ITeR@NET: una piattaforma innovativa per l'istruttoria in Tempo Reale On-Line

Antonio Carosi, Andrea Cilia
CILEA, Roma

Abstract
ITeR@NET è una piattaforma di servizi informatici destinata alla gestione e al monitoraggio delle domande di finanziamento, dalla pubblicazione del bando all'estinzione della pratica. Tale piattaforma si propone come uno strumento altamente personalizzabile, configurabile ed estendibile secondo esigenze applicative specifiche ed è realizzato adottando consolidate pratiche di progettazione e sviluppo.

ITeR@NET is a platform of computer services destined to the management and the monitoring of requests for funding, from the publication of the Program to the end of the project. Such platform is proposed as a customizable, configurable and extensible tool according to demands application specifications and it's realized adopting most consolidate practices of design and development.

Keywords: Istruttoria, Pattern, Nhibernate, Pubblica amministrazione, configurazione.

Introduzione

ITeR@NET (Istruttoria in Tempo Reale On Line) si presenta come una piattaforma di servizi informatici che permette la gestione automatica e il monitoraggio dell'intero iter dei processi di selezione di domande, dalla predisposizione del bando alla valutazione dei risultati al termine della pratica.

In base all'esperienza acquisita nella gestione delle domande di finanziamento tramite la piattaforma SIRIO, nello sviluppo di ITeR@NET si è tenuto conto dei seguenti fatti:

- l'iter seguito da una domanda spesso non è lineare, ma soggetto a eccezioni e ricircoli dovuti a richieste di integrazioni da parte dei valutatori o a richieste di supplementi istruttori da parte dell'Amministrazione;
- gli attori che ruotano attorno al sistema sono diversi, ognuno con un ruolo particolare che ne identifica le autorizzazioni a registrare o visionare dati;
- l'iter per cui passa una domanda può durare anche molto tempo prima di terminare: in questo lasso di tempo, il contesto reale può cambiare sostanzialmente, per esempio alcuni proponenti di una domanda possono rinunciare al progetto o essere estromessi dall'Amministrazione, oppure può verificarsi un cambiamento nella composizione di un comitato di valutatori;
- ogni attività finanziabile presenta le proprie peculiarità, esigenze e un proprio iter istruttorio.

Da quanto detto, per poter gestire un'istruttoria è necessario che la piattaforma sia la meno rigida possibile e metta in grado l'Amministrazione di poter facilmente gestire la totalità dei casi che si possono presentare.

Elenchiamo le principali caratteristiche di ITeR@NET:

- elevato grado di configurabilità e personalizzazione, anche dopo il rilascio del sistema: i moduli del sistema, a qualsiasi livello, basano le loro funzionalità principali secondo i dati specificati in un file di configurazione XML. È possibile modificare tali dati e permettere un diverso settaggio del comportamento del sistema agendo unicamente su tale file, evitando quindi di dover ricompilare il codice e ripubblicarlo in produzione;
- l'utilizzo di un file di configurazione in cui inserire i parametri critici, su cui si basa il sistema, permette anche una sua rapida estensibilità: le classi che implementano le funzionalità del sistema (come dettaglieremo nelle note tecniche) vengono istanziate dinamicamente durante l'esecuzione, secondo le

specifiche contenute nel file di configurazione. È quindi possibile implementare nuove funzionalità in una nuova classe e richiamarla nell'opportuna sezione del file di configurazione;
- oltre che per determinare il comportamento del sistema, il file di configurazione XML viene anche utilizzato come *repository* della specifica del flusso istruttorio utilizzata per rendere automatico il passaggio delle domande da uno stato a quello successivo: anche in questo caso, le modifiche al suo contenuto si rispecchiano in tempo reale al *runtime* dell'applicazione. Questo rende possibile, anche a un non addetto ai lavori, di poter modificare eventualmente l'iter istruttorio per un'attività senza la necessità di rivolgersi agli sviluppatori;
- elevata modularità delle funzionalità del sistema: è possibile assemblare i moduli funzionali di cui è costituito il sistema, in modo da venire incontro alle specifiche esigenze di un'Amministrazione;
- la suddivisione del sistema nei livelli di interfaccia utente, *business logic* e accesso ai dati, permette di avere un sistema altamente scalabile, che meglio si adatta ai volumi di utilizzo di una qualsiasi Amministrazione.

