UNIVERSITÀ DEGLI STUDI DI PADOVA

Corso di Laurea Triennale in Ingegneria Informatica

Progetto portale web con UML

9 settembre 2010

Relatore : Ennio Buro

Laureando: Luca Tumelero 575411-IF

Anno accademico 2009/2010

Capitolo 1 **Sommario**

Capitolo 2 **Introduzione**

2.1 Scopo

In Nextep è partito un nuovo progetto web: realizzare un portale europeo che metta in relazione le aziende produttrici di articoli sportivi con i negozi e con gli agenti del settore. Al fine del conseguimento dell'obbiettivo con l'aiuto del capo progettista della parte web Marco De Toni andremo a progettare l'architettura del portale e a scrivere la documentazione da fruire al cliente.

2.2 Introduzione alla modellazione

In ambito tecnico la parola "modello" indica una rappresentazione di un'entità in modo semplificato, astraendo da alcuni aspetti o dettagli irrilevanti allo scopo a cui esso è destinato, in modo da potersi concentrare maggiormente sui dati importanti e manipolarli per estrarne informazioni utili.

Un esempio di modellazione con cui bene o male tutti hanno avuto a che fare è la cartina stradale. In una carta stradale si rappresenta un territorio con chiarezza tralasciando alcuni dettagli, come l'altimetria oppure la larghezza delle strade, allo scopo di orientarsi lungo le strade. Con questo proposito le informazioni tralasciate non incidono sull'uso della cartina per orientarsi lungo le strade, ma la rendono quasi inutilizzabile per affrontare problemi di altro tipo come ad esempio l'analisi catastale dei terreni.

Affinché un modello sia utilizzabile, è necessario che sia espresso in una notazione comprensibile a chi se ne deve servire. Naturalmente quanto più

è diffusa è la notazione, tanto maggiore è la sua importanza e facilità d'uso.

La modellazione software è fondamentale sia per comprendere la realtà all'interno della quale si collocheranno i sistemi (software) da produrre, sia per progettare tali sistemi.

La scelta di un particolare linguaggio di modellazione dipende anzitutto da che cosa s'intende modellare. Infatti, ciascun linguaggio è basato su un vocabolario di termini che rappresentano i concetti principali del dominio che s'intende modellare. Alcuni linguaggi, quindi, possono essere del tutto inadatti a rappresentare alcune entità o quanto meno a rappresentarne gli aspetti di interesse. Ad esempio i simboli delle carte stradali non possono essere usati per descrivere la composizione del suolo e quindi non si può usare una carta stradale per modellare un territorio ai fini dell'analisi dell'inquinamento. Anche i linguaggi in grado di esprimere i concetti rilevanti per il tipo di modellazione in esame, non tutti sono fra loro equivalenti. I principali aspetti da considerare in un'eventuale scelta sono la facilità d'uso del linguaggio e del modello ottenuto, nonché la sua diffusione.

Con facilità d'uso del linguaggio s'intende sia la semplicità di espressione, cioè il fato che sia facile produrre modelli, sia l'immediatezza della comprensione, ovvero quanto sia facile leggere e capire un modello.

Per quanto riguarda la modellazione informatica è molto importante che i linguaggi di modellazione usati siano comprensibili, almeno parzialmente, da non esperti per quanto riguarda i modelli del dominio (quelli che rappresentano la struttura su cui si va ad aggiungere il nuovo prodotto, in

pratica lo status quo del sistema informatico presente) e dell'analisi, perché devono venire discussi con i clienti. Inoltre deve essere facile produrli e afferrarne il significato perché l'attività di modellazione è soggetta a forti vincoli temporali.

2.2.1 Alcuni modelli

2.2.1.1 OMT: Object-modeling technique

Il linguaggio di modellazione OMT fu sviluppato da James Rumbaught nel 1991 e costituì per qualche periodo uno standard de facto nella modellazione software.

OMT si può pensare come un'estensione al paradigma ad oggetti del paradigma entity-relationship utilizzato diffusamente per la progettazione di database.

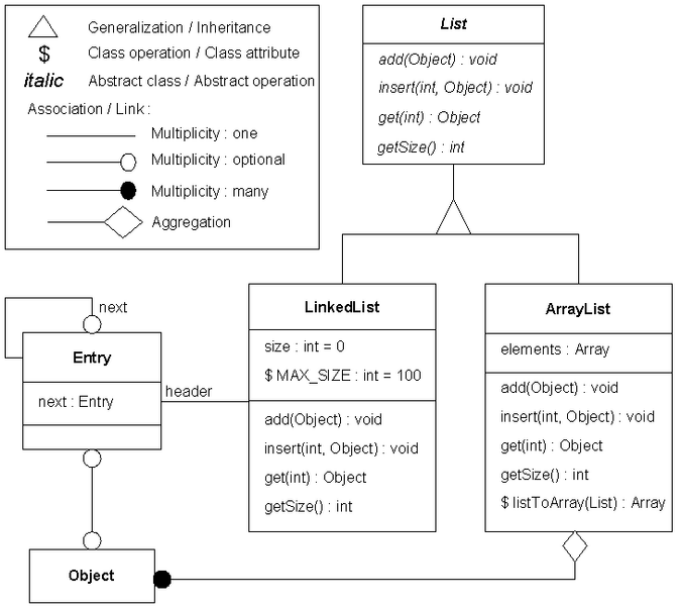

Figure 1 OMT: un esempio

2.2.1.2 Il metodo Booch per l’analisi Object-Oriented

Il metodo Booch per l’analisi Object-Oriented, sviluppato da Grady Booch nel 1992 utilizza notazioni simili a quelle che poi sono state applicate nel linguaggio UML anche se più grezze ed incomplete. L’avvento di UML di fatto fece cadere in disuso questo linguaggio, ma molti aspetti del processo RUP sono direttamente tratti da questa metodologia.

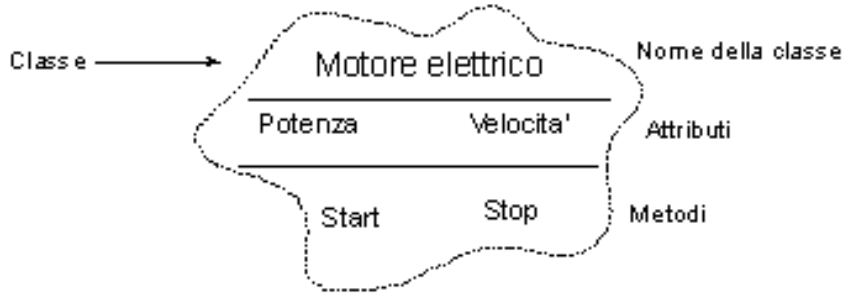

Figure 2 La rappresentazione della classe secondo Booch

2.3 Unified Modeling Linguage

A partire dagli anni ’70, a seguito del crescente successo della programmazione object oriented, furono proposti molti metodi di sviluppo object-oriented, al punto che all’inizio degli anni ’90 più di cinquanta di essi si contendevano il mercato, utilizzando molte notazioni diverse, spesso simili ma non uguali. La cosiddetta *battaglia dei metodi object-oriented* si rifletteva quindi in una battaglia delle notazioni, benché alcuni punti comuni si stessero affermando, primo tra tutti l’opportunità di adottare una notazione visuale sebbene, ad esempio, per rappresentare gli oggetti si proponevano rettangoli, anziché cerchi o nuvolette.

