UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI INGEGNERIA ELETTRONICA

TESI DI LAUREA

IMPLEMENTAZIONE DEL PROTOCOLLO EPP PER LA COMUNICAZIONE CON IL NIC

Relatore:

Prof. L. G. Paccagnella

Laureando:

Costantino Pietro

ANNO ACCADEMICO 2010-2011

INDICE:

CAPITOLO 1: INTRODUZIONE

1.4 Le reti di telecomunicazione

La necessità di creare una rete di telecomunicazioni nasce dal problema di far comunicare le persone anche a grandi distanze, utilizzando dei dispositivi interconnessi tra loro (telefoni, computer, palmari...). A tutti gli effetti una rete di telecomunicazioni altro non è che una rete di dispositivi e dei loro collegamenti, i quali consentono la trasmissione e la ricezione di qualsiasi tipo di informazione, anche trovandosi in punti geograficamente molto distanti.

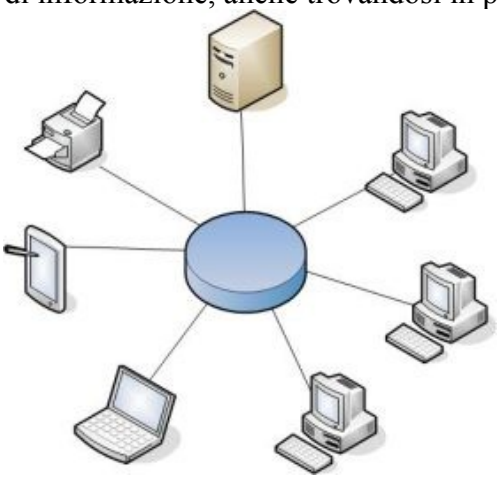

fig. 1.1 : Rete di calcolatori

La trasmissione è possibile tramite il trasferimento di dati attraverso cavi, sistemi radio, oppure qualsiasi altro sistema elettromagnetico o ottico. Ogni dispositivo connesso a questa rete di telecomunicazioni si differenzia dagli altri per mezzo di un indirizzo univoco proprio, che appunto per la sua peculiarità lo distingue da tutti gli altri connessi.
La più grande rete di telecomunicazioni si chiama “Internet” (dal latino *inter*, "tra" e dall'inglese *net*, "rete", *tra la rete*); si tratta di una rete mondiale ad accesso pubblico che al giorno d'oggi rappresenta il principale mezzo di comunicazione di massa. Internet è nata negli anni sessanta nell'ambito militare; sono stati gli americani a progettarla allo scopo di creare un fitto sistema di controspionaggio e di difesa contro i paesi sovietici durante la guerra fredda. L'antenato di Internet fu il progetto “Arpanet”, finanziato dall'”Agenzia per i Progetti di Ricerca per la Difesa Avanzata” (“Darpa”) americana, un'agenzia militare statunitense finalizzata alla difesa militare del paese. La prima rete di computer fu teorizzata nel 1962 dagli scienziati Joseph Licklider e Welden E. Clark col nome di "Intergalactic Computer Network", ma ebbe piena realizzazione solo nel 1969. Essa collegava quattro nodi: l'Università della California di Los Angeles, l'SRI di Stanford, l'Università della California di Santa Barbara, l'Università dello Utah e viaggiava ad una velocità di 50 Kbps. In pochi anni Arpanet si allargò di molto, estendendosi fino all'Europa. In Francia venne realizzata “Cyclades” sotto le direttive di Louis Pouzin, in Norvegia “Norsar” che collegava Arpanet con l'università di Londra.
Nel 1991 il ricercatore Tim Bernes-Lee al CERN di Ginevra definì il protocollo Http, il primo sistema per la lettura di documenti ipertestuali. Nacque quindi il concetto di “link” o “collegamento ipertestuale”, contestualmente il governo americano varò la *High performance computing act,* legge con cui per la prima volta veniva prevista la possibilità di ampliare, ad opera dei privati e con finalità di sfruttamento commerciale, una rete che ebbe la veste ufficiale di Internet. Nel 1993 venne realizzato il primo browser chiamato “Mosaic”, esso rivoluzionò completamente il mondo di Internet, consentendo la nascita del “World Wide Web”, una delle pietre miliari su cui si basa l'attuale sistema ipertestuale. Infatti nel World Wide Web le informazioni sono contenute in librerie o “pagine” a cui si può accedere mediante l'utilizzo di appositi software, quali i browser, e si può scaricare file, accedere a immagini, filmati, tracce audio ….
Oggi Internet conta più di 1,5 miliardi di utenti, offre i più svariati servizi tra cui quello di posta elettronica; chiunque può accedervi purché disponga di un computer e di un browser, appoggiandosi ad un Internet Service Provider il quale gli fornisce un accesso tramite una linea di telecomunicazione dedicata, ad esempio l'adsl. Allo scopo di mettere in comunicazione diverse entità l'Internet Engineering Task Force, società che si occupa dell'evoluzione tecnica e tecnologica di Internet, ha stabilito alcune “regole” o “protocolli”, ovvero una serie di istruzioni che devono essere eseguite da qualsiasi elemento connesso alla rete. Il protocollo più importante è il “Protocollo IP” (Internet Protocol) il quale definisce il linguaggio costituente i messaggi scambiati in rete.

Un altro protocollo importante è il TCP (Transmission Control Protocol), un protocollo di trasporto che stabilisce un canale di comunicazione affidabile tra due dispositivi interconnessi i quali consentono uno scambio sicuro di informazioni con “ricevuta di ritorno”.
Tutti i dispositivi connessi alla rete vengono denominati “host”; con in suddetto termine si possono intendere il computer o il palmare con cui ci si connette ad Internet, oppure il modem o ancora la stampante, tutti dispositivi che ospitano i programmi adatti alla navigazione.
Un host può essere essenzialmente di due tipi: “client” oppure “server”. Un client, o cliente, è un elemento connesso alla rete che accede ai servizi o alle risorse di un altro elemento, anch'esso connesso alla rete, predisposto a condividere i suoi contenuti e che può essere di tipo hardware (un personal computer) o di tipo software (un programma di posta elettronica). Un server è questo secondo elemento, connesso a decine/centinaia di client, ovvero un componente informatico che fornisce vari tipi di servizi e permette la condivisione di dati. Un server, essendo un computer utilizzato per fornire servizi ad altri computer, necessita di una struttura hardware più complessa rispetto ad un semplice client, come ad esempio molta più memoria o la possibilità di sostenere più connessioni con altri pc. Per semplificare la comunicazione e la ricerca di informazioni all'interno della rete gli “utenti”, ovvero coloro che vogliono condividere dei dati, creano la propria “pagina web” o “sito internet”, un insieme di pagine correlate tramite una struttura ipertestuale di facile consultazione, che permettono di accedere con semplicità ai dati presenti. L'insieme di dati condivisi e l'architettura a pagine web risiede fisicamente nella memoria di un server connesso alla rete secondo regole stabilite dai protocolli.
Ciascun host connesso ad Internet, per essere univocamente riconosciuto, riceve dal proprio provider un codice chiamato “indirizzo ip”, costituito da 4 terzetti di cifre divise tra loro da un punto. Ogni terzetto comprende cifre che variano tra 0 e 255; essendo il numero totale di indirizzi limitato, ad un elemento client connesso alla rete viene assegnato un indirizzo “dinamico”, ovvero ad ogni connessione alla rete il provider assegna al client un indirizzo diverso. Diverso è per un server, il quale deve essere sempre riconoscibile all'interno della rete, a cui viene assegnato un indirizzo “statico”, cioè permanente anche in caso di disconnessione dalla rete, che è salvato all'interno di un database. Non a caso avviene questa scelta: la rete può essere rappresentata da un grafo, i cui nodi sono gli host; ogni client che vuole stabilire una connessione con un server, oltre ad avere il suo indirizzo ip, deve conoscere la strada per poterlo raggiungere. I provider contengono la “tabella di routing”, cioè una tabella contenente i percorsi più brevi per potersi muovere da un punto a un altro della rete e, di conseguenza, poter raggiungere i server connessi. Quando un client si rivolge al suo provider per richiedere la connessione con un qualsiasi server, ne fornisce l'indirizzo e ne consulta la tabella per individuarne il percorso. Se l'indirizzo di arrivo non fosse statico, il server sarebbe irrintracciabile.

1.2 I domini .it

Per un calcolatore è semplice memorizzare delle cifre, non lo è altrettanto per l'uomo. Viene pertanto utilizzato un sistema che associa stringhe alfanumeriche facili da ricordare a indirizzi numerici: il “DNS”. Con tale termine (domain name system) si intente un insieme di regole che permettono di associare ad un indirizzo un “nome a dominio”, regole stabilite dal W3C (World Wide Web Consortium). Un nome è costituito da una serie di stringhe separate da punti, ad es. www.ing.unipd.it. La parte più importante, denominata “dominio di primo livello”, è la prima contrassegnata a destra “*.it*”, a sinistra vi è il “dominio di secondo livello” costituito da due parti “*unipd.it*” e così via. In questo esempio si arriva al massimo ad un “dominio di quarto livello”.Il dominio di primo livello è quindi l'ultima parte di un nome a dominio, la sigla alfanumerica che segue il punto più a destra. La Internet Assigned Numbers Authority (IANA) classifica attualmente i domini di primo livello in tre tipologie:

- domini di primo livello nazionali (country code top-level domain o ccTLD), utilizzati da uno

stato o da un territorio, essi sono costituiti da due lettere (it, fr, eu...).
- domini di primo livello generici (generic top-level domain o gTLD), impiegati da particolari classi di organizzazioni (com, edu, mil...).
- domini di primo livello infrastrutturali, attualmente "arpa" è l'unico esistente.

fig. 1.2 Logo del Registro.it

Il dominio di primo livello nazionale assegnato all'italia è il .it. Il 23 dicembre 1987 IANA cedette la gestione del controllo sui ccTLD italiani al Centro Nazionale Universitario del Calcolo Elettronico, un istituto del CNR di Pisa, il quale registrò il suo primo dominio nel 1988. Dal 1992 al 1997 il servizio fu curato dal Network Information Sevice (NIS) ed in seguito dall'istituto per le Applicazioni Telematiche (IAT), anch'esso organo del CNR. Nel 2002 una fusione di più istituti del CNR fece nascere l'Istituto di Informatica e Telematica, il quale prese in consegna l'onere di mantenere il registro dei domini .it. Fu solo nel 2009 che nacque il Registro.it, l'anagrafe di tutti i domini .it. Chiunque può creare il proprio sito internet, purché disponga delle risorse necessarie (server collegato alla rete), ed in seguito registrarlo all'anagrafe dei domini. Nonostante la sigla .it sia il simbolo del "Made in Italy" l'estensione a qualunque entità (cittadino o società) dell'autorizzazione a registrare un nome a dominio dà la possibilità anche a stranienri di utilizzare questa targa, abolendo al suffisso .it qualsiasi forma di appartenenza geografica.
Per poter creare, modificare o cancellare uno più domini .it è necessario rivolgersi al Registro.it o NIC (Network Information Center), l'anagrafe dei domini.it, il quale su richiesta degli utenti provvede ad associare all'indirizzo ip statico assegnato al server di riferimento il nome a dominio scelto (purché compatibile con le politiche adottate). Il NIC non si occupa direttamente della registrazione di nuovi indirizzi per conto degli utenti, ma delega questo compito a delle società che stipulano un contratto con lui, lasciandogli piena autonomia di gestione. Queste imprese agiscono da intermediarie tra i privati e il NIC applicando tariffe e scadenze a loro piacimento; chiunque quindi volesse registrare un nome a dominio, si deve rivolgere ad una di queste ultime, lasciando loro il compito di comunicare con il Registro.it.

1.3 Epp, Registrar, Registrant

Un utente, cittadino privato o società, che vuole registrare un dominio per sé, prende il nome di registrante o "Registrant"; egli, non potendo comunicare direttamente con il NIC, deve rivolgersi a una società che ha stipulato un contratto, la quale a sua volta prende il nome di Registrar. La comunicazione con il NIC, per la registrazione dei domini fino al 1 Luglio 2010, avveniva in modalità asincrona (o cartacea). Il Registrar inviava un fax con tutti i dati necessari alla registrazione, rispettando le politiche del sistema, la metodologia e i tempi d'attesa richiesti. La risposta di avvenuta operazione, però, non era immediata, bensì giungeva solo in un secondo momento, spesso dopo qualche giorno (i tempi d'attesa dipendevano dalla tipologia di richiesta). Per questo, a partire dal 1 Luglio 2010, il Registro.it mise a disposizione un server (all'indirizzo epp.nic.it) per la gestione delle richieste in modalità sincrona, in modo che ciascuna di esse avvenisse telematicamente con tempi di risposta molto brevi (se non immediati). La modalità sincrona è quella tutt'oggi utilizzata dal NIC per la maggior parte delle richieste, fatta eccezione per qualcuna che utilizza la vecchia modalità asincrona. In rete, come già detto, per consentire di mettere in comunicazione macchine differenti situate in locazioni geografiche diverse, è necessario stabilire delle regole o protocolli, che rappresentano un linguaggio comune di interazione. Per relazionarsi con il server NIC la regola di comunicazione è rappresentata dal "protocollo EPP" (Extensible Provisioning Protocol). Si tratta di uno dei numerosi protocolli di rete, atto a comunicare l'inserimento di oggetti all'interno di database; per questa sua peculiarità viene utilizzato da numerosi registri di nomi a dominio (tra cui il Registro.it), e da altri come l'ENUM, il registro dei prefissi di telefonia nazionale. Il protocollo, creato nel 2004 dalla IETF, è basato sullo

XML, un particolare formato di testo molto utilizzato nella comunicazione tra le macchine; si tratta di un protocollo “stateful” nel quale parte delle informazioni scambiate include il concetto di stato ed entrambi i sistemi tengono traccia della sessione di comunicazione e come protocollo di trasporto utilizza il TCP.

