**Università degli Studi di Roma "Tor Vergata"**

FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Informatica

# Prova Curricolare

Candidato:
**Andrea Graziani**
**Matricola 0189326**

Relatore:
**Francesco Lo Presti**

**Anno Accademico 2017–2018**

# Indice

# 1 Ingegneria del software e Progettazione Web

Il corso di *"Ingegneria del software e Progettazione Web"* tenuto dal prof. *Giovanni Cantone* è focalizzato sull'apprendimento di un insieme di nozioni di base riguardante la documentazione, la progettazione e l'implementazione di applicazioni software usando una paradigma di **programmazione orientato agli oggetti**.

## 1.1 Descrizione del progetto

Il progetto del corso prevedeva lo sviluppo di **un'applicazione software dedicata al commercio elettronico**, vale a dire un software di *e-commerce* basata su database per la gestione degli articoli e delle anagrafiche dei clienti.

La descrizione delle funzionalità del sistema e delle varie soluzioni analitiche e progettuali proposte durante lo sviluppo dell'applicazione dovevano essere opportunamente documentate (attraverso, per esempio, il *Documento di Visione* o il *Documento di analisi e progettazione*) e corredate da grafici e diagrammi realizzati con il **linguaggio di modellazione UML** (come, ad esempio, il *class diagram* o il *sequence diagram*).

Le specifiche di progetto prevedevano che il *back-end* dell'applicazione, ovvero la componente software contenente sia la **logica di business**, vale a dire la logica applicativa che rende operativa un'applicazione, sia la **gestione dei meccanismi di persistenza e memorizzazione dei dati**, fosse realizzata con un **paradigma di programmazione orientato agli oggetti** adottando il linguaggio **Java**.

Per quanto riguarda il *front-end*, ossia la componente software visibile all'utente e con cui egli può interagire, erano previste due differenti versioni:

- Una **versione per desktop** la quale prevedeva lo sviluppo di un'interfaccia grafica attraverso l'uso del framework **Swing** o, in alternativa, della piattaforma **JavaFX**.
- Una **versione per Web** in veste di una *pagina web dinamica* gestita attraverso il web server **Apache Tomcat** sfruttando le tecnologie **JavaServer Pages** (**JSP**) e **servlet**.

## 1.2 Il processo di sviluppo software

Lo sviluppo del sistema software proposto ha richiesto lo svolgimento di numerose attività nonché lo studio e l'applicazione di diverse metodologie, paradigmi e pratiche di analisi, progettazione e programmazione.

Come noto, il tipo, la scansione temporale e il numero di attività da svolgere sono subordinate alla scelta di un **modello** (o **processo**) **di sviluppo software**; quest'ultimo definisce un approccio per progettare, implementare ed eseguire la manutenzione di un dato sistema software.[1]

Nel nostro caso abbiamo adottato un processo di sviluppo software di tipo *iterativo* denominato **Rational Unified Process** o **RUP** la quale, pur non definendo un singolo, specifico processo, bensì un framework adattabile che può

[1] Cfr. Craig Larman - *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition*, Addison Wesley Professional, pp. 56

dar luogo a diversi processi di sviluppo software, scompone il ciclo di sviluppo in un certo numero di *fasi* caratterizzate da un certo insieme di obiettivi che si concludono con la realizzazione di un qualche prodotto. Le fasi sono ulteriormente scomposte in *iterazioni*, che sono associate a periodi temporali e hanno scadenze precise.[2]

## 1.3 L'attività di analisi

Qualunque sia il processo di sviluppo adottato, una delle attività più importanti è senza dubbio quella di **analisi** la quale si focalizza in generale sullo studio dei problemi e dei requisiti piuttosto che sulle soluzioni.[3]

Per essere più precisi questa fase comprende la determinazione, l'analisi, la documentazione, la validazione e la gestione dei requisiti del sistema software ovvero dell'insieme delle funzionalità che l'applicazione deve essere capace di offrire per soddisfare le esigenze degli *stakeholders*. Definire le necessità dell'utente e le caratteristiche del sistema implementato, in altri termini, fornire una *specifica dei requisiti*, rappresenta un'attività critica dello sviluppo di un sistema software in quanto permette di evitare la realizzazione di funzionalità non richieste che comporterebbero aumento dei costi e allungamento dei tempi di sviluppo e manutenzione.