Nei primi anni ’90 la competizione cominciò ad operare una selezione ed a far emergere i metodi più promettenti, che furono raffinati incorporando gli aspetti migliori dei vari metodi, convergendo quindi naturalmente verso una base comune. Alla fine del 1994 la Rational Software Corporation assunse James Rumbaught[1]. Dato che nella stessa azienda lavorava anche Grady Booch[2] si venne a creare il clima ideale per la cooperazione che combinando gli aspetti migliori delle due proposte elaborasse un metodo unificato. Nell’autunno del 1995 al gruppo si unì Ivan Jacobson e i tre divennero subito noti come i “Three amigos” per le loro continue e vivaci discussioni sulla metodologia. Dai loro sforzi congiunti nacque una prima proposta di modellazione comune, in grado di supportare facilmente i rispettivi approcci, UML 0.9, sottoposta nel giugno 1996 al giudizio della

[1] Autore della famosa tecnica di design Object Oriented OMT (Object Modelling Technique)

[2] Autore del metodo Booch, uno dei più promettenti per l’analisi object-oriented

comunità scientifica per ottenere pareri e suggerimenti per possibili migliorie. Sotto la spinta della Rational, che considerava la frammentazione dei metodi come un ostacolo all'adozione di un processo object-oriented nella pratica industriale, i Three Amigos guidarono un consorzio di produttori ed integratori di sistemi informatici, denominato UML Patners, grazie ai cui sforzi si arrivò alla denominazione di UML 1.0. Contemporaneamente alla notazione di UML, i tre svilupparono un meta-modello di processo unificato RUP (rational unified process) che ha tra i suoi punti cardine l'uso di UML.
Nell'agosto del 1997, la versione 1.1 del linguaggio, rivista sulla base delle osservazioni del gruppo per la definizione semantica, venne sottoposta al OMG (Object Management Group), un'organizzazione senza scopo di lucro fondata per stabilire gli standard legati al mondo object-oriented, e nel novembre dello stesso anno fu adottata come standard. Negli anni seguenti, il linguaggio è stato rielaborato sulla base dei commenti e della pratica, fino all'attuale versione UML 2.1 (2007).
Il punto di forza principale di UML è l'essere una famiglia di notazioni visuali integrate fra loro dal fatto di essere tutte object-oriented, progettate per descrivere, seppure in linguaggi diversi per i molteplici aspetti classi, oggetti e i loro comportamenti. La scelta di non adottare una singola notazione ma una famiglia permette di supportare un approccio "multiview" alla modellazione, ovvero di descrivere un'entità da modellare attraverso diversi punti di vista specializzati ed espresse tutte con una notazione consona.

Le vere innovazioni di UML sono sostanzialmente due: avere integrato varie le varie rappresentazioni utilizzando sempre la stessa terminologia per lo stesso concetto, basandosi su una sorta di glossario comune garantendo uniformità; e l'aver descritto tutti i vari aspetti all'interno di UML, sia a livello di semantica statica e dinamica, mediante un meta modello espresso in linguaggio object-oriented.

2.3.1 UML: Use Case Diagram

Il diagramma dei casi d'uso è uno dei diagrammi più semplici di UML, ma anche uno dei più usati e significativi soprattutto per persone non del settore. Gli use case diagram servono per rappresentare ad alto livello i casi d'uso relativi a un'entità, mettendo in evidenza gli attori, cioè le persone esterne all'entità che si sta modellando e che interagiscono con essa, ed a quale entità si riferiscono i singoli casi d'uso.

Gli attori vengono indicati con il simbolo di un uomo stilizzato, mentre i casi d'uso singoli sono rappresentati da un ellisse al cui interno viene posizionato un titolo riassuntivo del caso d'uso.

La facilità di lettura del diagramma e il suo approccio molto ad alto livello permette di essere capito da qualsiasi tipologia di cliente.

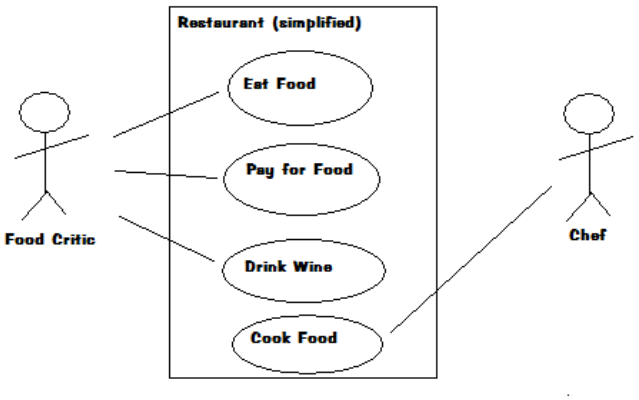

Figure 3 un esempio di use case diagram

2.3.2 UML: Class Diagram

Il diagramma delle classi è sicuramente il diagramma più conosciuto e più diffuso dell'intera suite di raffigurazioni UML. Il diagramma ha molteplici funzioni e diversi gradi di dettaglio, in generale il diagramma descrive le classi e le interfacce che verranno utilizzate per tipizzare gli oggetti del sistema.

In UML a differenza di altri linguaggi di modellazione, vi è una netta distinzione tra operazioni e metodi. Un'operazione è una specifica di un metodo, comprende la firma (nome, parametri, tipo di ritorno) mentre un metodo è l'implementazione di un'operazione, cioè una descrizione algoritmica che soddisfa i vincoli sintattici e semantici imposti dall'operazione.