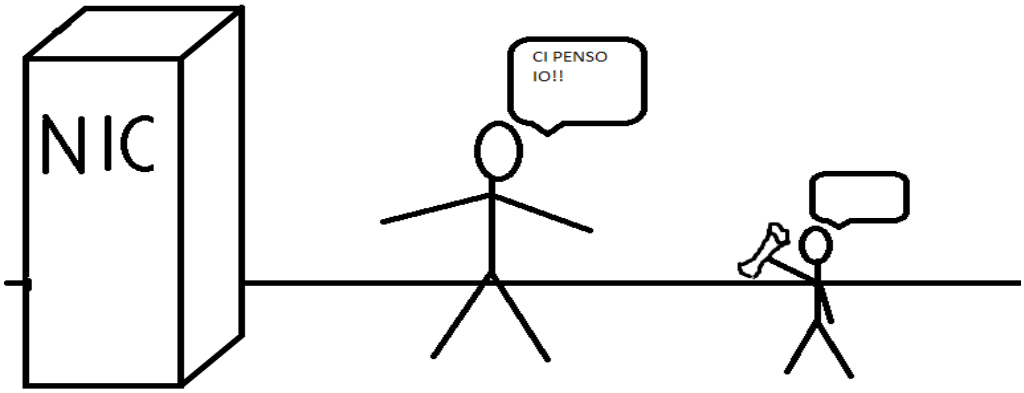

fig. 1.3 Una Registrant non può comunicare direttamente con NIC per registrare un nome a dominio ma deve passare attraverso la consulenza di un registrar

L'interazione è basata su una serie di scambi “richiesta-risposta”, ad ogni richiesta inviata corrisponde una risposta da parte del server. Il protocollo EPP prevede che ogni richiesta contenga i seguenti elementi:

- Un elemento iniziale standard:
  “<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">”
- Un elemento che può essere di due tipi:
  - “<hello>” per inviare un comando hello
  - “<command>” per inviare la richiesta di una qualsiasi altra operazione.
    Tale elemento contiene anche i seguenti:
    - Un elemento opzionale “<extension>” il quale può essere utilizzato per le estensioni definite dal server alle richieste dei comandi.
    - Un elemento opzionale “<clTRID>”, che può essere utilizzato dal client per identificare logicamente una transazione, non è altro che una stringa alfanumerica es. “<clTRID>ABC-12345</clTRID>”
- Un elemento finale standard “</epp>”

Ogni risposta inviata dal server contiene i seguenti elementi:

- un elemento iniziale standard
  “<?xml version="1.0" encoding="UTF-8" ?>
  <epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">”
- una serie di elementi diversi in base alla richiesta ricevuta
  - un elemento “<greeting>” in risposta all'hello

- uno o più elementi “<result>” che documentano il successo o il fallimento dell’esecuzione del comando richiesto
- un attributo code: codice di ritorno dell’operazione richiesta
- un elemento “<msg>” con una descrizione testuale del codice di ritorno
- zero o più elementi “<value>” che identificano gli elementi
- zero o più elementi “<extValue>” che possono essere utilizzati per fornire informazioni diagnostiche addizionali
- un elemento “<value>” che identifica un elemento inserito nella richiesta che ha causato un errore.
- un elemento “<reason>” con una descrizione testuale della ragione dell’errore

- un elemento opzionale “<msgQ>” che descrive i messaggi presenti nella coda di polling del Registrar.
- un elemento opzionale <resData> che contiene gli elementi specifici della risposta associata al comando richiesto
- un elemento finale standard “</epp>”

Le tipologie di richieste, chiamate anche comandi sono in tutto 18:

- crea contatto
- crea dominio
- modifica contatto
- modifica dominio
- modifica registrante
- modifica registrar
- modifica registrar con contestuale modifica del registrante
- cancella contatto
- cancella dominio
- ripristina dominio
- check contact
- check domain
- info contact
- info domain
- intorrogazione coda di polling
- login
- logout
- hello

Tutti questi comandi verranno ripresi e spiegati più avanti.
Concludendo, per comunicare con il server del NIC è necessario comporre un documento di testo con estensione XML secondo le specifiche del protocollo EPP descritte precedentemente e implementare uno dei comandi prestabiliti, (a seconda del comando il codice del testo è differente), inviare il file (ad esempio tramite richiesta “post”) al server, ricevere ed interpretare la risposta.
Di seguito è proposta un esempio di richiesta:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
<command>
<login>
<clID>esempioid</clID>
<pw>esempiopw</pw>
```

<options>
<version>1.0</version>
<lang>en</lang>
</options>

...

</login>
</command>
</epp>

Come si può notare dal tag <command >" il comando è di login, è presente l'elemento iniziale "<epp>" mentre la username (esempioid) è stata inserita all'interno dei tag "<clID>" e la password "<pw>".
Tutte le definizioni di comandi o oggetti utilizzati nell'implementazione del protocollo EPP sono contenuti in alcuni XML Schema:

- XML Schema standard del protocollo epp:
- epp-1.0.xsd: Extensible Provisioning Protocol v1.0 schema
  - domain-1.0.xsd: Extensible Provisioning Protocol v1.0 domain provisioning schema
  - contact-1.0.xsd: Extensible Provisioning Protocol v1.0 contact provisioning schema
  - eppcom-1.0.xsd: Extensible Provisioning Protocol v1.0 shared structures schema
- XML Schema che riguarda l'estensione per la gestione del grace period adottata dal Registro:
  - rgp-1.0.xsd: Extensible Provisioning Protocol v1.0 domain name extension schema for Registry grace period processing
- XML Schema che riguardano le estensioni al protocollo EPP definite dal Registro:
  - extepp-1.0.xsd: IT-NIC Extensible Provisioning Protocol v1.0 EPP extension.
  - extcon-1.0.xsd: IT-NIC Extensible Provisioning Protocol v1.0 domain extension
  - extdom-1.0.xsd: IT-NIC Extensible Provisioning Protocol v1.0 contact extension

Di seguito è presentata una parte dell' XML Schema "epp-1.0.xsd":

<?xml version="1.0" encoding="UTF-8"?>

...

```
 <element name="epp" type="epp:eppType" />

 <complexType name="eppType">
  <choice>
   <element name="greeting" type="epp:greetingType" />
   <element name="hello" />
   <element name="command" type="epp:commandType" />
   <element name="response" type="epp:responseType" />
   <element name="extension" type="epp:extAnyType" />
  </choice>
 </complexType>

 ...
```

```
  <complexType name="commandType">
    <sequence>
      <choice>
        <element name="check" type="epp:readWriteType" />
        <element name="create" type="epp:readWriteType" />
        <element name="delete" type="epp:readWriteType" />
        <element name="info" type="epp:readWriteType" />
        <element name="login" type="epp:loginType" />
        <element name="logout" />
        <element name="poll" type="epp:pollType" />
        <element name="renew" type="epp:readWriteType" />
        <element name="transfer" type="epp:transferType" />
        <element name="update" type="epp:readWriteType" />
      </choice>
      <element name="extension" type="epp:extAnyType" minOccurs="0" />
      <element name="clTRID" type="epp:trIDStringType" minOccurs="0" />
    </sequence>
  </complexType>

  <complexType name="loginType">
    <sequence>
      <element name="clID" type="eppcom:clIDType" />
      <element name="pw" type="epp:pwType" />
      <element name="newPW" type="epp:pwType" minOccurs="0" />
      <element name="options" type="epp:credsOptionsType" />
      <element name="svcs" type="epp:loginSvcType" />
    </sequence>
  </complexType>

...

  <simpleType name="pwType">
    <restriction base="token">
      <minLength value="6" />
      <maxLength value="16" />
    </restriction>
  </simpleType>

...

  <simpleType name="transferOpType">
    <restriction base="token">
      <enumeration value="approve" />
      <enumeration value="cancel" />
      <enumeration value="query" />
      <enumeration value="reject" />
      <enumeration value="request" />
    </restriction>
  </simpleType>
```

```
...

 <complexType name="trIDType">
  <sequence>
   <element name="clTRID" type="epp:trIDStringType" minOccurs="0" />
   <element name="svTRID" type="epp:trIDStringType" />
  </sequence>
 </complexType>

 <complexType name="msgQType">
  <sequence>
   <element name="qDate" type="dateTime" minOccurs="0" />
   <element name="msg" type="epp:mixedMsgType" minOccurs="0" />
  </sequence>
  <attribute name="count" type="unsignedLong" use="required" />
  <attribute name="id" type="eppcom:minTokenType" use="required" />
 </complexType>

...

