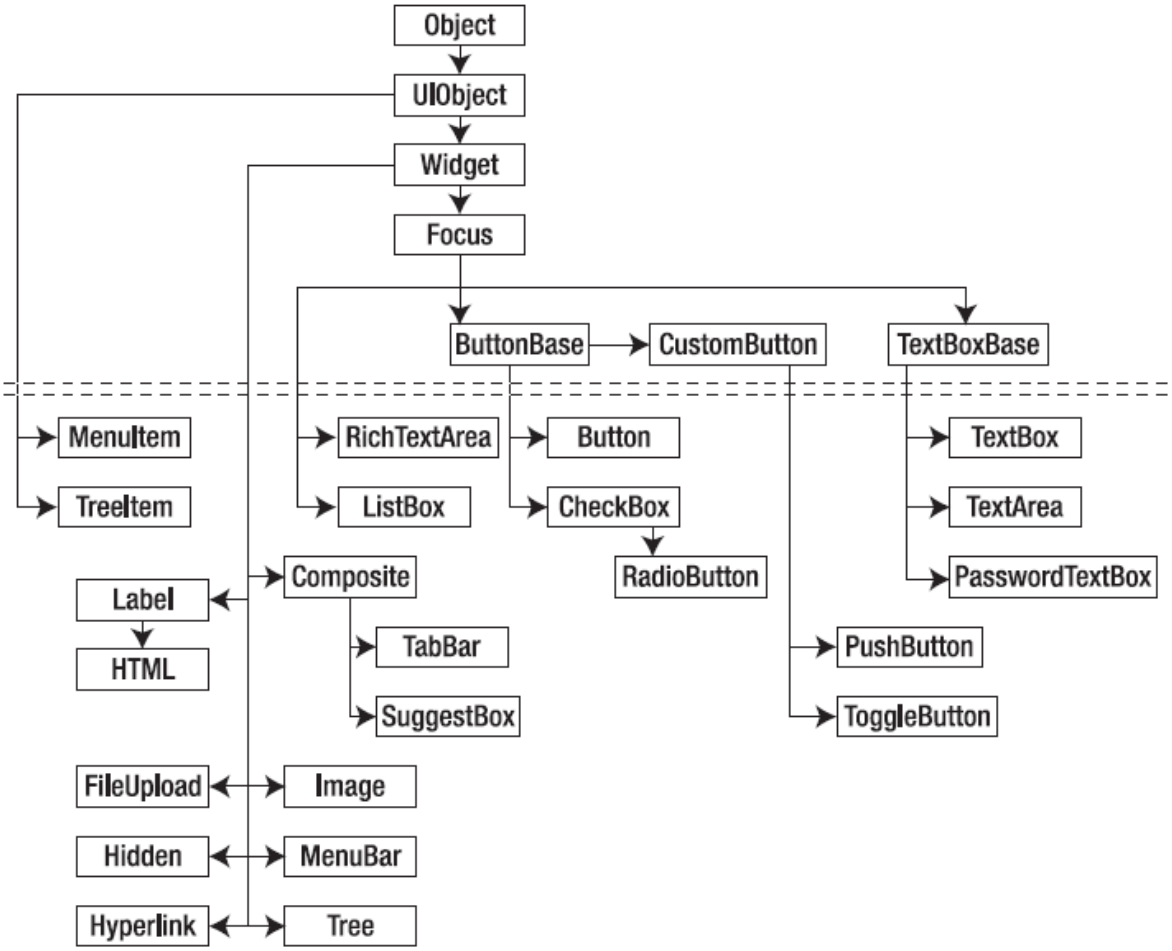


Figura 17 - Widgets in GWT

La figura[42] riporta l'albero completo delle classi che *implementano*[43] l'interfaccia grafica utente. Al di sopra della linea tratteggiate sono classi astratte e sotto classi effettive.

È vero che ogni componente può cambiare il suo stile in modo piuttosto semplice con il metodo *setStyle* ma è anche vero che gli stili pre-confezionati non erano molto gradevoli.

Creare da zero fogli di stile (vedi pag. **Errore. Il segnalibro non è definito.**) che trasformino una pagina/applicazione Internet da *piatta* ad *accattivante* è la chiave, nelle strategie di marketing, per *bloccare* il visitatore il più a lungo possibile su una certa pagina.

Una *bella* pagina internet è il frutto di un mix di tecnologie, gusto, arte ed altro. Ed una parte fondamentale è la stilizzazione che può richiedere molto tempo.

[42] Vedi Vipul Gupta, New York, Apress, 2008, p. 82

[43] Da quando Treccani ha *sdoganato* il verbo "implementare" (http://www.treccani.it/vocabolario/implementare/) mi sento meglio. I linguaggi sono vivi ed i vocabolari devono riflettere il più possibile la realtà attuale. Non posso però fare a meno di pensare alle discussioni degli accademici sull'opportunità o meno di considerare *to implement* e la sua italianizzazione appunto in implementare. In inglese significa infatti: "rendere effettiva una decisione". Secondo me è un po' azzardata ma non posso fare a meno di inchinarmi di fronte al cognome Treccani.

Forse era questo l'aspetto un po' trascurato nella versione iniziale del tool GWT.

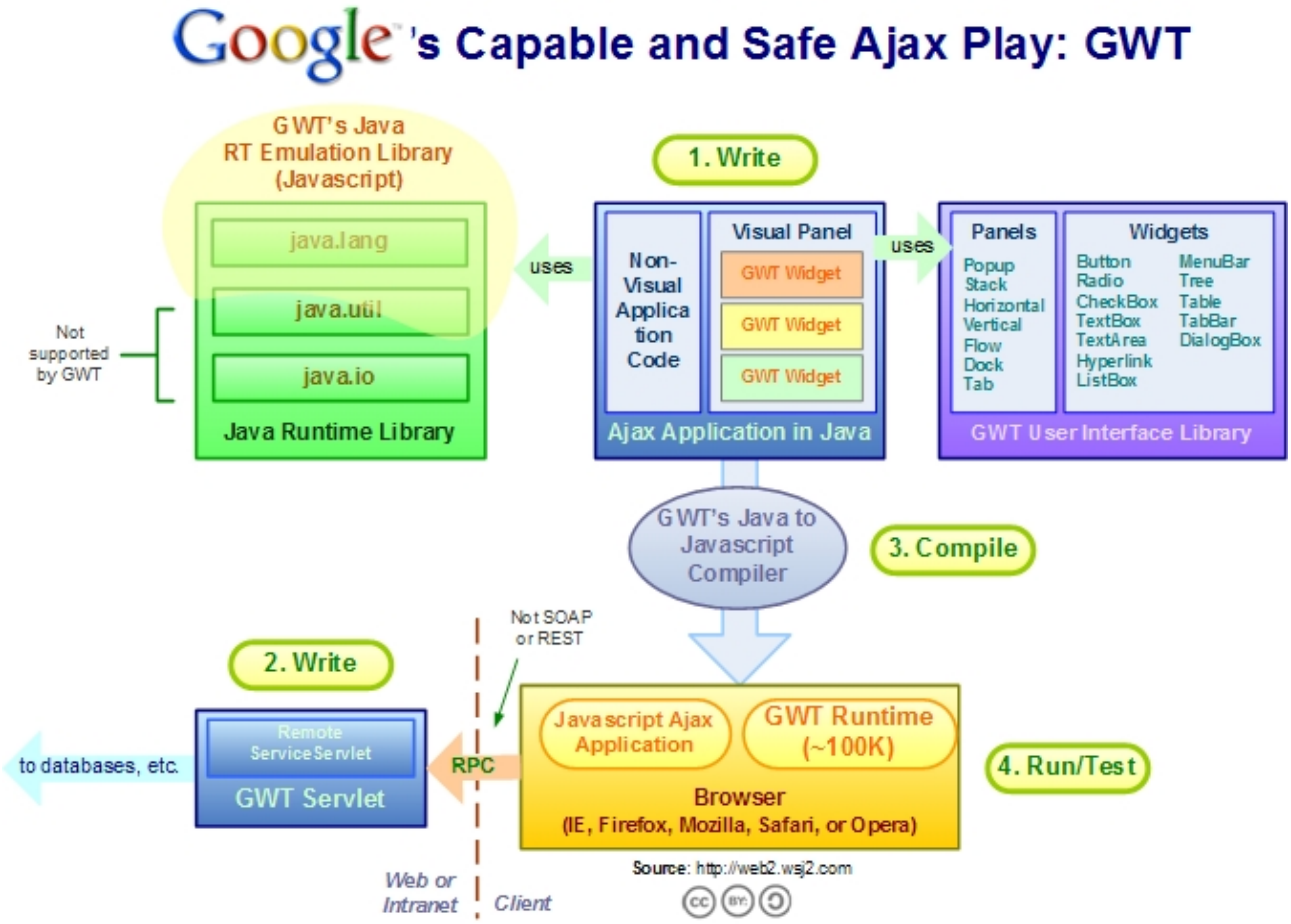


Figura 18 - Sguardo riassuntivo d'insieme su GWT[44]

[44] Vedi http://images.51cto.com/files/uploadimg/20091028/1049431.jpg

4.5.1.5 Vaadin[45]

Lo sviluppo di questo framework iniziò nel 2002 utilizzando come base di partenza un altro framework precedente (Millstone 3). Come punti chiave aveva, ed ha, AJAX *client-side* per la comunicazione e per il *rendering* dell'interfaccia grafica. Nel 2007 l'*engine* di *rendering* proprietario fu abbandonato in favore di GWT perché semplificava lo sviluppo di nuovi *widgets* rispetto alla soluzione proprietaria.

Rientra di diritto nella lista[46] dei *frameworks RIA*[47] e, come GWT, che estende, si basa sul modello di programmazione lato server (*server-side programming model*). Notevole salto in avanti rispetto a GWT, specialmente, con l'ultima versione di pubblico dominio, Vaadin 7.3.5.

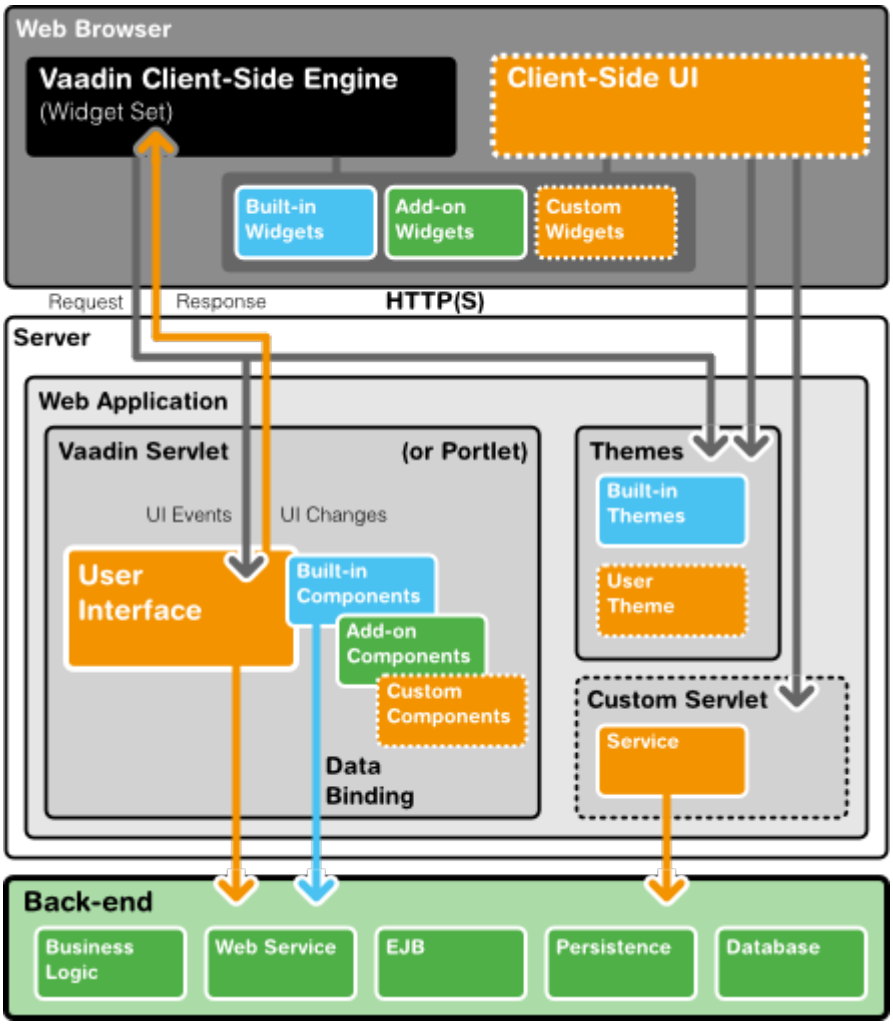


Figura 19 - Vaadin architettura generale

[45] È il termine finlandese per la femmina di renna: https://vaadin.com
[46] http://en.wikipedia.org/wiki/List_of_rich_Internet_application_frameworks
[47] Rich Internet Applications – Applicazioni internet arrichite

A differenza di GWT, il framework cura particolarmente l'aspetto della presentazione dei dati integrando attualmente la tecnologia SASS per lo sviluppo di temi personalizzati in modo semplice ed efficace.

Un ottimo framework basato su Java, con queste prerogative, come evidenziato nel sito:

- Framework basato su Java per sviluppare moderne applicazioni web che sono piacevoli da vedere ed usare
- Completamente open-source, sotto licenza Apache 2 License[48]
- Estendibile: i componenti compresi nel framework sono progettati per la facilità d'uso da parte dell'utente ma se non dovessero bastare ce ne sono altri 400 e più nella *directory* Vaadin[49]

La versione attuale è un enorme passo avanti rispetto a quella da me provata all'inizio del progetto, che non prevedeva, fra l'altro, l'integrazione con SASS.

Il progetto è in generale molto ben organizzato e spicca in particolare la documentazione che prevede un vero e proprio libro consultabile on-line gratuitamente o acquistabile in versione cartacea.

A questi livelli, il confronto Java⇔JavaScript si accende: è veramente un ottimo prodotto.

[48] http://www.apache.org/licenses/
[49] https://vaadin.com/directory

4.5.1.6 PrimeFaces

Sviluppato a partire dal 2009 dalla software house turca, PrimeTek, accreditata al comitato internazionale *Java Community Process Expert Group*[50] che definisce le specifiche di **JSF** (vedi pag. 37) si è affermato nel corso degli ultimi 5 anni come miglior componente *UI* per *JSF*.

Partito nello stesso periodo del precedente Vaadin, ha acquisito popolarità crescente grazie anche ad articoli apparsi sul blog di Oracle[51], alla pubblicazione di vari libri, e molto altro ancora.

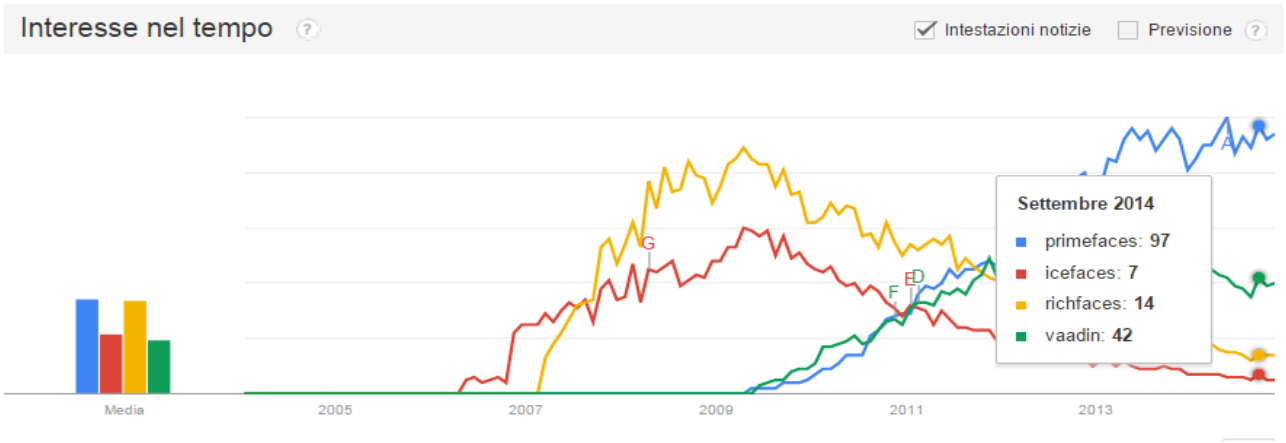


Tabella 4 – Trends Google search

Figura 20 – Esempio di emulazione di desktop Mac OS X

[50] https://jcp.org/en/home/index
[51] http://www.oracle.com/technetwork/articles/java/java-primefaces-pt2-2340750.html

```
<p:dialog header="About This Mac" draggable="true" widgetVar="info" showEffect="fade" hideEffect="fade">
    <h:form>
        <h:panelGrid columns="1" style="width:100%;text-align:center;">
            <p:graphicImage value="/images/macosx/apple_logo.png" />
            <h:outputText value="MOCK OS X" style="font-size:16px"/>
            <h:outputText value="Version 10.5.7" />
            <p:commandButton value="Software Update" async="true" onclick="return false;"/>
        </h:panelGrid>

        <h:panelGrid columns="2">
            <h:outputText value="Processor" style="font-weight:bold"/>
            <h:outputText value="2.4 GHz Intel Core 2 Duo" />

            <h:outputText value="Memory" style="font-weight:bold"/>
            <h:outputText value="4 GB 667 MHz DDR2 SDRAM" />

            <h:outputText value="Startup Disk" style="font-weight:bold"/>
            <h:outputText value="Macintosh HD" />
        </h:panelGrid>

    </h:form>
</p:dialog>
```

Tabella 5 - Esempio di dialogo in Primefaces

Questo frammento di codice implementa il pannello “About this Mac” della figura alla pagina precedente.

La tecnologia JSF combinata a *User Interfaces* come questa permette notevoli realizzazioni in modo molto semplice lato client. Richiede però molto più impegno lato server.

4.5.2 JavaScript

Il linguaggio nasce con il browser Netscape Comunicator col nome di Mocha che fu cambiato successivamente in LiveScript e quindi JavaScript.

“Java” in questa definizione è poco pertinente. È come dire che il gatto è un’evoluzione del cane. Certo, sono mammiferi a quattro zampe con una coda e due occhi frontali. Ma le somiglianze si fermano qui o vanno poco oltre: un gatto, se si offende, è capace di voltarti le spalle per ore fissando un muro. Il cane si appiattisce come uno zerbino o corre a nascondersi.

In questo caso il *brand* ha generato confusione portando a pensare che JavaScript si dovesse intendere come la versione interpretata del più *nobile* Java.

Ovviamente, nulla di più falso.

