

DODO.NET

SDD – System Design Document

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 2.0 |
| Data | 15/11/2021 |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Alfonso Cuomo  Simone Farina  Francesco Della Monica |
| Approvato da |  |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 15/11/2021 | 1.0 | Stesura introduzione | [tutti] |
| ?/?/2021 | 1.1 | Stesura obiettivi del sistema,  Design Goals e Trade-off | [tutti] |
| 9/01/2021 | 1.2 | Stesura dell’architettura di sistemi simili e di sistema proposto con la decomposizione in sottosistemi e diagramma di deployment | [tutti] |
| 10/01/2021 | 1.3 | Aggiunti il mapping h/s, la gestione dei dati persistenti e il controllo degli accessi,sicurezza e flusso globale del sistema | [tutti] |
| 19/01/2021 | 1.4 | Aggiunti condizioni limite, start-up, terminazione e fallimenti e il servizio del sottosistema | [tutti] |
| 04/08/2021 | 1.5 | Aggiornamento diagrammi | [tutti] |
| 05/08/2021 | 2.0 | Revisione finale | [tutti] |

Sommario

[**1. Introduzione**](#_heading=h.dt75p9h78b) **4**

[1.1 Obiettivi del sistema](#_heading=h.5nnx4bpkxp2f) 4

[1.2 Design Goals](#_heading=h.l6nsikbn76ev) 4

[1.2.1 Design Trade-off](#_heading=h.1b5hv1sgr6dn) 4

[1.3 Definizioni, acronimi e abbreviazioni](#_heading=h.6bvbggccxrtd) 5

[1.4 Riferimenti](#_heading=h.kqwliwblzkaj) 5

[1.5 Panoramica](#_heading=h.q2yqswuepxwy) 5

[**2. Architettura dei sistemi simili**](#_heading=h.nau31dr1t5jg) **5**

[**3. Architettura del sistema proposto**](#_heading=h.fywbhaycd3ha) **5**

[3.1 Panoramica](#_heading=h.px2hyyj1fbzx) 5

[3.2 Decomposizione in sottosistemi](#_heading=h.4c1luntn1ye8) 5

[3.2.1 Diagramma di Deployment](#_heading=h.1spikhwy5b3k) 7

[3.3 Mapping hardware/software](#_heading=h.w11dx6ojhudt) 7

[3.4 Gestione dati persistenti](#_heading=h.7eyz3lnh7sj0) 7

[3.5 Controllo degli accessi e sicurezza](#_heading=h.x9g0kcf1bfnj) 8

[3.6 Controllo flusso globale del sistema](#_heading=h.xzwajiomwdtz) 8

[3.7 Condizioni limite](#_heading=h.lav5py4gmrcs) 8

[3.7.1 Start-up](#_heading=h.2xckgwtrxxdo) 8

[3.7.2 Terminazione](#_heading=h.3a0ve8th1enq) 8

[3.7.3 Fallimento](#_heading=h.q98kj7u9xb29) 8

[**4. Servizi dei Sottosistemi**](#_heading=h.up5mcyggscps) **9**

[**5. Glossario**](#_heading=h.5ru40tciv2c2) **10**

# 1. Introduzione

## 1.1 Obiettivi del sistema

Il sistema proposto vuole fornire una piattaforma web proiettata alla gestione dell’inventario dei committenti (ossia la libreria *Dodo*) e accondiscendere alle esigenze dei clienti a cui essi si rivolgono. Per fare ciò il sistema supporterà un’area per l’amministratore del sito di tipo e-commerce (con le funzionalità di gestione dei prodotti) e l’interfaccia di uno store classico in cui poter visualizzare e acquistare i prodotti forniti.

Riassumendo dunque le funzionalità del sito possono essere raggruppate in tre categorie principali:

* *Gestione account*: area che si occupa della gestione dei dati personali, l’autenticazione e delle interazioni con il catalogo (l’utente può infatti visualizzare i vari prodotti e acquistarli).
* *Gestione store:* in questo particolare categoria ci si occupa della rimozione e dell’aggiunta dei prodotti allo store nel quale si procederà ad una categorizzazione per genere.
* *Gestione acquisto:* in questa sezione si intende gestire la procedura d’acquisto dei prodotti scelti, la quale avviene in maniera sicura e trasparente per l’utente.

