Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un sistema informativo a supporto delle attività di un concessionario di automezzi

Design and implementation of an information system to support the activities of a Heavy Goods Vehicles dealer

Relatore

Prof. Domenico Ursino

Candidato

Luca Chiavarini

Anno Accademico 2018-2019

Indice

Introduzione 3

1 Introduzione alla realizzazione dei sistemi informativi 7
1.1 I sistemi informativi 7
1.2 Database e DataBase Management System (DBMS) 9
1.3 Linguaggi per DBMS ed SQL 11
1.4 Software per lo sviluppo dei database: MySQL 12

2 Il contesto di riferimento e analisi dei requisiti 17
2.1 Contesto di riferimento 17
2.2 Analisi dei requisiti 18
2.2.1 Requisiti funzionali e diagrammi dei casi d'uso 19
2.2.2 Requisiti non funzionali 23

3 Progettazione del sistema 25
3.1 Progettazione della componente dati 25
3.1.1 Progettazione concettuale 26
3.1.2 Progettazione logica 30
3.1.3 Progettazione fisica 43
3.2 Progettazione della componente applicativa 45
3.2.1 Definizione dell'architettura del sistema 45
3.2.2 Definizione dei mock-up 48
3.2.3 Progettazione dei Process Flow 59

4 Implementazione del sistema 65
4.1 Implementazione della base di dati 65
4.2 Implementazione del sito Web 69
4.2.1 Sezione "public" del sito 76
4.2.2 Sezione "user" del sito 81
4.2.3 Sezione "staff" del sito 85
4.2.4 Sezione "admin" del sito 95

5 Discussione in merito al sistema realizzato 97
5.1 Punti di forza del sistema realizzato 97
5.2 Punti di debolezza del sistema realizzato 98
5.3 Lezioni apprese 98

6 Conclusioni e sguardo al futuro 101

Riferimenti bibliografici 103

Elenco delle figure

1.1 Classica rappresentazione di una base di dati; a sinistra viene simboleggiato il salvataggio dei dati mentre a destra viene rappresentato il loro recupero 10
1.2 Rappresentazione di due esempi che differiscono per la struttura fisica (ma non per quella logica) 10
1.3 Architettura di MySQL 15

2.1 Rappresentazione di un generico caso d'uso 21
2.2 Diagramma dei casi d'uso relativi all'attore "Amministratore" 21
2.3 Diagramma dei casi d'uso relativi all'attore "Dipendente" 22
2.4 Diagramma dei casi d'uso relativi all'attore "Cliente" 22
2.5 Classificazione gerarchica dei requisiti non funzionali 23

3.1 Diagramma E/R 27
3.2 Dizionario delle Entità per lo schema E/R di Figura 3.1 28
3.3 Dizionario delle Relazioni per lo schema E/R di Figura 3.1 29
3.4 Vincoli relativi allo schema E/R di Figura 3.1 30
3.5 Ristrutturazione della porzione dello schema E/R relativa ai vincoli . . 31
3.6 Schema E/R ristrutturato 32
3.7 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "contatta" 32
3.8 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "prenota" 33
3.9 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "effettua (in entrata)" 34
3.10 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "effettua (in uscita)" 35
3.11 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "relativo a (materia prima)" 36
3.12 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "relativo a (veicolo in entrata)" 37
3.13 Porzione dello schema E/R di Figura 3.1 relativo alla relazione "relativo a (veicolo in uscita)" 38