Architettura e note tecniche

La piattaforma è stata concepita per soddisfare esigenze specifiche.
- **Scalabilità**
  Ogni tipologia di attività finanziabile può avere un bacino diverso nel numero di utilizzatori, così come una diversa quantità e strutturazione di informazioni.
- **Modularizzazione**
  Come detto in precedenza, ogni attività ha un suo specifico iter istruttorio. Modularizzare e parametrizzare le diverse fasi dell'iter permette di gestire e manutenere più facilmente le diverse attività gestite.
- **Riusabilità**
  La corretta modularizzazione e parametrizzazione del sistema invoglia lo sviluppatore a riutilizzare codice già creato e testato, abbattendo i tempi di sviluppo e facilitando un *deployment* di servizi sicuri e affidabili.
- **Configurabilità ed estendibilità**
  ITeR@NET è un prodotto di cui si conosce la tipologia di fruitore (Amministrazioni più o meno grandi e complesse), ma di cui non si conoscono le specifiche esigenze nel dettaglio. È necessario quindi che l'architettura sviluppata agevoli la personalizzazione delle funzionalità in base alle specifiche esigenze.
- **Sicurezza**
  Il sistema applica vincoli di sicurezza e di privacy sui dati e sulle informazioni di carattere riservato tramite l'utilizzo di certificati, connessioni sicure, firma digitale e crittografia. Ogni pagina contenente dati critici sarà accessibile solamente previa autenticazione, fornendo *username* e *password*, tramite il servizio di *Single Sign On*.
- **Usabilità e accessibilità**
  Rispetto delle normative vigenti riguardo i principi di accessibilità e usabilità dei siti Web.
- **Integrabilità**
  Il sistema eventualmente deve poter interfacciarsi con sistemi esterni (per esempio un sistema di gestione documentale).

Il sistema si basa sul .NET Framework 2.0, in particolare è stato sviluppato in C# secondo le più note *best practices* riguardanti lo sviluppo di applicazioni *n-tier*. Per venire incontro alle esigenze ricordate sopra, ad alto livello la piattaforma si presenta costituita da 2 strati funzionali:
- **Sottosistema delle funzioni**
  Riguarda i moduli che sono a più stretto contatto con la funzionalità specifica del sistema, cioè gestire l'iter istruttorio delle domande. Si è scelto di creare un modulo che gestisca uno specifico stato in cui può trovarsi una domanda.
- **Sottosistema dei servizi**
  Riguarda i moduli che hanno funzionalità indipendenti dalla gestione dell'Istruttoria, che sono utilizzati trasversalmente nel sistema e che possono interfacciarsi con sistemi esterni. Esempi di tali moduli sono il modulo di gestione del flusso di lavoro o il gestore delle notifiche.

Ogni modulo funzionale, a sua volta, ha un'architettura divisa logicamente nei livelli descritti di seguito.

User Interface Layer (UIL)

Il livello di presentazione costituisce l'interfaccia del sistema con l'utente finale. Questo livello non ha nessuna conoscenza dei dettagli della logica applicativa o dell'accesso ai dati; deve occuparsi soltanto di acquisire e visualizzare i dati ricevuti dal livello sottostante (BLL), configurare l'aspetto visuale dei controlli utilizzati, accettare e convalidare l'input dell'utente,

richiamare le funzionalità del BLL in base al verificarsi di determinati eventi.