Il catalogo della libreria sarà quindi salvato in maniera persistente in un database relazionale che terrà traccia anche delle altre informazioni derivanti dalle attività effettuate sul sistema.

Le varie funzionalità fornite saranno accessibili dopo un processo di autenticazione tramite login di e-mail e password. Ovviamente, questo sarà possibile solo dopo una registrazione dei propri dati presso il sistema.

## 1.2 Design Goals

I Design Goals sono organizzati in cinque categorie: Performance, Dependability, Cost, Maintenance, End User Criteria. I Design Goals identificati nel nostro sistema sono i seguenti:

* *Criteri di performance*
  + *Tempo di risposta:* il sistema dovrà essere fluido e pienamente interattivo per gli utenti finali, per questo i tempi di risposta dovranno essere minimi
  + *Throughput:* il sistema deve permettere la fruizione in contemporanea di più utenti
* *Criteri di affidabilità*
  + *Robusto:* il sistema dovrà fornire all’utente l’assistenza necessaria per completare correttamente i moduli richiesti
  + *Disponibile:* il sistema dovrà essere disponibile agli utenti ogni qualvolta ne sia richiesto l’utilizzo
  + *Tolleranza ai guasti:* il sistema dovrà essere in grado di gestire errori sia lato client che lato server, come ad esempio il sovraccarico dei server per le troppe richieste.
  + *Sicurezza:* l’accesso al sistema avviene tramite il login di e-mail e password. Inoltre, gli utenti non potranno compiere accessi illegali a funzioni per cui non sono autorizzati.
* *Criteri di costi*
  + *Sviluppo:* Il costo complessivo di 200 ore per la progettazione e lo sviluppo del sistema (il che risulta in 50 ore per i singoli membri).
* *Criteri di manutenzione*
  + *Portabile:* il sistema sarà fruibile su più browser.
  + *Leggibilità del codice:* il sistema dovrà essere scritto in maniera pulita per consentire successivi aggiornamenti anche da parte di team di sviluppo diversi. Per fare ciò si necessita una documentazione rilevante e non banale dei metodi.
  + *Tracciabilità dei requisiti:* i requisiti saranno tracciabili tramite una matrice di tracciabilità
* *Criteri di usabilità*
  + *Utile:* il sistema intende essere utile per gli utenti finali che potranno velocizzare l’acquisto presso la libreria indipendente che commissiona la piattaforma
  + *Usabile:* il sistema vuole essere usabile da tutti tramite le interfacce intuitive

### 1.2.1 Design Trade-off

* *Prestazioni vs Affidabilità*: In questa prima versione del sistema si è preferito dare priorità ai tempi di risposta del sistema a discapito dell’affidabilità (questo non porterà a problemi di sicurezza in quanto i dati sensibili non saranno salvati sul sistema).
* *Tempo di rilascio vs Funzionalità*: Si è scelto di implementare tutte le funzionalità definite a discapito dei tempi di rilascio.

## 1.3 Definizioni, acronimi e abbreviazioni

* **GUI:** acronimo per Graphical User Interface

## 1.4 Riferimenti

* NC05\_RAD\_V\_2
* Slide del corso presenti sulla piattaforma e-learning
* Object-Oriented Software Engineering Using UML Patterns And Java, Third Edition,   
  Bernd Bruegge & Allen H. Dutoit
* <https://www.ibs.it>

## 1.5 Panoramica

Nel secondo capitolo verrà presentata l’architettura dei sistemi simili a quello proposto nel documento.

Nel terzo si procederà dunque alla presentazione dell’architettura del sistema proposto in cui sarà gestita la decomposizione in sottosistemi, il mapping hardware/software, i dati persistenti, il controllo degli accessi e la sicurezza, il controllo del flusso globale del sistema e le condizioni limite quali start-up, terminazione e fallimento.

Nel quarto capitolo si procederà ad una presentazione dei servizi dei sottosistemi. In calce al documento sarà presente un glossario dei termini tecnici utilizzati.

# 2. Architettura dei sistemi simili

Nel nostro caso non è presente un’architettura software già esistente, pertanto abbiamo analizzato le architetture di sistemi simili. La piattaforma presa in considerazione è IBS, che offre la possibilità di acquistare i libri.