JavaScript si dovrebbe definire come un linguaggio **interpretato** basato su **prototipazione** con **tipizzazione dinamica** e **funzioni di prima classe**.

Credo che sia l’unico linguaggio multi-paradigmatico a supportare uno stile di programmazione **orientato agli oggetti**, **imperativo** e **funzionale**.

Un concetto importante da capire è che, a prescindere dalla tecnologia usata per costruire una pagina web, cioè che sia basata o meno, su JSF o puro html, l’unico linguaggio che fa “funzionare” il web moderno, dopo la dipartita delle applet java e di flex, è per l’appunto il JavaScript.

Come dettagliato nei paragrafi precedenti, i frameworks java preparano ed inviano al client, ***markups*** e **codice JavaScript**. Full stop[52].

A capo dell’albero genealogico dei codici di markups possiamo mettere senz’altro SGML che a sua volta discende da GML, definito nel 1970 da IBM.

Il General Markup Language, vero capostipite della famiglia, fu realizzato da IBM per un suo prodotto di scrittura, SCRIPT/VS con il compito di *marcare* i vari elementi del testo con lo scopo di poter stampare su laser o *line* printer o di visualizzare su monitor cambiando solo un *profilo* di stampa.

Il suo successore, divenne standard ISO e di qui l’aggiunta dell’aggettivo *standard*.

Tutti gli altri linguaggi di marcatura discendono in un modo o dall’altro da questi, compresi HTML ed XML e loro derivati.

[52] Con markups intento tutta la famiglia di linguaggi con marcatori, quali. HTMl, DHTML, SGML, XML, XHTML, MATHML ecc. Diamo per defunte le applets, e il tag object per l’inclusione è sicura fonte di guai

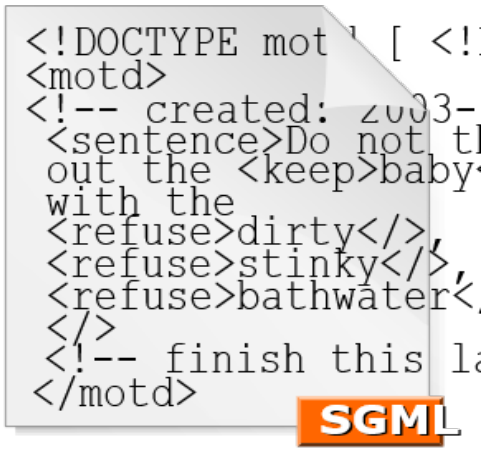


Figura 21 – Esempio di Standard General Markup Language

Detto questo vediamo più in dettaglio cos'è il JavaScript.

1. È un marchio registrato di Oracle Corporation
2. È standardizzato in ECMA-262[53] (ultima versione ufficiale, 5.1 in *draft* la 6.0 ed in preparazione la 7.0)
3. È **imperativo** e **strutturato**
   - Imperativo: gli *statements* del programma vengono eseguiti uno dopo l'altro
   - Strutturato: usa *strutture di controllo* quali *selezione* (**if-then-else, switch**) *cicli* (**for, while**), *sottoprogrammi* (**function)**
4. Ha **tipizzazione dinamica** delle variabili: il tipo di dato è associato al *valore* e non alla *variabile*
5. Basato sugli oggetti: gli oggetti sono *array associativi* integrati dalla prototipizzazione. I nomi delle proprietà degli oggetti sono stringhe di caratteri e chiavi di accesso. Si può fare riferimento ad un metodo/proprietà con la ***notazione punto***, persona.nome = 'Diego' o con la ***notazione parentesi quadre***: persona['nome'] = 'Diego'. Le proprietà possono essere quindi *aggiunte, cambiate o rimosse run-time*.
6. Interprete *run-time*: il linguaggio può interpretare una stringa come un programma con la funzione ***eval()***
7. Le funzioni sono ***cittadini di prima classe***: sono a tutti gli effetti degli oggetti e pertanto possono avere metodi ed attributi. Se una funzione è *nidificata* cioè contenuta in un'altra si parla di *chiusura lessicale* o ***closure*** poiché la funzione interna è creata ogni volta che viene invocata quella esterna. La cosa interessante che lo *scopo lessicale*, ***lexical scope*** della funzione esterna e cioè gli argomenti di chiamata, le variabili locali e le costanti diventano parte dello stato di ogni funzione interna anche dopo la conclusione dell'esecuzione della funzione esterna.

[53] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

8. Supporta le *funzioni anonime*, ***anonymous function*** altrimenti conosciute come *funzioni letterali* o *astrazioni lambda*: es.:
   var **add** = function **add**(a,b) { return a+b; }; // non è anonima, ha nome add e viene memorizzata in add
   var **add** = function (a,b) { return a+b; }; // è anonima, viene memorizzata in add
9. **Prototipi**, ***prototypes***: è il modo di JavaScript di intendere l'ereditarietà che in altri linguaggi è usata nelle classi
10. Funzioni come costruttori di classe: se la funzione viene prefissata con ***new*** verrà creata un'istanza di un prototipo che erediterà proprietà e metodi dal costruttore fino all'oggetto base **Object**
11. Normalmente lo script viene eseguito in un ambiente di esecuzione quale il browser che provvede oggetti e metodi con i quali può interagire ma con **node.js** è usato ***server-side*** **come server web ed altro**.
12. Supporta nativamente la sintassi Json
13. Supporta nativamente le espressioni regolari

```
<!DOCTYPE html>
 <meta charset="utf-8">
<title>Esempio</title>
 <h1 id="header">Esempio in JavaScript</h1>
 <script>
    document.body.appendChild(document.createTextNode('Hello World!'));
    var h1 = document.getElementById('header');
    h1 = document.getElementsByTagName('h1')[0];
</script>
 <noscript>Manca supporto a JavaScript o è inattivo.</noscript>
```

Figura 22 – Esempio di JavaScript in una pagina web

Con questa breve introduzione al JavaScript voglio solo rimarcare che questo linguaggio è molto, molto più potente, completo e complesso, di quanto comunemente si creda. Dopo due anni di continue prove non mi ritengo ancora completamente padrone di questa tecnologia ma solo poco più di un novizio, ad esempio, alcune particolarità del linguaggio, non aiutano di certo a fare chiarezza.:

var obj = {
"attr1" : "val1",
"attr2" : "val2"
};
	OK

var obj = {
attr1 : val1,
attr2 : val2
};
	OK, I nomi delle proprietà possono essere indicate senza apici se non vanno in conflitto con le parole riservate del linguaggio

var obj = {
"if" : "val1",
"else" : "val2"
};
	OK, i nomi delle proprietà sono racchiusi fra apici

var obj = {
if : "val1",
else : "val2"
};
	Errore: *if* e *else* sono parole riservate se inserite senza apici

Tabella 6 - Esempio di definizione di oggetto in JavaScript

Come detto sopra, tutti gli oggetti sono collegati ad un prototipo dal quale possono ereditare proprietà e metodi. Un metodo particolare è il metodo "prototype" che permette di definire

dinamicamente nuove proprietà. Se l'oggetto è definito da un *literal* eredita le sue proprietà e metodi dall'oggetto **Object**. Il JavaScript ha una serie di *oggetti predefiniti* e per interagire col browser:

CORE	HTML 5 ELEMENT			
• Strings • Numbers • Math • Date • Arrays • Booleans • RegExpl	• Abbreviation • Address • Anchor • Area • Article • Aside • Audio • BR	• Embed • Emphasized • Fieldset • Figcaption • Figure • Footer • Form • HGroup	• InputRange • InputReset • InputSearch • InputSubmit • InputText • InputTime • InputURL • InputWeek	• Parameter • Pre • Progress • Quote • S • Samp • Script • Section
BROWSER	• Base • Bdo • Blockquote	• HR • HTML • Head	• Ins • Italic • Kbd	• Select • Small • Source
• Window • Navigator • Screen • History • Location	• Body • Bold • Button • Canvas • Caption	• Header • Heading • IFrame • Image • InputButton	• Keygen • Label • Legend • Li • Link	• Span • Strong • Style • Subscript • Summary
HTML DOM	• Cite • Code	• InputCheckbox • InputColor	• Map • Mark	• Superscript • Table
• Document • Element • Attributes • Events	• Column • ColumnGroup • DFN • DList • DT • Datalist • Del • Details • Dialog • Div	• InputDate • InputDatetime • InputDatetimeLocal • InputEmail • InputFile • InputHidden • InputMonth • InputNumber • InputPassword • InputRadio	• Menu • MenuItem • Meta • Meter • Nav • Ol • Option • OptionGroup • Output • Paragraph	• TableData • TableHeader • TableRow • Textarea • Time • Title • Track • Ul • Underline • Variable • Video

Tabella 7 - Elenco oggetti predefiniti JavaScript

Ed anche:

PROPRIETÀ GLOBALI	FUNZIONI GLOBALI	
• Infinity • NaN • undefined	• decodeURI() • decodeURIComponent() • encodeURI() • encodeURIComponent() • escape() • eval()	• isFinite() • isNaN() • Number() • parseFloat() • parseInt() • String() • unescape()

Tabella 8 - Elenco proprietà e funzioni globali JavaScript

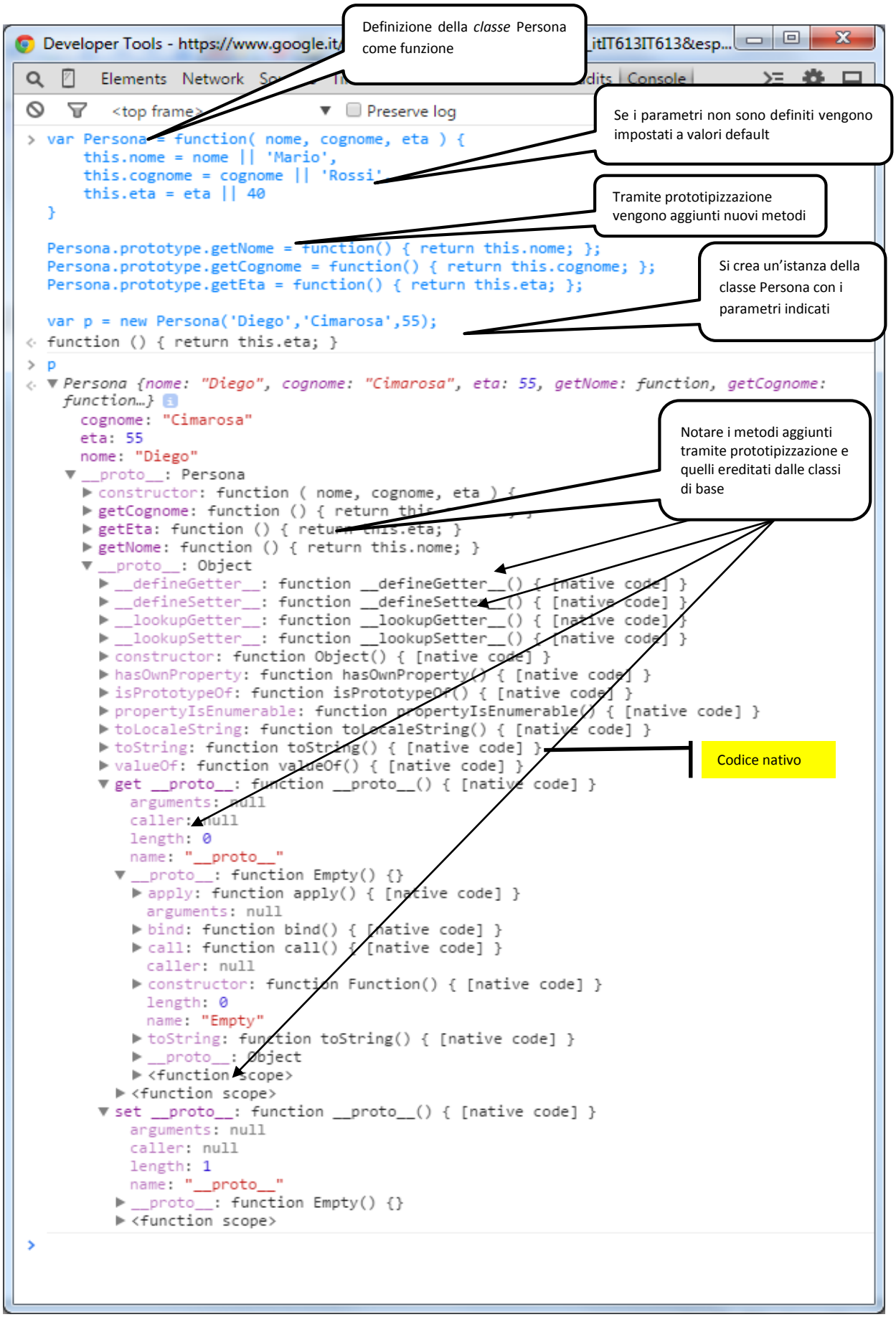


Figura 23 - Esempio di codice JavaScript nella console del browser

Nella pagina precedente riporto un esempio di JavaScript eseguito nella console del browser, in questo caso Chrome, evidenziando nei fumetti, come sia possibile aggiungere dinamicamente nuovi metodi con la prototipizzazione e una parte dell'albero delle classi.

In rilievo, con sfondo giallo, evidenzio che la chiamata alla funzione *toString* è *codice nativo* nel browser, e questo ovviamente, per ogni funzione dove il corpo è costituito da "{ [native code]}".

4.5.2.1 Considerazioni sulle prestazioni di JavaScript

Questo introduce una digressione sul tema della velocità di esecuzione del codice JavaScript.

La fama di linguaggio "lento" che ancora il JavaScript si porta addosso è dovuta alle scarse prestazioni dell'interprete di Internet Explorer 6 che era un'interprete *puro*, del tipo *line-by-line* con ovvie ricadute sulle performances.

Ripercorrendo l'evoluzione del browser Microsoft, che ricordo ancora, comportò un budget di circa 100.000.000$ per anno di sviluppo, dal 1995 al 2003, vediamo che l'introduzione del linguaggio di script ***JScript*** fu nel Marzo 1996 nella versione IE 3.0 Alpha 2.

JavaScript era un marchio registrato di Sun e Microsoft per problemi legali sviluppò il suo interprete JavaScript che aderiva alle specifiche ECMA-262 con alcune estensioni e supportando inoltre la sua tecnologia proprietaria di *Active Scripting* (o *ActiveX*[54] *Scripting*) che permette l'integrazione di oggetti tramite *OLE Automation*.

Fino alla versione IE 8, l'interprete incorporato di JavaScript non cambiò molto, ma dalla versione IE 9, Microsoft introdusse, sotto la pressione dei browsers rivali, Chrome, Firefox e Opera, un nuovo *engine* denominato *Chakra* che supportava la compilazione *Just In Time* e molto altro[55].

Similmente, i principali *competitors* rilasciarono vari *engines*:

- *V8 – Google Chrome, node.js*
- *SpiderMonkey – per la Mozilla, Firefox*
- *SquirrelFish (o Nitro) – Apple WebKit in Safari*
- *Tamarin – Adobe Flash*
- *Chakra – Microsoft IE9*

L'engine V8 è un **vero compilatore** in quanto compila il sorgente JavaScript in codice nativo per la CPU ospite (IA-32, X86-64, ARM, MIPS) che viene eseguito dal browser ed ottimizzato e ri-

[54] Evoluzione delle tecnologie Microsoft: OLE 2.0 e COM. Con Internet Explorer 3.0 una pagina che conteneva una tag *object* che referenziava un oggetto ActiveX poteva scaricarlo ed installarlo con interazione minima da parte dell'utente. Le critiche più forti riguardavano la sicurezza e l'impossibilità di utilizzo in sistemi non Microsoft

[55] Accesso alla GPU, 3D ecc.

ottimizzato dinamicamente tramite tecniche euristiche di profilazione dell'esecuzione del codice run-time, *inling expansion* ed altre avanzate tecniche.

L'aspetto degno di nota del *JavaScript engine* di Google è che è alla base di un (relativamente) nuovo ed innovativo server per il **real-time**.

Il tema delle performances del JavaScript è *hot-topic* ed è facile intuirne il motivo. Più difficile è concordare un unico indice che misuri in modo oggettivo le prestazioni; tema del resto spinoso in ogni ambito tecnologico.

Ad esempio: dopo 14 anni di giudizio[56], una *class action*[57] di una cittadina americana, tale *Janet Skold* ed altri, intentata contro Intel Corporation e Hewlett-Packard Company, perché secondo la querelante, Intel aveva manipolato i *benchmarks* del processore Pentium IV ed HP taciuto a riguardo, si sentiva ingannata nella scelta dei migliori processori AMD dell'epoca, e pretendeva un rimborso che fu riconosciuto nella somma di 15$.

Agli americani, evidentemente le "bugie", politiche o commerciali, proprio non piacciono.

Probabilmente, un giudizio che è arrivato dopo 14 anni, non può incidere in alcun modo oggi ma se ci fosse stato un sistema "garantito ed universale" allora per la misurazione delle prestazioni delle CPU (e della durata delle lampadine, degli alzacristalli delle autovetture e di qualunque altro prodotto industriale) forse Intel non sarebbe oggi leader mondiale con circa 53Mld di dollari di fatturato contro i 5.3mld di AMD

Se è difficile paragonare le prestazioni di un *hardware*[58] noto e misurabile, figuriamoci di un software. Lo stesso algoritmo può essere implementato in molti modi diversi in funzione dell'abilità e delle conoscenze del programmatore ma anche dell'ambiente in cui il programma verrà eseguito e del contesto in cui opera.

A mio parere, Java e JavaScript globalmente si equivalgono nelle prestazioni, e sono poco distanti dal C++, inadatto però al Web e come tale destinato a scomparire.

[56] Tanti anche per i tribunali italini!
[57] Dettagli del caso qui: https://www.intelpentium4litigation.com/Home/CaseDocs
[58] Siamo sicuri che la Ferrari non vince perché è inferiore alla Mercedes o che Alonso c'entri in qualche modo?

4.5.2.2 AJAX

L'acronimo AJAX significa "Asynchronous JavaScript And XML", in italiano "JavaScript asincrono ed XML". Il termine compare per la prima volta nel 2005[59] definendo l'insieme delle tecnologie che permisero a Google di rilasciare Google Maps, Gmail ed altri servizi innovativi anche se la tecnologia alla base di Ajax apparve nell'applicazione Outlook Web Access nel 1998.