3.14 Porzione dello schema E/R di Figura 3.1 relativo alla relazione “è (veicolo in entrata)” 39
3.15 Porzione dello schema E/R di Figura 3.1 relativo alla relazione “è (veicolo in uscita)” 40
3.16 Porzione dello schema E/R di Figura 3.1 relativo alla relazione “fornisce (veicolo)” 41
3.17 Porzione dello schema E/R di Figura 3.1 relativo alla relazione “fornisce (materia prima)” 42
3.18 Elenco degli indici primari relativi ad ogni tabella 44
3.19 Schema funzionale che rappresenta la mappa dell’applicazione web . . . 46
3.20 Schema dei link relativo alla sezione “pubblica” dell’applicazione web 46
3.21 Schema dei link relativo alla sezione “utente” dell’applicazione web . . 47
3.22 Schema dei link relativo alla sezione “staff” dell’applicazione web 47
3.23 Schema dei link relativo alla restante sezione “staff” dell’applicazione web (non specificata nella figura precedente) 48
3.24 Schema dei link relativo alla sezione “admin” dell’applicazione web . . 48
3.25 Mock-up relativo alla home page della sezione “public” del sito 49
3.26 Mock-up relativo alla pagina “catalog” della sezione “public”; in questa pagina è possibile visualizzare i veicoli e/o filtrarli secondo molteplici parametri 50
3.27 Mock-up relativo alla pagina “where” della sezione “public”; in questa pagina troviamo tutte le informazioni relative al luogo in cui è dislocata l’azienda 50
3.28 Mock-up relativo alla pagina “who” della sezione “public”; in questa pagina troviamo tutte le informazioni relative alla storia, all’organizzazione e all’obiettivo dell’azienda 51
3.29 Mock-up relativo alla pagina “contactus” della sezione “public”; in questa pagina troviamo tutti i contatti dell’azienda 51
3.30 Mock-up relativo alla pagina “signin” della sezione “public”; questa pagina consente ad un utente di registrarsi al sito 52
3.31 Mock-up relativo alla pagina “login” della sezione “public”; questa pagina consente ad un utente di accedere al sito 52
3.32 Mock-up relativo alla home page della sezione “user” del sito 53
3.33 Mock-up relativo alla pagina “catalog” della sezione “user”; da questa pagina il cliente, oltre alle funzionalità già descritte per il livello di utenza “public”, può prenotare la prova gratuita del veicolo selezionato 53
3.34 Mock-up relativo alla pagina “profile” della sezione “user”; da questa pagina l’utente, dopo essersi autenticato, può visualizzare e modificare le proprie informazioni personali. In essa, inoltre, è possibile visualizzare il riepilogo delle prenotazioni effettuate 54
3.35 Mock-up relativo alla home page della sezione “staff” del sito 54
3.36 Mock-up relativo alla pagina “catalog” della sezione “staff”; tramite questa pagina un componente dello staff può aggiungere, modificare o eliminare un veicolo 55

3.37 Mock-up relativo alla pagina “users” della sezione “staff”; tramite questa pagina un componente dello staff ha la possibilità di visualizzare, aggiungere, modificare o eliminare clienti iscritti al sito . 55
3.38 Mock-up relativo alla pagina “personale” della sezione “staff”; questa pagina contiene le informazioni relative a tutti i componenti dello staff (quindi di tutto il personale dell’azienda). Tramite essa è possibile, inoltre, aggiungere, modificare o eliminare un componente dello staff 56
3.39 Mock-up relativo alla pagina “fstaff” della sezione “staff”; questa pagina racchiude i link per le funzionalità più importanti di un utente di livello “staff” 56
3.40 Mock-up relativo alla pagina “listreservation” della sezione “staff”; questa pagina contiene la lista di tutte le prenotazioni e, per ciascuna di essa, vengono mostrate le informazioni sul cliente che l’ha effettuata, sul veicolo prenotato e sulla prenotazione ricevuta 57
3.41 Mock-up relativo alla pagina “orderplaced” della sezione “staff”; questa pagina contiene la lista di tutti gli ordini effettuati dall’azienda e dà la possibilità di aggiungere, modificare o eliminare un ordine 57
3.42 Mock-up relativo alla pagina “orderreceived” della sezione “staff”; questa pagina contiene la lista di tutti gli ordini ricevuti dall’azienda (che, quindi, sono stati effettuati dal cliente) e dà la possibilità di aggiungere, modificare o eliminare un ordine 58
3.43 Mock-up relativo alla pagina “supplierslist” della sezione “staff”; questa pagina contiene la lista di tutti i fornitori da cui l’azienda acquista le materie prime utili all’allestimento degli automezzi. Da qui, inoltre, è possibile aggiungere, modificare o eliminare un fornitore 58
3.44 Mock-up relativo alla pagina “dealershiplist” della sezione “staff”; questa pagina contiene la lista di tutte le concessionarie e di tutte le case produttrici da cui l’azienda acquista i veicoli (nuovi o usati). Da qui, inoltre, è possibile aggiungere, modificare o eliminare una concessionaria o una casa produttrice 59
3.45 Process Flow relativo all’attività di ricerca di un veicolo da parte di un utente appartenente a qualsiasi livello di utenza 60
3.46 Process Flow relativo all’attività di registrazione da parte di un nuovo cliente 61
3.47 Process Flow relativo all’attività di login di un utente 61
3.48 Process Flow relativo all’attività di prenotazione della prova gratuita di un veicolo da parte di un cliente già registrato 62
3.49 Process Flow relativo alle attività di creazione, modifica e cancellazione di un veicolo da parte di un componente dello staff 62
3.50 Process Flow relativo alle attività di creazione, modifica e cancellazione di un utente da parte di un componente dello staff 63
3.51 Process Flow relativo alle attività di creazione, modifica e cancellazione di un fornitore da parte di un componente dello staff . . . 63