## 1.4 L'attività di progettazione

La progettazione di un sistema software è un'attività che comprende l'elaborazione di una *soluzione concettuale*, piuttosto che implementativa, che permetta di soddisfare i requisiti individuati durante la fase di analisi; questa fase riguarda, ad esempio, la descrizione di uno schema di una base dati o l'organizzazione degli oggetti software.[4]

Durante la fase di progettazione abbiamo studiato e applicato un approccio denominato **responsibility-driven design** o **RDD** (in italiano **progettazione guidata dalle responsabilità**) in accordo alla quale gli oggetti software vengono pensati in termini di **responsabilità**, **ruoli** o **collaborazioni**.[5] Secondo tale approccio gli oggetti software vengono organizzati sia in base alle *informazioni* che possono incapsulare, condividere, derivare e calcolare, sia in base alle *operazioni* eseguite in modo autonomo o in collaborazione con altri oggetti.[6]

Un concetto chiave della fase di progettazione di un sistema software è quello di **pattern**. Un pattern rappresenta una descrizione di una soluzione generale a un problema di progettazione ricorrente applicabile in contesti diversi a cui viene attribuito un nome, contenente consigli su quando e come può essere applicato nonché possibili implementazioni, varianti e altro ancora.[7]

Infatti, l'operazione di individuazione ed assegnazione delle responsabilità degli oggetti software è avvenuta tenendo in considerazione un insieme di pattern o principi di progettazione denominati **General responsibility assignment**

[2] Cfr. ivi, pp. 56, 58
[3] Cfr. ivi, pp. 41
[4] *ibid.*
[5] Cfr. ivi, pp. 416
[6] *ibid.*
[7] Cfr. ivi, pp. 420

**software patterns** (o **principles**) o **GRASP**: si tratta di un insieme di nove pattern[8] il rispetto delle quali ha consentito di ottenere un sistema software caratterizzato da un bassa dipendenza tra gli oggetti, ottenendo in tal modo un basso impatto in seguito ai cambiamenti, migliorando contemporaneamente l'incapsulamento e facilitando il riuso e la manutenzione degli oggetti stessi.

Al fine di ottenere una migliore organizzazione degli oggetti software sviluppati durante le attività di progettazione e programmazione del sistema, sono stati studiati e implementati alcuni dei pattern descritti all'interno di una grande raccolta contenuta in un famoso libro intitolato "**Design Patterns: Elements of Reusable Object-Oriented Software**" scritto da un insieme di quattro autori chiamati **Gang of Four** (**GoF**).

L'identificazione degli oggetti e delle loro iterazioni è stara resa possibile seguendo anche un pattern speciale denominato **Entity-Control-Boundary Pattern** il quale, oltre a separare la logica di presentazione dei dati dalla logica di business, garantisce un buon design per lo sviluppo del sistema e la soddisfazione dei requisiti classificando gli oggetti in:

- Oggetti *Entity* che rappresentano i concetti del dominio.
- Oggetti *Control* che modellano la logica applicativa.
- Oggetti *Boundary* responsabili di gestire le interazioni tra sistema e tutti gli attori.

## 1.5 Alcune attività di sviluppo

Una parte del sistema software è stata sviluppata facendo ricorso ad una pratica di sviluppo denominata **test-driven development** o **TDD** (in italiano **sviluppo guidato dai test**) secondo la quale la stesura dei test automatici deve avvenire prima di quella del software che deve essere sottoposto a test.[9]

Tale pratica ha garantito una migliore qualità del software sviluppato poiché la grande quantità di test automatici, realizzati attraverso l'uso del framework **JUnit**, ha aiutato a ridurre il numero di bug presenti nel codice.