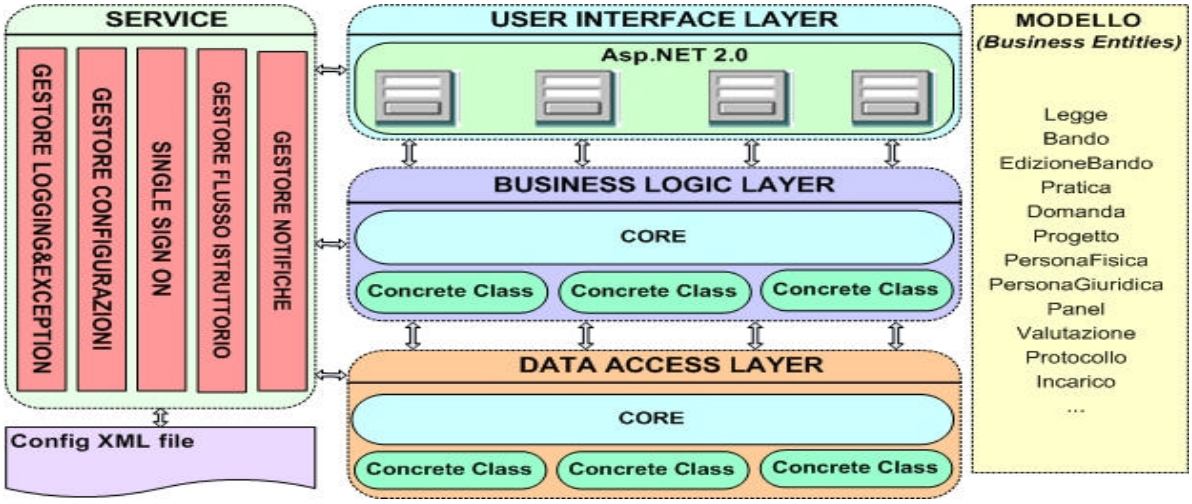

Fig. 1 - Architettura di un modulo funzionale

Pensando principalmente ad un'implementazione web e considerando che la piattaforma di riferimento è Microsoft .NET, si punta ad un'implementazione del livello di presentazione per mezzo dell'infrastruttura ASP.NET relativa alla versione 2.0 del framework della casa di Redmond. Il meccanismo di code-behind delle pagine ASP.NET permette un'ulteriore separazione in questo livello tra la parte deputata al layout (la pagina *.aspx*) e la parte di codice (il corrispondente file *.aspx..cs*) che specifica le funzionalità fornite.

Dalle caratteristiche sopra citate, è possibile modificare in qualunque momento il look delle pagine agendo solamente sugli elementi di questo livello, senza che questo influisca sugli altri livelli del sistema. Oltre che del BLL, di cui richiama la funzionalità, l'UIL conosce anche gli elementi del Modello (di cui si parlerà dopo), dei quali visualizza le proprietà o demanda agli strati sottostanti il loro settaggio.

Business Logic Layer (BLL)

Nel livello deputato alla gestione della Business Logic risiede la logica dell'applicazione: esso contiene le regole aziendali, quelle per la conversione dei dati e le chiamate alle funzioni di persistenza, (Data Access Layer DAL).

Questo livello si compone di unità funzionali "atomiche", nel senso che ognuna di esse ricopre responsabilità applicative relative ad una singola attività dell'iter che si ritiene indipendente da un punto di vista logico. Sono definite tutte le funzionalità logiche per ogni modulo: tali funzionalità vengono attuate sulle entità del Modello. Mentre queste, come dettaglieremo di seguito, sono costituite per lo più da proprietà, la parte funzionale risiede nei moduli del BLL.

Quindi questo livello si interfaccia con il Modello, oltre che con il DAL a cui demanda le funzionalità di persistenza.

Le esigenze di estendibilità del software hanno motivato l'adozione di principi progettuali che garantiscano una certa elasticità rispetto alla possibilità di evoluzione e mutamento delle necessità applicative da soddisfare. In tal senso, ogni modulo del BLL è costituito da:

- un **Core**, che comprende un'interfaccia che espone le funzionalità del modulo, un Proxy che rappresenta la facciata del modulo verso l'interfaccia utente e una Factory che crea dinamicamente le istanze concrete delle classi in base alle informazioni presenti nel file XML di configurazione;
- **classi concrete** che forniscono la specifica implementazione della funzionalità.

Quindi il legame fra interfaccia e implementazione di un modulo si realizza attraverso opportune classi software che, alla richiesta di fornire un oggetto che esponga i metodi di una determinata interfaccia, esaminano le informazioni di configurazione in funzione delle informazioni in entrata (inerenti, per quanto sopra detto, le caratteristiche del bando di cui ci si occupa) e recuperano la descrizione dell'imple-

mentazione da restituire, provvedendo così alle operazioni di istanziazione. In tal modo, si rende il sistema pronto a configurare il suo comportamento in situazioni diverse: basta modificare poche righe del file di configurazione e creare, se non esistono, le classi concrete che implementano le nuove funzionalità richieste, senza dover modificare le classi già esistenti.