Esso è accessibile tramite sito web raggiungibile con qualsiasi web browser e dispone di una companion app per dispositivi mobile. Ciò che si è evinto dall’analisi è che alla base di questa piattaforma è presente un database per la memorizzazione dei dati persistenti interrogato da applicazioni web progettate tramite diverse tecnologie (tra queste HTML, CSS, jQuery, JavaScript e Java). Deduciamo che sia quindi sviluppata secondo un’architettura a tre strati, ma non siamo in grado di capire se ci sono solo tre livelli oppure differenti tier. Per quanto riguarda il controllo degli accessi, quello che si evince è che il sistema presenta un autenticazione con username e password in modo da rendere disponibili le funzionalità che offre solamente agli utenti autorizzati, mentre non possiamo conoscere i criteri di sicurezza per la gestione dei dati sensibili (cifratura delle credenziali o simili).

# 3. Architettura del sistema proposto

## 3.1 Panoramica

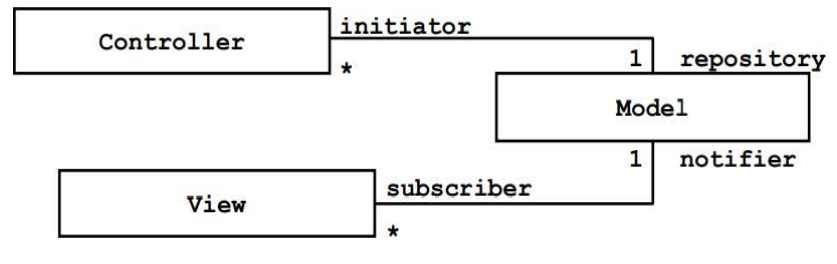
Il sistema da noi proposto consiste in una applicazione web, eseguita in locale per motivi di sicurezza, che memorizza i dati in maniera persistente in un database relazionale.

## 

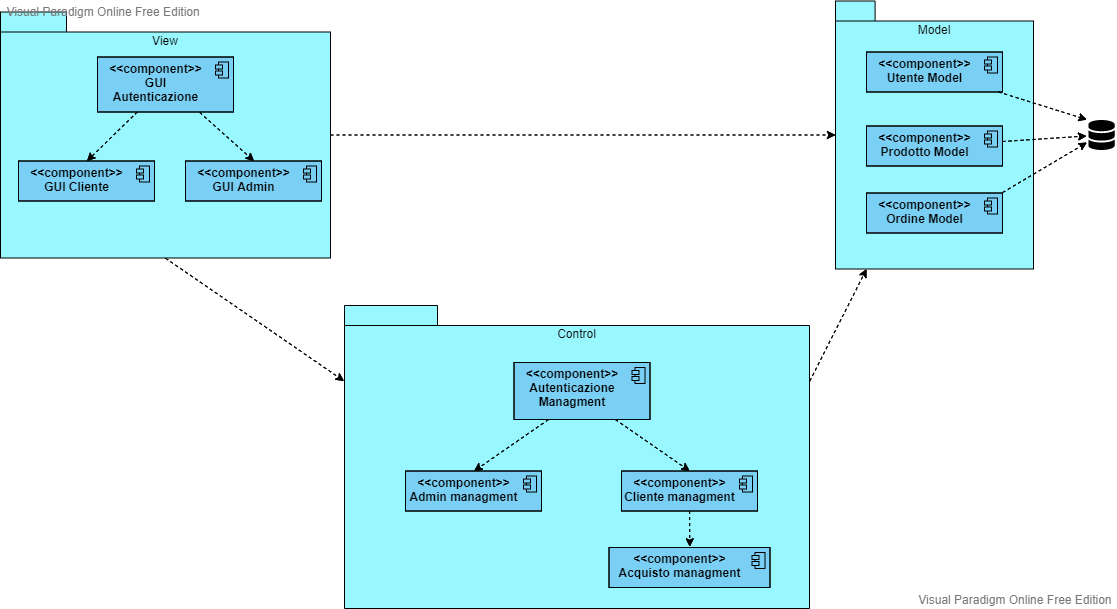
## 3.2 Decomposizione in sottosistemi

Per il sistema si prevede una decomposizione a tre layer, più comunemente descritta dal design pattern MVC, che suddivide le funzionalità come di seguito:

* *Model*: è il layer che si occupa di comunicare col database;
* *View*: è il layer che si occupa di gestire la parte visiva del sistema che permette agli utenti di interagire con esso;
* *Controller*: è il layer che gestisce la logica di business del sistema.