Un'altra attività molto importate è stata quella di **refactoring** ovvero una pratica volta a migliorare il design del codice già scritto senza alterarne il comportamento esterno che ha permesso di ripulire il codice e minimizzare la quantità di bug in esso presenti.[10]

[8] Cfr. ivi, pp. 423

[9] Cfr. ivi, pp. 530

[10] Cfr. Martin Fowler - *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Professional, pp. 8

# 2 Mobile Programming

Il corso di *"Mobile Programming"* tenuto dal prof. *Massimo Regoli* è stato cruciale per acquisire un insieme di conoscenze di base necessarie allo sviluppo di applicazioni per dispositivi mobili.

Il corso prevedeva lo svolgimento di due progetti intermedi e di uno finale riguardanti la progettazione e lo sviluppo di applicazioni mobili per la piattaforma *Android*, il più diffuso sistema operativo per dispositivi mobili, utilizzando, come piattaforma di sviluppo, l'IDE *Android Studio* che integra l'*Android SDK*, ovvero un insieme di librerie e strumenti per sviluppare e testare applicazioni Andriod.

## 2.1 Primo progetto intermedio

Il primo progetto intermedio richiedeva la progettazione e lo sviluppo di una semplice applicazione per la gestione di un distributore automatico di bevande con la possibilità, da parte dell'utente, di selezionare e personalizzare uno o più prodotti dal catalogo messo a disposizione.

Nonostante l'enorme semplicità dell'applicazione, il suo sviluppo fu cruciale per comprendere le dinamiche alla base dello sviluppo di un'applicazione mobile che hanno riguardato:

- Lo studio dell'**architettura** generale di un'applicazione Android riguardante, ad esempio, lo studio del concetto di *Activity* o del ruolo del file *manifest*.
- La gestione del **ciclo di vita delle applicazioni Android** ovvero lo studio dei metodi invocati dal sistema operativo quando si naviga tra i vari stati del ciclo di vita della nostra applicazione (`onCreate()`, `onStart()`, `onResume()`, ecc.)
- La **gestione dei permessi** per l'accesso alle risorse del sistema come la memoria, la fotocamera, i contatti e così via.
- Lo studio delle nozioni fondamentali per la realizzazione dell'**interfaccia grafica** affinché sia il più possibile chiara ed intuitiva per la maggior parte degli utenti. A tale scopo fu cruciale acquisire familiarità con i meccanismi di gestione degli elementi grafici nonché con la grande quantità di componenti e servizi messi a disposizione dell'Android SDK (es. *layout*, pulsanti, ecc.). L'interfaccia è stata realizzata mediante la scrittura in codice XML nonostante la presenza di un editor visuale integrato nell'ambiente di sviluppo.
- Lo studio delle nozioni elementari alla base del concetto di **localizzazione** delle applicazioni Android.

## 2.2 Secondo progetto intermedio

Il secondo progetto intermedio riguardava lo sviluppo di un progetto che sfruttasse una delle tecnologie messe a disposizione dal sistema operativo (tecnologia NFC, geolocalizzazione, i *widget*, meccanismi di notifica, ecc.)

Il progetto assegnatomi dal professore riguardava la realizzazione di un'applicazione capace di impostare un sfondo animato sulla schermata *home* del sistema operativo, ovvero, si richiedeva lo sviluppo di un *live wallpaper*.

L'applicazione sviluppata offre all'utente la possibilità di personalizzare alcuni aspetti dello sfondo animato come la velocità di *refresh*[11] o la quantità e il tipo di elementi geometrici da disegnare sullo schermo; è anche possibile impostare un'immagine GIF predefinita come sfondo animato.

Lo sviluppo del progetto ha richiesto lo studio di un insieme di API per la gestione delle operazioni di disegno sullo schermo e per l'impostazione e personalizzazione dello sfondo, facendo ricorso alle conoscenze apprese nei corsi di ingegneria del software per la gestione dei thread deputati alla gestione delle animazioni.