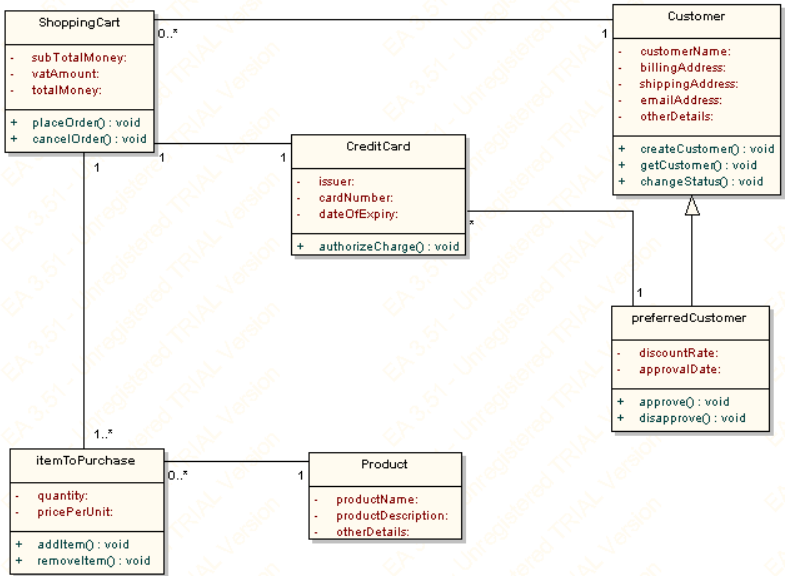

Figure 4 un esempio di class diagram

2.3.3 UML : activity diagram

Gli activity diagram modellano le attività del sistema dal punto di vista del flusso di dati e di controllo. L'idea base sottesa al diagramma è quello di descrivere come si succedono le azioni nell'ambito di un'attività complessa trasmettendosi informazioni e focus di controllo. Ogni azione è rappresentata con un rettangolo con bordi arrotondati contenente una descrizione dell'azione che nella forma più semplice può essere semplicemente il nome dell'azione stessa. In un activity diagram sono sempre presenti un cerchio annerito e un cerchio doppio che rappresentano inizio e fine del flusso.

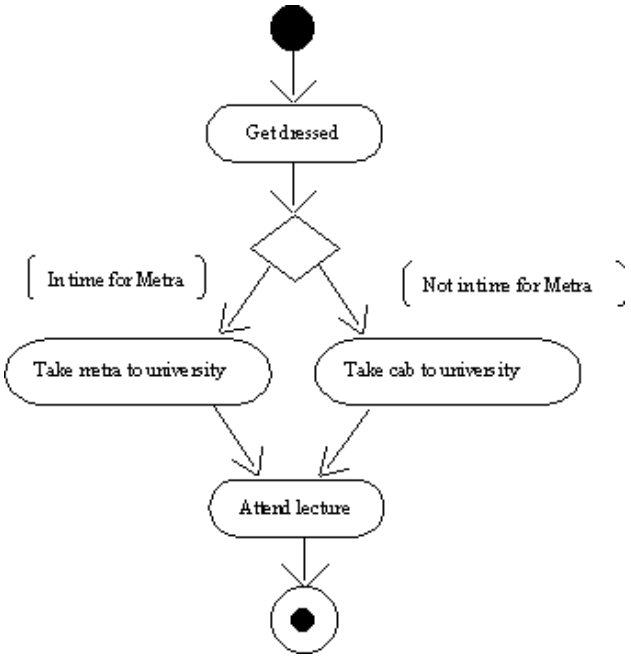

Figure 5 Un esempio di activity diagram

2.4 Strumenti Software

2.4.1 Eclipse

Eclipse è un ambiente di sviluppo software integrato multi - linguaggio e multipiattaforma free e open source: si passa infatti da un completo IDE per il linguaggio Java a un ambiente di sviluppo per il linguaggio C++ e a plug-in che permettono di gestire XML, PHP e persino di progettare graficamente una interfaccia grafica per un'applicazione JAVA. Una delle principali forze di Eclipse è la possibilità di aggiungere un considerevole numero di plug-in aggiuntivi che permettono allo sviluppatore di rendere più snello e veloce il processo produttivo. Il programma è scritto in linguaggio Java, ma anziché basare la sua interfaccia su Swing, il toolkit grafico di Sun Microsystems, si appoggia a SWT, librerie di nuova concezione che conferiscono ad Eclipse un'elevata reattività.
Il progetto originale è partito da un'idea di IBM che nel novembre 2001 apre il progetto Eclipse supportato da un consorzio di rivenditori.
Nel 2004 nasce la Eclipse Foundation, una corporation indipendente e no-profit allo scopo di permettere al progetto Eclipse di attirare sviluppatori al di fuori delle aziende fondatrici. Eclipse è in grado di fornire un completo IDE (*Integrated Development Environment*: ambiente di sviluppo integrato) per la programmazzione in Java, estendendola poi a C++ e altri linguaggi (come i citati PHP e XML). Al momento attuale nel panorama mondiale Eclipse si pone come il migliore o comunque uno dei migliori insieme a Netbeans IDE per java. La scelta di utilizzare questo software è stata effettuata tenendo bene presente che tutto il team di sviluppo conosceva molto bene la piattaforma di sviluppo in quanto praticamente

tutti i progetti degli ultimi due anni sono stati sviluppati con questo software. Inoltre la possibilità di avere plug-in ad hoc per Jboss e Seam di pregevole fattura e usabilità faceva propendere ancora di più la scelta verso questa soluzione a discapito delle altre.

2.4.2 Jboss

Jboss è un Application Server che implementa l'intera suite di servizi di Sun Microsistem Java EE. Anche questo software è open source e multipiattaforma e dal 2006 è stato acquisito da Red Hat inc. che ha proseguito nello sviluppo fino alla corrente versione 5.1.
Jboss è la soluzione server scelta per il progetto che si occuperà di processare tutti gli script java e che quindi è stato montato nel server dove verrà ospitato il portale nonché nel server interno per il betatest.

2.4.3 Seam

Seam framework è una piattaforma di sviluppo free open source per applicativi web in java. L'uso di Seam permette di sveltire il processo di creazione delle classi grazie all'uso di tag appositi che permettono di auto-generare funzioni standard tipo getter e setter di variabili o altro. Seam, inoltre, forza una standardizzazione delle classi, dei metodi e dell'architettura che facilita il rengineering e la gestione del codice lungo il ciclo di vita dell'applicazione.

2.5 La scelta di usare un CMS

2.5.1 Introduzione ai CMS

Un content management system, in acronimo CMS, letteralmente "sistema di gestione dei contenuti", è uno strumento software installato su un server web studiato per facilitare la gestione dei contenuti di siti web, svincolando l'amministratore da conoscenze tecniche di programmazione Web.

Esistono CMS specializzati, cioè appositamente progettati per un tipo preciso di contenuti e CMS generici, che tendono ad essere più flessibili per consentire la pubblicazione di diversi tipi di contenuti.

Tecnicamente un CMS è un'applicazione lato server che si appoggia su un database preesistente per lo stoccaggio dei contenuti e suddivisa in due parti: la sezione di amministrazione (*back end*) e la sezione pubblica altrimenti detta applicativa (front end).

Il back end serve agli amministratori del CMS per creare, gestire e supervisionare la produzione dei contenuti mentre il front end è quella parte che l'utente web usa per fruire dei contenuti e delle applicazioni del sito.

I CMS possono essere realizzati tramite programmazione in vari linguaggi web tra cui più comunemente in ASP, PHP, .NET. Alcuni linguaggi rendono il CMS multipiattaforma, mentre altri lo rendono usufruibile solo su piattaforme proprietarie.

2.5.2 Opzioni possibili

In questa sezione introduciamo una breve descrizione di tutti CMS presi in analisi per il progetto Sport4Trade.com cercando di metterne in evidenza gli aspetti positive e negativi.