Si è dunque deciso di suddividere il sistema nei seguenti sottosistemi, garantendo un basso accoppiamento ed un’elevata coesione tramite l’utilizzo di un’architettura aperta così da favorirne la modificabilità.



Il livello di View comprende i 3 sottosistemi:

* GUI Autenticazione: Permette la visualizzazione agli utenti non registrati e la loro autenticazione (ossia il login e la registrazione);
* GUI Cliente: Permette la visualizzazione del sistema dal punto di vista del cliente con cui può interagire;
* GUI Admin: Permette la visualizzazione del sistema dal punto di vista dell’admin con cui può interagire.

Il livello di Control comprende i 6 sottosistemi:

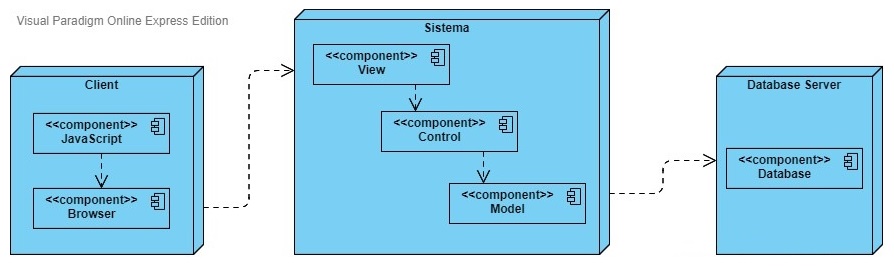
* Autenticazione management: Gestisce la logica di business che permette l’autenticazione e la visualizzazione del sito senza autenticazione;
* Cliente management: Gestisce la logica di business che permette la gestione delle funzionalità del cliente (aggiunta e rimozione dei prodotti dal carrello);
* Admin management: Gestisce la logica di business che permette la gestione delle funzioni dell’admin;
* Acquisto management: Gestisce la logica di business relativa agli ordini;

Il livello di Model comprende il seguente sottosistema:

* Utente Model: gestisce la comunicazione del sistema con la tabella Utente del database.
* Prodotto Model: gestisce la comunicazione del sistema con la tabella Prodotto del database.
* Ordine Model: gestisce la comunicazione del sistema con la tabella Ordine del database.

### 3.2.1 Diagramma di Deployment

L’utente interagisce con il sistema tramite la view , che viene visualizzata dal client con l’ausilio di un programma software capace di visualizzare le pagine web e che permetta l’interpretazione e l’esecuzione di codice JavaScript (browser). Il sistema inoltre interagisce con il database tramite la parte di Model del sistema.

****

## 3.3 Mapping hardware/software

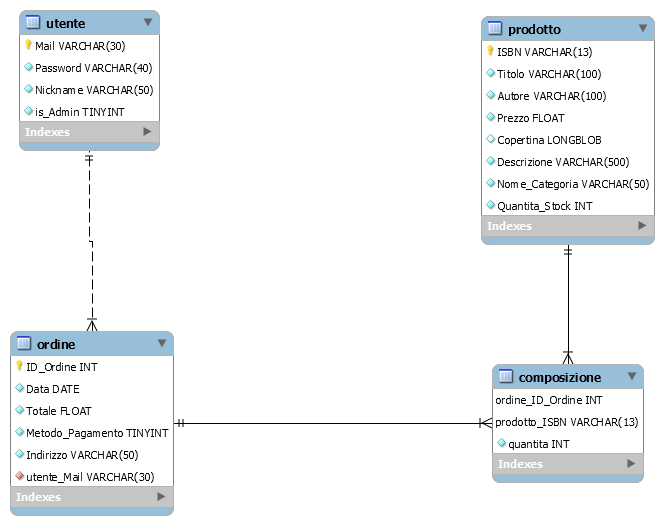
Il sistema si compone di due parti:

* Web-app, in cui saranno presenti i tre layer
* Database, in cui verranno salvati i dati persistenti.

Il sistema necessita di una macchina in grado di supportare Apache Tomcat, al fine di garantire l’operabilità del sito web e MySQL, per garantire l’operabilità del database.