## 2.3 Progetto finale

Essendo il progetto finale incentrato sulla realizzazione di un'applicazione scelta liberamente, è stata mia intenzione progettare e sviluppare un semplice **videogioco 2D** ispirato ad uno dei più famosi videogiochi della storia: *Pac-Man*!

In modo analogo alla versione prodotta dalla *Namco*, il giocatore, evitando il contatto con i fantasmi che infestano l'area di gioco, deve accumulare il maggior numero possibile di punti mediante l'uccisione dei fantasmi o la raccolta della frutta, nel rispetto delle meccaniche viste nel videogioco originale. La vera differenza consiste nell'assenza del labirinto e nella possibilità da parte del giocatore di muoversi liberamente nell'area di gioco in tutte le direzioni. Il giocatore, minacciato dall'arrivo di un'orda di fantasmi che si muovono a velocità e da posizioni completamente casuali, deve semplicemente sopravvivere il più a lungo possibile.

Il motivo per cui ho scelto di sviluppare tale progetto risiedeva nella volontà di voler comprendere le dinamiche alla base della realizzazione di un semplice videogioco sfruttando tutte le conoscenze accumulate durante tutto il corso di studi; è importante sottolineare che la progettazione ed implementazione del videogioco è avvenuta sfruttando esclusivamente le librerie messe a disposizione dall'*Android SDK*, ovvero senza usufruire di alcuno dei numerosi e potenti strumenti di sviluppo offerti da uno dei qualsiasi motori grafici oggi esistente i quali, benché avrebbero garantito risultati migliori, mi avrebbero impedito di comprendere a fondo la logica, intesa da un punto di vista architetturale, di un videogioco.

A dispetto dell'enorme semplicità del videogioco, lo sviluppo fu molto complesso poiché ha richiesto la risoluzione di una lunga serie di problematiche di natura molto diversa come:

- Riuscire a progettare ed implementare un efficiente algoritmo per **rilevare le collisioni** fra gli elementi del gioco.[12]
- **Progettare e gestire le animazioni** del background dell'area di gioco e dei vari oggetti che popolano lo schermo come il giocatore stesso o i

[11] Si definisce **frequenza d'aggiornamento** (a volte semitradotto dall'inglese *velocità di refresh*), il numero di volte in un secondo in cui viene ridisegnata l'immagine su un display.

[12] Per **rilevazione di una collisione** (in inglese *collision detection*) s'intende un problema computazionale che consiste nel rilevare l'intersezione fra due o più oggetti.

fantasmi tenendo presente che le animazioni differiscono in base allo stato del giocatore (se vivente, morente, ecc.)

- **Progettare e gestire le operazioni di disegno** dei vari elementi di gioco sullo schermo (background, HUD[13], oggetti) tenendo in considerazione il problema delle differenti dimensioni e risoluzioni degli schermi dei dispositivi mobili.
- **Gestire la musica e gli effetti sonori** prodotti durante la sessione video-ludica.
- **Progettare, scrivere e testare la logica di gioco.**

Ovviamente, oltre alle conoscenze acquisite durante il corso, al fine di ottenere un buon prodotto, fu necessario ricorrere ai paradigmi, alle metodologie e alle tecniche apprese durante il corso di *"Ingegneria del software e Progettazione Web"*: per esempio, oltre all'organizzazione degli classi, che ha richiesto il ricorso ad un considerevole numero di pattern, fu cruciale una buona gestione di un certo numero thread (ognuno dei quali adibito ad un compito ben preciso come la gestione delle animazioni, la rilevazione delle collisioni, il disegno ecc.) che fu necessario coordinare in modo opportuno al fine di evitare *race condition* e aumentare le prestazioni complessive dell'applicazione.

[13]Per **HUD**, dall'inglese *heads-up display*, s'intende l'insieme delle informazioni costantemente visibili durante il gioco in sovrimpressione, come punti, tempo di gioco, ecc.