2.5.2.1 Linguaggio PHP

2.5.2.1.1 Joomla

Joomla è un CMS opensource nato nel 2005, distribuito in licenza GNU GPL versione 2.0. Il codice sorgente è stato ereditato al momento della nascita dal CMS Mambo, ma attualmente è in rapido sviluppo, sotto la guida di un gruppo di sviluppatori (per buona parte ex-sviluppatori di Mambo) riuniti nell'associazione no-profit *Open Source Matters*. Il software è scritto internamente in PHP e necessita di un database MySql per poter funzionare. Attualmente il CMS è alla versione 1.5.
Joomla offre un valido strumento che permette l'aggiunta di numerosi plug-in che, grazie alla comunità molto nutrita sono in grado di aggiungere quasi tutte le funzionalità necessarie per la realizzazione del portale.
Dal punto di vista grafico, però, è non all'altezza: molte visualizzazioni, infatti, sono presentate grazie all'ausilio di tabelle HTML cosa che da qualche anno a questa parte sono state accantonate in favore di formattazioni di pagina che utilizzano diversi tag HTML (i "*div*").
Oltre a questo non si riscontra nell'azienda Nextep un uso avanzato del linguaggio PHP.

2.5.2.1.2 Drupal

Creato originariamente da Dries Buytaert come bulletin board system, divenne un progetto libero nel 2001. Il nome *Drupal* è la traslitterazione inglese per la parola olandese *druppel* che significa *goccia*. Il nome nasce dal defunto drop.org, sito il cui codice si evolse lentamente fino a trasformarsi in Drupal. Buytaert voleva chiamare il sito «dorp» (In olandese «villaggio», riferendosi all'orientamento «per community» del progetto), ma commise un errore di digitazione quando controllò la disponibilità del dominio. Rileggendo, decise che Drupal suonava meglio. Seppure più anziano di Joomla, Drupal, giunto attualmente alla versione 6 (è attesa nel giro di qualche mese la versione 7) ha saputo rinnovarsi ed evolversi in modo continuo, proponendo un software in grado di gestire tutte le problematiche standard per la creazione di portali web compreso l'e-commerce. Anche dal punto di vista della costruzione grafica il CMS segue tutte le ultime "tendeze" in fatto di stilizzazione dell'HTML (mediante CSS). Anche Drupal richiede il supporto di un database MySql per funzionare.

I dubbi legati a Drupal sorti in fase si analisi sono stati i rallentamenti anche vistosi del sistema standard se sottoposto ad un flusso di visite importante, cosa che ha fatto propendere insieme alla già citata anche per Joomla poca familiarità del team di sviluppo con la programmazione ad oggetti PHP.

2.5.2.2 Linguaggio JAVA

2.5.2.2.1 DotCMS

DotCms viene distribuito da dotCMS inc. in due versioni: community e professional. Il software è giusto recentemente alla versione 1.9. Analizzando il prodotto si nota subito l'assenza di un'integrazione a "plug-in" gestita in modo veloce e semplice, cosa che unita alla giovane età del prodotto hanno portato a non scegliere questo software. Si è, inoltre, notato che la comunità su cui si basa la versione "community" è poco nutrita, cosa che costringerebbe ad acquisire la versione a pagamento con annessa lievitazione dei costi di produzione del portale.

2.5.2.2.2 Alfresco

Alfresco è più che un CMS un repository di contenuti unito a funzionalità di portlets che lo rende molto simile ad un gestore di contenuti tradizionale. Il prodotto è disponibile in due versioni, enterprise (versione a pagamento con assistenza) e community (gratuita e senza assistenza garantita). Il software è modulare, quindi è possibile costruire o installare i propri moduli personalizzati al fine di raggiungere i propri scopi. Alfresco è un prodotto molto valido anche nella sua distribuzione community e fino all'ultimo è sembrato uno dei migliori candidati per l'utilizzo.

2.5.2.2.3 LifeRay Portal

Liferay è un CMS prodotto dall'omonima azienda (Liferay inc.) distribuito in due differenti versioni: una gratuita distribuita con licenza open source con rilasci ogni 12 mesi e una versione enterprise distribuita a pagamento con contratto di assistenza che garantisce rilasci trimestrali e supporto

esteso. Il prodotto è pensato per gestire portali di grandi dimensioni e con grande traffico, garantendo una continuità di fruizione del servizio ottima. La versione attuale è la 5 (quella che è anche stata adottata per il progetto), ma proprio per settembre è atteso il lancio della versione 6. Questo CMS annovera, inoltre, molti clienti importanti come Cisco system o T-mobile solo per citarne alcuni. Il software dispone di applet standard di pregevole fattura che facilitano la costruzione di portali di molto, nonché prevede la possibilità di installarne di personalizzati. La solidità e l'età matura del software (sono circa 5 anni che il progetto è avviato) unito all'esperienza già maturata in un altro progetto da parte di Allos s.p.a. (ditta assorbita da Nextep nel 2006) hanno fatto sì che la scelta per il progetto sport4trade.com ricadesse su questo strumento.

Capitolo 3 **L’azienda: Nextep**

Nextep s.r.l. è una ditta specializzata che lavora nel campo dell’IT da oltre 10 anni. Nel 2006 l’azienda acquisisce il settore web da Allos s.p.a. spostando la propria sede operativa da Cittadella a Carmignano di Brenta dove opera tutt’ora. L’acquisizione della parte web permette a Nextep di offrire un portafoglio di soluzioni superiore e allargare il suo mercato. Tutt’oggi le aree principali di cui si occupa l’azienda sono la Sistemistica, le Infrastrutture Tecnologiche, Il Web Design, lo sviluppo di Web Application e lo sviluppo di Software Verticale.

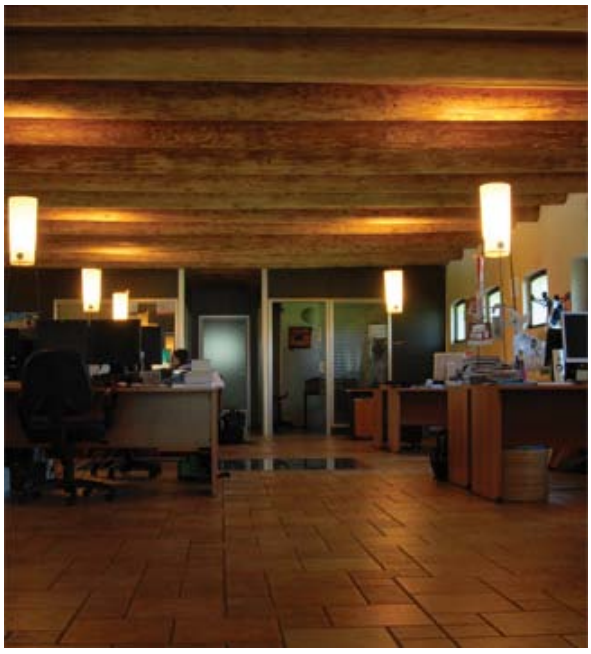

Figure 6 Nextep sede di Carmignano di Brenta

Capitolo 4 **Sport4Trade.com**

4.1 Cosa è sport4trade.com

Sport4trade vuole essere un portale rivolto alle aziende che producono e distribuiscono abbigliamento, accessori e attrezzatura sportiva, ai negozianti che rivendono e devono comprare tali articoli ed infine agli agenti che rappresentano le aziende.