Infatti, quell'applicazione fu possibile quando Microsoft introdusse in Internet Explorer 5 un'oggetto ActiveX (v. nota 54) che permetteva l'accesso all'interfaccia[60] IXMLHTTPRequest, implementata da Microsoft per la seconda versione delle libreria MSXML[61], usando il wrapper XMLHTTP.

```
var xmlHttpReq = new ActiveXObject("MSXML2.XMLHTTP.6.0");
xmlHttpReq.open("GET", "http://localhost/books.xml", false);
xmlHttpReq.send();
WScript.Echo(xmlHttpReq.responseText);
```

Figura 24 – Chiamata XMLHttpRequest attraverso componente ActiveX

Altri browsers come Firefox, Opera e Safari implementarono questa tecnologia come oggetto JavaScript denominandolo XMLHttpRequest. A dispetto del nome, siamo nel 2000 e lo standard era XML, le richieste http potevano e possono essere espresse in formato Json o semplice testo.

Il concetto chiave della tecnologia AJAX è semplice: comunicare col server in background senza interferire con lo stato corrente della pagina. Nel complesso erano/sono usate queste tecnologie:

- HMTL (o XHTML) e CSS per la presentazione dei dati
- Il DOM (Document Object Model) per interagire con i dati
- XML per lo scambio di dati e XSLT per il trattamento
- L'oggetto XMLHttpRequest per la comunicazione asincrona
- Il JavaScript come collante fra tutte

Oggi si preferisce spesso una struttura dei dati più leggera rispetto all'XML, nativa di JavaScript e denominata JSON, per l'appunto JavaScript Object Notation.

La novità di questa richiesta dati al server in background, fu la possibilità di aggiornare parti della pagina senza richiederne una totalmente nuova. Oltre ad alcuni problemi che espongo dopo, questa tecnologia affermò JavaScript come asse portante del Web.

E' facoltà comunque dell'utente di attivare o meno, nel browser, il supporto per JavaScript.

[59] Articolo di J.J. Garret: http://www.adaptivepath.com/ideas/Ajax-new-approach-web-applications/

[60] Si intende un'*interface*, ovvero una classe che definisce metodi ed attributi che vengono implementati da un'altra classe. In C++ un'interfaccia a metodi definiti come *virtual* and in C# la classe viene definita con *interface*

[61] Microsoft XML Core Service: è un set di servzi che permettono ad applicazioni scritte in C, C++,JScript, VBScript di realizzare applicazioni native Windows basate su XML

Alcuni problemi legati alla tecnologia Ajax:

- Problemi di indicizzazione nei motori di ricerca
- Difficoltà nell'uso dei *bookmarks*
- Impossibilità di funzionare se l'utente disattiva il supporto JavaScript o nei browser che non implementano XMLHttprRequest
- Blocco delle chiamate da siti diversi per la "regola della stessa origine" *Same origin policy*
- Maggior complessità dell'applicazione

A parte l'ultimo punto, gli altri sono stati affrontati ed in vari modi risolti e non rappresentano difficoltà insormontabili; la natura *asincrona* di queste richieste, in effetti, qualche grattacapo lo può creare.

Immaginiamo una web-applicazione in cui un operatore, per completare una certa azione, abbia bisogno del contenuto di alcuni files memorizzati sul server quali, ad esempio, dei messaggi email inviati dagli agenti di vendita e strutturati secondo una convenzione standard per essere elaborati automaticamente a richiesta del client. Un approccio *naif* potrebbe essere questo:

```
FUNZIONE CHIUSURA-CASSA
        n = 1;
        ripeti : leggi messaggio n sul server
                se fine-messaggi
                        esci
                altrimenti
                        estrai dati dal messaggio;
                        aggiorna risultati;
                        n = n + 1;
                fine-se
        fine-ripeti;
FINE-FUNZIONE-CHIUSURA-CASSA
```

In un mondo *sequenziale* il pseudo-codice illustrato funziona, almeno a livello logico, ma quando si richiama un servizio asincrono quale XMLHttprequest, no.

Se "leggi messaggio n sul server" è una richiesta asincrona, vuol dire che il *thread* principale gestito dal browser e che ospita la nostra pagina web col nostro codice JavaScript, continua la sua esecuzione *indipendentemente* dalla richiesta avviata al server. Nell'esempio sopra può portare ad un crash del browser, ad un loop infinito o ad altri risultati strani.

Una soluzione migliore è questa:

```
FUNZIONE CHIUSURA-CASSA
        ripeti: leggi messaggio sul server
                effettua la chiusura lessicale in CALLBACK
        fine-ripeti
FINE-FUNZIONE-CHIUSURA-CASSA

CALLBACK: parametri dal server
        Se fine-messaggi
                Segnala all'operatore
        altrimenti
                Estrai dati dal messaggio
                Aggiorna risultati
        Fine-se
FINE-CALLBACK
```

Quello che succede in questo caso è che la funzione "FUNZIONE CHIUSURA-CASSA" terminerà il proprio *loop* in un tempo X (pochi milionesimi di secondo) e dopo un tempo Y (dove Y >> X) arriveranno alla funzione CALLBACK i dati inviati dal server. Spesso, nella realtà, non si sa a priori quante chiamate al server sono necessarie ed in tal caso è necessario effettuare una prima *closure* per determinarne il numero ed una seconda per effettuare le richieste.

Una soluzione elegante al problema di sincronizzare le chiamate asincrone o di parallelizzarle a piacere è quella di utilizzare il modulo **async.js**[62] pensato per il server node.js, ma utilizzabile tranquillamente lato client.

[62] https://github.com/caolan/async

4.5.2.3 jQuery ed altri frameworks JavaScript

Dopo l'introduzione di Ajax, le pagine Web diventano sempre più dinamiche, tendendo sempre di più alla facilità d'uso ed interazione al quale ci hanno abituato le applicazioni desktop.

Nuovi componenti quali il *canvas* e più avanti le *websocket*, *webrtc* per arrivare al 29 Ottobre 2014 al rilascio definitivo delle specifiche di HTML 5 hanno da un lato reso enormemente più ricca la programmazione web e dall'altro, ovviamente, molto più complessa.

Per cercare di facilitare il lavoro degli sviluppatori, sono nati moltissimi progetti open-source o commerciali in cui la gara per diventare "lo" strumento di sviluppo, è sempre più accesa.

Fra questi vorrei citare:

- Angular.js: https://angularjs.org/

  Progetto open-source di Google è da intendersi come *web application framework*.
  Orientato alle sfide lanciate dalle moderne *one-page-application* implementa un'architettura MVC basata sull'uso di direttive simili a quelle già viste per JSF a pag. 37 (par. 4.5.1.1) e sul supporto del *data-binding* bidirezionale.

- Dojo Toolkit: http://dojotoolkit.org/

  Rilasciato sotto licenza BSD o Academic Free License, è sviluppato da un'organizzazione no-profit, The Dojo Foundation, sin dal 2004.

  Suddiviso in vari moduli (dojo: core system, dijit: UI, dojox: moduli sperimentali, util: vari) include *widgets* e *componenti* per migliorare l'aspetto delle pagine web ed un sistema di *packaging* per facilitare la modularizzazione dei componenti. Ha un *data-package* per gestire *datastores* sul server piuttosto specializzati, quali:

  - CsvStore: lettura files *comma separated values* (export di Excel)
  - OpmlStore: lettura dati gerarchici da files OPML[63]
  - YahooStore: lettura dati dal servizio web Yahoo Service
  - DeliciousStore: lettura bookmarks dal servizio web del.icio.us
  - RdfStore: lettura-scrittura di dti RDF[64]

  Ha inoltre la particolarità di poter essere utilizzato in applicazioni JavaScript basate su Adobe Air.

  Ha fama, come del resto Angular.js di essere piuttosto difficile da utilizzare

[63] Outline Processor Markup Language: formato XML per gi *outlines* o *strutture*
[64] Specifica W3C: Resource Description Framework => metadata data model

- JQuery: http://jquery.com/

  Come riportato in home page: JQuery è una libreria JavaScript piccola, veloce e ricca di funzionalità. É considerata la libreria JavaScript più diffusa a livello mondiale. Licenziata sotto la Mit License, è completamente free ed open-source.

  La sintassi è molto semplice per le query DOM, creare animazioni gestione degli eventi ed in genere lo sviluppo di applicazioni Ajax. È stata inclusa da Microsoft in VisualStudio per essere utilizzata nel framework ASP.NET Ajax.

  Pur restando sostanzialmente un tool per la manipolazione del DOM, può essere arricchita di funzionalità tramite *plugins*.

```
$(document).ready(function(){
    $('img').click ( function() {
      // handle the click event on any img element in the page
    });
})
```

Figura 25 - Esempio di codice JQuery

  Come riporta il commento, questo semplice codice, reagisce al click effettuato su qualsiasi immagine presente sulla pagina (definita cioè nel tag <img>).

Purtroppo non è pensabile di esaminare in dettaglio tutte le librerie/frameworks esistenti poiché sono semplicemente troppe. Ho riportato solo quelle che ho approfondito dopo lunghe ricerche e prove.

4.5.2.4 Flex

Flex 1.0 fu rilasciato nel 2004 da Macromedia come piattaforma server di livello enterprise per lo sviluppo di applicazioni *data-driven* su tecnologia Flash e con un costo *enterprise*.

Basato su due (+1) elementi principali:

- MXML[65] per la UI
- ActionScript[66]
- Flex Server/Coldfusion/BlazeDS/LiveCycle lato server

L'ultima tipologia di server, LiveCycle, distribuita come applicazione Java EE, permetteva ad un'applicazione Flex sul client di:

- invocare metodi definiti negli oggetti Java remoti
- implementare il *pattern* 'sottoscrivi' e 'pubblica/sottoscrivi', ovvero: il client Flash può pubblicare eventi verso un argomento definito sul server e sottoscrivere eventi trasmessi dal servizio di messaggistica. Il caso più comune è lo streaming real-time di informazioni finanziarie
- *bind* di *data sets* trasmessi al client. L'applicazione può sincronizzare in automatico i dati locali col server e viceversa.
- Generazione di documenti PDF combinando i dati clienti o grafici con moduli memorizzati sul server

Senza dubbio, la tecnologia Flash e di riflesso Flex per lo sviluppo, hanno rappresentato un punto di eccellenza delle tecnologie informatiche ed è difficile trovare un computer desktop, Windows, Apple o Linux, senza il plugin di Flash.

Ma nell'economia dell'Universo nulla è per sempre. Nasce e muore il Tempo, lo Spazio, l'Energia e la Materia … figuriamoci il software!

La fine di Flash, è noto, fu decretata dal post di Steve Jobs: "Thoughts on Flash" nell'Aprile del 2010: https://www.apple.com/hotnews/thoughts-on-flash/.

Ricorda molto un epitaffio, stile "Il cinque Maggio" di Manzoniana memoria.

In sostanza, in questo post, Steve Jobs rispedisce ad Adobe le accuse che Apple sia un sistema chiuso quando rimosse Flash dall'Apple store ribadendo che Flash è al 100% un sistema closed-source di cui solo Adobe ha le chiavi e che al contrario, ed è qui la vera ragione, Apple ha sposato la nuova tecnologia HTML5. Adobe sosteneva che il 75% dei video in Internet erano basati su Flash e Jobs affonda sostenendo che la quasi totalità dei video sono fruibili nel formato H.264 integrato

[65] Macromedia XML user interface markup language
[66] Da considerare come estensione del JavaScript

nei prodotti Apple. Sul fatto che iPhone ed iPad non possano far girare i giochi Flash Jobs è, ovviamente, d'accordo, ma 'tocca' con decisione ribadendo che su Apple Store ci sono 50.000 giochi per lo più gratuiti. Citando i reports di Symantec del 2009 in cui si afferma che Flash è il problema più grosso per la sicurezza e che ha scarse prestazioni sui dispositivi mobili, segna sicuramente un punto. Un altro lo guadagna dimostrando semplicemente che la decodifica di un video fatta da Flash via software è molto più onerosa per la durata della batteria di quello realizzato via hardware da chip dedicati nello standard H.264. Continua con una lucidità micidiale, scrivendo che i dispositivi mobili funzionano con le dita delle persone, *touch screen*, e che i siti web che usano flash devono essere riscritti per supportare i dispositivi mobili. A quel punto perché non usare le tecnologie moderne quali HTML, CSS e JavaScript? Secondo lui, però, il punto più importante è la disponibilità di tools di sviluppo adeguati ai tempi e non vuole che gli sviluppatori per iPhone … iQualcosa, dipendano da altri vendors.

La chiosa finale è da riportare per intero:

"Forse Adobe dovrebbe focalizzarsi di più nel creare grandi tools HTML 5 per il futuro, e meno nel criticare Apple che si sta gettando il passato alle spalle"

IPSE DIXIT.

Talmente forte che Adobe cambiò rotta di 180°, donò Flex all'Apache Foundation … e si mise di gran lena a realizzare "grandi tools HTML 5".

Peccato! A me Flash piaceva e l'ho sperimentato per circa due mesi …

4.5.2.5 ExtJS

Nel corso di questa lunga sperimentazione ho incontrato o per meglio dire mi sono *s*contrato con questo framework verso la fine del 2010. Gli esempi disponibili sul sito erano semplicemente fantastici lasciando intravedere grandi possibilità di sviluppo. Rimasi stupito all'inizio quando approfondii i primi esempi:

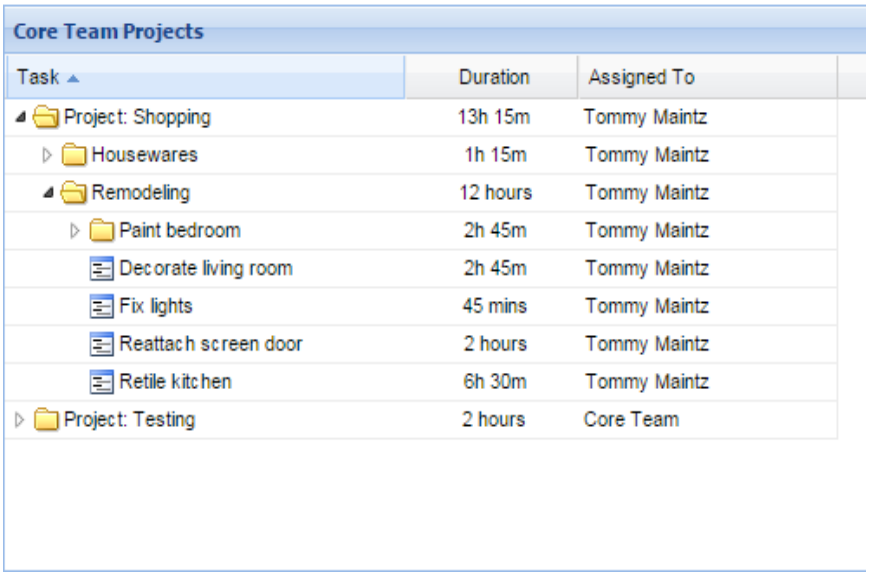


Figura 26 - Esempio di TreeGrid in ExtJs

Ovviamente non è possibile riprodurre l'interazione della *tree-grid* su carta ma effettuare il *drag-n-drop* dei nodi col mouse con la gestione dell'auto-scroll mi sembrava una funzionalità decisamente avanzata. Non conoscendo affatto il framework pensavo che fosse simile a dojo o jQuery: la pagina index.html avrebbe riportato la struttura della tabella ed il javaScript ne gestiva l'interattività.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>TreeGrid Example</title>
    <!-- ext CSS includes -->
    <link rel="stylesheet" type="text/CSS" href="../../resources/CSS/ext-all.CSS" rel="stylesheet" />
    <link rel="stylesheet" type="text/CSS" href="../ux/treegrid/treegrid.CSS" rel="stylesheet" />
    <!-- ext JavaScript includes -->
    <script type="text/JavaScript" src="../../adapter/ext/ext-base-debug.js"></script>
    <script type="text/JavaScript" src="../../ext-all-debug.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGridSorter.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGridColumnResizer.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGridNodeUI.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGridLoader.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGridColumns.js"></script>
    <script type="text/JavaScript" src="../ux/treegrid/TreeGrid.js"></script>
    <script type="text/JavaScript" src="tree-grid.js"></script>
  </head>
  <body style="padding: 50px;"></body>
</html>
```

Figura 27 - Index.html per esempio 26

Sorpresa! Dov'è il codice HTML? Cosa sono tutti quei files *.js ?