</schema>
```

Si noti come lo schema imponga come primo elemento “<epp>”, il quale poi può essere seguito da elementi quali “<greeting>, <hello>, <command>”, ecc...

CAPITOLO 2: IL PROGETTO

2.1 Edistar s.r.l.

fig. 2.1 Logo di Edistar

Una delle società che fornisce il servizio di intermediazione tra Registrant e NIC è EDISTAR S.R.L. Fondata nel 1993 per opera di Marco Trevisan e Maurizio Jorge Vincenzi nasce a Castelfranco Veneto come centro servizi specializzato nella gestione automatica di applicazioni telefoniche; nel 1995 si trasferisce a Vedelago dove tuttora risiede con i suoi uffici.
Edistar è una società di telecomunicazioni i cui servizi sono mirati ad aiutare le aziende a comunicare nel miglior modo possibile con la propria clientela tramite soluzioni multicanale che riescono a migliorare i contatti riducendo i costi.
A livello applicativo gli ambiti spaziano dai servizi telefonici, come la messaggistica e i call center, dove vengono monitorate le linee telefoniche fornendo statistiche e risorse utilizzate, fino ai concorsi a premi, con la realizzazione di siti web e la messa a disposizione di server.
La strategia è quella di colpire diversi mercati attraverso soluzioni in outsourcing e white label: outsourcing significa letteralmente “esternizzazione”, ricorrere all'aiuto di altre imprese per lo svolgimento di alcune fasi del processo produttivo; è il concetto opposto al “fai date”. Quando si necessita di un servizio particolare di alta qualità ma non si dispone delle risorse necessarie, invece di affrontare pesanti costi ed aggiornamenti si può acquistare il servizio dall'esterno con la garanzia di un ottimo prodotto. Edistar è una società che lavora in outsourcing non producendo dei beni ma fornendo dei servizi a pagamento, sfruttando le proprie risorse mirate allo scopo. Il white label significa letteralmente “etichetta bianca”, una strategia di mercato mirata allo sfruttamento di rivenditori esterni. Ogni commerciale vende ai suoi clienti i prodotti Edistar sostituendo al marchio di copypight il proprio, ma senza per questo violarne il diritto d'autore, in quanto per ogni copia venduta una parte dei guadagni torna alla società di produzione.
Per quanto riguarda il personale la società Edistar è suddivisa in 3 settori d'impiego: la parte tecnica costituita da esperti tecnici informatici e periti che è il cuore dell'azienda, il nucleo operativo dove vengono sfornati i prodotti di marchio Edistar; una parte Amministrativa-commerciale composta da commerciali interni e segretari; infine una fitta rete di commerciali esterni dediti a mantenere i rapporti coi clienti.

2.2 Introduzione al software per la comunicazione col NIC

Edistar è un'azienda che ha stipulato un contratto con il Registro.it, di conseguenza ha dovuto adattarsi alla nuova modalità sincrona per la registrazione di nomi a dominio. Tutto ciò ha portato allo sviluppo di un software finalizzato a eseguire tutta una serie di operazioni di registrazione e modifica in automatico da una qualsiasi postazione computer di un dipendente.
Il progetto, che comprende una semplice interfaccia grafica codificata in html che ne consente la navigazione, è scritto interamente in codice php. Esso è strutturato in modo da elaborare, in modo semplice e veloce, i dati inseriti tramite le interfacce grafiche, compilare i file XML ed inviare la richiesta al server NIC. Il programma interagisce anche con il database “postgreSQL”, in modo da tener traccia delle operazioni effettuate ed avere il quadro completo della situazione con domini e contatti, registrati in ogni momento. Inoltre, il database è utile al programma durante le operazioni di modifica, nelle quali è necessario conoscere i valori precedentemente impostati, piuttosto che ripristinare i domini eliminati. PostrgreSQL utilizza il linguaggio SQL per eseguire i comandi o “query” sui dati. Questi ultimi sono conservati come tabelle con chiavi primarie e chiavi esterne. La chiave primaria è costituita da una o più colonne della tabella, che identifica l'unicità del record. Ogni tabella deve avere almeno una chiave primaria. Inoltre non possono esserci due o più record

uguali nella stessa colonna, la quale è definita come chiave primaria. La chiave esterna, invece, è un vincolo referenziale tra due tabelle, identifica una colonna o un insieme di colonne di una stessa tabella che, a loro volta, referenzia una colonna o un insieme di colonne di un'altra tabella. I valori di un record delle colonne referenzianti devono essere presenti in un record della tabella referenziata.

2.3 Istruzioni per l'uso

I presenti argomenti illustrano le istruzioni per l'utilizzo del software; sono specificate in modo esaustivo le operazioni fondamentali, corredate di riferimenti ad immagini e tabelle.
La prima fase del lavoro consiste nell'effettuare la login: inserire username e password negli appositi spazi, come illustrato in fig. 2.2 e selezionare Invia. Se non si dispone di username e password è necessario contattare il Registro.it tramite email al conatto epp@nic.it.
Se le credenziali inserite sono corrette, si accederà alla schermata successiva (il menù principale) fig. 2.3 dove è possibile effettuare le varie operazioni; diversamente verrà ripresentata la schermata di login.

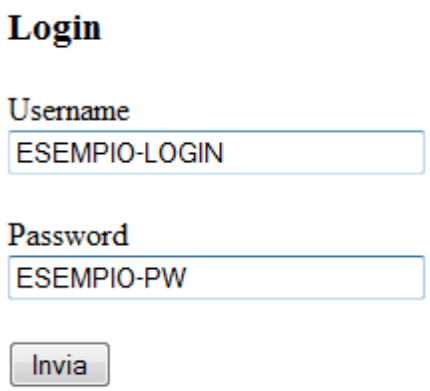

fig. 2.2 login

Il menù principale presenta diverse opzioni: gestione account, gestione domini, sincronizza contatti, sincronizza domini, altro e logout.

fig.2.3 Menù principale

Vediamole nel dettaglio.
Gestione account è una sezione dedicata alle operazioni sui contatti, qui si possono creare nuovi account, modificare i presenti e cancellarli. La schermata è composta da una tabella contenente i contatti registrati, fig. 2.4 (sono presenti quelli registrati e salvati all'interno del database locale; nulla toglie che per un qualsiasi errore gli account presenti all'interno del database del Registro.it contengano qualche piccola differenza), sui quali è possibile effettuare le operazioni di modifica. Analoga a questa sezione è *gestione domini* fig. 2.5; qui è possibile eseguire varie operazioni sui domini registrati (creazione, cancellazione e modifica). Vi è inoltre la tabella dei domini eliminati, i quali possono però essere ripristinati.

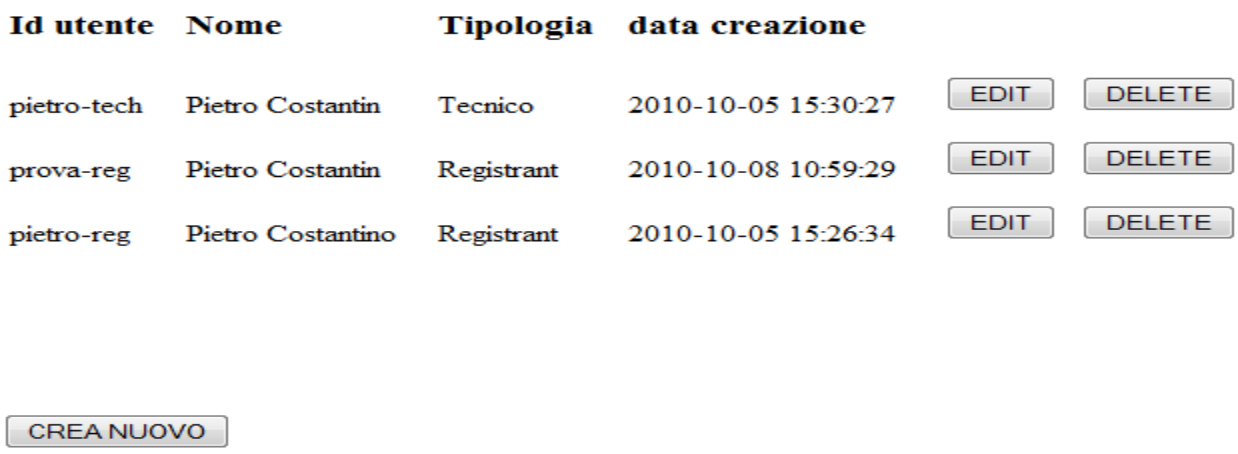

fig 2.4 Gestione contatti

fig. 2.5 Gestione domini, sono presenti pure i domini eliminati

Il link a *logout*, effettua la logout dal sistema, mentre *Altro* è una sezione dedicata alle operazioni asincrone rispetto al database; in questa pagina è possibile gestire richieste particolari descritte in seguito fig. 2.6. Selezionando *Sincronizza Contatti* e *Sincronizza Domini* il software inizia una serie di operazioni per aggiornare il database locale con quello remoto, correggendo eventuali differenze.

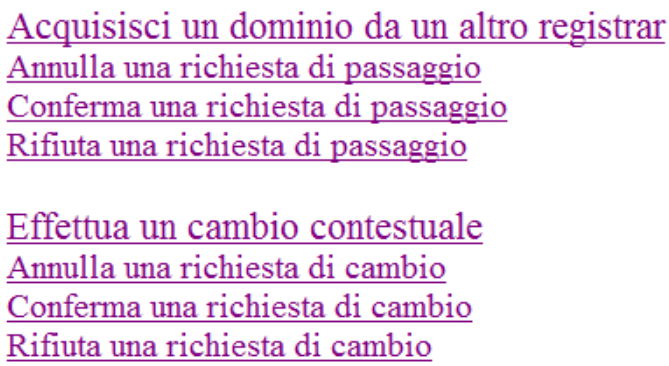

fig. 2.6 Altro

Crea contatto: andare su *gestione contatti->crea nuovo*; viene visualizzata una schermata simile a quella di fig. 2.7, compilare i campi e inviare il form. Segue una tabella riassuntiva dei dati da inserire e delle cardinalità (con 0 elemento non obbligatorio, con 1 obbligatorio, con n,m n obbligatori fino ad un massimo di m).
I contatti registrabili sono di 3 tipi: Registrant, Admin e Tech, Il contatto di tipo Registrant corrisponde all'utente al quale vengono assegnati i nomi a dominio da registrare e, solo in questo caso, vanno compilati gli ultimo tre campi presenti nella tabella.

CAMPO	CARDINALITA'	LUNGHEZZA	VALORE
ID del contatto	1	1-16	A-Z, a-z, 0-9, -
Nome	1	1-255	
Organizzazione	0	1-255	
Via/Piazza	1,3	1-128	
Città	1	1-128	
Provincia	1	1-128	Se nazione = IT allora deve

			contenere la sigla a 2 lettere di una provincia italiana
CAP	1	1-16	CAP
Nazione	1	2	Deve essere riportato in codice ISO 3166-1 a 2 lettere
Telefono	1		Deve essere indicato in formato ISO (+39.123456789)
Telefono interno	0	1-10	
Fax	1		Deve essere indicato in formato ISO (+39.123456789)
Fax interno	0	1-10	
Email	1		Deve essere indicato nel formato RFC2822
Authinfo	0		
ConsentForPublishing	1	1	Valori ammessi: 1 = vero, 2 = falso
Tipo	1		
Nazionalità	0	2	Obbligatorio per contatto di tipo Registrant
EntityType	0	1	Obbligatorio per contatto di tipo Registrant
RegCode	0	1-36	Obbligatorio per contatto di tipo Registrant

Nel caso venga inserito un valore non corretto, compare una stringa di errore a fianco dell'apposita casella di input. Una volta terminata l'operazione, l'esito positivo viene confermato mediante una stringa di successo e un link al menù principale (o al logout), in caso contrario il NIC fornisce una risposta differente in base al tipo d'errore commesso.