Nel portale si potranno trovare informazioni esclusive inerenti alle aziende, ai loro prodotti, ai loro cataloghi, ai loro eventi , alle loro campagne vendite, le offerte di lavoro degli agenti e le informazioni sulle fiere del settore.

Il portale è gestito da uno staff specializzato che gestirà i contenuti per conto dei clienti che saranno abbonati. Sport4trade.com offrirà inoltre una serie di servizi collaterali come newsletter, analisi di mercato e dei trend economici oltre che ad una ricca raccolta di articoli riguardanti gli sport dal punto di vista dei negozianti.

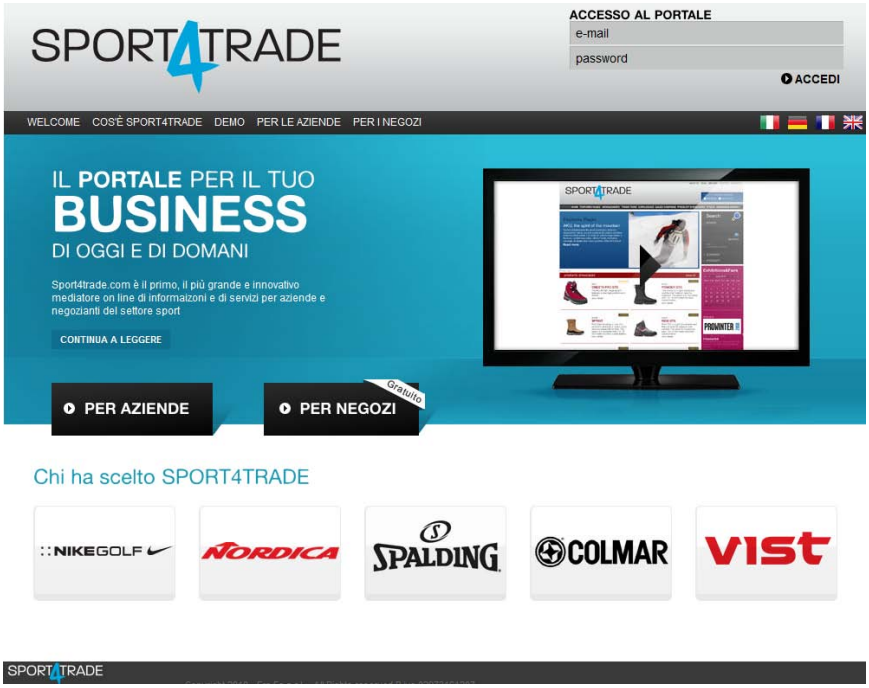

Figure 7 l'home page del sito realizzato

4.2 Analisi Requisiti

Sin da una prima analisi della relazione sull'analisi dei requisiti svolta da Nextep appare chiara la presenza di più attori tra cui spiccano l'azienda, il negozio e l'agente, oltre ai ruoli di amministratore e fiera. Ognuno di questi attori ha bisogni e compiti diversi, anche se è possibile trovare in essi delle somiglianze e dei punti in comune.

L'azienda è a tutti gli effetti chi produce o distribuisce all'ingrosso l'abbigliamento e gli articoli sportivi. L'interesse principale sarà quindi quello di essere presente sul portale tramite un company profile, composto dalla propria anagrafica, una scheda con tutte le informazioni sulla produzione e commercializzazione, e una vetrina prodotti multimediale.

L’azienda avrà la possibilità, grazie al portale, di far conoscere ai negozianti in anteprima, tramite la sezione *vetrine*, tutte le novità da loro prodotte o distribuite; nella sezione *cataloghi/brochure* di consentire ai negozi di avere i cataloghi sempre aggiornati; nella sezione *campagne vendita* di informare distributori, negozi e agenti dei prodotti diventando un ottimo strumento di promozione e supporto alla vendita; la sezione *stock* è invece un valido supporto di vendita con la possibilità di visualizzare i prodotti con prezzi e disponibilità offrendo in promozione determinati stock di prodotti.

L’azienda, inoltre, potrà utilizzare il portale inserendo i filmati delle proprie sfilate, degli eventi da loro organizzati, delle interviste, venendo coinvolta negli speciali creati dalla redazione, e visionare informazioni sui propri concorrenti, agenti, fiere ed eventi.

Un apposita bacheca consentirà alle aziende di esporre richieste di ricerca personale o nuove aree di vendita per i loro prodotti.

Il negozio è chi vende al consumatore finale l’abbigliamento e l’articolo sportivo, è quindi il principale destinatario del portale.

Il negoziante potrà usare il sistema come fonte di informazione per l’acquisto dei nuovi prodotti, identificare nuovi brand, conoscere le novità, essere sempre aggiornato sui marchi e le relative aziende che li vendono o distribuiscono, scaricare i cataloghi, e conoscere eventuali promozioni per acquistare determinati prodotti.

Il negoziante inoltre, grazie al portale, potrà inoltre essere sempre presente, anche se non fisicamente, ai vari eventi organizzati dalle aziende, alle

principali fiere del settore tramite i servizi filmati e gli approfondimenti realizzati dalla redazione di Sport4Trade.
Gli agenti sono i motori del mercato in quanto presentano ai negozianti i nuovi prodotti, curano il rapporto tra negozio e azienda, oltre che riportare alle aziende i dati relativi alle tendenze del mercato.
Utilizzeranno il portale per visualizzare le aziende, per trovare nuove occasioni di mercato o nuovi prodotti da rappresentare. Leggeranno nel area apposita le richieste delle aziende.
Le fiere sono il principale punto d'incontro tra chi vende e chi acquista quindi tra aziende, distributori e negozianti. Nel portale potranno essere presenti con una scheda dettagliata sugli eventi e le caratteristiche della fiera oltre che con tutte le informazioni di supporto: cataloghi, brochure, schede di adesione, informazioni utili.

4.2.1 Funzionalità per attore

Utente anonimo

Landing Page: Home page user anonimo

Funzionalità per questa tipologia di utente:

- Gestione Login (Accesso/reinvio password)
- Gestione Registrazione
- Visualizzazione delle pagine de di filmati introduttivi

Utente azienda

Landing Page: home page azienda

Funzionalità per questa tipologia di utente:

- Ricerca:
    - Ricerca prodotto con visualizzazione lista prodotti ordinabili per data inserimento alfabetico e marchio di appartenenza.
    - Ricerca azienda con visualizzazione delle aziende in ordine alfabetico.
    - Ricerca marchio con visualizzazione delle aziende distributrici in ordine alfabetico.
- Visualizzazione della scheda azienda.
- Visualizzazione della scheda prodotto vetrina.
- Visualizzazione e modifica del proprio profilo utente.
- Visualizzazione fiere ed eventi.
- Visualizzazione news.
- Visualizzazioni sezione “in Primo Piano”.
- Visualizzazione offerte aziende (una bacheca testuale di annunci di richieste da parte delle aziende o di proposte da parte degli agenti).
- Visualizzazione, modifica e inserimento proprie proposte agenti.
- Visualizzazione degli articoli della sezione “Sport Economy”.