## 3.4 Gestione dati persistenti

Per la gestione dei dati persistenti il sistema si interfaccia con un database relazionale gestito tramite MySQL. La struttura dei dati segue il seguente schema:



## 3.5 Controllo degli accessi e sicurezza

Il controllo degli accessi è garantito tramite l’utilizzo di email e password, salvate nel database, per gli utenti del sistema che si distinguono in clienti e admin che interagiscono secondo le loro singole facoltà per evitare accessi non autorizzati ad informazioni sensibili o a funzionalità non consentite. Ricordiamo che il sistema non prevede un metodo di recupero o di modifica delle password, almeno nella sua prima versione.

Si ricorrerà all’utilizzo della sessione del server per tenere traccia dell’utente loggato e per salvare gli elementi inseriti nel carrello. Per motivi di budget il payment check, ossia il sistema di convalida del metodo di pagamento, non verrà implementato, almeno nella prima versione.

Le operazioni che gli utenti possono effettuare sugli oggetti sono riportate nella tabella che segue:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Utenza | Prodotto | Carrello | Acquisto |
| Utente | Login  Logout | Visualizza |  |  |
| Cliente |  | Acquista | Inserisci prodotto  Rimuovi prodotto | Acquista prodotti |
| Admin |  | Aggiungi libro Rimuovi libro |  |  |

## 

## 3.6 Controllo flusso globale del sistema

Il sistema adotta un controllo del flusso globale di tipo thread-driven poiché il web container (Tomcat) permette l'interazione concorrente tra il sistema e più client tramite l’intercettazione di eventi generati dai browser. Ossia, ogni richiesta da parte di un utente verrà eseguita su un thread dedicato. Questa soluzione permette al sistema di poter rispondere a più utenti contemporaneamente ma richiede più attenzione alla gestione delle sezioni critiche.

## 3.7 Condizioni limite

### 3.7.1 Start-up

Per il primo avvio del sistema lato admin si dovrà avviare il web server Apache Tomcat collegato ad un Database MySQL. I client potranno richiedere dunque l’homepage al server e, in seguito, tramite l’interfaccia di login potranno accedere alle funzionalità richieste.

### 3.7.2 Terminazione

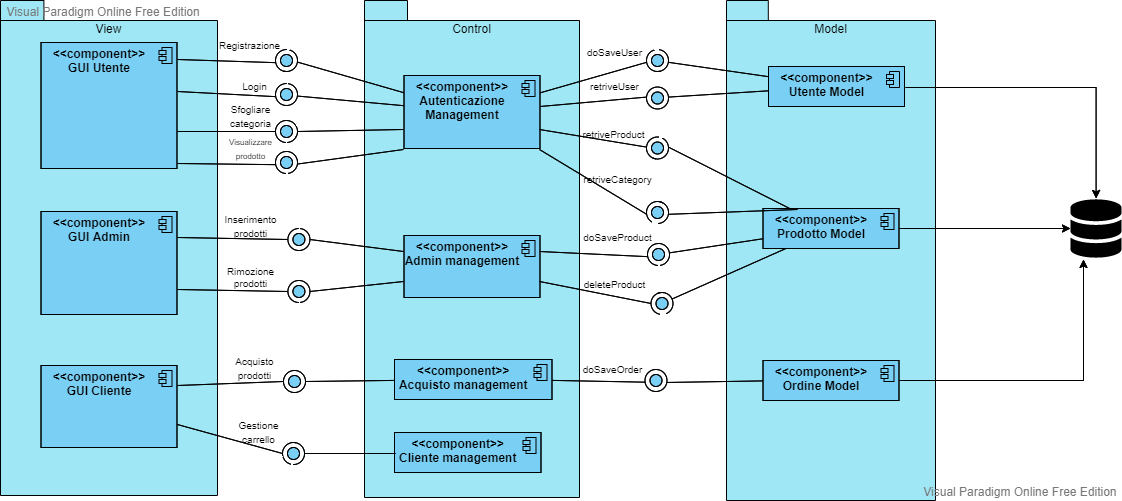
Per la corretta chiusura dell’applicazione, il client potrà effettuare una procedura di log-out. Per terminare il server, l’admin dovrà effettuare la procedura di terminazione del server, dopo la quale nessun client potrà connettersi al sistema.