Data Access Layer (DAL)

Il DAL permette di separare la logica di accesso ai dati dalla logica di *business*. Inoltre permette l'astrazione dalla specifica fonte dati utilizzata per persistere i dati. Le comunicazione tra BLL e DAL avvengono tramite l'interscambio di istanze di classi *custom* le quali sono raggruppate in un loro proprio *assembly* condiviso sia dal BLL che dal DAL. Tale contenitore di classi *custom* è il Modello: esso contiene le classi che rappresentano il dominio di interesse dell'applicazione. Su tali entità, il DAL effettua le operazioni di persistenza (le tipiche operazioni CRUD[1]).

Il livello Data Access è stato realizzato in accordo a principi di progettazione analoghi a quelli che hanno guidato lo sviluppo del livello BLL. Si fa ancora ricorso alla separazione netta fra interfacce (il Core) e implementazioni (classi concrete) e alla definizione dei legami fra esse a *runtime* sulla base dell'analisi delle informazioni di configurazione. Modificando le informazioni di tale file nella sezione che si riferisce al DAL, è possibile modificare la fonte dati da utilizzare e le modalità di interazione con essa.

Diversamente dal BLL, costituito da moduli rappresentanti una singola attività dell'iter istruttorio, il DAL è costituito da moduli che racchiudono le funzionalità a livello di entità del modello di dominio. In tal modo è possibile raggruppare le funzionalità CRUD a livello di entità, agevolando il riuso e la manutenibilità del codice. Questo implica che l'implementazione di una singola funzionalità di un modulo nel BLL possa richiedere la collaborazione con più classi di accesso ai dati.

Dal punto di vista delle implementazioni fornite, ci si è concentrati sull'esplorazione della versione .NET di un popolare *framework Object/Relational* per la gestione della persistenza, ossia Nhibernate (versione 1.0.2). Tramite questo *framework* risulta possibile manipolare dati persistenti operando unicamente su oggetti di dominio, senza conoscere le caratteristiche realizzative del supporto di memorizzazione (DB relazionale o altro) e semplificando significativamente il codice scritto per accedere a tali dati, visto che lo strumento utilizzato carica su di sé la responsabilità di gestire l'*impedance mismatch* che determina il divario di rappresentazione delle informazioni fra modello a oggetti e modello relazionale. Si è voluto approfondire l'utilizzo del *framework* in esame nella convinzione che esso possa semplificare la scrittura del codice di accesso ai dati e permettere il persistere delle informazioni in modo coerente al modello a oggetti, allo stesso tempo astraendosi dalle caratteristiche di rappresentazione e manipolazione dei dati del supporto persistente.

Modello

Il modello raggruppa le entità del dominio di interesse ed è costituito da classi *custom,* che rappresentano gli oggetti del mondo reale che si vuole formalizzare. Tali entità costituiscono gli oggetti attraverso cui comunicano i vari livelli del sistema.

In base alle *best practices* nello sviluppo delle applicazioni *n-tier*, si è preferito non includere funzionalità avanzate nelle entità. Esse fungono principalmente da contenitori di proprietà, oltre ad avere eventuali metodi accessori che operano localmente su esse. In tal modo, viene creato un disaccoppiamento tra le entità, racchiuse nel modello e i processi che operano su di esse, inclusi nei moduli del BLL.

Poiché viene utilizzato Nhibernate come motore di persistenza, il modello è costituito dalle classi rappresentanti le entità, costituite per lo più dalle funzionalità *getter* e *setter* per l'interfacciamento con le loro proprietà, ognuna accoppiata a un file di *mapping* (*.hbm.xml*) che traduce la rappresentazione relazionale dei dati nel modello a oggetti. Tramite questo *mapping*, è possibile effettuare nel DAL le operazioni CRUD in modo indipendente dalla fonte dati (in pratica, non viene usato SQL) e si rimane coerenti con il modello a oggetti.

In figura 2 viene descritto schematicamente il flusso delle chiamate tra i livelli del sistema, dalla richiesta dell'utente fino all'esecuzione dell'operazione sulla fonte dati.

[1] CRUD (Create Read Update Dolete).