```
/*!
 * Ext JS Library 3.3.1
 * Copyright(c) 2006-2010 Sencha Inc.
 * licensing@sencha.com
 * http://www.sencha.com/license
 */
Ext.onReady(function() {
    Ext.QuickTips.init();

    var tree = new Ext.ux.tree.TreeGrid({
        title: 'Core Team Projects',
        width: 500,
        height: 300,
        renderTo: Ext.getBody(),
        enableDD: true,

        columns:[{
            header: 'Task',
            dataIndex: 'task',
            width: 230
        },{
            header: 'Duration',
            width: 100,
            dataIndex: 'duration',
            align: 'center',
            sortType: 'asFloat',
            tpl: new Ext.XTemplate('{duration:this.formatHours}', {
                formatHours: function(v) {
                    if(v < 1) {
                        return Math.round(v * 60) + ' mins';
                    } else if (Math.floor(v) !== v) {
                        var min = v - Math.floor(v);
                        return Math.floor(v) + 'h ' + Math.round(min * 60) + 'm';
                    } else {
                        return v + ' hour' + (v === 1 ? '' : 's');
                    }
                }
            })
        },{
            header: 'Assigned To',
            width: 150,
            dataIndex: 'user'
        }],

        dataUrl: 'treegrid-data.json'
    });
});
```

Figura 28 - Modulo ExtJs TreeGrid

Dopo un po' capii che il modulo principale era questo e che i dati della griglia provenivano dalla riga in grassetto nella figura sopra: 'treegrid-data.json'

```
[{
  task:'Project: Shopping',
  duration:13.25,
  user:'Tommy Maintz',
  iconCls:'task-folder',
  expanded: true,
  children:[{
    task:'Housewares',
    duration:1.25,
    user:'Tommy Maintz',
    iconCls:'task-folder',
    children:[{
      task:'Kitchen supplies',
      duration:0.25,
      user:'Tommy Maintz',
      leaf:true,
      iconCls:'task'
……..
```

Ne riporto a lato un frammento per completezza.

Non capivo però dov'era finito il codice HTML. Dopo aver letto buona parte della documentazione, realizzai che in effetti veniva generato *run-time* dal framework ExtJs.

Altra cosa che mi lasciò molto perplesso fu il codice sorgente dell'esempio: ma che linguaggio è? Assomigliava un poco al JavaScript ma pensai che fosse un'estensione … assai estesa!

Questa volta avevo indovinato: ExtJs significa per l'appunto **Extended JavaScript**.

ExtJs viene definito come un framework applicativo in puro JavaScript ed ha come progenitore il progetto *Yahoo! User Interface Library* (**YUI**) che fu rilasciata come Open Source nel Febbraio del 2006. Breve cronistoria del progetto:

- Luglio 2006: Jack Slocum[67] fonda ExtJs
- Settembre 2006, prime versione beta, basate sul framework YUI
- Marzo 2007, v. 1.0: riscrittura completa eliminando il codice YUI. Sono presenti vari *widgets* ed un primo *component* per la grafica vettoriale
- Luglio 2008, v.2.0[68]: nuovi componenti, ciclo di vita ben definito
- Luglio 2009, v.3.0[69]: nuovi componenti e migliorie. Supporto per il protocollo **REST**[70] ed **Ext.Direct**, architettura per usare *client-side* metodi *server-side*. Il server può essere basato su diverse tecnologie: Java, PHP, ColdFusion
- Giugno 2010, ExtJs si fonde con altre due librerie (jQTouch per il supporto dei dispositivi mobile e Raphaël per la grafica vettoriale. La società cambia nome in Sencha Inc.
- Aprile 2011, v.4.0[71]: rivisitazione completa del framework. Supporto SVG e VML, introduzione dell'architettura **MVC**
- Ottobre 2013, Jack Slocum lascia ExtJs, viene rilasciato un plugin per Eclipse
- Giugno 2014, v.5.0: supporto mobile, MVVM, two-way-data-binding
- Agosto 2014: Yahoo annuncia ufficialmente di chiudere lo sviluppo di YUI

ExtJs viene rilasciato sotto licenza GNU GPL License V3 per progetti Open Source, licenza commerciale per essere proprietari del codice sviluppato e licenza OEM per chi vuole rivendere l'SDK. Progetti collaterali ad ExtJs:

- **Touch**: framework per il mobile sotto licenza GPLv3
- **GXT**: UI widgets integrabili con GWT per lo sviluppo in ambiente java
- **Architect**: ide per Windows, Mac e Linux per la creazione di applicazioni ExtJs/Touch. Disponibile in versione di prova. Richiede licenza commerciale
- **Animator**: tool sulla falsariga di Flash per la creazione di complesse animazioni in HTML/CSS. Richiede licenza commerciale
- **Cmd**: tools per la generazione dello scheletro di applicazioni, compilazione e packaging anche in codice nativo (mobile)
- **Space**: online *envinronment* per la gestione integrata delle applicazioni ExtJs/Touch. Richiede licenza commerciale

[67] Vedi: http://jackslocum.com/blog/
[68] Vedi: http://www.sencha.com/learn/overview-of-extjs-2#Component_Life_Cycle
[69] Vedi: http://www.sencha.com/learn/overview-of-extjs-3/
[70] Representational State Transfer: V.i tesi di dottorato di Roy Fieldin: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm e http://it.wikipedia.org/wiki/Roy_Fielding
[71] http://docs.sencha.com/extjs/4.0.0/#!/guide

4.6 Scelta del framework

Dopo questa lunga ricerca le idee su cosa usare per lo sviluppo si sono concretizzate in questi punti:

- Frameworks Open Source
- Vasta comunità a supporto
- Evitare assolutamente effetti lock-in
- Ambienti di sviluppo adatti ad uno/pochi sviluppatori
- Ambienti di "produzione" alla portata di budget ridotti
- Minor uso possibile di tecnologie diverse

Gli ultimi due punti sono stati decisivi nell'abbandonare la tecnologia Java in favore di JavaScript ed l'unico framework completo, dalla gestione dei temi dell'applicazione alla struttura delle classi, era ExtJs anche se con molti dubbi.

Il target di questo prodotto è la piccola/media software house che sviluppa applicazioni commerciali per conto terzi e non la vasta platea di singoli sviluppatori. Basti pensare che una licenza che dà diritto al supporto completo per il prodotto, parte da 4.825$ per un pacchetto di 5 licenze ed un anno di supporto.

Purtroppo, i tools aggiuntivi, che fanno parte delle licenza commerciale quali Sencha Architect, Sencha Space e Sencha Animator, sono disponibili in *trial* per un solo mese, quindi oltre al framework, lo sviluppo si è basato su un editor free (Notepad++) e gli ambienti di debug dei browsers Chrome, Internet Explorer 11 e Firefox. Quest'ultimo con l'ottimo plugin FireBug che offre molte funzionalità nel debug.

Lato server, rinunciando a Java, ed aver abbandonato Apache/PHP in favore di Node.js, ho usato come debugger, il modulo node-inspector e semplici display a console.

5 L’APPLICAZIONE

Per chiarezza riporto le definizioni corrette di alcuni termini fondamentali:

- programma: sequenza di codice in esecuzione nel computer. Quando non è in esecuzione è semplicemente un file memorizzato su hard-disk o qualche altro device. Si parla comunemente del file “Word.exe” e del programma “Word”.
- Programma applicativo: un programma od un insieme di programmi destinato/i a risolvere un qualche tipo di problema comunemente di tipo “gestionale”. Sinonimi: applicativo, applicazione, software. E’ importante la distinzione fra il singolo programma e la catena dei programmi. Per complicare le cose, un programma/software può fare uso di moduli e librerie
- App: programma per dispositivi mobile

Il software che presento qui è da intendersi come “applicativo web” perché è un insieme di circa 500 moduli JavaScript provenienti da diverse librerie (170 ***kloc)*** e 130 scritti da me (14 kloc) per un totale di 184.000 linee di codice circa. Trattandosi di files JavaScript e CSS il termine “compilazione” non è del tutto corretto perché a differenza di ciò che asserisce Google[72] riguardo al loro “closure compiler”, essendo il linguaggio oggetto lo stesso del linguaggio sorgente, non si ha “compilazione” nel senso storicamente dato a questo termine in ambito informatico, ma di “ottimizzazione” a meno che non si voglia intendere “better JavaScript” un linguaggio diverso da “JavaScript”.

Come riportato al capitolo precedente, la libreria principale di sviluppo è **ExtJs** con l’aggiunta delle seguenti:

- jQuery 2.1.1: http://jquery.com/
  Libreria compatta per facilitare la gestione degli eventi, della manipolazione del DOM del browser, le animazioni e molto altro. E’ richiesta da alcuni dei moduli aggiuntivi sotto riportati
- toastr.min.js: https://github.com/CodeSeven/toastr
  Modulo jQuery per la notifica di messaggi di applicazione. L’ho preferita rispetto all’equivalente della libreria base per il miglior impatto visivo del messaggio
- fabric.js: http://fabricjs.com/
  Libreria per la gestione del canvas ad alto livello. Permette una gestione completa di

[72] Vedi Closure Tools: https://developers.google.com/closure/compiler/

oggetti grafici. La utilizzo nel modulo GesVol.view.editor.FotoEditor.js

- CryptoJS v3.1.2: https://crypto-js.googlecode.com/files/CryptoJS%20v3.1.2.zip

  Modulo utility rilasciato da Google che implementa i principali algoritmi di crittografia correntemente utilizzati.

L’applicazione **GesVol** è da intendersi un ***work-in-progress*** perché si pone l’obbiettivo di riunire la gestione di quante più associazioni di volontariato possibile conservando, per ognuna di esse, le prerogative principali, ma riunendo in un unico archivio i dati dei volontari per poter offrire alle Autorità Pubbliche, in primo luogo la Protezione Civile, ma anche Guardie Forestali e Vigili del Fuoco un punto di informazione, di dialogo, di raccolta, disponibile via web.

5.1 Architettura

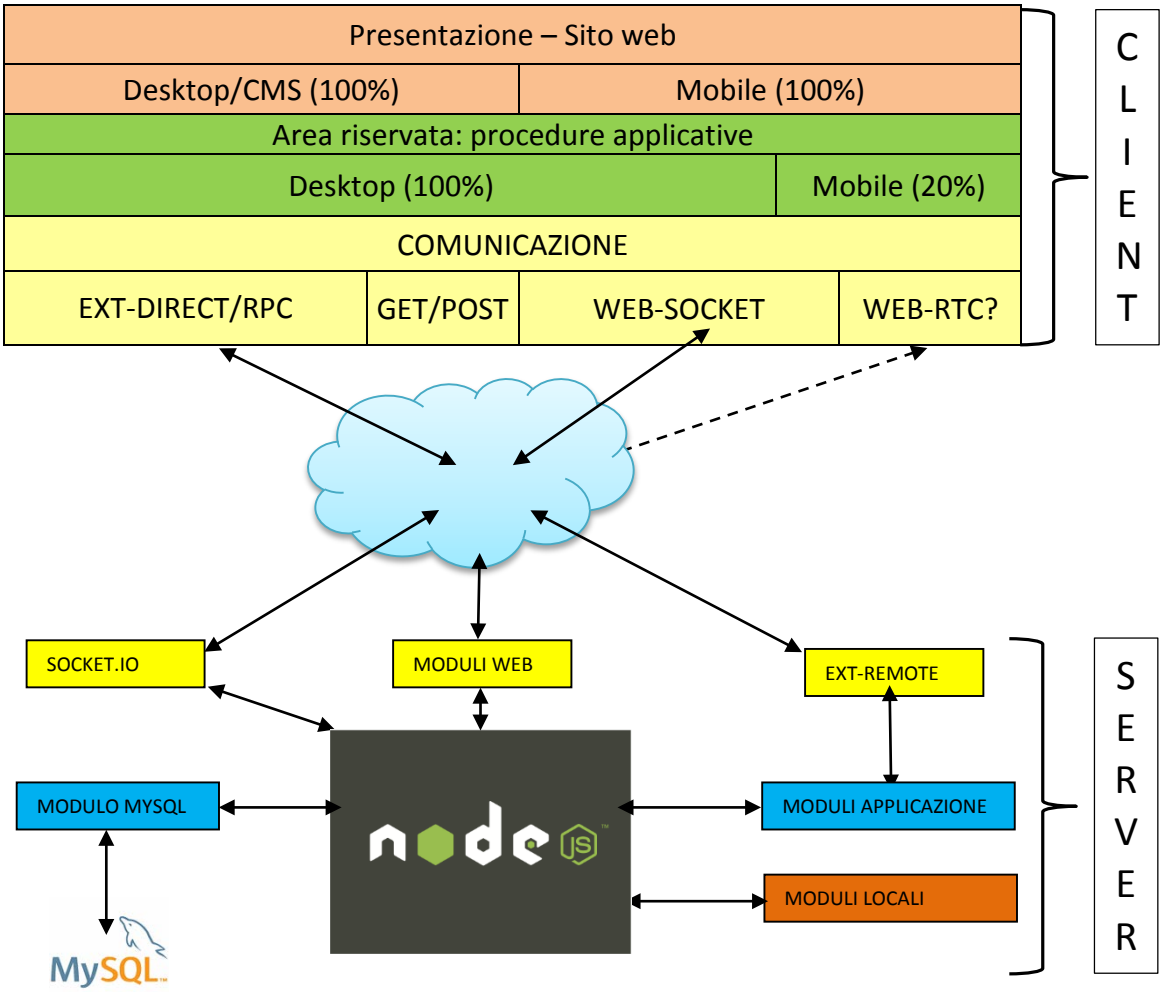


Figura 29 - Schema concettuale applicazione GesVol

Per soddisfare le esigenze fino qui esposte, ho suddiviso logicamente l’applicazione nel modo indicato in figura ed adesso dettaglio:

- Sezione client
  - Un *layer* di presentazione: sito web dove troveranno posto le pagine web dinamiche relative alle varie associazioni. Dalla Home Page, proposta automaticamente all’indirizzo dell’applicazione, l’utente potrà sfogliare le pagine di GesVol, vista come associazione virtuale, e cioè esistente solo in questa applicazione, pensata per raggruppare le informazioni comuni a tutte le associazioni, oppure *navigare* alla Home dell’associazione, selezionandola in vari modi. Le pagine, lette da DB, sono costruite con componenti ExtJs standard, scritti ad hoc o degli equivalenti Touch per il mobile o semplice codice Html. Sono comunque in due versioni: **D**esktop e **M**obile. Le percentuali del 100% significano semplicemente che tutte le pagine disponibili per la versione desktop saranno disponibili, opportunamente rivisitate, per la versione mobile.
  - Un *layer* applicativo riservato agli utenti registrati, in cui, ognuno, verrà inserito in uno o più gruppi e sulla base di questa appartenenza visualizzerà un menu personalizzato con le funzionalità proprie del suo ruolo/associazione-i. È previsto, come succede in realtà, che un utente possa essere socio di una o più associazioni ricoprendo ruoli/incarichi diversi in ciascuna di esse. Per non complicare eccessivamente la gestione ho pensato ad un elenco (combobox) dove l’utente seleziona l’associazione con la quale in quel momento si vuole collegare all’area riservata in modo da limitare la gestione del desktop a quattro casi:
    - Utente di una associazione (caso normale)
    - Utente *guest* o temporaneo o ospite
    - Utente Ente Pubblico
    - Utente aministratore

    Le percentuali del 100% e 20% significano che solo una frazione delle funzionalità previste per il desktop sarà realizzata anche per la versione mobile.

  - Un *layer* di comunicazione col server basato su tre componenti fondamentali ed una in forse:
    - GET/POST: sono le richieste “normali” effettuate da una pagina Html per richiedere od inviare dati al server. In questa applicazione servono in pratica solo a richiedere i files necessari alla stessa (CSS, immagini, icone, ecc.)
    - Ext-DIRECT/RPC: tutti i *data store* dell’associazione usano modelli che tramite *proxy* richiedono/trasmetto dati dal/al server via RPC. Mi è sembrata la scelta più flessibile in quanto si possono usare le chiamate proprio del modello o dello store (che richiamano via proxy i metodi remoti) o direttamente i metodi remoti. Questa flessibilità, dettagliata più avanti, ha portato diversi problemi in quanto i dati di risposta, vengono o meno rivisti dallo stack ExtJs per cui bisogna approntare procedure diverse
    - Web-socket: ad un certo punto dello sviluppo, avevo scritto dei moduli che intercettavano le chiamate alla XMLHttpRequest (v. par. AJAX a pag. 62) e le

ridirezionavano verso la web-socket. Vari esperimenti mi hanno convinto che sarà la tecnologia del futuro in quanto permette un miglioramento delle prestazioni da un minimo del 50% al 90%. Purtroppo, l’avvento della versione 5.0 di ExtJs ha complicato molto la gestione degli eventi legati alla preparazione dei dati prima dell’invio al server e quella successiva alla ricezione per cui ho deciso di rinunciare a web-socket come tecnologia primaria di trasmissione dati e di riservargli *solo* l’inoltro di alcuni messaggi particolari e di dati in certe situazioni.

        - Webrtc: dopo alcuno esperimenti mi sono arreso. Molta fatica con scarsi risultati. Vi è ancora poco supporto nel mobile

- Sezione server
    - La scelta di Node.js è stata indubbiamente un azzardo perché la programmazione, *totalmente asincrona* che implica, richiede un cambio di visione dei problemi a 180°. Non si può ragionare in modo sequenziale (imperativo) ma è necessario pensare in modo totalmente diverso: se faccio questo, allora, *prima o poi* otterrò quello. È il “prima o poi” che complica le cose.

      Come evidenziato in figura, ci sono tre componenti per l’interazione con i clients:

        - Moduli web: è una serie di componenti di NodeJs per gestire le richieste in arrivo, gli errori, il *parsing* delle richieste HTTP/HTTPS, *cookies* ed altro
        - Socket.io: modulo per la comunicazione via web-socket
        - Ext-remote: è l’interfaccia ExtJs per le Remote Procedure Call in JavaScript. Ne esistono altre per PHP, Java ed altre tipologie di server[73]

      Ed alcuni moduli:

    - Uno per l’interrogazione del DB MySql
    - Vari per interagire col sistema (lettura files e directory, *hash table* utenti, sessione ed altro)
    - Vari moduli che realizzano il *core-business* dell’applicazione. Anche se il Data Base è piuttosto articolato (32 entità al momento ed in crescita), sfruttando una particolarità del modulo sql su node.js ho potuto scrivere solo 10 moduli per la gestione di tutte le tabelle semplificando molto lo sviluppo della parte server

Nei prossimi paragrafi dettaglierò, funzione per funzione, l’applicazione GesVol.

[73] Vedi: http://www.sencha.com/products/extjs/extdirect

5.1.1 L’ambiente di sviluppo

Per lo sviluppo non ho optato per un ambiente locale, cioè un PC di mia proprietà, ma ho preferito affittare un *server virtuale* tramite un service provider italiano. Ovviamente è di fascia *entry-level*, 1 processore virtuale, 512Mb RAM e 15Gb di spazio disco. Su questa VM avevo installato la versione 12.04 LTS di Ubuntu aggiornando in seguito il sistema alla versione 14.04.1 LTS.

Le ragioni sono molteplici, ma innanzitutto la connessione permanente ad Internet e con banda più che sufficiente per i miei scopi sperimentali.

Dopo aver installato da remoto, con CPanel[74] fornito dal provider, ho installato il pacchetto XAMPP per Linux che racchiude in un'unica installazione i seguenti softwares:

- Web server Apache
- Data base MySql
- Ftp server Filezilla
- Mail server Mercury
- Java Server Tomcat
- PHP
- Perl
- phpMyAdmin

Tipico scenario di un’ambiente di sviluppo ‘casalingo’ anche se ospitato su server remoto.

Completano l’ambiente di sviluppo, le applicazioni **putty**[75] per accesso via linea di comando al server tramite *ssh* e l’editor Notepad++[76] con alcune estensioni.

Tutto, Open Source e free. Sicuramente il “migliore degli ambienti di sviluppo possibile”[77].

[74] Software prodotto da: http://cpanel.net/ e largamente usato dai providers Internet
[75] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[76] http://notepad-plus-plus.org/
[77] Pangloss sarà sicuramente d’accordo

5.1.2 Il Data Base

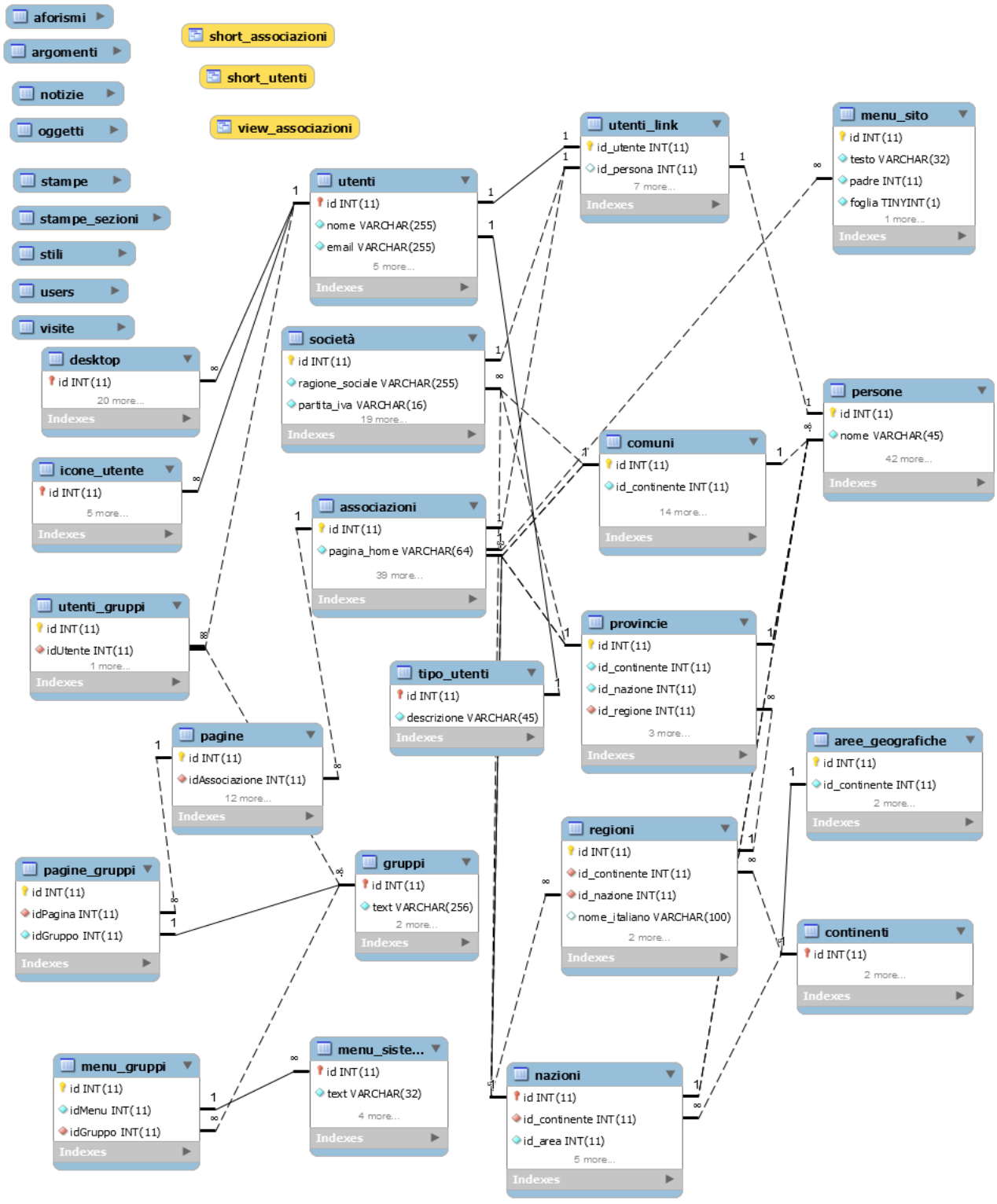
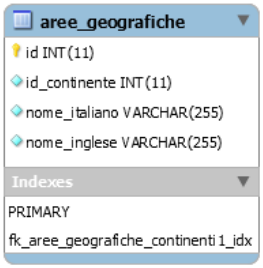
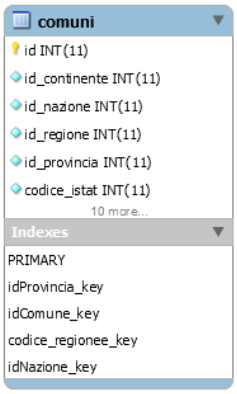


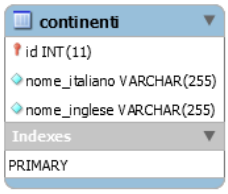
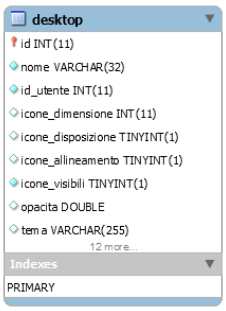
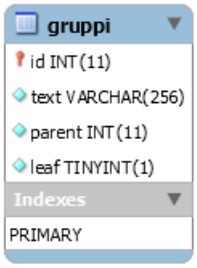
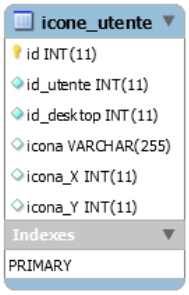
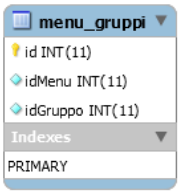
Figura 30 - Diagramma entità applicazione

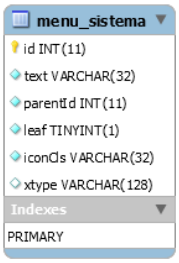
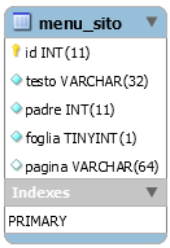
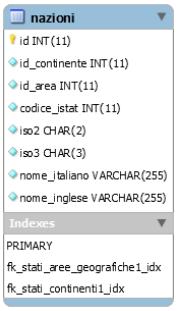
Il dettaglio nelle prossime pagine.

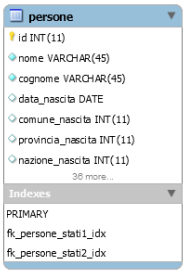
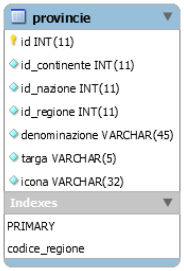
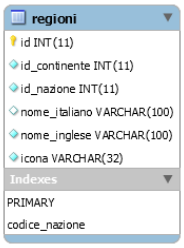
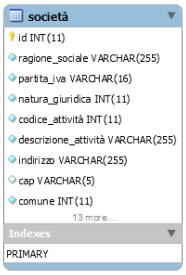
5.1.2.1 Le entità

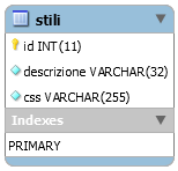
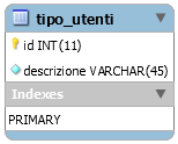
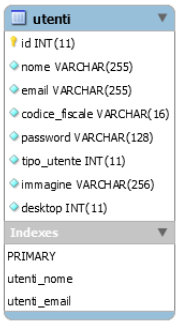
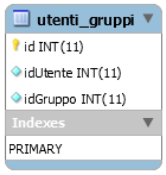
In questa sezione riporto le entità definite nel DataBase ed utilizzate nell’applicazione.

5.1.2.1.1 Aforismi 	Semplice tabella a due colonne: chiave primaria e valore. Viene usata in home page quale ‘messaggio del giorno’ anche se viene letta dal client 1 volta al minuto.
5.1.2.1.2 Aree geografiche 	Tabella di transcodifica
5.1.2.1.3 Associazioni 	Una delle tabelle cardini del progetto. Riporta la sede legale ed operativa dell’associazione, le rispettive coordinate WGS84 ed UTM (che saranno utilizzate in alcune funzioni quale il calcolo dei percorsi dei mezzi di servizio o dei tragitti dei soci dalle loro residenze alle sedi legali od operative) e varie informazioni per il web
5.1.2.1.4 Comuni 	Referenziata per decodifica da varie altre tabelle: • Persone • Guardie • Associazioni • Enti • Società Ha ridondanza in quanto ogni tupla riporta i codici delle varie entità a cui appartiene gerarchicamente dove ci si potrebbe limitare solo a quella di gerarchia immediatamente superiore

5.1.2.1.5 Continenti 	Tabella di transcodifica
5.1.2.1.6 Desktop 	Rappresenta il desktop utente Ha attributi per la gestione delle icone, del tema, dello sfondo, del bordo, del salvaschermo ed altro
5.1.2.1.7 Gruppi 	Gerarchia di gruppi per stabilire chi può fare cosa nell’ambito della gerarchia dell’associazione. Es.: il responsabile del magazzino dell’associazione X vedrà un menu funzioni diverso dal responsabile automezzi. Vige il principio di gerarchia: più in basso uno è collocato meno funzionalità vedrà
5.1.2.1.8 Icone 	Memorizza le icone create dall’utente sul desktop che sarà visualizzato come dataview di queste icone.
5.1.2.1.9 Menu_gruppi 	Determina quale gruppo può visualizzare la voce di menu

5.1.2.1.10 Menu_sistema  	E’ il menu che viene visualizzato cliccando il pulsante ‘Start’ del desktop. È strutturato gerarchicamente e filtrato in base a quali gruppi è inserito l’utente
5.1.2.1.11Menu_sito  	Come il menu sistema anche il menu sito, è strutturato gerarchicamente. Ogni click di un pulsante sulla main_toolbar comporta il caricamento della pagina specificata nel nodo se non nulla e se esistente nella tabella pagine e di una nuova barra menu se questo nodo ha figli (foglia = false)
5.1.2.1.12Nazioni  	Tabella di transcodifica per varie altre entità
5.1.2.1.13Notizie  	Ultime notizie dalle associazioni. Per uniformare la presentazione, anche se rimane possibile, per ogni associazione creare un menu totalmente personalizzato, si cercherà di forzare alcune voci standard, fra le quali sezioni chi siamo, notizie, agenda, bacheca
5.1.2.1.14 Pagine  	Tabella fondamentale per il “sito”. Editabile direttamente dall’utente autenticato ed abilitato. Si è convenuto di abilitare tre “responsabili informatici” per ogni associazione che cureranno l’inserimento e la modifica della pagina corrente nella sezione “sito” dopo essersi autenticati. L’editor fornito, permetterà la modifica di una sezione alla volta della pagina ed è basato su TinyMce. La modifica/creazione del campo “azioni” richiamerà un elenco di componenti predefiniti che potranno essere inseriti con un semplice click. In questa versione alpha, il “responsabile informatico” è libero di inserire qualsiasi codice JavaScript voglia. In versioni future ci sarà un controllo efficiente ed una crittografia del codice

5.1.2.1.15 Pagine_gruppi 	Determina quale gruppo può modificare quale pagina
5.1.2.1.16 Persone 	Varie informazioni anagrafiche sulla persona Uno degli obbiettivi del progetto è fornire ai responsabili delle associazioni, degli enti di protezione civile, e degli altri enti coinvolti, un quadro real-time dei volontari disponibili in un certo territorio e del loro operato. In seconda battuta, più prosaicamente, è la stampa dei pass in occasioni di esercitazioni o calamità che richiedono lunghi tempi di permanenza e quindi di avvicendamento di volontari
5.1.2.1.17 Provincie 	Tabella di transcodifica
5.1.2.1.18 Regioni 	Tabella di transcodifica
5.1.2.1.19 Società 	Questa tabella è presente per un duplice scopo: • creare gruppi di acquisto per le associazioni • gestire/integrarsi con procedure contabilità
5.1.2.1.20 stampe 	Per alcune entità è stato previsto di gestire le stampe tramite templates docxml (preparati in Word o programmi office in grado di produrre formati compatibili). Sul server nodejs, alcuni moduli effettueranno il ***print merge*** del set di dati richiesto dal client. Questa tabella contiene attributi utili a questa gestione

5.1.2.1.21 Stili 	Elenco dei temi dell’applicazione (fogli di stile CSS sostituibili dinamicamente a richiesta dell’utente)
5.1.2.1.22 Tipo_utenti 	Tabella di transcodifica
5.1.2.1.23 Utenti 	Utenti dell’applicazione. E’ identificato da 4 chiavi univoche (ID, nome, email, codice_fiscale) e si può autenticare con ognuna di esse
5.1.2.1.24 Utente_gruppi 	Associa l’utente ad 1 o più gruppi e sulla base di questa associazione l’utente riceverà un menu “personalizzato” che gli permetterà o meno certe funzionalità in funzione dell’appartenenza ad un dato gruppo

5.1.3 Lancio/attivazione

Come detto in precedenza, attualmente il sito è ospitato su server virtuale Ubuntu e risponde agli indirizzi web:

- http://37.9.231.173:3000/ws/gvd TEST
- http://37.9.231.173:3000/ws/build/production/GesVol PRODUZIONE

i percorsi sono il risultato delle fasi di generazione della struttura e del *build* dell'applicazione e del tutto provvisori.

5.1.3.1 Home Page

Figura 31 - Home Page GesVol

I vari componenti della pagina sono:

1. Area immagine: può essere un logo, logotipo o immagine. Ha una dimensione fissa di 128x128px
2. Area titolo pagina: corrisponde al nome dell’associazione; è previsto un sottotitolo a discrezione dell’associazione con corpo ridotto e centrato sotto il principale
3. Barra menu sito: sono previsti al massimo 8 sezioni. Nessun effetto di *rollover* o altro. L’utente deve cliccare per poter cambiare pagina
4. Area pulsanti dinamica: alcuni pulsanti compaiono solo dopo l’autenticazione. Quando ritenuto necessario e generalmente nei pulsanti senza testo esplicativo[78], l’evento *mouseover* visualizzerà un *tooltip* fluttuante che aiuta nell’identificazione della funzionalità. Il pulsante più a destra, quello con la matrice di puntini, è il menu del sito. Dettaglio avanti.
5. È un componente che ho chiamato “Aforismi”. Presenta un detto/motto/proverbio che dovrebbe ricordare in qualche modo l’importanza di fare volontariato. Dettaglio avanti.
6. Componente “ViewAssociazioni”: riporta l’elenco delle associazioni registrate in archivio. Sulla prima riga sono presenti tre pulsanti per riordinarlo secondo il numero di volontari, la sigla dell’associazione o la data di costituzione. Dettaglio avanti.
7. Gruppo di quattro pulsanti per cambiare la modalità di vista dell’elenco: l’icona dei pulsanti dovrebbe essere sufficiente ad identificare la funzionalità che comunque viene evidenziata al passaggio del mouse. Dettaglio avanti.
8. Piede della pagina: in questo caso riporta un link verso il mio blog personale

Gli unici componenti fissi della pagina, Home Page in questo caso, ma vale per qualunque altra pagina del sito, sono i contenitori header (1 e 2), menu (3) e piede (4) che hanno altezza fissa. Il corpo della pagina (5,6, e 7) è ‘elastico’, con altezza in funzione dell’altezza della finestra del browser ma con larghezza fissa a 1024px.

Una discreta percentuale di monitor ancora in uso ha risoluzione 1024x768 e mantenere fisso questo parametro semplifica il rendering della pagina e massimizza il numero di potenziali utilizzatori. Una risoluzione al di sotto di questa per un sistema desktop significa che l’hardware è così vecchio che probabilmente utilizza ancora Windows 98 o Windows 95 o addirittura Windows 3.1. In ogni caso incompatibili col web moderno.

Inoltre, una larghezza fissa di 1024px è molto usata nel web perché dà modo di utilizzare la parte centrale per le informazioni essenziali del sito e di sfruttare quelle laterali (se rimane spazio a sufficienza, per banner pubblicitari od *ads* di diverso tipo). In questo sito, al momento, le bande laterali sono semplicemente sfondi colorati in attesa di un utilizzo migliore.

[78] Regola valida in tutta l’applicazione

5.1.3.2 Funzionalità della Home Page

Brevemente le funzioni principali disponibili nella Home Page.

Come accennato sopra, la Home Page del sito, come tutte le altre, è dinamica, in quanto letta da database e per mezzo di classi ExtJs costruita e visualizzata dal client. Il processo iniziale è piuttosto elaborato e comporta queste fasi:

1. L'utente digita l'indirizzo del web server
    a. Il server risponde col file index.html