All'interno del database del Registro.it viene creato un oggetto di tipo contact nello stato di “ok”, viene inserita la data di registrazione e, se il contatto è di tipo “registrant” ed è una persona fisica (cioè EntityType = 1), il campo Organizzazione, se vuoto, viene forzato al valore del campo Nome.

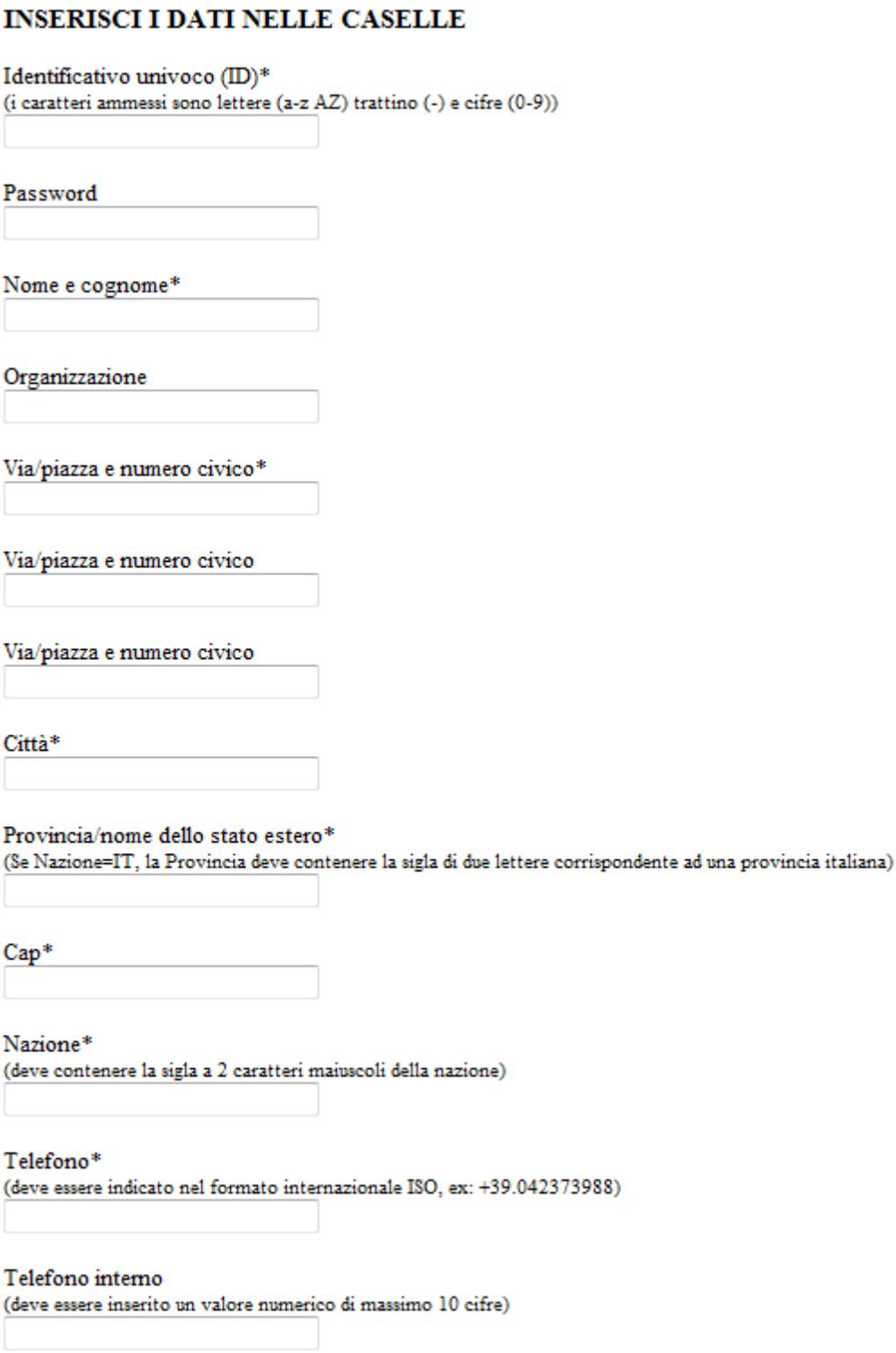

fig. 2.7 crea contatto

Modifica contatto: selezionare *edit* a fianco del suddetto, si presenta una schermata simile a quella di fig. 2.8 dove i campi sono già compilati con i valori precedentemente impostati. I valori modificabili sono: nome, organizzazione, via, città, provincia, cap, nazione, telefono, fax, email, ConsentForPublishing, Nazionalità, EntityType, RegCode.

Inoltre è possibile aggiungere lo status “clientDeleteProhibited”, ovvero assegnarvi una protezione per impedirne l'eliminazione. Quest'aggiunta, assegnabile solo in fase di modifica, è possibile spuntando la checkbox corrispondente in testa alle finestre di input. Nel caso lo status sia già presente vi è la possibilità di rimuoverlo nel medesimo modo. A seguito di quest'operazione il database del NIC verrà aggiornato con i nuovi valori e la data dell'ultima modifica.

fig.2.8 Modifica contatto

Cancella contatto: precisato che non è possibile eliminare un account a cui vi sono assegnati uno o più nomi a dominio, basta andare su *gestione contatti* e selezionare *cancella* a fianco del contatto da cancellare, premere *si* sulla schermata successiva.
Crea dominio: andare su *gestione domini->crea nuovo*, viene visualizzata una schermata simile a

quella di fig. 2.9, compilare i campi e inviare il form. Prima di registrare un nome a dominio bisogna accertarsi di aver inserito nel database del NIC almeno un contatto di tipo tecnico, e, in caso il “Registrant” non sia una persona fisica, anche un contatto di tipo “admin”.Segue una tabella riassuntiva dei dati da inserire con le rispettive cardinalità.

CAMPO	CARDINALITA'	VALORE
Nome del dominio	1	A-z, 0-9, -
Periodo	0	Di default 1
Unità di tempo	0	Valori ammessi: y = anno, m = mese
Nome dell'host	2-6.	Nome degli host da associare (min. 2)
Indirizzo IP	0	Indirizzi ip degli host associati
Tipo di indirizzo IP	0	Di default ipv4
Registrante	1	ID del registrante
Contatto admin	1	ID del contatto admin, se il registrante è una persona fisica vi coincide
Contatto tech	1-6	ID del contatto tecnico
Authinfo	1	

Nel caso venga inserito un valore non corretto, compare una stringa di errore a fianco dell'apposita casella di input. Una volta terminata l'operazione, l'esito positivo viene confermato mediante una stringa di successo e un link al menù principale (o al logout), in caso contrario il NIC fornisce una risposta differente in base al tipo d'errore commesso.
All'interno del database del Registro.it viene creato un oggetto di tipo domain, insieme alla sua data di creazione e la data di scadenza; viene addebitato al Registrant e disponibile per la fatturazione. Il dominio rimane in uno status di *inactive/DNSHold* per un periodo non superiore ai 30 giorni, durante i quali è possibile effettuare operazioni di modifica. In questi giorni il NIC si occupa di verificare periodicamente se la configurazione DNS per la lista degli host è positiva. Se allo scadere dei 30 giorni il controllo dovesse fallire il nome a dominio passa ad uno stato di *pending delete* e, dopo un periodo massimo di 5 giorni, viene rimosso.
Modifica dominio: selezionare *edit* a fianco del suddetto, si presenta una schermata simile a quella di fig. 2.10 dove il campo password è preimpostato con il valore precedente. In testa al form sono presenti una o più checkbox e alla fine una serie di finestre per la selezione dei contatti tecnici e degli host. Le operazioni ammissibili sono la modifica della password, l'eliminazione di contatti tecnici e di host spuntando le relative checkbox in alto e l'aggiunta di nuovi contatti. A operazione completata il NIC aggiorna il suo database con l'aggiunta della data di ultima modifica; se vengono modificati gli host, il dominio resta in uno stato di *pending update* per un tempo massimo di 5 giorni durante i quali il server NIC effettua i controlli sul DNS. Se al termine dei giorni il controllo DNS fallisce il database del registro non aggiorna gli host.
Cancella dominio: selezionare *delete* a fianco del suddetto, selezionare *si* nella schermata successiva. Il dominio rimane in uno stato di *pending delete* per un tempo massimo di 30 giorni, durante i quali compare all'interno della tabella sottostante. È possibile ripristinarlo selezionando *ripristina*.
Cambio Registrant: selezionare *Cambio Registrant*, in seguito compilare il form con gli appositi dati: user id del nuovo e impostare una nuova password.
Atri tipi di operazioni sono il passaggio da un Registrar ad un altro e contestualmente il cambio di registrant; queste operazioni mantengono in parte la vecchia metodologia asincrona, in quanto oltre

a compilare il form presente nell'apposita sezione (selezionare altro ed in seguito l'apposita richiesta) è necessario compilare un'apposita modulistica non trattata in questa sede.

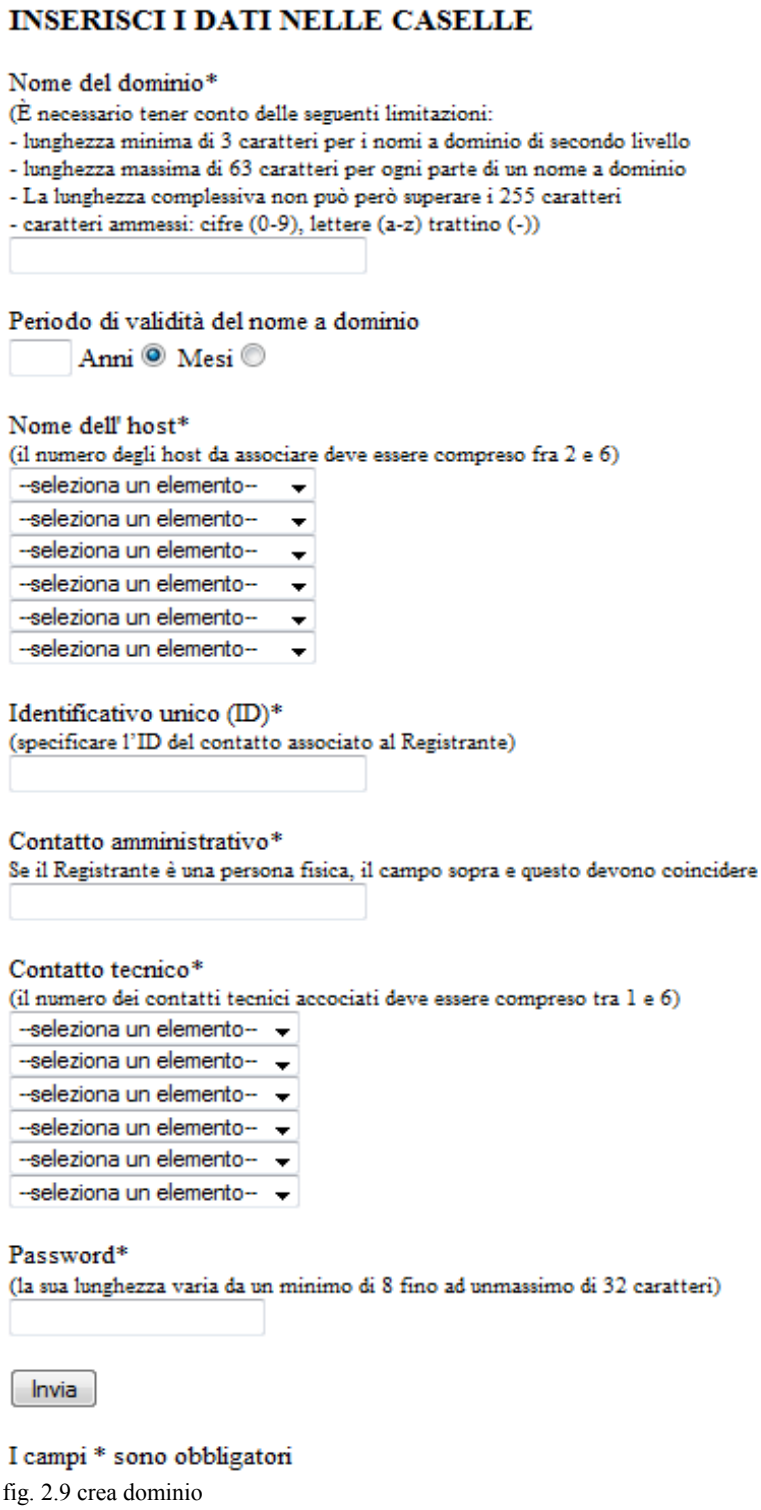

fig. 2.9 crea dominio

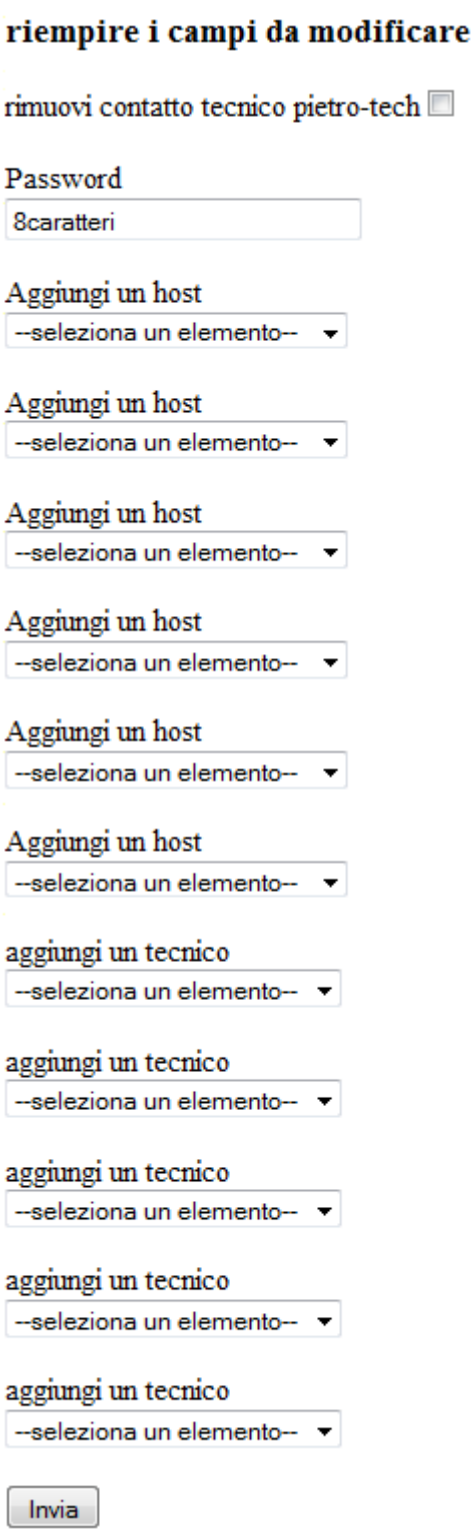

fig 2.10 modifica dominio

CAPITOLO 3: DESCRIZIONE

3.1 La struttura del progetto

Il programma, composto da oltre 95 file, è suddiviso logicamente in cinque livelli operativi come illustrato in fig. 3.1. Le classi che fanno parte dello stesso livello sono strutturate in modo simile e compiono le medesime operazioni.

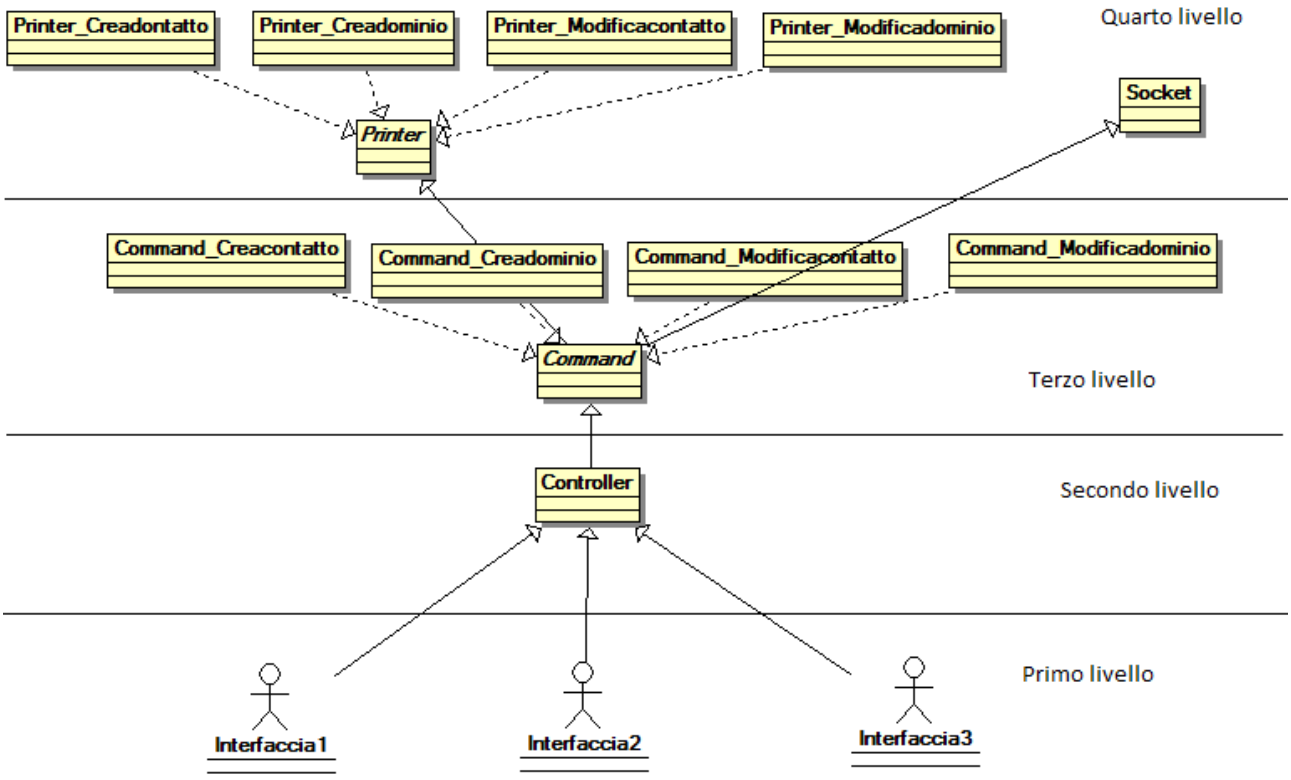

fig.3.1 Struttura del progetto

Il ciclo di vita dei dati inseriti da parte dell'utente parte dal primo livello, raggiunge il quarto e muore solo ad operazione conclusa quando, a seguito di un successo, viene aggiornato il database. I dati inseriti devono essere conformi alle politiche del Registro.it (lunghezza massima, caratteri ammessi, ...) quindi, per prima cosa, vengono controllati e, se qualche elemento non è corretto, viene segnalato nella schermata di input. Una volta superata la verifica, avviene la composizione del file XML, l'invio al server remoto e la ricezione della risposta. Questa stessa è interpretata e poi segnalata a video in una finestra di output.
Per la connessione remota e il collegamento al database è stata utilizzata una struttura di supporto: un framework, una serie di librerie e componenti sviluppati in php da Zend (società di software che lavora soprattutto nelle applicazioni web) che si possono scaricare gratuitamente dal sito www.zend.com.
Prima di iniziare la descrizione approfondita del progetto è necessaria una precisazione: la comunicazione con il server NIC avviene tramite richiesta "*post*" (come specificato dal Registro stesso); la struttura dei file che vengono scambiati è composta da diverse parti, gli "header" e il "body". I primi costituiscono l'intestazione del messaggio, i dati di controllo necessari al funzionamento della trasmissione; fanno parte degli header ad esempio i cookie, oppure lo status di connessione (un numero, la diagnosi della connessione, ad es. "200 connessione avvenuta correttamente", "404 il server remoto non risponde", ...).

Il “body” consiste nel messaggio vero e proprio inviato.
Dal momento che per qualsiasi tipo di richiesta si eseguono operazioni su contatti e domini, sono state create le classi “*Domrec_Datalogic_Contact*” e “*Domrec_Datalogic_Domain*”, le quali ne rappresentano gli oggetti stessi, con costruttore, variabili provate e metodi “*set*” e “*get*” per impostarne i valori.

3.2 Il primo livello

Il primo livello rappresenta quell'insieme di classi php e script html necessari per l'interazione con l'utenza. Le classi facenti parte di questa categoria si suddividono a loro volta in tre tipologie: classi di input, classi di output e classi di input-output.
Classi di input: fanno parte di questa categoria tutti gli script html creati per costituire le interfacce grafiche dedite all'inserimento dei dati da parte dell'utente attraverso finestre, menù a tendina, checkbox o bottoni. Per ogni script l'intero codice è collocato all'interno di un form che punta all'indirizzo “*include/starter.php*”, una struttura che raccoglie tutti i dati inseriti dall'utente, la quale compone un array con chiavi predefinite e lo invia tramite richiesta “*post*” all'url specificato.

“
```
<form method="post" action="../include/Starter.php">  *
```

Codice contenente la struttura di input:

```
<input type="submit" name="pulsante" value="Invia">
</form>
```
”

L'ultimo elemento prima della chiusura del tag “*form*” rappresenta il bottone “invia”, elemento grazie al quale i dati raccolti vengono inviati.
L'array comprende un elemento con chiave “command” che corrisponde alla richiesta da effettuare, un elemento con chiave “comefrom” che allo stato attuale ha attributo “web” e i vari elementi inseriti dall'utente, le cui chiavi dipendono da codice a codice.

COMMAND	COMEFROM	KEY_1	KEY_2	KEY_n
comando	web	ex_1	ex_2	ex_n

Gli elementi “comefrom” e “command” sono necessari per mantenere traccia della storia del vettore, la provenienza (in questo caso una pagina web) e l'arrivo o il comando con il quale è possibile scegliere il percorso da seguire.
Di seguito è illustrato un esempio di classe di primo livello: “*Newaccount.html*”.

“
```
<html>
<head>
<title>Creare un account</title>
</head>
<body>
<h3>INSERISCI I DATI NELLE CASELLE</h3>
<form method="post" action="../include/Starter.php">
     <input type="hidden" name = "comefrom" value="web" />
     <input type="hidden" name = "command" value="newaccount" />
```

```
Identificativo univoco (ID)*<br />
<small> (i caratteri ammessi sono lettere (a-z AZ) trattino
(-) e cifre (0-9))</small><br />
<input type="text" name="id" size="30"><big> <span
style="visibility:hidden; color:red">ID errato</span>
</big><br /><br />
Password <br />
<input type="text" name="authinfo" size="30"><big> <span
style="visibility:hidden; color:red"> password errata</span>
</big><br /><br />
Nome e cognome*<br />
<input type="text" name="name" size="30"><big> <span
style="visibility:hidden; color:red">nome errato</span>
</big><br /><br />
Organizzazione <br />
<input type="text" name="org" size="30"><big> <span
style="visibility:hidden; color:red">organizzazione
errata</span> </big><br /><br />
Via/piazza e numero civico*<br />
<input type="text" name="street1" size="30"><big> <span
style="visibility:hidden; color:red">via1 errato</span>
</big><br /><br />
Via/piazza e numero civico<br />
<input type="text" name="street2" size="30"><big> <span
style="visibility:hidden; color:red">via2 errato</span>
</big><br /><br />
Via/piazza e numero civico<br />
<input type="text" name="street3" size="30"><big> <span
style="visibility:hidden; color:red">via3 errato</span>
</big><br /><br />
Città*<br />
<input type="text" name="city" size="30"><big> <span
style="visibility:hidden; color:red">città errata</span>
</big><br /><br />
Provincia/nome dello stato estero*<br />
<small>(Se Nazione=IT, la Provincia deve contenere la sigla di
due lettere corrispondente ad una provincia italiana)
</small><br />
<input type="text" name="sp" size="30"><big> <span
style="visibility:hidden; color:red">provincia errata</span>
</big><br /><br />
Cap*<br />
<input type="text" name="pc" size="30"><big> <span
style="visibility:hidden; color:red">cap errato</span>
</big><br /><br />
Nazione*<br />
<small>(deve contenere la sigla a 2 caratteri maiuscoli della
nazione)</small><br />
<input type="text" name="cc" size="30"><big> <span
style="visibility:hidden; color:red">nazione errata</span>
</big><br /><br />
```

```
Telefono*<br />
<small>(deve essere indicato nel formato internazionale ISO,
ex: +39.042373988)</small><br />
<input type="text" name="voice" size="30"><big> <span
style="visibility:hidden; color:red">telefono errato</span>
</big><br /><br />
Telefono interno<br />
<small>(deve essere inserito un valore numerico di massimo 10
cifre)</small><br />
<input type="text" name="voiceint" size="30"><big> <span
style="visibility:hidden; color:red"> telefono interno
errato</span> </big><br /><br />
Fax <br />
<small>(deve essere indicato nel formato internazionale ISO,
ex: +39.042373988)</small><br />
<input type="text" name="fax" size="30"><big> <span
style="visibility:hidden; color:red">fax errato</span>
</big><br /><br />
Fax interno <br />
<small>(deve essere inserito un valore numerico di massimo 10
cifre)</small><br />
<input type="text" name="faxint" size="30"><big> <span
style="visibility:hidden; color:red"> fax interno
errato</span> </big><br /><br />
Email*<br />
<small>(deve essere indicato nel formato RFC2822 e succesivi
ex: user@domain.it)</small><br />
<input type="text" name="email" size="30"><big> <span
style="visibility:hidden; color:red">email errata</span>
</big><br /><br />
Do il mio conenso al trattamento dei dati personali*<br />
Si<input type="radio" name="authpub" value="true" checked>
No<input type="radio" name="authpub" value="false"><br
/><br />
Tipologia utente*<br />
<select name = "tipo">
<option value="0">--seleziona un elemento--</option>
<option value="1">Registrant</option>
<option value="2">Registrar</option>
<option value="3">tecnico</option>
</select><br /><br />
Questa parte è da compilare solo se la tipologia utente
corrisponde a Registrant<br />
Nazionalità<br />
<small>(deve coincidere con il campo Nazione se la tipologia
se la tipologia non coincide con persone
fisiche)</small><br />
<input type="text" name="nationalitycode" size="30"><big>
<span style="visibility:hidden; color:red">nazionalità
errata</span> </big><br /><br />
Tipologia<br /><select name="entitytype">
```

```
    <option value="0">--seleziona un elemento--</option>
    <option value="1">Persone fisiche italiane e
    straniere</option>
    <option value="2">Società / imprese individuali</option>
    <option value="3">Liberi professionisti / ordini
    professionali</option>
    <option value="4">enti no-profit</option>
    <option value="5">enti pubblici</option>
    <option value="6">altri soggetti</option>
    <option value="7">soggetti stranieri equiparati ai precedenti
    escluso le persone fisiche</option>
    </select><br /><br />
    Codice fiscale/numero documento identità/partita iva<br />
    <input type="text" name="regcode" size="30"><big> <span
    style="visibility:hidden; color:red">codice errato</span>
    </big><br /><br />
    <input type="submit" name="pulsante" value="Invia"><br
    /><br />
    <small>I campi * sono obbligatori</small>
    <br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
</form>
</body>
</html>
"
```

Tutti gli elementi di input sono situati all'interno del "*form*". In tale modo risulta impostata una pagina per la creazione di un nuovo account, infatti alla chiave "command" corrisponde l'attributo "newaccount".
Le chiavi dell'array corrispondono ai tag XML del protocollo EPP per la creazione di un nuovo account e solamente una classe appositamente creata per questo comando è in grado di riconoscere le chiavi ed elaborarne i contenuti.

NOME	CHIAVE
Provenienza	comefrom
Comando	command
ID	id
Password	authinfo
Nome	name
Organizzazione	org
Via 1	street1
Via 2	street2
Via 3	street3
Città	city
Provincia	sp
CAP	pc
Nazione	cc

Telefono	voice
Telefono interno	voiceint
Fax	fax
Fax interno	faxint
Email	email
ConsentForPublishing	authpub
Tipo	tipo
Nazionalità	nationalitycode
Entitytype	entitytype
RegCode	regcode

A comando diverso corrispondono chiavi dell'array diverse, fatta eccezione per le prime due.
Oltre a *"Newaccount.html"* fanno parte di questa categoria lo script per la login, la modifica dell'account, la creazione di un nuovo dominio, ... tutte classi molto simili (di cui qui non è riportato il codice).
Classi di output: ne fa parte solo "*Domrec_View_Browser*". Questa è scritta interamente in codice php e implementa due metodi: *"setResponse"* e *"getError"*. Entrambi non restituiscono niente ma generano una stampa a video: "operazione andata a buon fine", oppure "Errore!" con link al logout e al menù principale (per questo motivo"*Domrec_View_Browser*" fa parte della categoria "Classi di output"). Il metodo "*getError*" in ingresso necessita di due parametri: il numero dell'errore e la rispettiva descrizione. Riportiamo qui di seguito il codice della classe "*Domrec_View_Browser*".

"

```
<?php
class Domrec_View_Browser{

     public function setResponse(){

          echo "<html>
               <head>
               <title></title>
               </head>
               <body>
               <a2>Operazione andata a buon fine</a2>
               <a  href=\"http://localhost:2280/Index.html\">Torna
               al menù principale</a>
               <form method=\"post\"
               action=\"../include/Starter.php\">
               <input type=\"hidden\" name = \"command\"
               value=\"logout\" />
               <input type=\"submit\" style= \"width:300px;
               height:60px; color:#0000ff\" value=\"LOGOUT\">
               </form>
               </body>
               </html>";
     }
     public function getError($codice,$errore){
```

```
        echo "<html>
                <head>
                <title>Errore</title>
                </head>
                <body>
                <h1>$codice</h1>
                $errore[$codice];<br />
                <a
    href=\"http://localhost:2280/Index.html\">Torna   al   menù
    principale</a>
                <form method=\"post\"
    action=\"../include/Starter.php\">
                <input type=\"hidden\" name = \"command\"
        value=\"logout\" />
                <input  type=\"submit\"  style=  \"width:300px;
        height:60px; color:#0000ff\" value=\"LOGOUT\">
                </form>
                </body>
                </html>";
    }
}
?>
"
```

Classi di input-output: fanno parte di questa categoria le classi "*Gestionecontatti*" e "*Gestionedomini*", scritte in codice misto php e html. Questi due script costituiscono le pagine di output a cui si accede tramite la selezione all'interno del menù principale di "Gestione Account" oppure di "Gestione Domini" e che presentano le tabelle dei contatti e dei domini registrati.
Le due classi sono molto simili, ad eccezione di qualche elemento aggiuntivo presente in "*Gesionedomini*", ovverosia il tasto "Cambio Registrant" e una seconda tabella contenente i domini eliminati.
Come esempio viene proposto lo script della classe "*Gestionedomini*".

"

```
<?php
require_once 'Zend/Db/Adapter/Pdo/Pgsql.php';

try{
$db = new Zend_Db_Adapter_Pdo_Pgsql(array(   'host' => 'localhost',
                                             'username' => 'edi',
                                             'password' => 'edi',
                                             'dbname' =>
                                             'domrec'));
}
catch (Exception $e){
        echo "impossibile connettersi al database";
        die();
}