### 3.7.3 Fallimento

Il sistema può incorrere in vari casi di fallimento, riguardanti sia l'hardware che il software:

1. Sovraccarico o interruzione della fornitura elettrica del server: il sistema non prevede alcuna strategia di backup e ripristino dei dati.
2. Errori di tipo software: il sistema mostra un alert di errore.
3. Errore di tipo hardware: il sistema non prevede alcuna strategia per ovviare a questo tipo di problema.

# 4. Servizi dei Sottosistemi



Autenticazione management offre 2 servizi al layer View:

* Registrazione
* Login

Prodotti management offre 2 servizi al layer View:

* Sfogliare Categoria
* Visualizza Prodotto

Acquisto management offre 2 servizi al layer View:

* Acquisto Prodotti

Cliente management offre 3 servizi al layer View:

* Gestione Carrello

Admin management offre 4 servizi al layer View:

* Inserimento Prodotti
* Rimozione Prodotti

Utente model offre 2 servizi al layer Control:

* doSaveUser (per la registrazione)
* retrieveUser (per il login)

Prodotto model offre 4 servizi al layer Control:

* retrieveProduct
* retrieveCategory
* doSaveProduct
* deleteProduct

Ordine model offre 1 servizio al layer Control:

* doSaveOrder

# 5. Glossario

**Tomcat**: specifico Application Server, ossia sistema software per la gestione delle richieste/risposte provenienti dai client

**Apache:** È il nome di un server web libero sviluppato dalla Apache Software Foundation.

**Server:** Computer di elevate prestazioni che in una rete fornisce un servizio agli altri elaboratori collegati, detti client

**Client:** Programma o parte di un programma che permette di scambiare dati con un server.

**Hardware:** In informatica, l'insieme delle componenti fisiche, non modificabili di un sistema di elaborazione dati

**Software:** l'insieme delle procedure e delle istruzioni in un sistema di elaborazione dati

**Backup:** duplicazione di un file o di un insieme di dati su un supporto esterno al computer, per avere una copia di riserva.

**Alert:** Messaggio di errore generato dal sistema al fine di informare l’utente di un eventuale malfunzionamento

**CSS:** È un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML, ad esempio i siti web e relative pagine web.

**JavaScript:** è un linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi.

**JQuery:** È una libreria JavaScript per applicazioni web, distribuita come software libero, distribuito sotto i termini della Licenza MIT.

**Java:** È un linguaggio di programmazione ad alto livello.

**HTML:** È un linguaggio di markup. Nato per la formattazione e [impaginazione](https://it.wikipedia.org/wiki/Impaginazione) di documenti ipertestuali.

**E-Mail:** Casella di posta elettronica usata per ricevere mail.

**Password:** È una sequenza di [caratteri](https://it.wikipedia.org/wiki/Carattere_(informatica)) [alfanumerici](https://it.wikipedia.org/wiki/Alfanumerico) utilizzata per accedere in modo esclusivo a una risorsa informatica.

**Database:** archivio di dati strutturato in modo da razionalizzare la gestione e l'aggiornamento delle informazioni e da permettere lo svolgimento di ricerche complesse.

**MySQL**: tipo di sistema software per la gestione dei dati persistenti su database

**Throughput:** misura del numero di istruzioni compiute in un'unità di tempo

**Applicazione web:** programma accessibile tramite browser web ed in grado di elaborare richieste e risposte HTTP.

**Companion App:** applicazione mobile complementare o sostitutiva dell’applicazione web.

**E-Commerce:** commercio elettronico, acquisto di beni e servizi attraverso il web tramite server sicuri (di tipo HTTPS e sistemi di crittografia dei dati sensibili) con servizi di pagamento quali carta di credito o simili.

**Web Server**: macchina connessa alla rete dotata di un ambiente di esecuzione.

**Web Browser**: È un'applicazione per l'acquisizione, la presentazione e la navigazione di risorse sul web. Tali risorse (come pagine web, immagini o video) sono messe a disposizione sul World Wide Web (la rete globale che si appoggia su Internet), su una rete locale o sullo stesso computer dove il browser è in esecuzione.

**Start-up**: Fase iniziale dell’avvio del sistema

**RAD:** *Requirement Analysis Document*, documento della fase di analisi dei requisiti.

**Architettura aperta:** si definisce una architettura aperta una architettura scomposta in layer in cui un layer può accedere ai layer di livello più basso.