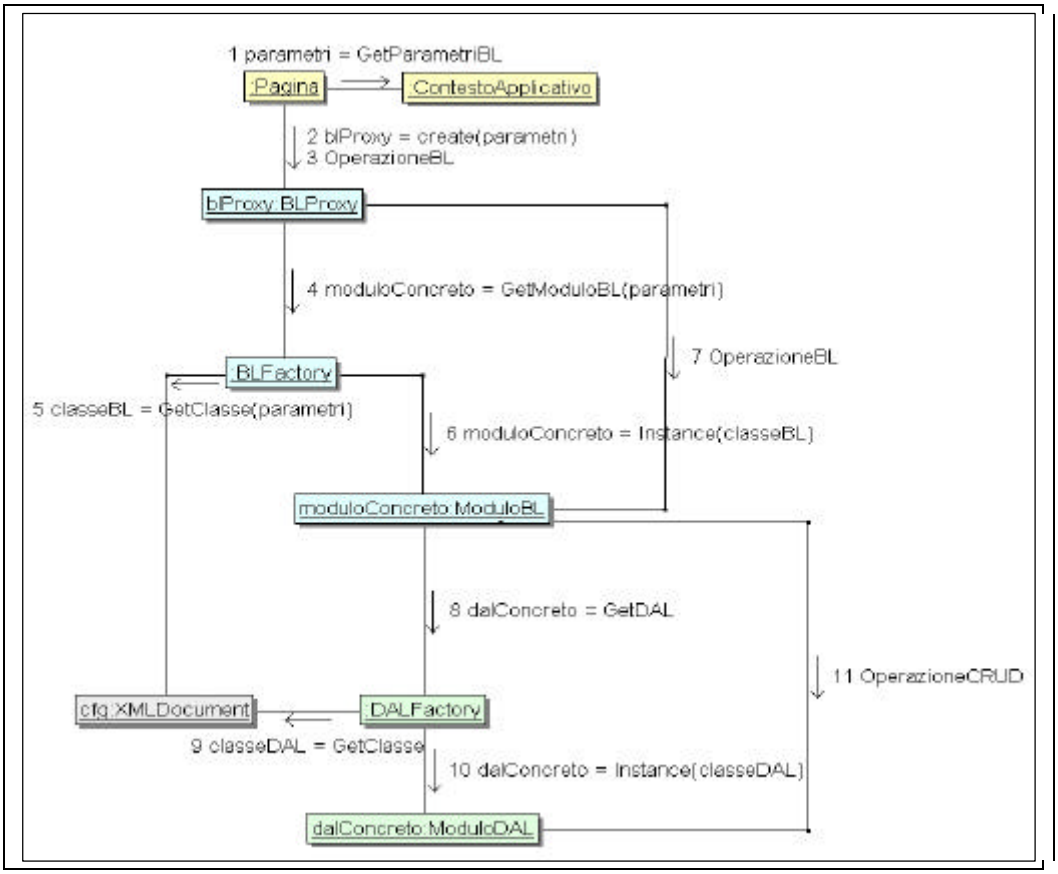

Fig. 2 – Flusso delle chiamate tra i vari livelli del sistema

Conclusioni

La piattaforma ITeR@NET permette a una generica Amministrazione di gestire, organizzare e monitorare in modo automatizzato e sicuro un intero flusso istruttorio. Il sistema è stato sviluppato secondo le normative vigenti e adottando gli standard tecnologici ed architetturali più evoluti. È stata data grande enfasi a caratteristiche quali la semplicità nella configurabilità ed estensibilità del prodotto, oltre alla maggiore flessibilità possibile per adattarsi nel modo migliore alle esigenze di una qualunque Amministrazione.

Il progetto è stato portato avanti secondo la metodologia dello *sviluppo iterativo*. In tale approccio, lo sviluppo è organizzato in una serie di brevi mini-progetti, detti iterazioni; il risultato di ogni iterazione è un sistema eseguibile, testato e integrato, anche se parziale. Ogni iterazione è comprensiva delle attività di analisi, progettazione, implementazione e test, oltre a eventuali revisioni.

Bibliografia

[1] Homepage di Microsoft Patterns&Practices
URL: http://msdn.microsoft.com/practices/
[2] Homepage del sito di NHibernate
URL: http://www.hibernate.org/343.html
[3] Nhibernate - Reference documentation
URL: http://www.hibernate.org/5.html#A17
[3] Best Practices with ASP.NET and Nhibernate.
URL: http://www.codeproject.com/aspnet/NHibernateBestPractices.asp
[4] Pattern DAO: (analogo delle classi DAL, ma per Java).
URL: www.mokabyte.it/2003/06/devj2ee-7.htm
[5] Larman C.(2005), "Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development".
[6] Pattern in .NET
URL: http://www.dofactory.com/