$datinotnull   =   $db->fetchAll('select   *   from   domain   where
data_cancellazione is null');
```

```
$datinull = $db->fetchAll('select * from domain where
data_cancellazione is not null');

?>

<html>
<head>
<title>Domini</title>
</head>
<body>
<h2>Di segito viene riportata la tabella contenente tutti i domini
contenuti nel database</h2>

<br/> <br/> <br/> <br/> <br/>
<table width="70%">

<tr>

     <td><h3>Nome dominio</h3></td>
     <td><h3>Nome </h3></td>
     <td><h3>data creazione</h3></td>
     <td><h3>status</h3></td>

<?php

for ($i=0; $i<count($datinotnull); $i++){
     $id = $datinotnull[$i]['id_registrante'];
                         $nome = $datinotnull[$i]['nome'];
                         $data = $datinotnull[$i]
     ['data_creazione'];
                         $status = $datinotnull[$i]['status'];

          echo"<tr>
               <td>$nome</td>
               <td>$id</td>
               <td>$data</td>
               <td>$status</td>
               <td><form method=\"post\"
     action=\"include/Starter.php\">
               <input type=\"hidden\" name = \"comefrom\"
     value=\"web\" />
               <input type=\"hidden\" name = \"command\"
     value=\"requiremodificadomain\" />
               <input type=\"hidden\" name = \"name\"
     value=\"$nome\" />
               <input type=\"submit\" value=\"EDIT\">
               </form>
               <td><form method=\"post\" action=\"Sicurezza.php\">
               <input type=\"hidden\" name = \"name\"
     value=\"$nome\" />
               <input type=\"hidden\" name = \"command\"
```

```
     value=\"deletedomain\" />
               <input type=\"submit\" value=\"DELETE\">
               </form>
               <td><form              method=\"post\"
     action=\"include/Starter.php\">
               <input   type=\"hidden\"   name   =   \"comefrom\"
     value=\"web\" />
               <input   type=\"hidden\"   name   =   \"command\"
     value=\"requiremodificaregistrant\" />
               <input    type=\"hidden\"    name    =    \"name\"
     value=\"$nome\" />
               <input         type=\"submit\"         value=\"CAMBIO
REGISTRANT\" />
               </tr>
               </form>";
}
?>
               <td><h3>Nome dominio</h3></td>
               <td><h3>Nome </h3></td>
               <td><h3>data eliminazione</h3></td>

<?php

for ($i=0; $i<count($datinull); $i++){

     $id = $datinull[$i]['id_registrante'];
                         $nome = $datinull[$i]['nome'];
                         $data            =            $datinull[$i]
['data_cancellazione'];


          echo"<tr>
               <td>$nome</td>
               <td>$id</td>
               <td>$data</td>
               <td><form              method=\"post\"
     action=\"include/Starter.php\">
               <input   type=\"hidden\"   name   =   \"comefrom\"
     value=\"web\" />
               <input   type=\"hidden\"   name   =   \"command\"
     value=\"ripristinodominio\" />
               <input    type=\"hidden\"    name    =    \"name\"
     value=\"$nome\" />
               <input type=\"submit\" value=\"RIPRISTINA\">
               </form>";
}
?>

</table>
<br/><br/><br/><br/><br/>
<form method="post" action="include/Starter.php">
```

```
<input type="hidden" name = "comefrom" value="web" />
<input type="hidden" name = "command" value="requiredomain" />
<input type="submit" value="CREA NUOVO">
</form>
</body>
</html>
```
”

Questo codice è suddiviso in varie parti. Nella prima si hanno il collegamento al database e il recupero dei domini eliminati e non, tramite la query diretta di “select” (i domini non cancellati vengono riconosciuti tramite l'assenza della data di eliminazione): vengono istanziati due vettori contenenti ciascuno i domini eliminati e quelli non eliminati. Nella seconda parte si hanno in sequenza un codice html, che imposta la prima tabella (tramite il costrutto html per la generazione di tabelle con i tag “<tr>” e “<td>”), e un blocco di codice php che, attraverso un “*ciclo for*”, la riempie. Analogamente, per i domini eliminati, è presente la sequenza codice html - codice php per istanziare ed infine riempire una seconda tabella. La terza parte è costituita dalle istruzioni per inserire il pulsante “crea nuovo”.

3.3 Il secondo livello

I dati inseriti dall'utente vengono inviati alla classe “*Starter*”, un elemento intermedio molto semplice che richiama due metodi: “*getRequestvars*”, della classe “*Tools*”, e “*run*”, della classe “*Domrec_Controller*”. La classe *“Starter”* è stata scritta prevedendo un'eventuale implementazione del software in cui, oltre al metodo “*post*”, vi siano dati inviati tramite un metodo “*get*”; in questo modo tutti gli array (provenienti da richieste “*get*” e “*post*”) vengono filtrati dalla classe e uniti in un unico vettore. Qui di seguito è riportato il codice della classe “*Starter*”:

“
```
<?php
require_once 'utils/Tools.php';
$a = Domrec_Utils_Tools::getRequestvars();
require_once ("Controller.php");
print "<br>";
DomRec_Controller::run($a);
?>
```
“

Il metodo “*getRequestvars*” è un metodo statico, senza parametri in ingresso che restituisce un array, raccoglie le richieste “*post*” e “*get*” tramite il costrutto “*if isset*” e costruisce un unico array tramite un ciclo *“foreach”,* mantenendo inalterate le chiavi. Segue il codice del metodo:

```
static function getRequestvars(){
     $retval = array();

     if(isset($_POST)){
               foreach ($_POST as $k=>$v){
                    $retval[$k]=$v;
               }
          }
          if(isset($_GET)){
               foreach ($_GET as $k=>$v){
                    $retval[$k]=$v;
```

```
            }
        }
        return $retval;
    }
```

Il metodo "*run*" è un altro metodo statico, che non restituisce niente, e lancia l'esecuzione dello script "*Domrec_controller*" inviandogli in ingresso il vettore contente i dati.
La classe sopra citata è l'unica appartenente al secondo livello, il cui unico metodo è *"run"*.
Il codice è composto da un costrutto di selezione "*switch-case"*, dove l'elemento analizzato è il contenuto del vettore alla chiave "command"; lo "*switch*"analizza il comando e sceglie il percorso da seguire. Ogni caso è costituito da un blocco di istruzioni che segue uno schema ben preciso, indipendentemente dal comando. Come esempio è riportata una parte del codice di "*Domrec_controller*".
Nell'esempio sono riportati i blocchi operativi dei casi "newaccount" e "newdomain".

"

```
class DomRec_Controller{
…
public static function run($arraydati){
…
switch ($arraydati["command"]){

            case "newaccount":

                require_once ("command/Creacontatto.php");

                    $viewrer = new Domrec_View_Browser();
                    $newcontact = new
                    Domrec_Command_Creacontatto();
                    $newcontact->execute($arraydati);
                    if ($newcontact->control()){
                        $newxml = $newcontact->getResult();
                        echo $newxml;
                        $a2x = new Domrec_Utils_ArrayToXML();
                        $newarray = $a2x->xml2ary($newxml);
                        $result = $newarray['epp']['_c']
                        ['response']['_c']['result']['_a']
                        ['code'];
                        if ($result > 1500){
                            $reason =  $newarray['epp']
                            ['_c']['response']['_c']
                            ['result']['_c']['extValue']
                            ['_c']['value']['_c']
                            ['reasonCode']['_v'];
                            $error =
                        Domrec_Utils_Tools::getErrorcode();
                        $viewrer->getError($reason,  $error);

                        }
                        else{
                            $newcontact->addDb($arraydati);
```

```
                    $viewrer->setResponse();

               }
               else {
                    $result = $newcontact->getResval();
                    $view = new
               Domrec_view_Error($result,"newcontact");
                    $view->view();}
          break;

          case "newdomain":

               require_once("command/Creadominio.php");

               $viewrer = new Domrec_View_Browser();
               $newdomain = new Domrec_Command_Creadominio();
               $newdomain->execute($arraydati);
               if ($newdomain->control()){
                         $newxml = $newdomain->getResult();
                         $a2x = new Domrec_Utils_ArrayToXML();
                         $newarray = $a2x->xml2ary($newxml);
                         $result = $newarray['epp']['_c']
                         ['response']['_c']['result']['_a']
                         ['code'];
                         if ($result > 1500){
                              $reason =  $newarray['epp']
                              ['_c']['response']['_c']
                              ['result']['_c']['extValue']
                              ['_c']['value']['_c']
                              ['reasonCode']['_v'];
                              $error =
                         Domrec_Utils_Tools::getErrorcode();
                         $viewrer->getError($reason, $error);
                         }
                         else {
                              $newdomain->addDb($arraydati);
                              $viewrer->setResponse();
                         }
               }
               else {
                    $result = $newdomain->getResval();

                    $view = new
               Domrec_view_Error($result,"newdomain");
                    $view->view();}
          break;
```

…
“

I due blocchi sono molto simili, infatti la sequenza di istruzioni è sempre la stessa, con differenze solo nelle classi che vengono richiamate. Inizialmente il vettore viene passato come parametro al

metodo “*execute*” di una delle classi della categoria “command”; questa, attraverso i metodi “*control*”, “*getResult*” e “*getResval*”, permette di verificare se i dati inseriti sono corretti o no e, nel primo caso, di recuperare la risposta del NIC, nel secondo di riottenere l'elenco dei valori errati.
La risposta del NIC è costituita da un file XML strutturato in base alle regole del protocollo EPP. Per riuscire a estrapolare con semplicità le informazioni utili, il file XML viene trasformato in un array tramite il metodo “*xml2ary*” della classe “*Domrec_Utils_ArrayToXML*”. Una volta mutato il file in un vettore, è possibile recuperare il risultato dell'operazione alla chiave “['epp']['_c']['response']['_c']['result']['_a']['code']”, e in caso d'errore, la “reason d'errore” alla chiave “['epp']['_c']['response']['_c']['result']['_c']['extValue']['_c']['value']['_c']['reasonCode']['_v']”.
Nella tabella che segue è riportata un esempio di risposta

CODICE >=	REASON
4000	Reason per errori generici
5000	Reason per errori riguardanti la sessione
6000	Reason per errori riguardanti l’accounting
7000	Reason per errori riguardanti la configurazione DNS proposta nei comandi Create Domain e Update Domain
8000	Reason per errori riguardanti l’oggetto contact
9000	Reason per errori riguardanti l’oggetto domain

Se il risultato è minore o uguale di 1500 non vi sono errori, i dati vengono inseriti nel database tramite il metodo “*addDb*”.

3.4 Il terzo livello

Il terzo livello è composto dalla classe astratta “*Domrec_Command*” e da tutte le classi che la implementano: “*Domrec_Command_Creacontatto*”, “*Domrec_Command_Creadominio*”, “*Domrec_Command_Modificacontatto*”, … Ognuna di esse è stata creata in funzione di uno specifico comando, ; la classe padre contiene il metodo “*validatepar*” e la firma del metodo “*execute*”, ogni classe figlio, inoltre, implementa il metodo “*getResult*”, “*getResval*”, “*control*” e “*addDb*”.
Il metodo “*execute*” riceve in ingresso i dati inseriti dall'utente e ne verifica la correttezza mediante il metodo della classe padre “*validatepar*”. Per ogni elemento scorretto, la chiave di questo viene accodata ad un vettore istanziato all'inizio. Se alla fine del blocco di controllo dati tutti i valori sono corretti, il metodo “*control*” restituisce una variabile booleana con valore “true”, altrimenti la stessa con valore “false”.
Il metodo “*getResval*” restituisce il vettore contenente i valori errati.
Come esempio è riportato il codice del metodo “*validatepar*”

“

```
public function validatepar($value,$mask,$l = null, $m = null){

          $regex = Domrec_Utils_Tools::getRegex();
          $reg  = $regex[$mask];
          if($l!=null && $m!=null) $reg = str_replace("rpl", "$l,
$m", $reg);
          if (preg_match($reg, $value)) $this->_validation = true;
          else $this->_validation = false;
```

```
            return $this->_validation;

        }
“
```

Attraverso il costrutto “*preg_match*” il metodo confronta l'elemento da verificare passato in ingresso con un'espressione regolare, una stringa contenente cifre, lettere e simboli.
Il costrutto restituisce “true” se il valore è corretto, diversamente “false”. L'insieme delle espressioni regolari è collocata in un vettore con chiavi predefinite all'interno della classe “*Tools*”, il cui metodo “*getRegex*” ne permette la consultazione; le chiavi dell'array sono predefinite:

```
“
id: chiave unica per validare un identificativo unico accettato
dal nic, caratteti dalla a alla z maiuscoli e minuscoli, cifre e
trattino, il carattere non deve iniziare per DUP o per trattino
all: tutti i caratteri sono ammessi
numeric: tutte le cifre da 0 a 9
upperalpha: tutti i caratteri dalla a alla z maiuscoli
loweralpha: tutti i caratteri dalla a alla z minuscoli
alpha: tutti i caratteri dalla a alla z maiuscoli e minuscoli
tel: formato internazionale ISO dei numeri telefonici
email: formato RFC2822 per email
alphanumeric: tutti i caratteri dalla a alla z maiuscoli e
minuscoli, cifre da 0 a 9
domain: chiave unica per validare i nomi a dominio .it ammessi dal
nic
ip: formato indirizzo ip
url: chiave unica per validare i nomi degli host
in tutti i casi dove presente il rimbolo rpl indica la parte da
sostituire nall'espressione
per la lunghezza della stringa

static private $_regex = array( "id" => "/^[^DUP][a-zA-Z0-9\-]
{rpl}$/",
                            "all" => "/^(\w|\W){rpl}$/",
                            "numeric" => "/^[0-9]{rpl}$/",
                            "upperalpha" => "/^[A-Z]{rpl}$/",
                            "loweralpha" => "/^[a-z]{rpl}$/",
                            "alpha" => "/^[A-Za-z]{rpl}$/",
                            "tel" => "/^\+[0-9]{2,2}\.[0-9]{8,10}$/",
                "email" => "/^([\w\-\+\.]+)@([\w\-\+\.]+).([\w\-\
+\.]+)$/",
                            "alphanumeric" => "/^[a-zA-Z0-9]{rpl}$/",
"domain" => "/^([a-z0-9][a-zA-Z0-9\-]{0,63}\.)?([a-z0-9\-]{3,63})?
(\.it)$/",
                "ip" => '/^[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]
{1,3}$/',
"url" => "/^([a-z0-9][a-zA-Z0-9\-]{0,63}\.)?([a-z0-9\-]{3,63}).
([a-z]{2,3})$/");
”
```

Alcune espressioni regolari definiscono anche la lunghezza minima e quella massima di caratteri, altre sono di lunghezza variabile; per queste ultime i limiti devono essere stabiliti all'ingresso del metodo "*validatepar*" unitamente all'elemento da confrontare e al codice di confronto.
Se tutti gli elementi sono corretti i dati vengono trasferiti al livello successivo per la composizione del file XML. Segue come esempio il codice della classe "*Domrec_Comand_Creacontatto*"

"

```
<?php

class Domrec_Command_Creacontatto extends Domrec_Command {

     ...

     public function execute(array $data){

          $newcontact = new Domrec_Datalogic_Contact();
          $socket = new Domrec_Connectors_Socket();

     if (!$this->validatepar($data['id'],"id",1,16))$this-
>_resval[] = 'ID';
     if (!$this->validatepar($data['name'],"all",1,255))$this-
>_resval[] = 'nome';
     if ($data['org'] != ""){
          if (!$this->validatepar($data['org'],"all",1,255))$this-
>_resval[] = 'organizzazione';}

     if (!$this->validatepar($data['street1'],"all",1,128))$this-
>_resval[] = 'via1';
     if ($data['street2'] != ""){
          if (!$this->validatepar($data['street2'],"all",1,128))
$this->_resval[] = 'via2';}
     if ($data['street3'] != ""){
          if (!$this->validatepar($data['street3'],"all",1,128))
$this->_resval[] = 'via3';}
     if (!$this->validatepar($data['city'],"alpha",1,128))$this-
>_resval[] = 'città';
     if (!$this->validatepar($data['sp'],"alpha",1,128))$this-
>_resval[] = 'provincia';
     if (!$this->validatepar($data['pc'],"numeric",5,5))$this-
>_resval[] = 'cap';
     if (!$this->validatepar($data['cc'],"upperalpha",2,2))$this-
>_resval[] = 'nazione';
     if (!$this->validatepar($data['voice'],"tel"))$this->_resval[]
= 'tel';
     if ($data['voiceint'] != ""){
          if (!$this-
>validatepar($data['voiceint'],"numeric",1,10))$this->_resval[] =
' telefono interno';}
     if ($data['fax'] != ""){
          if (!$this->validatepar($data['fax'],"tel"))$this-
>_resval[] = 'fax';}
```

```
     if ($data['faxint'] != ""){
     if (!$this->validatepar($data['faxint'],"numeric",1,10))$this-
>_resval[] = ' fax interno';}
     if (!$this->validatepar($data['email'],"email"))$this-
>_resval[] = 'email';
     if ($data['authinfo'] != ""){
          if (!$this->validatepar($data['authinfo'],"all",2,10))
$this->_resval[] = 'password';}
     if (($data['tipo']) == 1){
          if (!$this-
>validatepar($data['nationalitycode'],"upperalpha",2,2))$this-
>_resval[] = 'nazionalità';
          if (!$this->validatepar($data['regcode'],"all",1,36))
$this->_resval[] = 'codice';}
     if (count($this->_resval) ==0){

                $via[1] = $data['street1'];

                $via[2] = $data['street2'];

                $via[3] = $data['street3'];

                $newcontact->setId($data['id']);

                $newcontact->setNome($data['name']);

                $newcontact->setOrg($data['org']);

                $newcontact->setVia($via);
                $newcontact->setCity($data['city']);

                $newcontact->setProv($data['sp']);

                $newcontact->setCap($data['pc']);

                $newcontact->setNazione($data['cc']);
                $newcontact->setTel($data['voice']);

                $newcontact->setTelint($data['voiceint']);

                $newcontact->setFax($data['fax']);

                $newcontact->setFaxint($data['faxint']);

                $newcontact->setEmail($data['email']);

                $newcontact->setAuthinfo($data['authinfo']);

                $newcontact->setAuthpub($data['authpub']);

                $newcontact-
```

```
>setNationcode($data['nationalitycode']);
                $newcontact->setType($data['entitytype']);

                $newcontact->setRegcode($data['regcode']);


                $xmlfile =
Domrec_Printer::switchStrategy("newaccount", $newcontact);
                $socket->connect($xmlfile);
                $this->_result = $socket->getResult();

                }

     }

     public function getResult(){
          return $this->_result;
     }

     public function getResval(){
          return $this->_resval;
     }

     public function control(){
          if (count($this->_resval) ==0) return true;
          else return false;
     }
     public function addDb(array $data){
        if($data['authpub'] == true) $au=1;
        else $au=0;

        $dati = array(
                            'id' => $data['id'],
                                 'nome' => $data['name'],
                            'organizzazione' => $data['org'],
                                 'città' => $data['city'],
                            'provincia' => $data['sp'],
                            'cap' => $data['pc'],
                            'nazione' => $data['cc'],
                            'telefono' => $data['voice'],
                            'telefono_interno' => $data['voiceint'],
                            'fax' => $data['fax'],
                            'fax_interno' => $data['faxint'],
                            'email' => $data['email'],
                            'authinfo' => $data['authinfo'],
                            'consentforpublishing' => $au,
                            'tipo' => $data['tipo'],
                            'via1' => $data['street1'],
                            'via2' => $data['street2'],
                            'via3' => $data['street3'],
                            'data_creazione' => date('Y-m-d
```

```
H:i:s',time()),
                          'status' => "");
        try{
       $db = new Zend_Db_Adapter_Pdo_Pgsql(array(
                                'host' => 'localhost',
                                'username' => 'edi',
                                'password' => 'edi',
                                'dbname'   => 'domrec',
                                     'port' => '5432'));}
       catch (Exception $e){
           $this->_logger->errror('impossibile connettersi al
database');
           echo "impossibile connettersi al database";
           exit;
       }
       $tab = new User3($db);

       $tab->insert($dati);

      if ($data['tipo'] == 1){

                $datireg = array(
                          'id_registrant' => $data['id'],
                          'nazionalità' => $data['nationalitycode'],
                          'entitytype' => $data['entitytype'],
                          'regcode' => $data['regcode']);

           $tab = new User2($db);

           $tab->insert($datireg);
      }
      }
}
}
?>
"
```

I dati vengono utilizzati per creare un oggetto di tipo "*Domrec_Datalogic_Contact*"; quest'ultimo viene passato come parametro d'ingresso al metodo "*SwitchStrategy*" della classe"*Domrec_Printer.*", il quale restituisce il file XML pronto per l'invio sottoforma di stringa. Il file viene inviato tramite la classe "*Domrec_Socket*" al server NIC. Il metodo "*getResult*" restituisce la risposta.
Il metodo "*addDb*" aggiorna il database, in questo caso viene utilizzata la libreria "*Db_Adapter*" di Zend Framework per il collegamento, dopo aver impostato i vari parametri.
Il metodo presentato nell'esempio inserisce il nuovo oggetto contatto nel database i cui parametri di collegamento sono: host, username, password, nome, database e porta.

3.5 Il quarto livello

Fanno parte del quarto livello le classi dedicate alla composizione del file XML secondo i criteri del protocollo EPP. Come per le classi appartenenti al livello precedente vi è una classe astratta

“*Domrec_Printer*” con tutte le sue implementazioni; ognuna delle quali è concepita per il il proprio scopo. Poichè tutti i file XML inviati al server NIC per le diverse richieste contengono l'intestazione e la sezione conclusiva comune, parte del codice è presente solamente nella superclasse, e richiamata dalle sottoclassi.
La classe “*Domrec_Printer*.” contiene parte di codice salvato in variabili protette, un metodo statico “*switchStrategy*” per la scelta del “printer” da utilizzare e la firma del metodo “*printDatalogic*”.
L'esempio che segue rappresenta la classe “*Domrec_Printer_Printcontact*”, dove viene composto il corpo del file XML sotto forma di stringa, ed in seguito completato con introduzione e conclusione.

“

```
<?php

class Domrec_Printer_Printcontact extends Domrec_Printer{

      $id = $data->getId();
      $nome = $data->getNome();
      $org = $data->getOrg();
     $city = $data->getCity();
      $prov = $data->getProv();
      $cap = $data->getCap();
      $nazione = $data->getNazione();
      $tel = $data->getTel();
      $telint = $data->getTelint();
      $fax = $data->getFax();
      $faxint = $data->getFaxint();
      $email = $data->getEmail();
      $authinfo = $data->getAuthinfo();
      $authpub = $data->getAuthpub()    ;
      $nationcode = $data->getNationcode();
      $type = $data->getType();
      $regcode = $data->getRegcode();
      $via = $data->getVia();
      $via1 = $via[1];
      $via2 = $via[2];
      $via3 = $via[3];

     if ($org != "")$org = "<contact:org>$org</contact:org>";
      else $org = "";
      if ($telint != "")$tel = "<contact:voice
x=\"$telint\">$tel</contact:voice>";
      else $tel = "<contact:voice>$tel</contact:voice>";
      if ($fax != ""){
          if ($faxint != "")$fax = "<contact:fax
x=\"$faxint\">$fax</contact:fax>";
          else $fax = "<contact:fax>$fax</contact:fax>";}
      else $fax = "";
      if ($via2 != "")$via2 =
"<contact:street>$via2</contact:street>";
      else $via2 = "";
      if ($via3 != "")$via3 =
"<contact:street>$via3</contact:street>";
```

```
      else $via3 = "";
      if ($nationcode != "" && $regcode != "")$nationcode =
"<extcon:registrant>

     <extcon:nationalityCode>$nationcode</extcon:nationalityCode>

     <extcon:entityType>$type</extcon:entityType>

     <extcon:regCode>$regcode</extcon:regCode>

     </extcon:registrant>";
      else $nationcode = "";

          $this->corpo = "<create>
          <contact:create
          xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\"
          xsi:schemaLocation=\"urn:ietf:params:xml:ns:contact-1.0
          contact-1.0.xsd\">
          <contact:id>$id</contact:id>
          <contact:postalInfo type=\"loc\">
          <contact:name>$nome</contact:name>
          $org
          <contact:addr>
          <contact:street>$via1</contact:street>
          $via2
          $via3
          <contact:city>$city</contact:city>
          <contact:sp>$prov</contact:sp>
          <contact:pc>$cap</contact:pc>
          <contact:cc>$nazione</contact:cc>
          </contact:addr>
          </contact:postalInfo>
          $tel
          $fax
          <contact:email>$email</contact:email>
          <contact:authInfo>
          <contact:pw>$authinfo</contact:pw>
          </contact:authInfo>
          </contact:create>
          </create>
          <extension>
          <extcon:create
          xmlns:extcon=\"http://www.nic.it/ITNIC-EPP/extcon-1.0\"
          xsi:schemaLocation=\"http://www.nic.it/ITNIC-EPP/extcon-
1.0
          extcon-1.0.xsd\">


<extcon:consentForPublishing>$authpub</extcon:consentForPublishing
>
          $nationcode
```

```
          </extcon:create>
          </extension>";


          $a = $this->intro. "
          " .$this->corpo. "
          " .$this->ending;

          return $a;

     }
}
?>
“
```

Fa sempre parte del quarto livello la classe “*Domrec_Socket*”. Questa classe gestisce la comunicazione con il server NIC attraverso il metodo “*connect*”, invia il file e riceve la risposta che viene restituita tramite il metodo “*getResult*”. La comunicazione avviene tenendo traccia della sessione di lavoro, quest'ultima composta da una stringa alfanumerica è collocata all'interno della risposta alla login, e durante ogni tipo di richiesta deve essere impostata in un cookie.
Per la connessione remota la classe è supportata dalla libreria “*Http_Client*” di Zend Framework.