3.52 Process Flow relativo alle attività di creazione, modifica e cancellazione di una concessionaria da parte di un componente dello staff 64
3.53 Process Flow relativo alle attività di creazione, modifica e cancellazione di un ordine in entrata (o in uscita), da parte di un componente dello staff 64

4.1 Architettura del pattern MVC 70
4.2 Rappresentazione del funzionamento dell'ORM di Zend Framework . . 70
4.3 Schermata relativa alla home page della sezione "public" del sito 78
4.4 Schermata relativa alla pagina "where" della sezione "public"; in questa pagina troviamo tutte le informazioni relative al luogo in cui è dislocata l'azienda 78
4.5 Schermata relativa alla pagina "who" della sezione "public"; in questa pagina troviamo tutte le informazioni relative alla storia, all'organizzazione e all'obiettivo dell'azienda 79
4.6 Schermata relativa alla pagina "contactus" della sezione "public"; in questa pagina troviamo tutti i contatti dell'azienda 79
4.7 Schermata relativa alla pagina "catalog" della sezione "public"; in questa pagina è possibile visualizzare i veicoli e/o filtrarli secondo molteplici parametri 80
4.8 Schermata relativa alla pagina "signin" della sezione "public"; questa pagina consente ad un utente di registrarsi al sito 81
4.9 Schermata relativa alla pagina "login" della sezione "public"; questa pagina consente ad un utente di accedere al sito 81
4.10 Schermata relativa alla home page della sezione "user" del sito 83
4.11 Schermata relativa alla pagina "catalog" della sezione "user"; da questa pagina il cliente, oltre alle funzionalità già descritte per il livello di utenza "public", può prenotare la prova gratuita del veicolo selezionato 84
4.12 Schermata relativa alla pagina "profile" della sezione "user"; da questa pagina l'utente, dopo essersi autenticato, può visualizzare e modificare le proprie informazioni personali. In essa, inoltre, è possibile visualizzare il riepilogo delle prenotazioni effettuate 85
4.13 Schermata relativa alla home page della sezione "staff" del sito 90
4.14 Schermata relativa alla pagina "catalog" della sezione "staff"; tramite questa pagina un componente dello staff può aggiungere, modificare o eliminare un veicolo 91
4.15 Schermata relativa alla pagina "users" della sezione "staff"; tramite questa pagina un componente dello staff ha la possibilità di visualizzare, aggiungere, modificare o eliminare clienti iscritti al sito . 92
4.16 Schermata relativa alla pagina "personale" della sezione "staff"; questa pagina contiene le informazioni relative a tutti i componenti dello staff (quindi di tutto il personale dell'azienda). Essa, inoltre, consente di aggiungere, modificare o eliminare un componente dello staff 92