# 3 Ingegneria di Internet e del Web

Il corso di *"Ingegneria di Internet e del Web"* tenuto dal prof. *Francesco Lo Presti* è focalizzato nella descrizione delle metodologie e dei principi architetturali per la progettazione di reti di calcolatori, con particolare enfasi ai protocolli del livello applicativo, di trasporto e di rete, fornendo un insieme di conoscenze di base per lo sviluppo di applicazioni di rete.

## 3.1 Descrizione generale del progetto

Il progetto ha riguardato lo sviluppo in linguaggio di programmazione **C**, usando l'API del socket di Berkeley, di un'**applicazione di tipo client-server** per il trasferimento di dati in modo affidabile tra due o più host collegati in una rete che permetta una serie di funzionalità come il download, l'upload e la visualizzione dei file presenti sul server.

Le specifiche di progetto prevedevano che il trasferimento dei dati tra host avvenisse attraverso l'uso del protocollo di trasporto **UDP** il quale, come noto, offre un modello di servizio di tipo **best-effort** in accordo alla quale non viene garantita né la consegna dei pacchetti né il rispetto dell'ordine originario e non garantisce neppure l'integrità dei dati all'interno dei pacchetti trasmessi[14].

L'uso di un protocollo non affidabile e senza connessione come UDP ha richiesto, al fine di garantire una trasmissione affidabile dei dati, l'implementazione, a livello applicativo, di **un protocollo a ripetizione selettiva** (**SR**, *selective-repeat protocol*) che eviti le ritrasmissioni non necessarie facendo ritrasmettere al mittente solo quei pacchetti su cui esistono sospetti di errore (ossia, smarrimento o alterazione).[15].

Per simulare in modo adeguato la natura inaffidabile della rete Internet e studiare le prestazioni del protocollo sviluppato quando avviene la perdita di uno o più pacchetti, è stato implementato un meccanismo in accordo alla quale i pacchetti trasmessi, soggetti ad un certo valore di *probabilità di perdita* configurabile dall'utente, potessero essere automaticamente scartati dal sistema benché correttamente ricevuti.

## 3.2 Problematiche connesse allo sviluppo del protocollo di rete

In base a quanto detto lo scopo del progetto consisteva nel progettare, implementare e testare un proprio **protocollo di rete**, operante a livello applicativo, capace, nel rispetto dei requisiti del progetto, di garantire una comunicazione affidabile tra host simulando la naturale inaffidabilità delle odierne reti di calcolatori. Come noto, progettare un protocollo di rete ha richiesto una descrizione dettagliata e completa delle modalità di interazione tra calcolatori in una rete, attraverso *la definizione di tutte le regole e i meccanismi che dominano l'intero processo di comunicazione.*

La risoluzione di un'ampia gamma di problemi riscontrati durante lo sviluppo è stata facilitata dallo studio dei vari protocolli di rete avvenuto durante il corso

[14]Cfr. Andrew S. Tanenbaum & David J. Wetherall - *Reti di calcolatori 5/Ed*, Pearson, pp. 179

[15]Cfr. *ivi*, pp. 212

poiché molti dei meccanismi, delle soluzioni e delle tecniche studiate sono state adottate nel progetto.

Per esempio uno dei problemi più importanti e complessi era riuscire a rilevare e a gestire lo **smarrimento dei pacchetti**, un evento non raro sulle odierne reti di calcolatori, e decidere cosa fare quando ciò avviene. Adottando la soluzione prevista in molti protocolli di rete, se dopo la trasmissione di un certo pacchetto il mittente non ricevesse alcun riscontro da parte del destinatario dopo un certo lasso di tempo, quest'ultimo deve provvedere a ritrasmettere il pacchetto smarrito. A tale scopo è stato pertanto necessario implementare un meccanismo di ritrasmissione basato sul tempo capace di segnalare al mittente l'avvenuta scadenza di un dato lasso di tempo, progettando il mittente in modo tale da essere in grado di rispondere adeguatamente all'avvenuto timeout provvedendo a ritrasmettere il pacchetto smarrito.