Utente negozio

Landing Page: Home Page Negozio

Funzionalità per questa tipologia di utente:

- Ricerca:

    - Ricerca Prodotto.
    - Ricerca Azienda.
    - Ricerca Marchio.
- Visualizzazione della scheda azienda.
- Visualizzazione della scheda prodotto vetrina.
- Visualizzazione e modifica del proprio profilo.
- Visualizzazione fiere ed eventi.
- Visualizzazione news.
- Visualizzazioni sezione “in Primo Piano”.
- Visualizzazione campagne vendita delle aziende.
- Visualizzazione degli stock disponibili da parte delle aziende.
- Visualizzazione cataloghi delle aziende da poter scaricare.
- Visualizzazione degli articoli della sezione “Sport Economy”.

Utente agente

Landing Page: home page agente

Funzionalità per questa tipologia di utente:

- Ricerca:
    - Ricerca prodotto.
    - Ricerca azienda.
    - Ricerca marchio.
- Visualizzazione della scheda azienda.
- Visualizzazione della scheda prodotto.
- Visualizzazione e modifica del proprio profilo.

- Visualizzazione fiere ed eventi.
- Visualizzazione news.
- Visualizzazioni sezione “in Primo Piano”.
- Visualizzazione offerte aziende.
- Visualizzazione, modifica, inserimento proprit proposte di rappresentanza per le aziende.

Utente fiera

Landing Page: home page fiera

Funzionalità per questa tipologia di utente:

- Ricerca:
  - Ricerca Prodotto.
  - Ricerca Azienda.
  - Ricerca Marchio.
- Visualizzazione della scheda azienda.
- Visualizzazione della scheda prodotto in vetrina.
- Visualizzazione e modifica del proprio profilo utente.
- Visualizzazione fiere ed eventi.
- Visualizzazione news.
- Visualizzazioni sezione “in Primo Piano”.
- Visualizzazione degli articoli della sezione “Sport Economy”.

Aministratore

Landing Page: Console Amministrazione

Funzionalità: Nella prima fase del progetto tutti i contenuti sono inseriti e gestisti da un amministratore generale. Gli utenti, escluso la possibilità di modificare i propri dati del profilo e per gli agenti l'inserimento degli annunci, non possono aggiornare contenuti

4.3 Rappresentazione grafica: casi d'uso

Il primo obiettivo del mio tirocinio è stato produrre un diagramma UML dei casi d'uso. Dopo un'attenta lettura dell'analisi dei requisiti si sono potuti isolare con facilità i ruoli-utente: fiera, agente, negozio e azienda e dato che la maggior parte dei casi d'uso erano comuni a tutti questi ruoli ho provveduto a generalizzare l'utente e collegare direttamente ad esso tutti i casi d'uso comuni e lasciando collegate alle specializzazioni solo i casi d'uso relativi alle sole specializzazioni dell'utente (figura 8).

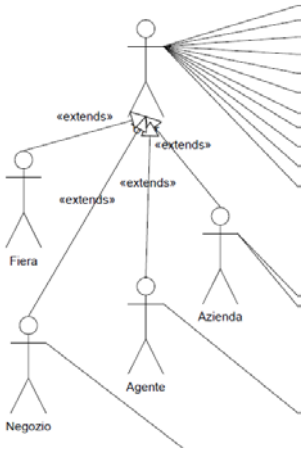

Figure 8 La generalizzazione degli attori-utenti

Fino a questo momento tutto procedeva per il meglio fino al sorgere delle prime perplessità riguardanti alla rappresentazione dell'accesso al portale degli utenti. A rigor di UML il fatto che ogni caso d'uso richiedesse l'utilizzo di una procedura di login dovrebbe essere rappresentata con un caso d'uso secondario che tramite una freccia <<*uses*>> si riferisce ad ogni caso d'uso ma, questa rappresentazione sarebbe risultata molto pesante dal punto di vista visivo e dato che il destinatario del lavoro doveva essere il committente si è preferito adottare una soluzione dal minor impatto visivo ma comunque chiara e ordinata seppur non legale secondo gli standard UML 2.0. Questa soluzione che è riportata in figura 3 è quella di collegare il sottocaso *login* alla cornice che rappresenta il sistema portale sport4trade.com.

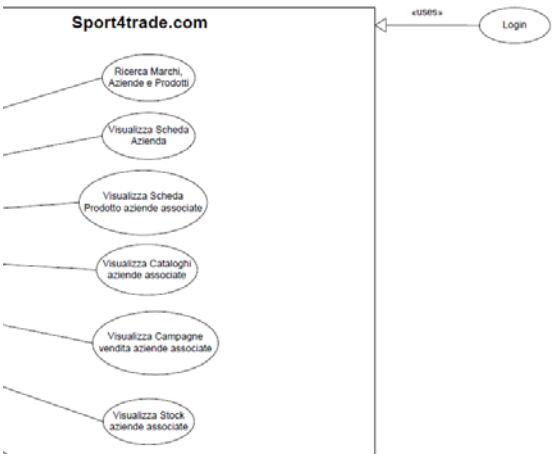

Figure 9 Una soluzione non del tutto legale nel linguaggio UML

4.4 Rappresentazione grafica: permessi di visualizzazione

I permessi di visualizzazione sono qualcosa che in un certo qual modo è direttamente collegata ai casi d'uso, la differenza fondamentale è che non rappresentano che cosa può fare un determinato attore, ma se un attore può o non può visualizzare, modificare o inserire un determinato oggetto del portale. Questa rappresentazione è chiaramente pensata a scopo interno, serve per i programmatori in modo che possano impostare i vari permessi di accesso agli oggetti del sito che nel nostro caso sono il contenuto di determinate pagine ovvero i prodotti, le schede delle aziende, i brand, i cataloghi, gli stock, le campagne vendita, gli annunci, gli articoli sportivi e gli eventi fieristici.

Questa rappresentazione è stata di gran lunga la più difficile per me non tanto a livello contenutistico perché questo era ben conosciuto e sviscerato, ma a livello di rappresentazione, è infatti molto complesso trovare un modo per rappresentare questo modello con un linguaggio visuale nonchè con UML.

Un primo tentativo naturale è stato quello di rappresentare tale vincoli mediante un secondo diagramma di casi d'uso, ma questa implementazione non era molto sensata in quanto quello che si rappresentava non era un caso d'uso vero e proprio a meno di non definire dei casi d'utilizzo fittizi come ad esempio "visualizzo *xxx*" dove *xxx* è il nome di un oggetto di cui si sta descrivendo le politiche di accesso. Questo diagramma però non forniva l'immediatezza desiderata e diventava complicato da leggere all'aumentare degli oggetti collegati e all'inserimento dei permessi di modifica e inserimento.