```
“
<?php

require_once 'Zend/Http/Client.php';

class Domrec_Connectors_Socket{

     private $_response;

     public function connect($newxml){

          $client = new Zend_Http_Client('
          https://pub-test.nic.it',array('maxredirects' => 0,
                                         'timeout'      => 3));
          $session = file_get_contents("../session.txt");
          $client->setCookie("JSESSIONID",$session);
          $client->setRawData($newxml, "text/xml");
          $client->setMethod(Zend_http_client::POST);
          try{$response = $client->request();}
          catch (Exception $e){echo "<center><h5>Impossibile
connettersi</h5></center>"; die();}

          if ($response->getStatus() != 200){
               $this->_response = "Il server non risponde";
               echo "<center><h1>  impossibile connettersi al
server provare in un secondo momento </h1></center>";
               die();
          }
          else $this->_response = $response->getBody();
```

```
	}

	public function getResult(){

		return $this->_response;

	}
}
?>
“
```

La sessione di lavoro (a seguito della login) è salvata nel file “session.txt”e, ad ogni richiesta, viene recuperata. I metodi di Zend_Framework utilizzati sono: “*setCookie*” per impostare il cookie contenente la sessione di lavoro, “*setRowdata*” per impostare il file XML all'interno del “*body*”, “*setMethod*” per impostare la richiesta “post”, “*request*” per inviare e ricevere la risposta e “*getBody*” per recuperare il testo dal “body”.
Il metodo “*getResult*” restituisce la resposta del NIC.

3.6 Il database

Il database creato, di nome domrec, contiene sette tabelle: contact (contenente i contatti registrati), domain (contenente i domini registrati), registrant (contenente i dati riguardanti i contatti di tipo registrant), usertype (i cui valori sono 1,2,3 in corrispondenza di registrant, admin e tech), host (contenente i dati relativi agli attributi degli host), mux1 e mux2.
Di seguito è riportato il codice SQL della creazione del database:

```
“
create database domrec with template editemplate;

create table registrant(
id_registrar text not null,
nazionalità text not null,
entitytype smallint not null,
regcode text not null,
primary key (id_registrar));

create table usertype(
id smallint not null,
descrizione text not null,
primary key (id));

create table contact(
id text not null,
nome text not null,
organizzazione text not null,
via1 text not null,
via2 text not null,
via3 text not null,
città text not null,
provincia text not null,
cap integer not null,
```

```
nazione text not null,
telefono text not null,
telefono_interno integer not null,
fax text not null,
fax_interno integer not null,
email text not null,
authinfo text not null,
consentforpublishing boolean not null,
tipo smallint not null,
primary key (id),
foreign key (tipo) references usertype(id));

create table host(
id_host text not null,
ip_host text not null,
ip_type text not null,
primary key (id_host));

create table domain(
nome text not null,
periodo smallint not null,
tempo text not null,
id_registrante text not null,
id_admin text not null,
authinfo text not null,
status text not null,
primary key (nome));

create table mux1(
nome text not null,
id text not null,
primary key (nome, id),
foreign key (nome) references domain(nome),
foreign key (id) references host (id_host));

create table mux2(
nome text not null,
id text not null,
primary key (nome, id),
foreign key (nome)references domain(nome),
foreign key (id) references contact(id));
”
```

La tabelle sono reciprocamente relazionate, vi sono due tipi di relazioni: la relazione *uno a molti* e la relazione *molti a molti*. La relazione *uno a molti* è instaurata tra le tabelle usertype (uno) e contact (molti). La colonna “tipo” di contact è referenziata con chiave esterna alla colonna “id” di usertype. Un valore della colonna “id” può corrispondere a diversi contatti e il valore del record di “tipo” non può non essere contenuto nella tabella usertype.
La relazione *molti a molti* è instaurata tra le tabelle domain, host e contact. Infatti ad un dominio possono essere associati più host, ma uno stesso host può essere associato a più domini, analogamente con i contatti di tipo tech.

Per questo motivo sono state create le tabelle mux1 e mux2, al fine di caratterizzarne il rapporto di reciprocità; esse infatti hanno la doppia chiave primaria e la doppia chiave esterna. Contengono l'accoppiamento tra host e dominio e tra contatto tecnico e dominio.

CAPITOLO 4: OSSERVAZIONI E CONCLUSIONI

4.1 Il metodo seguito durante il progetto

Il progetto è stato sviluppato in un arco temporale di circa 500 ore, nel corso di un tirocinio presso l'azienda Edistar (dal 07/07/2010 al 07/10/2010). La prima fase del lavoro, durata circa un terzo delle ore, è stata di studio e preparazione, mentre i restanti due terzi sono stati dedicati alla stesura del codice vero e proprio. Durante la prima fase di studio l'attenzione è stata rivolta soprattutto all'analisi del protocollo EPP, che il software ha dovuto innanzitutto implementare e successivamente alla realizzazione della struttura (a livelli) del progetto, a cui è stato dedicato una buona parte del tempo di studio. Come primo approccio è stato necessario consolidare alcuni prerequisiti:

2. Conoscenza elementare della programmazione ad oggetti
3. Conoscenza del linguaggio di programmazione php
4. Concetti fondamentali di reti e protocolli
5. Conoscenza del linguaggio html
6. Conoscenza di linguaggio SQL

Per quanto riguarda la programmazione ad oggetti, i due corsi di java sostenuti sono stati esaustivi, mentre il php, l'SQL e l'html sono stati oggetto di studio nel primo periodo al fine di consolidarne la conoscenza. Gli elementi fondamentali di reti erano già patrimonio di cultura personale. Il php è un linguaggio di scripting "object oriented" nato per la programmazione web, ma utilizzato principalmente nelle applicazioni web lato server. È molto simile a Java, per questo motivo è stato semplice da apprendere; le differenze fondamentali tra i due linguaggi consistono nel fatto che, mentre Java deve essere compilato prima che ne sia lanciata l'esecuzione, il php è un linguaggio interpretato e gli eventuali errori di compilazione sono evidenziati al momento dell'esecuzione. Nel php, a differenza che in Java, le variabili non devono essere dichiarate e gli array sono molto più semplici da gestire; tutte le variabili sono di tipo "*string*" e per questo, quando vengono utilizzate, non deve esserne specificato il tipo. La stessa cosa accade per gli array; invece nelle operazioni aritmetiche è l'interprete php ad eseguire la trasformazione. Trattandosi di un linguaggio di scripting è inoltre possibile avviare programmi senza dover lanciare l'esecuzione di un metodo, in java al contrario è necessario eseguire il metodo statico "*main*". Per quanto riguarda il linguaggio html è stato necessario apprenderne gli elementi fondamentali per la creazione di pagine web elementari; oltre ai semplici "link", una struttura molto utilizzata è stata il "form", con tutti gli elementi di input, le variabili e i loro attributi. Come per l'html anche l'SQL è stato oggetto di studio; oltre alla creazione di tabelle sono state studiate le "query" per l'inserimento di dati e la consultazione delle tabelle stesse. L'azienda ha messo a disposizione degli strumenti che si sono rivelati necessari per lo sviluppo del progetto; essendo il software un'applicazione web, è stato necessario simulare l'interlocuzione con il server, oltre ad avere un ambiente di sviluppo ed un interprete php. La simulazione è stata realizzata grazie ad una "VirtualBox". Quest'ultima (Oracle VM VirtualBox versione 3.2.6) consiste in un software per l'emulazione di ambienti, una macchina virtuale che ospita un sistema operativo indipendente da quello utilizzato. In questo caso è stato installato "SUSE linux" come sistema operativo emulato, la versione server fig. 4.2. Per sviluppare php è stato utilizzato l'interprete "Apache" (un software installato nella macchina virtuale in grado di compilare ed eseguire script php) mentre come ambiente di sviluppo è stato impiegato eclipse, versione Helios. L'eclipse è una multipiattaforma sviluppata dall'Eclipse Foundation, ideata per la programmazione in php, html, xml, java e C++.

fig. 4.1 logo php

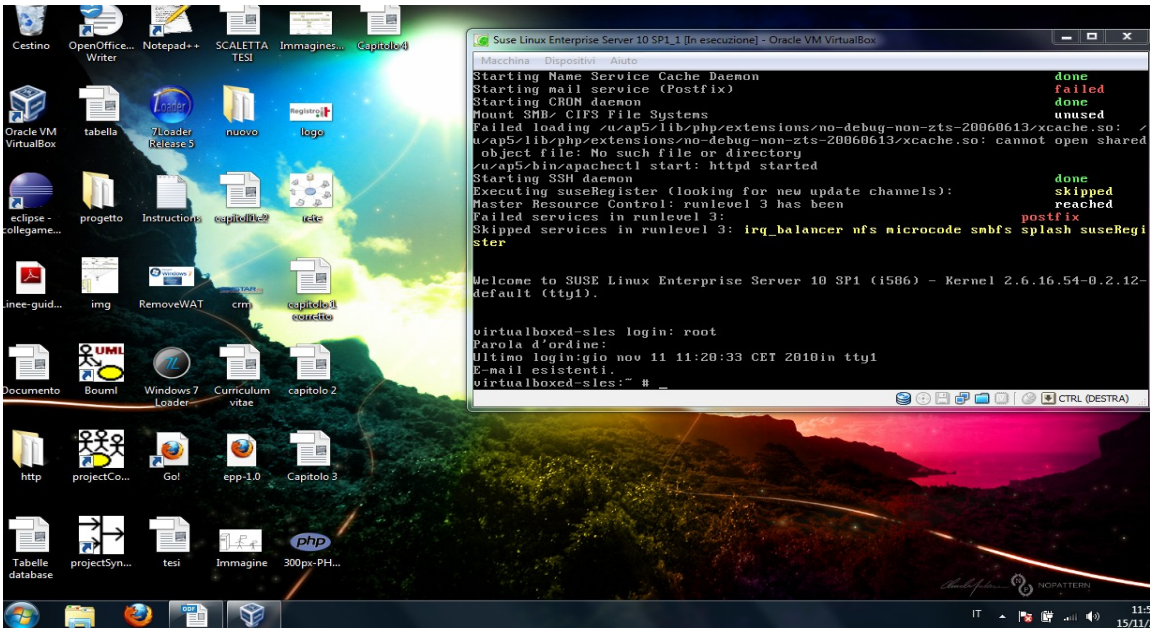

fig 4.2 immagine dei windows 7 con la VirtualBox in funzione


fig 4.3 logo di eclipse

Gli script php sono stati salvati in una cartella condivisa tra il sistema operativo Windows e quello emulato, in questo modo è stato possibile lavorare sul sistema operativo conosciuto. Per eseguire le classi è bastato collegarsi al virtual server, tramite browser, all'indirizzo "http://localhost:2280/nomefile" sulla falsariga del collegamento ad un sito internet. Una volta completato il progetto, il NIC ha fornito un account di test per consentire di effettuare delle prove. Gli account e i domini registrati sono stati effettivamente creati su un server di test all'indirizzo "test.epp.it", ma non sono stati fatturati. In questo modo è stato possibile effettuare le varie operazioni come da regolamento, rispettando le modalità e le tempistiche. Solo dopo aver superato tutti i test il software è stato installato su di un server dell'azienda.

4.2 Difficoltà incontrate

A conclusione dell'esperienza, la difficoltà più grande è stato l'impatto con un progetto di grandi dimensioni non finalizzato ad uso personale. Infatti il software è stato progettato per funzionare sui server di un'azienda, in previsione che chiunque, anche non conoscendolo bene, lo possa utilizzare. Per questo motivo il programma doveva rispondere in maniera positiva e comprensibile a qualsiasi input, anche se logicamente errata. Di conseguenza tutte le eccezioni dovevano essere controllate, con il costrutto "*try-catch*". C'è inoltre da dire che, affrontare un progetto con degli strumenti nuovi mai utilizzati prima, ha necessitato di un periodo di "rodaggio". Un ruolo importante è stato ricoperto dal tutor Alessandro Giacomella, il quale ha via via seguito il sottoscritto nello sviluppo ed è stato sempre disponibile in chiarimenti anche riguardanti il php stesso.

4.3 Conclusioni

L'esperienza è stata sicuramente positiva, l'adesione all'intero progetto e ad ogni sua parte è stata di pieno entusiasmo. Motivante è stata la possibilità di lavorare con un'altra persona la cui esperienza professionale è stata di grande aiuto. Sono state acquisite molte abilità nel campo informatico oltre che essere stata effettuata una prima esperienza lavorativa. Le abilità acquisite spaziano dai linguaggi veri e propri appresi, ai pattern di progettazione informatica, tra cui il concetto di

organizzare il lavoro schematicamente, prima di iniziare la stesura del codice.

BIBLIOGRAFIA

Siti internet consultati:

- http://www.nic.it/conosciamoci
- http://www.edistar.com/it/pg_gen_company_profile.shtml
- http://it.wikipedia.org/wiki/Domain_Name_System
- http://it.wikipedia.org/wiki/Internet
- http://it.wikipedia.org/wiki/Rete_di_telecomunicazioni
- http://it.wikipedia.org/wiki/Dominio_di_primo_livello

Manuali consultati

- Gestione delle operazioni sincrone sui nomi a dominio nel ccTLD.it Linee Guida versione 1.2

- A. S. TANENBAUM, Reti di calcolatori, Bruno Mondadori

- V. VASWANI, PHP: a beginner's guide, McGraw-Hill