Questo approccio introduce, come noto, la presenza di pacchetti duplicati nel canale di trasmissione[16]. Dal momento che il destinatario non può sapere a priori se un pacchetto in arrivo contenga dati nuovi o rappresenti una ritrasmissione, adottando la soluzione usata dal protocollo TCP, è stato necessario utilizzare dei **numeri di sequenza**. Ciò ha richiesto di definire opportune **intestazioni** (o **header**) da apporre ai pacchetti da trasmettere all'interno del quale è stato possibile specificare, oltre ai numeri di sequenze, una serie di informazioni utili a garantire il corretto funzionamento del protocollo SR e delle procedure di instaurazione e chiusura delle varie trasmissioni.

Per lo sviluppo dell'applicazione è stato necessario ricorrere alle conoscenze acquisite durante il corso di *"Sistemi Operativi"* e studiare un insieme di nuove API UNIX per poter gestire adeguatamente un insieme di componenti software come i *socket* o i *timer UNIX*.

[16] Cfr. *ivi*, pp. 202

# 4 Ingegneria degli Algoritmi

Il corso di *"Ingegneria degli Algoritmi"* tenuto dal prof. *Giuseppe Italiano* è incentrato sullo studio di una serie di nozioni per progettare, analizzare, implementare ed ingegnerizzare algoritmi e strutture dati, usando **Python** come linguaggio di riferimento.

## 4.1 Descrizione generale dei progetti

Il corso prevedeva due progetti, uno intermedio ed uno finale, che richiedevano entrambi la progettazione e l'analisi di algoritmi o strutture dati idonei a risolvere un qualche problema di interesse applicativo.

In particolare, il progetto intermedio era focalizzato sulla progettazione ed implementazione di una speciale *variante* di un **albero binario di ricerca di tipo AVL**. Questa variante, pur mantenendo inalterate le operazioni di inserimento e ricerca dei nodi, adotta un approccio denominato *lazy* per le operazioni di eliminazione dei nodi in accordo alla quale, invece di rimuovere fisicamente un nodo, procedura che potrebbe comportare costi aggiuntivi per mantenere il bilanciamento dell'albero, l'eliminazione del nodo avviene semplicemente contrassegnando il nodo stesso come *eliminato*. Il progetto finale era focalizzato invece sulla progettazione ed implementazione di un **grafo orientato aciclico** per risolvere un insieme di problemi applicativi sfruttando una serie di nozioni apprese durante il corso riguardante le forme di rappresentazione dei grafi nonché di vari algoritmi per eseguire efficientemente operazioni di visita e calcolo di cammini minimi.

Durante le fasi di progettazione ed analisi degli algoritmi elaborati nei due progetti, effettuata secondo le modalità e le tecniche studiate durante il corso, fu molto importante comprendere i pregi e i difetti delle strutture dati sviluppate in base al contesto di utilizzo attraverso **l'analisi del costo in termini computazionali degli algoritmi stessi**, scegliendo di volta in volta la migliore implementazione per risolvere efficientemente un certo problema applicativo.

Per esempio l'albero binario di ricerca sviluppato durante il progetto intermedio risultava molto efficiente durante le operazioni di cancellazione, ricerca e inserimento in quei scenari d'utilizzo dove avvengono molti reinserimenti capaci di rimpiazzare nodi precedentemente eliminati, mostrando invece un degradamento delle prestazioni in altri scenari di utilizzo. Un altro esempio riguarda lo sviluppo del progetto finale dove, per risolvere il problema applicativo proposto, fu necessario, per il corretto funzionamento dell'algoritmo sviluppato, privilegiare una forma di rappresentazione del grafo che fosse più adatta per una ricerca più efficiente del successore di un nodo.

Sono state effettuate inoltre una serie di **analisi sperimentali** volte a misurare il tempo di esecuzione di una serie di test per osservare il comportamento degli algoritmi sviluppati e confermare sperimentalmente le analisi teoriche. Per esempio, durante lo sviluppo del progetto intermedio, si è potuto facilmente osservare dai dati sperimentali il degrado delle prestazioni della struttura dati a causa della crescita senza controllo del numero di nodi contrassegnati come eliminati, eventualità possibile qualora il numero di inserimenti in grado di sostituire i suddetti nodi sia insufficiente.