Un’altra soluzione pensata è stata suggerita dall’essenza degli elementi in gioco: oggetti ovvero qualcosa che ha molto a che vedere con le classi del diagramma delle classi UML. Questa soluzione rappresenta in modo corretto gli elementi in gioco, ma come rappresentare in questo modo i vari tipi di utenti? E come fare a rappresentare il vincolo di visualizzazione che sta alla base del portale ovvero “vedo solo gli articoli e affini delle aziende collegate al mio utente”? Queste domande apparentemente facili sono risultati essere degli scogli troppo complicati da aggirare mantenendo la rappresentazione lineare e semplice in quanto alcune soluzioni possibili erano quelle di rappresentare la generalizzazione degli utenti e tramite una classe di associazione rappresentare il vincolo di relazione con le aziende tramite una lista, ma è semplice rendersi conto che questa soluzione oltre al non essere legale per UML 2.0 potrebbe essere del tutto forviante dal punto di vista del programmatore che si trova di fronte un diagramma delle classi che non descrive le vere e proprie classi del sistema (almeno non in tutti i casi).
Dopo una serie di discussioni e riflessioni si è giunti alla conclusione di abbandonare lo standard UML per puntare a qualcosa di diverso che potesse rendere nel modo migliore quanto richiesto.
La prima soluzione pensata da me è stata un mix di use-case per la rappresentazione degli attori e gli insiemi di Venn per la rappresentazione del vincolo di associazione utente-aziende (figura 10).

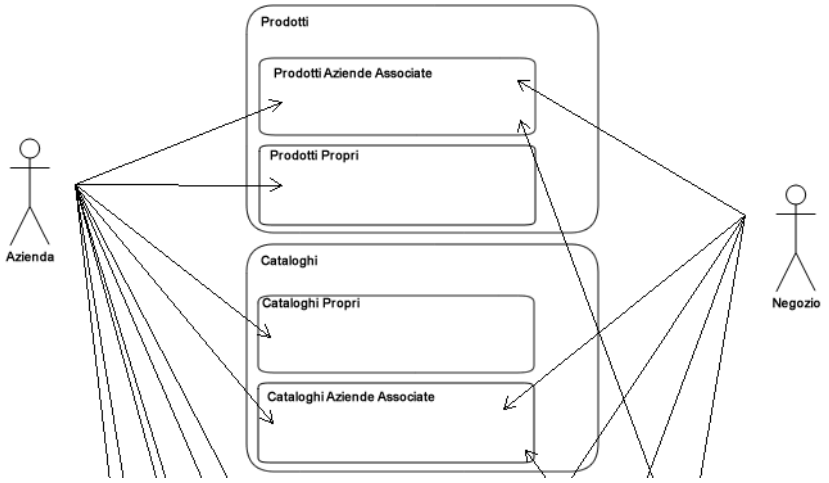

Figure 10 una rappresentazione mista usecase – diagrammi di Venn

Come si vede bene dalla figura 10 si è deciso di eliminare la generalizzazione degli utenti per rendere più immediato il diagramma che risulta leggibile e immediato.

Un'altra strada intrapresa è stata quella di abbandonare la rappresentazione grafica degli attori per passare ad una rappresentazione tabellare; scelta che si è dimostrata vincente in quanto con semplicità si può costruire una tabella che rappresenti in modo esaustivo i permessi di accesso ai numerosi oggetti presenti (figura 11).

Sicurezza: Visualizzazioni

		Company Profile	Brand	Prodotti	Cataloghi	Campagne Vendita	Stock	News & Eventi
Azienda	Tutti	☑	☑					☑
	Aziende a cui ha accesso			☑	☑	☑	☑	
	Propri			☑	☑	☑	☑	
Negozio	Tutti	☑	☑					☑
	Aziende a cui ha accesso			☑	☑	☑	☑	
Agente	Tutti	☑	☑					☑
	Aziende a cui ha accesso			☑	☑	☑	☑	
Fiera	Tutti	☑	☑					☑
	Aziende a cui ha accesso			☑	☑	☑	☑	
Admin	Tutti	☑	☑	☑	☑	☑	☑	☑

Figure 11 rappresentazione tabellare dei permessi di accesso agli oggetti

4.5 Rappresentazione grafica: classi di database

Le classi di database sono le classi java le cui istanze rimangono memorizzate fisicamente nella memoria formando un database ad oggetti persistente. Questo è possibile in quanto le classi suddette implementano *javax.persistence*. La progettazione del database è stata svolta in collaborazione tra gli analisti che avevano delineato i requisiti e il capo-progetto. Il mio compito durante la delineazione delle classi è stato quello di prendere nota e redigere il diagramma delle classi in uml.

Durante la riunione si è proceduto con una prima rilettura dei requisiti strutturati e individuazione delle entità chiave che sono risultate essere:

- utente
- azienda
- negozio
- agente
- fiera

- prodotto
- catalogo
- stock
- evento
- articoli sportivi (features pages)
- news
- articoli economici sullo sport
- annunci

L’utenza identificata è come traspare chiaramente dalle specifiche il motivo principale del portale: è possibile visualizzare tutto il contenuto solo una volta autenticato, quindi sembra ovvio partire nella progettazione del database da questa entità.