```
1   <!DOCTYPE HTML>
2   <html manifest="">
3   <head>
4       <meta http-equiv="X-UA-Compatible" content="IE=edge">
5       <meta charset="UTF-8">
6       <title>GesVol</title>
7       <script src='/socket.io/socket.io.js' type='text/JavaScript'></script>
8       <script type="text/JavaScript"
9       src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBC7zYs8-
10      4zX4UrjUkzwZr07Kz0M2tJjGQ&v=3&sensor=false"></script>
11      <script type="text/JavaScript" src="http://google-maps-utility-library-
12      v3.googlecode.com/svn/trunk/markerclusterer/src/markerclusterer.js"></script>
13      <script id="microloader" type="text/JavaScript" src="bootstrap.js"></script>
14      <noscript>
15      <h2>Questo sito richiede JavaScript ma il tuo browser non lo supporta</h2>
16      <h2>Controllare che sia attivo</h2>
17      <h3>Grazie</h3>
18      </noscript>
19  </head>
20  <body>
21  </body>
22  </html>
```

Figura 32 - Pagina index.html

Questa è *l'unica pagina html dell'applicazione* che ha il compito di caricare i componenti necessari e merita una trattazione approfondita:

1. Dichiara una pagina html aderente alle specifiche HTML 5
2. Il documento non è disponibile per il *browsing offline* in quanto l'attributo *manifest* della cache è nullo
3. Tag intestazione della pagina
4. I meta-tags definiscono le informazioni aggiuntive, *metadati* associati alla pagina; in questo caso servono ai browsers Microsoft (IE 6 e segg.) di selezionare la modalità di visualizzazione del documento che in questo contesto è *edge,* ovvero la più recente possibile
5. Si deve usare il set di caratteri "UTF-8"
6. Titolo della pagina che compare sulla barra della finestra
7. Caricare il file socket.io.js dal percorso specificato
8. Caricare le API di Google Maps con la chiave indicata
9. Continua precedente
10. Continua precedente
11. Caricare la libreria aggiuntiva di Google Maps
12. Continua precedente

13. Caricare il file di bootstrap dell’applicazione, *bootstrap.js*
14. Nel caso che il browser non supporti il JavaScript
15. Riportare messaggio opportuno
16. Continua precedente
17. Continua precedente
18. Fine tag *noscript*
19. Fine tag *head*
20. Definizione del *body* che verrà popolato dall’applicazione più avanti
21. Fine tag *body*

Il client, ricevuto *index.html* inizia a richiedere gli files come specificato sopra. Quando scarica i codice JavaScript lo esegue immediatamente.

Per *bootstrap.js* che è il risultato del comando “**sencha app refresh**” eseguito sul server questo comporta la richiesta del file *bootstrap.json* che contiene la sequenza ordinata dei files classi da caricare.

Ordinata non in senso alfabetico, ma di referenziazione da parte della gerarchia delle classi.

Considerando la complessità del framework vengono richiesti nel modo di test, circa 500 files, fra sorgenti JavaScript, icone ed altro e circa 10 nel modo di “produzione”; di conseguenza, una critica molto comune al framework ExtJs è la lentezza di caricamento ed esecuzione di alcune funzionalità base (griglie ed alberi).

Se è vero che nel modo di test ci vuole circa un minuto per caricare tutte le classi, in *produzione*, dopo l’assemblaggio eseguito col comando “**sencha app build production**” il numero dei files si riduce, utilizzando la tecnica degli *sprites* e componendo manualmente insieme le librerie aggiuntive ed i fogli di stile relativi, a 3 più le api Google.

Questi files rimasti sono comunque piuttosto grandi: il file *app.js* risultato del comando sopra (concatenazione in modo “ragionato” di tutte le classi, compressione eliminando spazi e caratteri ridondanti) supera normalmente 1.2Mb e con i fogli di stile ed immagini ci sono altri 800Kb circa da inviare al client per un totale di circa 2Mb.

Non sono pochi, ma da tener presente, che questo download si effettua una volta soltanto.

Sfruttando la cache locale del browser, infatti, il client, dalla seconda richiesta della pagina, preleverà localmente tutti i files necessari.

Ci sono comunque due tecniche aggiuntive per abbattere di circa l’80-85% le dimensioni dei files da inviare.

- Ricompilazione con Google closure compiler che è in grado di eseguire super-ottimizzazione del codice JavaScript quali: ridenominazione dei nomi delle funzioni e delle variabili nella forma più corta possibile, eliminazione di *dead code* o codice ridondante ed ovviamente dei commenti (già rimossi nel nostro caso dal compressore YUI)
- Compressione in formato gzip o similare

Queste due operazioni sono sufficienti a trasformare il codice del ***fat client*** in ***quasi-thin*** ed inviarlo al browser che provvedere a decomprimerlo ed ad eseguirlo.

Le prove che ho fatto mi hanno dato tempi complessivi inferiori ai 4 secondi pari ad una riduzione di circa il 93%. Nel mondo del web è comune intendere per *real time* un tempo di attesa non superiore a 3 secondi per cui, stando a questa definizione l’applicazione non può ritenersi tale ma la considerazione da fare è la seguente:

- Lo scopo è quello di creare **un’applicazione web**
- Le funzionalità del *sito* sono solo aggiuntive
- Le applicazioni web prenderanno il posto di quelle locali

Quale utente non sarebbe soddisfatto nel vedere che il proprio computer apre un documento Word od un foglio Excel in poco più di 3 secondi?

Col framework ExtJs è possibile caricare in background interi *packages*. Cioè: se la complessità dell’applicazione è tale che i files *di produzione* risultanti, dopo tutte le tecniche viste per ridurli, continuano ad essere troppo grandi e quindi lenti da scaricare, si può creare un file principale col compito di visualizzare la home page o poco più e sfruttare quei pochi secondi nel quale l’utente focalizza l’attenzione e/o inserisce le credenziali per collegarsi, per caricare in modo asincrono le altre classi, zippate, ottimizzate e quant’altro, necessarie all’applicazione.

Se a tutto questo si aggiunge il supporto delle web-socket si abbatteranno i tempi di download di circa il 50-70%.

Dulcis in fundo, la velocità di trasmissione dati della rete pubblica o dell’intranet.

Google ha cablato con fibra ottica a 10Gbits/sec alcune importanti città degi USA (Austin, Kansas City). Nel luglio 2014 alcuni ricercatori dell’Università Tecnica di Danimarca (DTU) hanno sperimentato velocità di **43** terabits/sec pari a circa un DVD in alta risoluzione in 1 secondo.

Questo significa che la velocità di trasmissione su rete pubblica è **molto** superiore alla velocità di scambio dati fra la CPU e la RAM del nostro PC/Tablet/Smartphone.

Non mi aspetto di vedere presto queste meraviglie, specialmente in Italia, ma è chiaro che la velocità della Rete continuerà ad aumentare almeno in modo tale da soddisfare la crescete domanda di bits.

2. Tornando al client: una volta ricevuti tutti i files, eventualmente decomprimendoli chiamerà alla fine, l'*entry point* dell'applicazione che nel nostro caso, e standard nelle applicazioni ExtJs, è il file **app.js**:

```
Ext.onReady(function () {

        Ext.Loader.setConfig({
                        disableCaching: false,
                        enabled: true,
                        paths: {
                                'GesVol'    : 'app'
                               ,'GesVol.ux' : 'app/ux'
                        }
        });

        Ext.require([
                'Ext.*'
               ,'Ext.data.*'
               ,'Ext.form.*'
               ,'Ext.grid.*'
               ,'Ext.layout.*'
               ,'Ext.panel.*'
               ,'Ext.toolbar.*'
               ,'Ext.tree.*'
               ,'GesVol.DirectAPI'
               ,'GesVol.store.MenuSito'
               ,'GesVol.ux.Util'
        ]);

        Ext.application({
                autoCreateViewport: false
               ,extend: 'GesVol.Application'
               ,name: 'GesVol'
        });
});
```

Figura 33 - App.js, entry point dell'applicazione

5.1.3.3 Entry point e naming convention

Con questo listato "entriamo" nel dettaglio dell'applicazione e del framework ExtJs: non è pensabile di dettagliare le circa 14.300 linee di codice finora scritte o peggio, le circa 170.000 della libreria ExtJs una ad una per cui ne darò, per questo e per gli altri sorgenti, una descrizione di massima.

Da notare Ext.onReady(function() { … }).

E' il *listener* che viene attivato quando il documento è pronto, quando scatta cioè l'evento "ready" del framework ExtJs. Viene eseguita la funzione anonima passata come parametro che imposta alcune configurazione del Loader delle classi e richiede i files specificati. L'asterisco in 'Ext.grid.*', ad esempio, significa: richiedo tutti i files JavaScript del percorso ext/grid/ e sottodirectory. Questa è una delle poche operazioni sequenziali, ma che, come visto può essere resa asincrona.

Quando tutti i files richiesti sono disponibili, ricordo che nel caso di *produzione* si ha un unico file applicazione con le classi concatenate nell’ordine giusto, la funzione Ext.application({…}) costruisce l’applicazione.

Se avete letto con attenzione la configurazione del Loader avrete visto che l’attributo ‘paths’ ha due elementi: ‘GesVol’ impostato ad ‘app’ e ‘GesVol.ux’ impostato ad ‘app/ux’.

Questi sono i percorsi, relativi al file index.html, dai quali leggere tutte le risorse necessarie.

Il framework usa un *naming convention* standard per comporre i nomi delle classi, praticamene identica a quella del Java. In questo caso, la riga evidenziata in giallo nel codice, “extend: ‘GesVol.Application’, va interpretata così:

la classe Ext, con la funzione Ext.application estende la classe GesVol.Application in cui il percorso del file sorgente si compone utilizzando il percorso prelevato dall’attributo *paths* del Loader che ha nome uguale al valore dell’attributo *name* definito nell’oggetto di configurazione passato come parametro (!).

Sembra uno scioglilingua e pressoché indecifrabile, ma in effetti è molto più semplice di quello che appare: in sostanza si richiede il file JavaScript che si trova nel percorso ottenuto concatenando, con l’opportuno separatore di directory, i valori evidenziati in giallo, ovvero: ‘app/Application.js’.

GesVol identifica il *namespace* dell’applicazione e non si riporta nel nome del file, ma solo nella definizione e nell’uso della classe che il file implementa. All’inizio dell’uso del framework, questo è fonte di frustrazione e sicuri mal di testa perché sembra che i files degli esempi siano caricati come per magia ma che sia impossibile farlo con i propri sorgenti.

Ovviamente non è così: utilizzando il comando “**sencha generate app”** in modo opportuno, viene creata automaticamente la gerarchia delle directory e lo scheletro dall’applicazione che poi viene estesa per implementare le funzionalità volute.

Il file GesVol.Application è di 136 righe, un po’ troppe per essere esaminate nel dettaglio; brevemente, ha questi compiti basilari:

- Definire un *global handle* per gli errori interni all’applicazione (qualche *syntax error* non catturabile durante la “compilazione”)
- Aggiungere alla classe String, nativa di JavaScript, tramite prototipizzazione il metodo *format* per comporre dinamicamente stringhe parametriche
- Definire alcuni parametri per la libreria jQuery: un conflitto fra le due librerie può causare un’errata inizializzazione di jQuery ed un blocco dell’applicazione ancora prima di attivarsi
- Attivare il gestore dei *tooltip*: piccoli messaggi fluttuanti al passaggio del mouse
- Definire un *global handler* per intercettare l’uscita, tramite chiusura della finestra o tasto Indietro del browser dall’applicazione in modo da uscire completando il *cleanup* eventualmente necessario

- Definire un *message bus* con ***new Ext.util.Observable()*** per poter disporre, a livello di intera applicazione, di un gestore eventi globale permettendo a moduli e *controllers* diversi di interagire fra loro con chiamate del tipo: *this.fireEvent(‘nuovo_evento’)* che si propaga globalmente permettendo da *handlers* diversi di intercettare l’evento e di agire di conseguenza
- Impostare i parametri per i messaggi di errore gestiti con jQuery
- Precaricare una serie di dati utili nella Home Page

Quando tutti gli *stores* di questa fase sono stati caricati, viene chiamato il costruttore di classi ExtJs che è un metodo dell’oggetto *singleton* **Ext**: Ext.create(‘GesVol.view.main.Main’). Ormai pratici della *naming convention*, troveremo il file Main.js nella directory ‘./app/view/main’. Lo esamino in dettaglio per approfondire alcune particolarità del framework:

```
1   Ext.define('GesVol.view.main.Main', {
2       controller: 'main_controller'
3      ,extend: 'Ext.container.Viewport'
4      ,id: 'main_view'
5      ,items: [{
6          flex: 1
7         ,region: 'west'
8         ,style: 'background-color: #ffff00;'
9         ,xtype: 'container'
10     },{
11         border: 1
12        ,items:[
13         {  xtype: 'main_header'},
14         {  xtype: 'main_toolbar'},
15         {  xtype: 'main_body_outer'},
16         {  xtype: 'main_footer'}
17        ]
18        ,layout: { align: 'start', pack: 'center', type: 'vbox'    }
19        ,region: 'center'
20        ,style: { borderColor: 'black', borderStyle: 'solid' }
21        ,width: 1024
22        ,xtype: 'container'
23     },{
24         flex: 1
25        ,region: 'east'
26        ,style: 'background-color: #00ffff;'
27        ,xtype: 'container'
28     }]
29     ,layout: { align: 'stretch', type: 'hbox' }
30     ,listeners: {
31        boxready:    'onViewportBoxReady',        scope: 'controller'
32     }
33     ,requires: [
34         'GesVol.view.editor.PageEditor'
35        ,'GesVol.view.main.Body'
36        ,'GesVol.view.main.Footer'
37        ,'GesVol.view.main.Header'
38        ,'GesVol.view.main.MainController'
39        ,'GesVol.view.main.Toolbar'
40        ,'GesVol.view.utente.Login'
41        ,'GesVol.ux.buttons.MenuButton'
42     ]
43     ,xtype: 'app_main'
44  });
```

Figura 34 - Il modulo Main.js

1. Il metodo *define* definisce la classe passando l’oggetto Json con i vari parametri
2. Questa *view* utilizza un *viewcontroller* denominato ‘main_controller’
3. Questa classe estende il *viewport* (l’area disponibile del browser)
4. Ha come identificatore: ‘main_view’
5. Come sottocomponenti[79] utilizza tre *containers*

6-9. Definisce un’area elastica di colore giallo a sinistra

11-22. Definisce l’area di lavoro dell’applicazione.
Include i 4 componenti principali e fissi; header, toolbar, body e footer che verranno disposti verticalmente secondo le regole specificate nel layout

24-27. Definisce un’area elastica di colore turchese a destra

30-32. Definisce un handler per l’evento ‘boxready’ da far gestire al *controller* associato

5.1.3.4 Pattern MVVM vs MVC

Questo punto è la chiave del *pattern* MVC (Model View Controller) anche se, per complicare le cose, si deve parlare in questo caso di *pattern MVVM* senza la ‘M’ in quanto non vi è alcun modello associato alla view. Ho sperimentato che questa versione del framework implementa il pattern MVVM in modo non proprio cristallino e, anziché semplificare lo sviluppo con il *two-way-binding*, lo complica. La visione dei progettisti Sencha, buona nel complesso, è ancora lacunosa in alcuni aspetti importanti: la visione *producer-consumer* per i *binding* automatico dei componenti è senz’altro corretta: la view definisce un modello per i dati di cui ha bisogno (che può implicare l’uso di diversi *data model* e *data stores* o *locali) ed il framework* richiederà automaticamente i dati al server.

Peccato che si siano dimenticati di gestire gli eventi causa di errori; forse danno per scontato che i server siano sempre attivi e che i databases non si fermino mai. In effetti si può sempre gestire l’errore definendo nel modello-view una handler nel proxy ma a quel punto la complessità cresce ed i benefici di questa nuova architettura diminuiscono. Alcune immagini ci aiuteranno a fissare questi concetti:

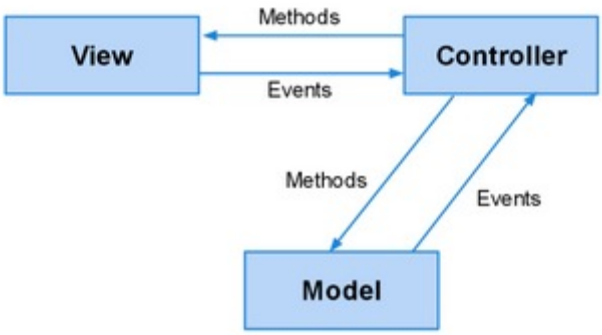


Figura 35 - MVC: Model View Controller

[79] Non si deve dimenticare che il framework *genera* il codice html, per cui i termini di View, Container, Panel, Window sono semplicemente, nel codice html, dei tags *div*. Quello che non è semplice è la gestione a livello logico di ciò che succede in questi *div*, delle strutture dati utilizzate e dei metodi che permettono l’interazione con l’utente

Il Controller gestisce gli eventi prodotti dal modello e dalla *view* e reagisce con gli opportuni metodi.

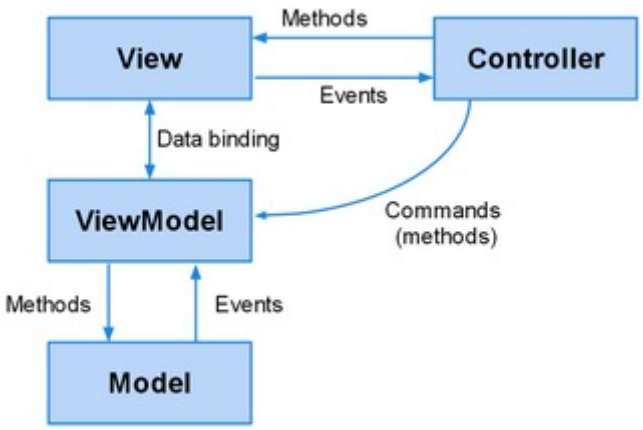


Figura 36 - MVVM: Model View - View Model

La View è collegata al ViewModel per ricevere/impostare in automatico i dati. Il ViewModel, a sua volta, ascolta gli eventi del Model e reagisce con opportuni metodi. L’interazione fra View e Controller rimane la stessa del pattern MVC ma il controller può interagire con nuovi metodi con il ViewModel.

Il binding dei dati si può propagare a tutti i figli *legittimi*[80] in modo gerarchico, che sommato alle funzionalità del *data session* rendono il framework particolarmente potente.

Scenario tipico: griglia con i dati di un ordine. Click sulla riga e si presenta il dettaglio, magari un’altra griglia che specifica tutti i componenti dell’articolo/bene riferito alla prima riga. Come aggiungere/modificare/cancellare i dati della seconda griglia senza interferire con la prima? Il controllo di sessione è pensato per questo: isolare gruppi di dati dipendenti fra loro ed effettuare il *commit* dei cambiamenti solo a richiesta, permettendo dei ripensamenti.

[80] Non è una battuta: ricordiamoci che il codice generato è sostanzialmente una gerarchia di tags *div* e che solo a livello logico assumono differenti funzionalità nel framework. Se si definisce un *viewmodel* in una *window*, tutti gli oggetti (leggi div) dipendenti potranno accedervi e tramite l’attributo ***bind*** collegare, per l’appunto, i dati del modello ai componenti. Se un componente od un controller creano con Ext.create() un nuovo componente, quest’ultimo non potrà, a meno di avventurosi *workarounds*, accedere al *viewmodel* dell’oggetto che l’ha creato e quindi non potrà utilizzare le funzionalità automatiche di *binding* vanificando buona parte della nuova architettura. In questo senso sono da intendersi i *figli illegittimi*. Quello che mi ha lasciato perplesso e che non sono riuscito a completare è il fatto di poter ereditare in tutte le windows create nel desktop, i dati del desktop stesso. Almeno con questa logica.

5.1.3.5 Caricamento dinamico della pagina

Abbiamo lasciato il caricamento della pagina in sospeso nella definizione dei componenti del *viewport* in attesa dell'evento 'boxready' assegnato al controller 'main_controller'.

A questo punto gli elementi grafici che compongono la pagina sono stati creati dall'*engine di rendering di ExtJs* ma non ancora popolati con alcun valore.

Vediamo cosa succede nel dettaglio:

- Si maschera l'intero *viewport* con un overlay grigio e con un messaggio di attesa
- Si associa lo *store* StiliStore al combobox per il cambio del tema dell'applicazione
- Si imposta una variabile globale di comodo per la pagina corrente al valore nullo
- Si richiama la funzione di caricamento pagina passando il nome della pagina richiesta, in questo caso 'home'
- La routine fa un test iniziale per controllare che la pagina richiesta non sia quella corrente, ed avvisa l'utente se ciò è vero
- Se falso, e cioè, se:
    - È la prima volta che viene chiamata
    - L'utente ha richiesto una pagina diversa da quella corrente
        - Viene effettuata una chiamata dirette al server tramite RPC: ExtRemote.DXAll.read dove i parametri sono:
            - table = 'pagine'
            - chiave = 'nome'
            - valore = <pagina richiesta>
            - se c'è stato un errore sul DB si richiama la routine di gestione errori definita nel controller main globale passandogli i parametri di ritorno
            - se la pagina richiesta non è stata trovata si segnala all'utente
            - se invece è stata trovata, si procede con l'elaborazione
- elabora pagina
    - si chiama ExtRemote.DXMenuSito.read per leggere il menu relativo all'associazione
    - se ok
        - si salva l'array di elementi in una variabile globale
        - si toglie la *maschera* alla *view*
        - si costruisce il menu orizzontale
        - si effettua il rendering della pagina, che nel dettaglio significa
            - *impostare logo, titolo, corpo e piede ai rispettivi valori HTML*
            - *eseguire il codice letto nel campo* ***azioni*** con la funzione **eval**: è questo punto a generare il contenuto *dinamico* delle pagine del sito

Di seguito alcuni *screenshots* del risultato.

5.1.3.5.1 Dataview

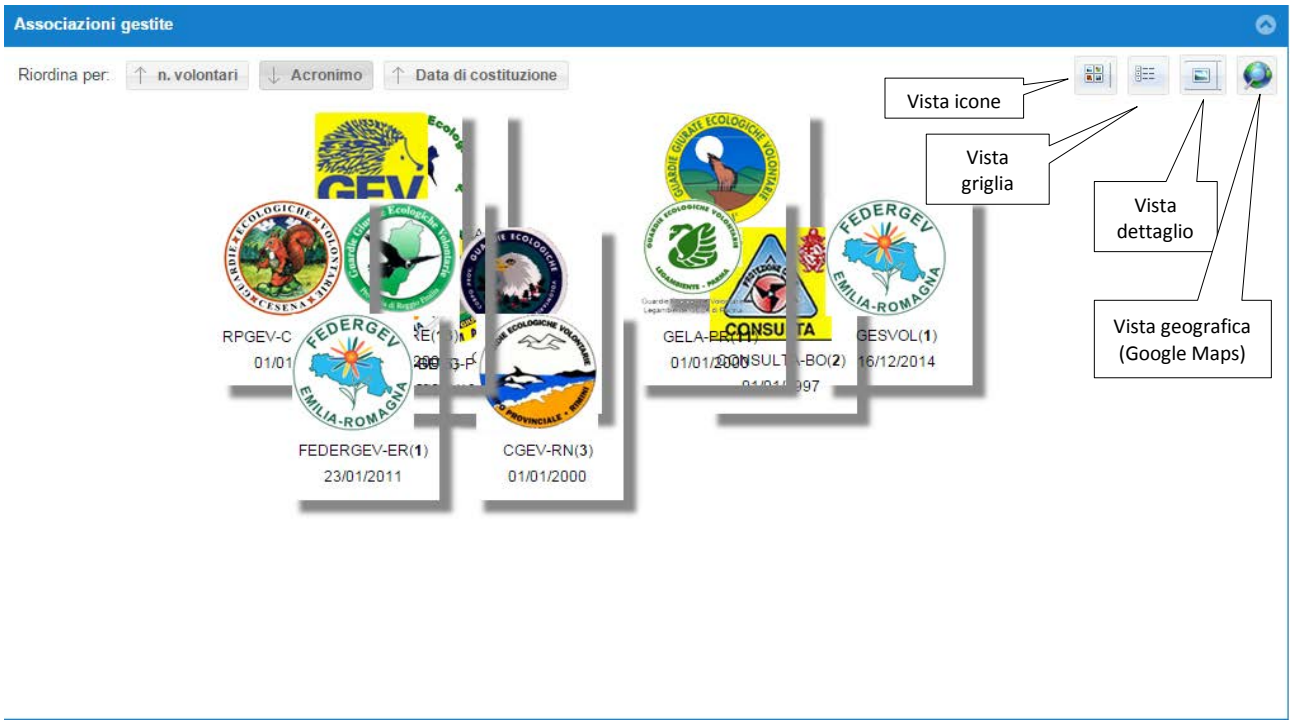


Figura 37 - Multisort con animazione DataView

Questa ‘istantanea’ non rappresenta un bug dell’applicazione ma un momento dell’animazione della *DataView*[81] “Associazioni gestite” dopo il click sul pulsante “Acronimo”. È un effetto un po’ scenografico ma anche utile. Il *multisort* funziona anche per le viste *griglia* e *dettaglio* senza però effetti di animazione in quanto legati alla D*ataView*.

[81] Questo ed altri termini propri del framework ExtJs verranno ripresi più volte nel corso della trattazione. Per uno sguardo più approfondito fare riferimento alla documentazione ufficiale: http://docs.sencha.com/extjs/5.0/apidocs/ o ai riferimenti bibliografici

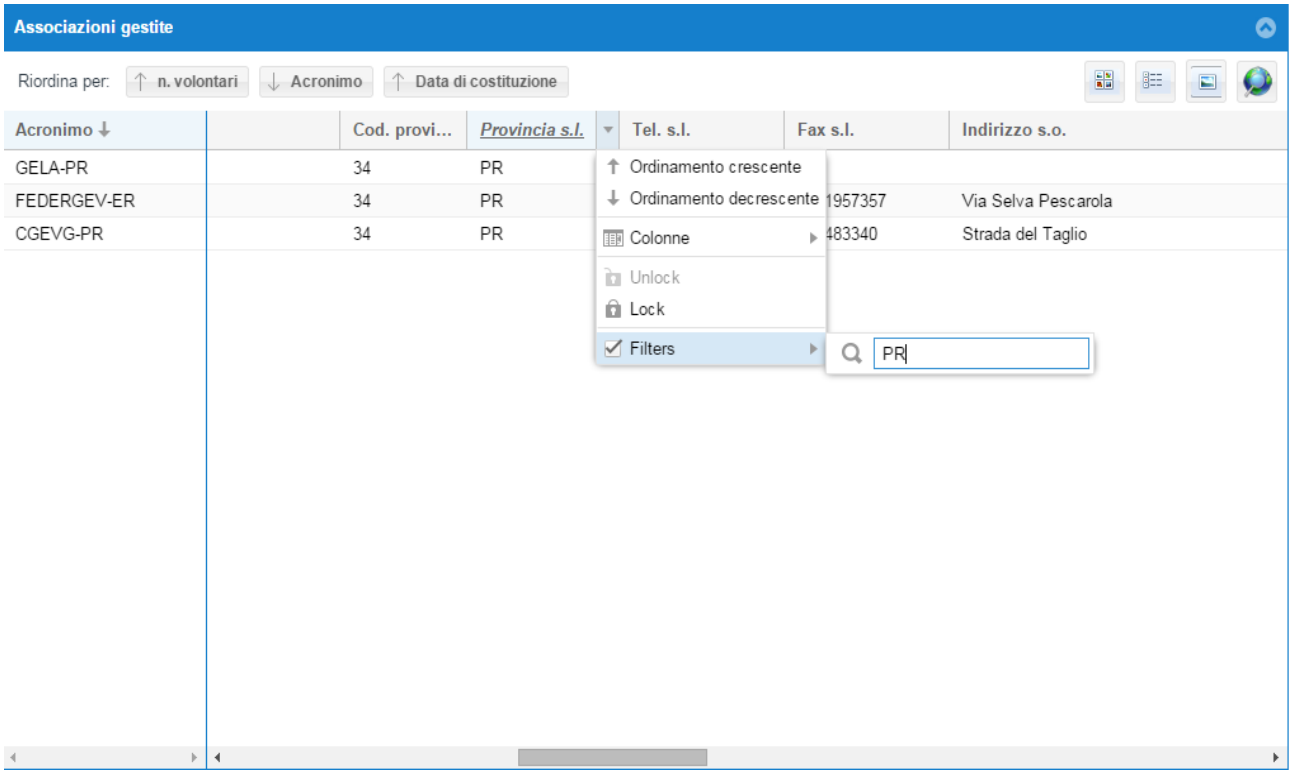


Figura 38 - Griglia colonna bloccata, scorrimento e filtro

In quest’altra immagine l’utente ha cliccato il pulsante “Vista griglia”, ha spostato verso sinistra le colonne della griglia per evidenziare la provincia dove ha la sede legale l’associazione ed ha cliccato sul piccolo triangolo a destra del nome della colonna visualizzando il menu contestuale che riporta le funzionalità di Ordinamento crescente/decrescente, submenu selezione/deselezione colonne, blocco/sblocco colonne, e filtro in base alla tipologia, stringa, in questo caso.

Ha quindi digitato le lettere ‘P’ ed ‘R’ nella casella e la *GridView* ha filtrato tutte le associazioni di Parma presenti in archivio.

Questo senza alcun intervento/chiamata da/verso il server ni quanto i dati sono stati già letti all’avvio.

Questo modello di esecuzione, viene detto ***fat client*** in quanto il client esegue la maggior parte del lavoro di presentazione ed analisi/manipolazione dei dati lasciando al server il solo compito di fornire i dati necessari e di aggiornare, su richiesta, il DataBase.

È in contrapposizione, ovviamente, al ***thin client*** che prevede esattamente l’opposto: fare eseguire al server tutte le operazioni ed inviare al client la pagina pronta per il rendering finale del browser.

È facile intuire che quest’ultimo *modus* operandi genera un elevato numero di richieste verso il server ed un indubbio sovraccarico. Ad ogni click dell’utente, praticamente, parte una richiesta di aggiornamento. Questo è come funziona JSF (v. JSF pag. 37). Frameworks più moderni come quelli esaminati (Vaadin e Primefaces) sono in grado di mischiare codice JavaScript che può eseguire molte funzioni lato client riducendo il numero di chiamate verso il server.

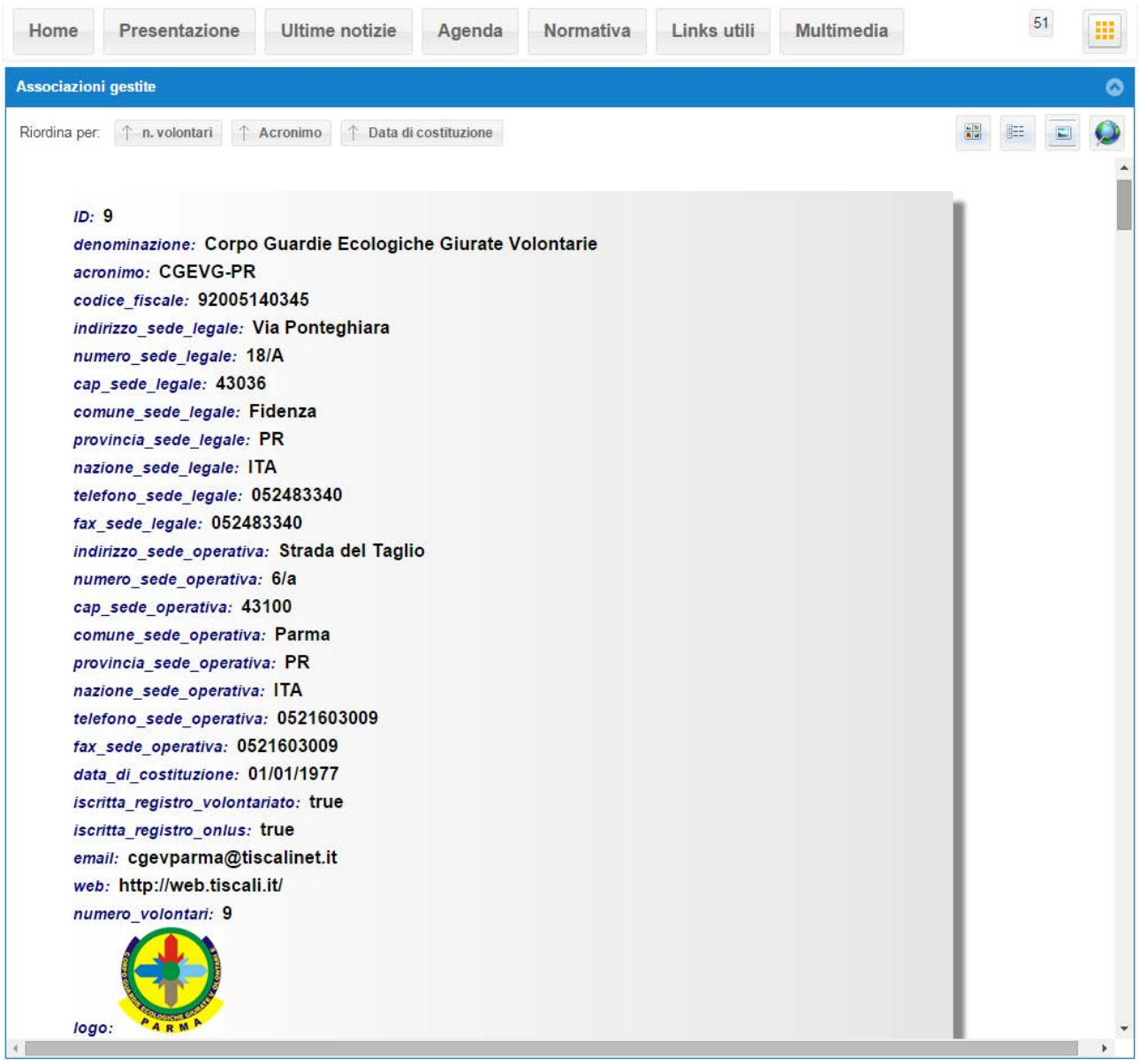


Figura 39 - Vista dettalglio associazione

La terza vista che si può attivare dalla Home Page è la vista dettagliata dei dati dell’associazione impaginata come se fosse una lettera tradizionale. I pulsanti *multisort* funzionano anche in questo contesto e dato che i dati sono pre-caricati, il sort è praticamente istantaneo.

Sono partito da queste considerazioni:

- Il numero iniziali di associazioni gestite è 12, cioè quelle che aderiscono a FederGev Emilia-Romagna
- La velocità di trasmissione dati è più che sufficiente per quasi tutte le esigenze informatiche
- I PC moderni hanno quantità di RAM paragonabili a quella dei supercomputers di 20 anni fa ad un costo migliaia di volte inferiore

Con queste premesse, utilizzando ExtJs è senz'altro più conveniente e facile da gestire, pre-caricare quanti più dati possibile e filtrarli, riordinarli in locale con le opportune features del framework.

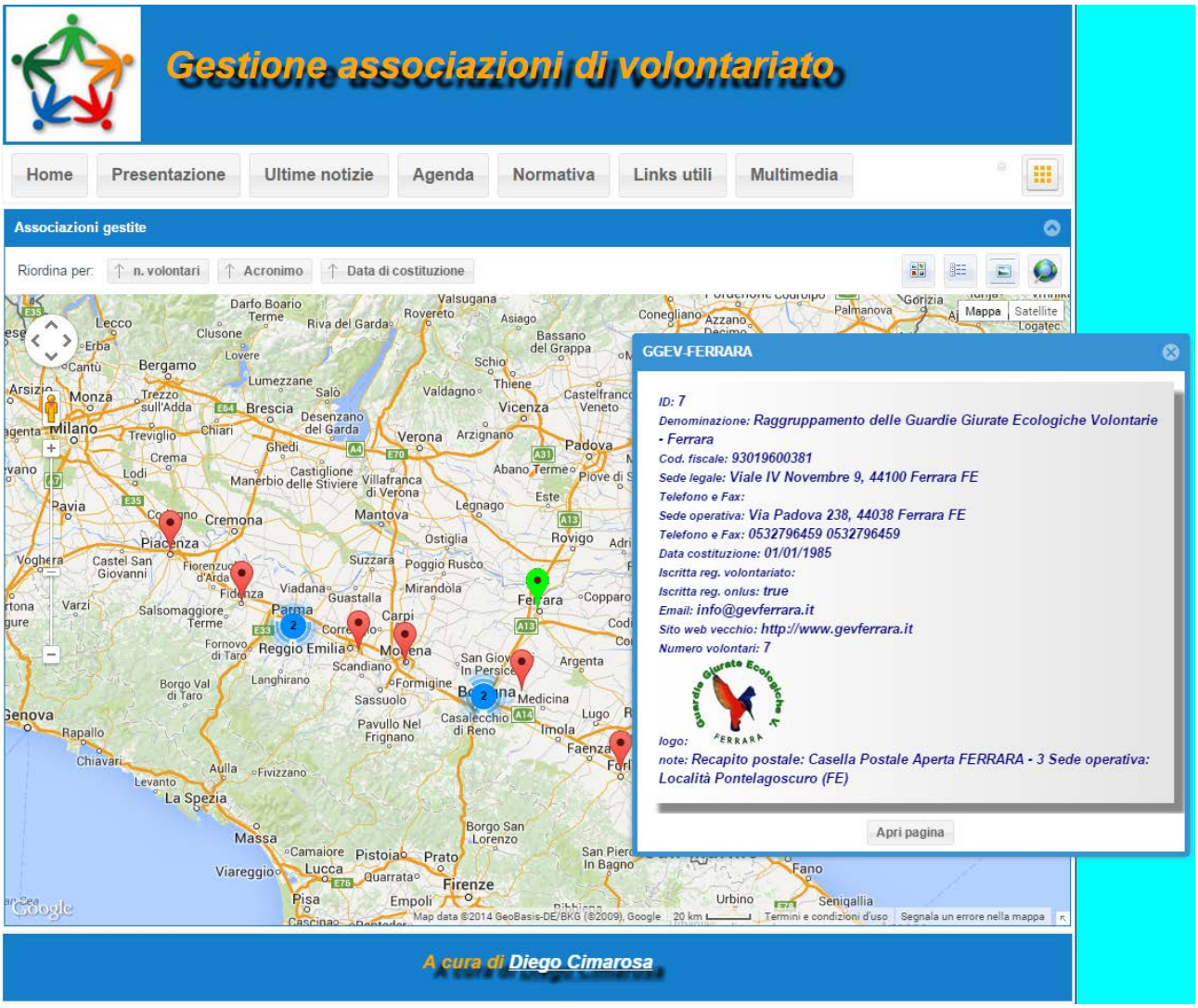


Figura 40 - Visualizzazione su Google Maps delle sedi delle associazioni e dettaglio di quella selezionata

L'utente ha cliccato sull'ultimo pulsantino disponibile sulla *DataView* e si è attivato il pannello GoogleMaps che evidenzia alcune particolarità interessanti:

- Le icone rosse, tradizionali di Maps, identificano la posizione geografica della sede delle associazioni il cui nome appare[82] come tooltip al passaggio del mouse
- L'utente ha cliccato sull'icona di Ferrara: l'applicazione 'ha detto' a Maps di centrare in quel punto e, per evidenziare il fatto, di usare un'icona verde anziché la solita rossa
- Parma e Bologna hanno entrambe cerchietti blu con un numerino; 2 in questo caso, e non è un bug dell'applicazione ma il modo di Google Maps per rendere un *cluster*[83] *di markers*

[82] Con un tasso del 50% di successo … qualche misterioso bug!
[83] *To cluster*: per l'appunto, *raggruppare*

in modo più intelligente. Se i punti da mappare sono al di sotto di una certa distanza, vengono raggruppati in un'unica icona col numero delle icone raggruppate.

- L’utente ha alfine cliccato (con Ctrl) sull’icona verde per evidenziare i dati dell’associazione in modo più gradevole

Figura 41 - Cambio tema applicazione

Un'altra funzionalità che di per sé non crea *valore aggiunto* ma che da un significato all’aggettivo “***rich***” nell’acronimo “***RIA***” è la possibilità di cambio tema/stile applicazione.

L’utente ha cliccato sul pulsante menu visualizzando il menu del sito con le seguenti funzioni:

- Accesso: richiesta di credenziali per accedere a funzionalità riservate
- Registrazione: richiesta di registrazione con wizard
- Password: richiesta nuova password in seguito a dimenticanza
- Elenco a discesa (*combobox*) per cambio tema: ogni voce corrisponde ad un foglio di stile CSS che viene sostituito dinamicamente al click del mouse. In questo caso l’utente ha cambiato il tema di default, *Neptune* in *Verde Amazzonia*, nome e tema di fantasia per richiamare il verde delle foresta

5.2 Desktop

L’idea di un desktop scritto in JavaScript potrebbe dar adito a seri dubbi su chi la presenta ma le tecniche già sviluppate permettono ben oltre.

Nel framework ExtJs c’è una lunga serie di esempi che sono la base di partenza per molte altre idee. Un esempio particolarmente ben riuscito è il web-desktop qui sotto interamente scritto in JavaScript – ExtJs:

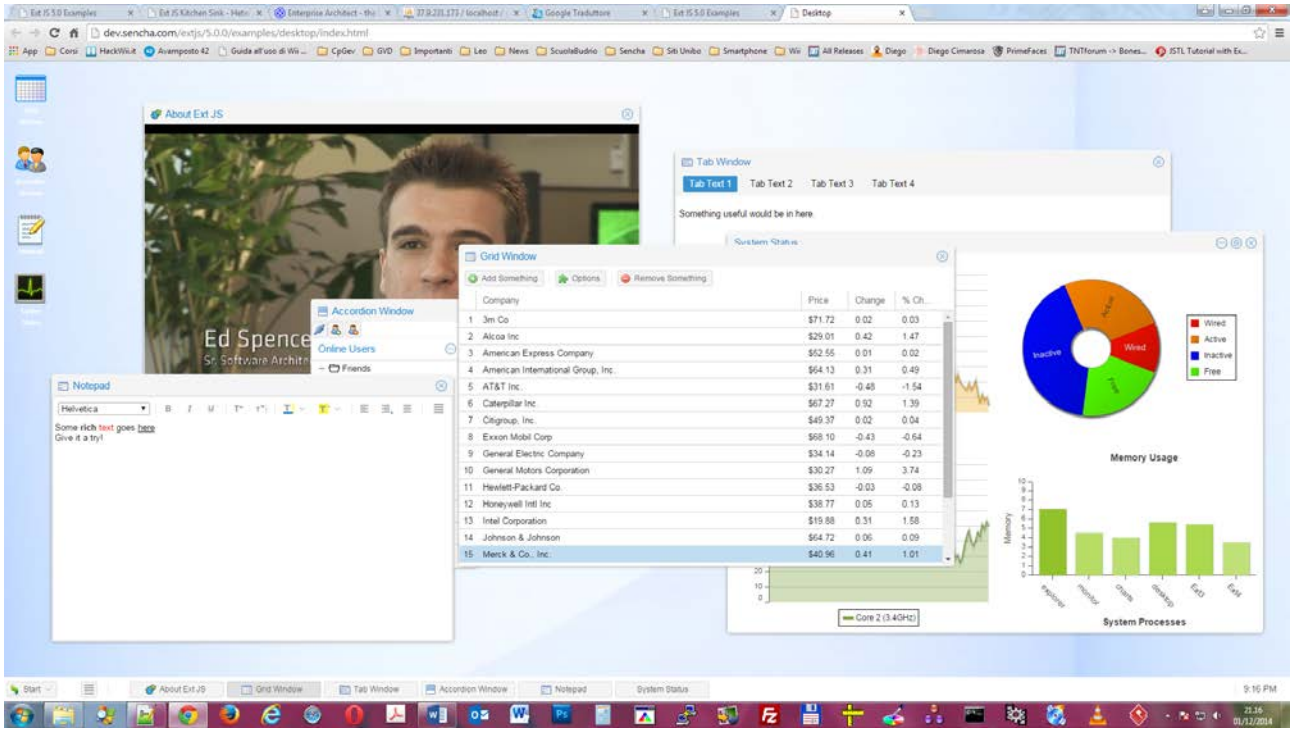


Figura 42 - Esempio di web desktop in ExtJs

Sulla base di questi esempi ero riuscito a sviluppare un sistema simile ma con qualche possibilità in più:

- gestione completa delle finestre con disposizione ad icone e cascata
- minimizzazione nella task-bar e ripresa di tute le finestre attive
- completa gestione dello sfondo: carico di immagine locale e memorizzazione sul server per visualizzazione
- gestione delle icone: dal menu start, drag’n’drop dell’iconcina della funzione direttamente sul desktop; riordino per tipologia, data, nome
- file explorer quasi completo: lista delle cartelle/files dallo spazio utente sul server, upload multiplo

purtroppo, un banale errore nel salvataggio ha azzerato gli ultimi tre mesi di sviluppo ed il desktop che sono riuscito a recuperare è molto più scarno:

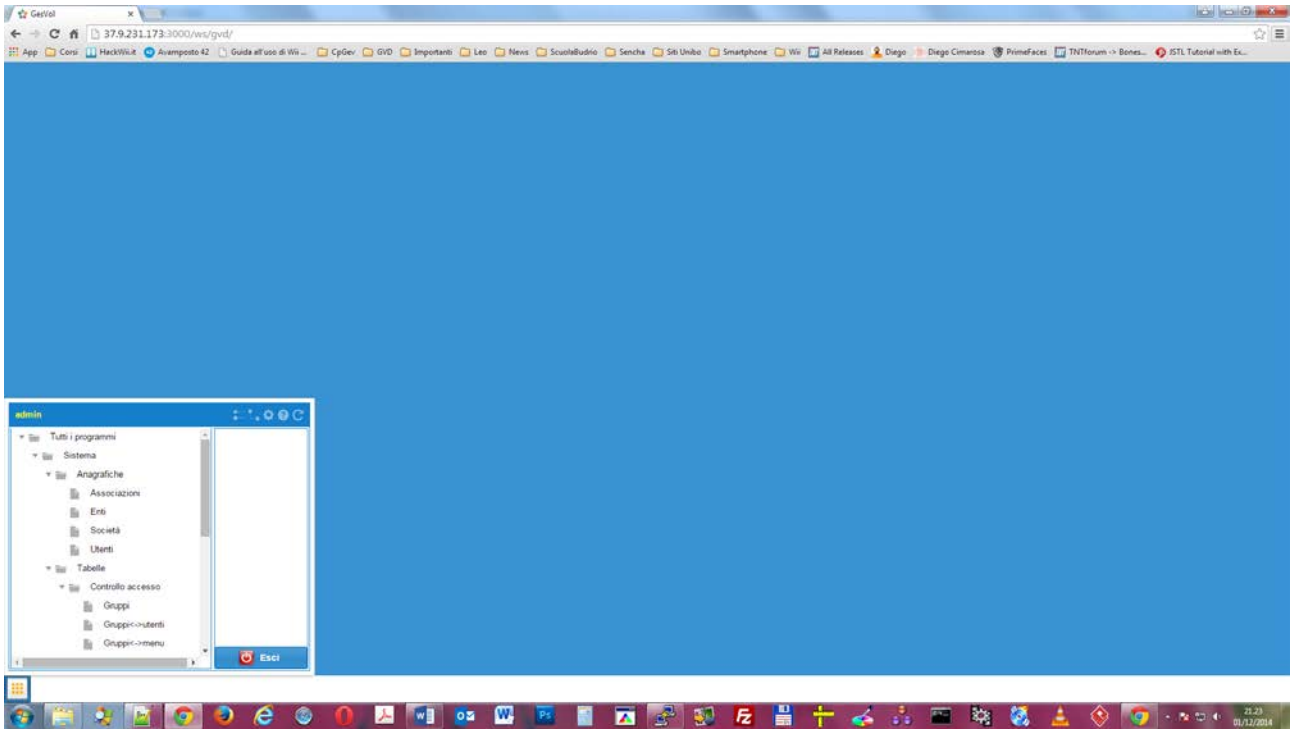


Figura 43 - Desktop GesVol

E’ disponibile solo il menu di sistema e mancano tutte le funzionalità della gestione delle finestre.

5.3 Mobile

Il nuovo framework ExtJs 5 ha alcune funzionalità speciali per adattarsi allo sviluppo mobile che vengono chiamate *responsive design*.

È possibile cioè cambiare il comportamento di un componente con la nuova opzione di configurazione, **responsiveConfig** resa disponibile tramite *plugin* a qualsiasi componente come in questo esempio[84]:

```
Ext.define('MyApp.view.Main', {
    extend: 'Ext.tab.Panel',
    plugins: 'responsive',
    responsiveConfig: {
        landscape: {
            tabPosition: 'left'
        },
        portrait: {
            tabPosition: 'top'
        }
    },
    items: [
        { title: 'Foo' },
        { title: 'Bar' }
    ]
});
```

Figura 44 - Esempio di *responsive design*

In pratica ciò viene sconsigliato perché ExtJs non è ottimizzato per i dispositivi mobili e risulterebbe troppo oneroso in termini di consumo di batteria e gestione del layout.

I componenti sono stati progettati per interagire con mouse e tastiera e non con le dita delle persone.

Per questo è necessario usare la versione mobile del framework, denominata Sencha Touch[85].

Questo framework è rilasciato sotto una licenza commerciale gratuita che permette legalmente di distribuire software proprietario senza condividere il codice sorgente come invece avviene con la licenza GNU GPL license v3.

Le differenze fra i due frameworks non sono molte: spiccano, però queste:

- uso dei profili per personalizzare al meglio per tablet e smartphone riutilizzando quanto più codice possibile
- supporto al pulsante "Indietro" del browser
- instradamenti o routes: il controller può intercettare un cambio di indirizzo nel browser
- supporto al nuovo standard XMLHttpRequest Level 2[86]
- una serie di componenti quali "carousel", "navigation view", "nested list" espressamente pensati per il mobile

[84] Vedi http://www.sencha.com/blog/designing-responsive-applications-with-ext-js
[85] Vedi http://www.sencha.com/products/touch/
[86] Vedi http://www.w3.org/TR/XMLHttpRequest2/

- nuovi temi che possono essere definiti in app.json in funzione della piattaforma
- icons fonts: nell’ottica di ottimizzazione, le icone sono rimpiazzate da caratteri di un font speciale

La vera differenza è nella possibilità di creare applicazioni native con uso di API native per l’accesso, ad esempio, alla macchina fotografica del dispositivo.

GesVol Mobile è per ora soltanto un’idea che verrò sviluppata in futuro anche in funzione di come evolveranno le possibilità offerte dal framework.

6 CONCLUSIONE

Questo progetto mi ha impegnato per più di tre anni e dopo tutto questo tempo è ben lungi dall'essere completato, e forse non lo sarà mai.

Lo scopo non era quello di iniziare e concludere un discorso, un'applicazione, un software. No.

Era solo curiosità ed un tentativo, forse un po' infantile, di rispondere dopo 64 anni ad un'affermazione che è passata alla storia come sua, ma che, come spesso accade, la Storia non ha mai saputo che lo fosse.

Mr. Thomas J. Watson, disse[87] che al mondo c'era posto solo per quattro o cinque computers. Forse nel 2020 il suo pensiero sarà realtà. Miliardi di persone collegate più o meno 24 ore al giorno a pochi clusters di super-computers. Chissà chi fra i giganti di oggi esisterà ancora fra cinque anni. Microsoft si fonderà con Google? Amazon comprerà IBM? A quanti Terabits/sec ci scambieremo informazioni? Come sarà il web fra cinque anni?

L'altro aspetto che mi ha colpito in questa ricerca è la distanza culturale fra gli USA ed il resto del mondo. La loro forza, a mio parere, non sta nella loro pur enorme economia, tecnologia o esercito, ma in una parola che specialmente qui in Italia sembra aver perso significato.

Futuro.

Loro lo sentono, come i cammelli che sentono l'acqua a chilometri di distanza, come i bambini che sentono il Natale. Loro sentono il futuro, e lo costruiscono, giorno dopo giorno, mattoncino dopo mattoncino, garage dopo garage. Cos'hanno che noi non abbiamo? Noi abbiamo la Storia, la Bellezza, ma dove abbiamo nascosto la Fantasia? I ragazzi di StreetMap girano il mondo in lungo ed in largo, perfino sott'acqua, nei *carrugi* di Genova o nelle calli di Venezia. Quanti in Italia avrebbero investito un centesimo in un progetto così? Chi avrebbe investito 1Mld di dollari nello sviluppo di un programma come ha fatto Microsoft?

- Qui base Italia, ci siete pianeta America?
- Qui pianeta America. Siamo in attesa.

Lasciamo sognare i giovani. E' l'ora di concludere.

Grazie per l'attenzione.

[87] Nel 2001 partecipai ad un interessantissimo convegno a tema "Da Gesù a Pinocchio, storia del mito" organizzato dalla Facoltà di Lettere dell'Università di Parma ed al quale l'invitato principale, Umberto Eco, "dimostrò" che non importa se una storia è vera o no, perché quando diventa mito diventa essa stessa Storia e Pinocchio è vero quanto Gesù.

7 BIBLIOGRAFIA

Adobe. *Adobe Flash Professional CS5 Classroom in a Book.* San Jose, California: Adobe Press, 2010.

Assemblea legislativa della Regione Emilia-Romagna - sito Demetra. «Normativa regionale.» *http://demetra.regione.emilia-romagna.it/.* 5 Luglio 1989. http://demetra.regione.emilia-romagna.it/al/monitor.php?vi=nor&st=doc&dl=4ac9aec5-61a1-9a12-2e3a-4e4cbebdd70e&dl_t=xml&dl_a=y&dl_id=10&ul=1&ev=1&bk=1 (consultato il giorno Novembre 13, 2014).

Assintel. 2013. http://www.assintel.it/eventi/assintel-report-2013/ (consultato il giorno Novembre 26, 2014).

Benington, Herbert D. *Wikipedia.* s.d. http://en.wikipedia.org/wiki/Waterfall_model#cite_note-2.

BUCKETT, CHRIS. *Dart in Action.* New York: Manning Publications Co., 2013.

Catlin, Hampton, and Michael Lintorn Catlin. *Pragmatic Guide to Sass.* 2011.

Centro Studi Consiglio Nazionale Geologi. «Rapporto sulle frane in Italia.» 2007.

Clark, John E., and Bryan P. Johnson. *Sencha Touch Mobile JavaScript Framework.* Birmingham B3 2PB, UK..: Packt Publishing Ltd., 2012.

Corriere della Sera. «In Italia il 70% delle frane europee.» *www.corriere.it.* 18 Marzo 2014. http://www.corriere.it/ambiente/14_marzo_17/italia-70percento-frane-europee-daa14c9e-ade7-11e3-a415-108350ae7b5e.shtml (consultato il giorno Novembre 13, 2014).

Crockford, Douglas. *JavaScript: The Good Parts.* Sebastopol, CA: O'Reilly Media, Inc.,, 2008.

Croft, Jeff, Ian Lloyd, and Dan Rubin. *Pro CSS Techniques.* New York: 2006, 2006.

ECMA International. *ECMAScript Language Specification.* Ginevra: Ecma International, 2011.

Fain, Yakov, Victor Rasputnis, e Anatole Tartakovsky. *Enterprise Development with Flex.* Sebastopol, Californa: O'Reilly Media, 2010.

GARCIA, JESUS, ANTHONY DE MOSS, and MITCHELL SIMOENS. *Sencha Touch in Action.* Shelter Island, NY 11964: Manning Publications Co., 2013.

GARCIA, JESUS, GRGUR GRISOGONO, and JACOB K. ANDRESEN. *Ext JS in Action, Second Edition.* Shelter Island, NY 11964: Manning Publications Co., 2014.

Groner, Loiane. *Mastering Ext JS.* Birmingham B3 2PB, UK..: Packt Publishing Ltd., 2013.

«http://www.alexa.com/siteinfo/http%3A%2F%2Fwww.protezionecivile.gov.it.» *Alexa.* s.d. 2014.

Kernighan, Brian. *http://it.wikipedia.org/wiki/Brian_Kernighan.* s.d. (consultato il giorno Novembre 25, 2014).

Kosmaczewski, Adrian. *Mobile JavaScript Application Development.* Sebastopol, CA: O'Reilly Media, Inc.,, 2012.

Kumar, Ajit. *Sencha MVC Architecture.* Birmingham B3 2PB, UK.: Packt Publishing Ltd., 2012.

macromedia. *Esercitazioni di Flash MX.* San Francisco, California: Macromedia, Inc., 2002.

Mednieks, Zigurd, Laird Dornin, G. Blake Meike, and Masumi Nakamura. *Programming Android.* O'Reilly, 2012.

Mohammad, Abdullah Al. *Ext JS 4 Plugin and Extension Development.* Birmingham B3 2PB, UK.: Packt Publishing Ltd., 2013.

Myer, Thomas. *Beginning PhoneGap.* Indianapolis, IN 46256: John Wiley & Sons, Inc., 2012.

Oracle Corporation. *Oracle® GlassFish Server 3.0.1.* Oracle Corporation, 2010.

—. *The Java EE 6Tutorial.* Oracle Corporation, 2010.

Powers, Shelley. *Learning Node.* Sebastopol, California: O'Reilly, 2012.

Prime technology. *Primefaces USER'S GUIDE 3.4.* Ankara, 2009.

Regione Emilia Romagna. «• PRONTUARIO delle violazioni accertabili dalle GUARDIE GIURATE ECOLOGICHE VOLONTARIE .» *http://www.federgev-emiliaromagna.it/files/2011-06_prontuario_a4_gev_regione.pdf.* 08 Giugno 2011. http://www.federgev-emiliaromagna.it/files/2011-06_prontuario_a4_gev_regione.pdf (consultato il giorno Novembre 13, 2014).

—. «Raccolta di leggi per i corsi di formazione delle Guardie Ecologiche Volontarie.» *http://ambiente.regione.emilia-romagna.it/.* Settembre 2010. http://ambiente.regione.emilia-romagna.it/parchi-natura2000/fruizione/volontariato/gev/guardie-ecologiche-volontarie/le-norme-e-gli-atti-in-vigore/LeggiGEV_vol1_settembre2010.pdf (consultato il giorno Novembre 13, 2014).

Statista. s.d. http://www.statista.com/statistics/203964/global-software-spending-forecast/ (consultato il giorno Novembre 26, 2014).

Stefanov, Stoyan. *Object-Oriented JavaScript.* Birmingham, B27 6PA, UK.: Packt Publishing Ltd., 2008.

Svennerberg, Gabriel. *Beginning Google Maps API 3.* Apress, 2010.

Villa, Crysfel, and Armando Gonzalez. *Learning Ext JS 4.* Birmingham B3 2PB, UK.: Packt Publishing Ltd., 2013.

Yaapa, Hage. *Express Web Application Development.* Birmingham B3 2PB, UK: Packt Publishing, 2013.

8 INDICE DELLE FIGURE