# 5 Basi Di Dati E Conoscenza

Il corso di *"Basi Di Dati E Conoscenza"* tenuto dal professoressa *Vittoria De Nitto Personé* è incentrato sullo studio di modelli, metodi e sistemi per la definizione, progettazione e realizzazione di sistemi software che gestiscano insiemi di dati di grandi dimensioni.

## 5.1 Descrizione generale del progetto

Il progetto del corso prevedeva lo sviluppo di un'applicazione software per **l'interrogazione di una base dati** di grandi dimensioni ospitante informazioni a carattere scientifico ed aerospaziale come sistemi satellitari, strumentazioni, agenzie aerospaziali, galassie, mappe e sistemi stellari.

In accordo alle tecniche e alle metodologie apprese durante il corso, lo sviluppo del sistema software ha richiesto lo svolgimento di una serie di attività tra cui:

- Un'attività di **analisi dei requisiti** al fine di individuare e definire in modo informale i dati e le relazioni di interesse applicativo.
- Un'attività di **progettazione concettuale** il cui scopo è quello di rappresentare le specifiche informali della realtà d'interesse in termini di una descrizione formale e completa, ma indipendente dai criteri di rappresentazione utilizzati nei sistemi di gestione della base dati; il prodotto di questa fase è chiamato **modello concettuale**.[17] In particolare fu elaborato un modello concettuale noto come **modello entità-relazione** (anche detto **modello E-R**) il cui scopo è quello di fornire una rappresentazione grafica facile da comprendere dei dati e delle loro relazioni attraverso un insieme di costrutti: le *entità*, le *associazioni* e gli *attributi*.[18] Dal momento che una parte delle proprietà delle entità e delle relazioni della base dati non fu rappresentabile direttamente all'interno dei modelli concettuali usati durante la progettazione, è stato necessario produrre un documento contenente le cosiddette **bussines rules** per meglio definire alcuni aspetti applicativi.
- Un'attività di **progettazione logica** il cui scopo è quello di fornire una traduzione dello schema concettuale in termini del modello di rappresentazione dei dati adottato dal sistema di gestione della basi dati utilizzata producendo il cosiddetto **schema logico**. Dal momento che il modello di rappresentazione dei dati adottato è di tipo relazionale, lo schema logico prodotto rappresentava la traduzione dello schema concettuale in un insieme di *tabelle*, opportunamente correlate fra di loro.[19]
- L'attività di **implementazione** del sistema software in linguaggio di programmazione **JAVA** sfruttando **PostgreSQL** come **DBMS** ricorrendo ai paradigmi, alle metodologie e alle tecniche apprese durante il corso di ingegneria del software per tutte le questioni relative alla progettazione e organizzazione degli oggetti.

[17] Cfr. Paolo Atzeni, Stefano Ceri, Piero Fraternali, Stefano Paraboshi, Riccardo Torlone - *Basi di dati, Modelli e linguaggi di interrogazione*, Quarta edizione, McGraw-Hill pp. 194

[18] Cfr. *ivi*, pp. 196-204

[19] Cfr. *ivi*, pp. 194

Ovviamente molte delle scelte progettuali adottate hanno richiesto l'analisi di due parametri che generalmente regolano le prestazioni di un sistema software: il *costo di una operazione* e l'*occupazione di memoria*.[20] Per questo motivo, sono state effettuate diverse operazione di **ristrutturazione del modello concettuale** al fine di privilegiare le operazioni di interesse applicativo ottimizzando ove possibile l'occupazione di memoria. Inoltre fu cruciale garantire l'integrità dei dati memorizzati nella base dati, prevenendo la presenza di ridondanza e incoerenza in seguito alle operazioni di inserimento, aggiornamento, e cancellazione; in altri termini si è fatto ricorso ad un procedimento noto come **normalizzazione** la quale ha certificato la qualità dello schema prodotto.

[20]Cfr. *ivi*, pp. 279