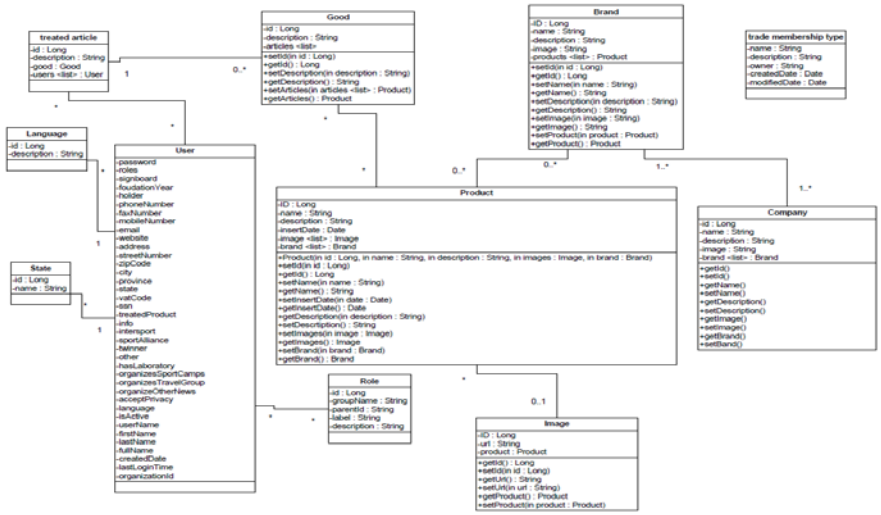

Figure 12 Class diagram

4.6 Rappresentazione grafica: flussi di accesso

A scopo chiarificatore si è deciso di realizzare un diagramma UML di attività per i tre tipi di user principali (negozio, azienda e agente). Come si può immaginare molto facilmente in realtà la navigazione su un sito internet non segue un flusso monodirezionale in quanto la sola presenza di un menu offre all'utente di scegliere in che zona del sito spostarsi. Questa moltitudine di possibilità non è stata però rappresentata in quanto lo scopo di questo diagramma è ancora una volta la miglior comprensione da parte del personale interno del flusso di navigazione "tipico" all'interno del portale. La navigazione, infatti, è caratterizzata da una parte comune a tutti gli utenti e non, ovvero l'arrivo sulla home-page dell'utente anonimo (not registered user landing page) dove sono previsti gli appositi strumenti per la registrazione di un account nuovo oppure dell'identificazione. Una volta autenticato (login) l'utente viene direzionato verso la home-page progettata per la sua tipologia di utenza da dove potrà scegliere in che modo consultare il sito. Una volta che l'utente ha scelto una pagina la struttura del portale prevede la possibilità praticamente per tutti gli "oggetti del sito" di visualizzarne il dettaglio, ovvero una pagina in cui ci sono ulteriori informazioni sull'oggetto scelto (ad esempio un prodotto od una campagna vendita o un'azienda).

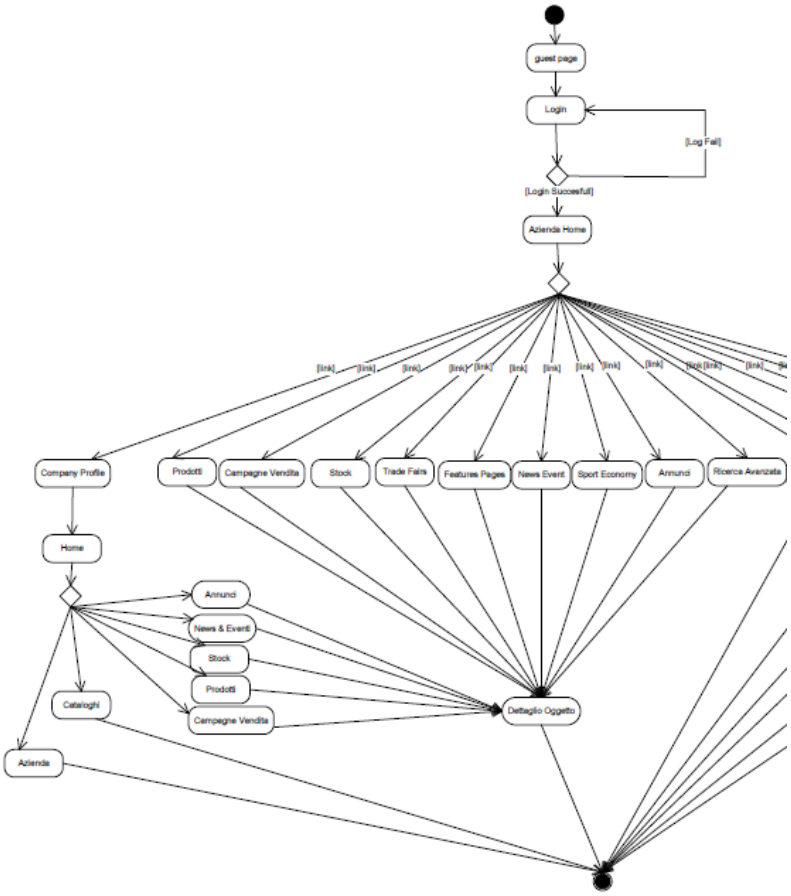

Figure 13 spaccato dell'activity diagram per azienda

Capitolo 5 **Conclusioni**

5.1 Considerazioni sull'ambiente

Il tirocinio si è svolto in un ambiente tranquillo e collaborativo, tutte le persone si sono mostrate gentili e disponibili.

5.2 Considerazioni sul progetto

Sport4Trade è stato un progetto importante, costato circa 9 mesi di lavoro tra raccolta requisiti, analisi, progettazione, realizzazione e messa in opera al fronte di una stima iniziale, accettata dal cliente di un anno. La riduzione del tempo disponibile per la realizzazione del progetto ha fortemente inciso sulle scelte progettuali e costruttive; le scelte iniziali erano di procedere con una metodologia iterativa che prevedeva la consegna di funzionalità per parti al cliente con scaglioni di distanza due mesi a partire da sei mesi dopo l'inizio del progetto. La scelta di riduzione dei tempi da parte del cliente ha influito non solo sulla metodologia di realizzazione del progetto, ma anche nel mio lavoro all'interno del progetto che doveva dapprima essere di ausilio nella progettazione e di realizzazione di test automatici con relativa esecuzione di essi e analisi dei risultati. L'adozione di Liferay Portal per ridurre i tempi di realizzazione del progetto ha cambiato anche i miei compiti che non hanno più previsto la scrittura di programmi per il testing di parti del progetto, ma l'implementazione delle policy di accesso per gli utenti sul portale e la creazione di modelli ad importazione automatica per la creazione di contenuti all'interno del portale (template).

Il progetto è stato un successo ed è possibile usufruire del portale dal primo di luglio di quest'anno sebbene in alcuni momenti si sono verificate delle incertezze sulla realizzazione dovuti anche alla relativa poca esperienza nella tecnologia utilizzata e al numero di programmatori che hanno contribuito al progetto, quasi una dozzina. Per l'istruzione dei programmatori sugli obiettivi e sulle funzioni del progetto sono stati utilizzati i diagrammi prodotti da me che si sono dimostrati esaustivi e velocemente assimilabili, permettendo di diminuire il tempo impiegato per l'istruzione di quelli che sono andati a realizzare effettivamente il codice. Per quanto riguarda la mia esperienza personale, il tirocinio si è rivelato essere ricco di stimoli e ha permesso la mia crescita a livello culturale nel settore grazie allo studio di prodotti professionali per la realizzazione di grossi progetti come Liferay, Jboss e Eclipse.

Capitolo 6 **Bibliografia**

Martin Fowler *UML Distilled*, terza edizione Pearson Education Italia

Roger S. Pressman *Principi di Ingegneria del Software*, quinta edizione McGraw-Hill

Richard Sezov *Liferay Portal Administrator's Guide*, 3rd Edition, Liferay Press

Jonas X. Yuan *Liferay Portal Enterprise Intranets*, pack publishing

6.1 Sitografia

www.nextep.it

www.wikipedia.org

www.eclipse.org

www.jboss.org

www.liferay.com

Capitolo 7 **Ringraziamenti**

Si ringraziano per il tempo dedicato e l’accoglienza calorosa lo staff di Nextep e in particolare Marco De Toni e Simone Peserico nonché Mirko Soffia per la disponibilità. Un sincero ringraziamento anche al tutor universitario professor Ennio Buro per la competenza e i consigli.

Capitolo 8 **Glossario delle immagini**