4.17 Schermata relativa alla pagina “fstaff” della sezione “staff”; questa pagina racchiude i link per le funzionalità più importanti di un utente di livello “staff” 93
4.18 Schermata relativa alla pagina “listreservation” della sezione “staff”; questa pagina contiene la lista di tutte le prenotazioni e, per ciascuna di essa, vengono mostrate le informazioni sul cliente che l’ha effettuata, sul veicolo prenotato e sulla prenotazione ricevuta 93
4.19 Schermata relativa alla pagina “orderplaced” della sezione “staff”; questa pagina contiene la lista di tutti gli ordini effettuati dall’azienda e dà la possibilità di aggiungere, modificare o eliminare un ordine 94
4.20 Schermata relativa alla pagina “orderreceived” della sezione “staff”; questa pagina contiene la lista di tutti gli ordini ricevuti dall’azienda (che, quindi, sono stati effettuati dal cliente) e dà la possibilità di aggiungere, modificare o eliminare un ordine 94
4.21 Schermata relativa alla pagina “supplierslist” della sezione “staff”; questa pagina contiene la lista di tutti i fornitori da cui l’azienda acquista le materie prime utili all’allestimento degli automezzi. Da qui, inoltre, è possibile aggiungere, modificare o eliminare un fornitore 95
4.22 Schermata relativa alla pagina “dealershiplist” della sezione “staff”; questa pagina contiene la lista di tutte le concessionarie e di tutte le case produttrici da cui l’azienda acquista i veicoli (nuovi o usati). Da qui, inoltre, è possibile aggiungere, modificare o eliminare una concessionaria o una casa produttrice 95
4.23 Schermata relativa alla home page della sezione “admin” del sito 96
4.24 Schermata relativa alla pagina “users” della sezione “admin”; tramite questa pagina l’amministratore del sito è in grado di creare, modificare od eliminare qualsiasi utente (a prescindere dalla classe di utenza alla quale quest’ultimo appartiene) 96

Elenco dei listati

4.1 Creazione del database 65
4.2 Creazione della tabella PERSONALE 65
4.3 Creazione della tabella UTENTE 65
4.4 Creazione della tabella ORDINE_IN_ENTRATA 66
4.5 Creazione della tabella ORDINE_IN_USCITA 66
4.6 Creazione della tabella MATERIA_PRIMA 66
4.7 Creazione della tabella VEICOLO 66
4.8 Creazione della tabella VEICOLO_IN_ENTRATA 66
4.9 Creazione della tabella VEICOLO_IN_USCITA 66
4.10 Creazione della tabella CASA_PRODUTTRICE_CONCESSIONARIA 67
4.11 Creazione della tabella FORNITORE 67
4.12 Creazione della tabella CONTATTA 67
4.13 Creazione della tabella PRENOTA 67
4.14 Creazione della tabella RELATIVO_A(MAT. PRIMA) 67
4.15 Creazione della tabella RELATIVO_A(VEICOLO_IN_ENTRATA) 67
4.16 Creazione della tabella RELATIVO_A(VEICOLO_IN_USCITA) 68
4.17 Creazione della tabella FORNISCE(VEICOLO) 68
4.18 Creazione della tabella FORNISCE(MAT. PRIMA) 68
4.19 Implementazione del vincolo di unicità sul campo NumeroTelaio della tabella VEICOLO 68
4.20 Implementazione dell'indice sul campo Regione della tabella FORNITORE 68
4.21 Implementazione dell'indice sul campo Regione della tabella CASA PRODUTTRICE O CONCESSIONARIA 68
4.22 Implementazione dell'indice sul campo Nome della tabella MATERIA PRIMA 68
4.23 Implementazione dell'indice sui campi CostoTotale e Data della tabella ORDINE IN ENTRATA 68
4.24 Implementazione dell'indice sui campi CostoTotale e Data della tabella ORDINE IN USCITA 69
4.25 Implementazione dell'indice sul campo Cognome della tabella CLIENTE 69
4.26 Codice PHP contenuto nel file include/connect.php che consente la connessione con il database 71

4.27 Definizione della funzione `_initDbParms()` contenuta nella classe `Bootstrap` 71
4.28 Definizione della classe `Application_Model_DBContext` che estende la classe `Application_Model_Abstract` 72
4.29 Definizione della classe `Application_Resource_Concessionaria` che estende la classe `Zend_Db_Table_Abstract` 73
4.30 Definizione della classe `Application_Resource_Fornitore` che estende la classe `Zend_Db_Table_Abstract` 73
4.31 Definizione della classe `Application_Resource_Ordineinentrata` che estende la classe `Zend_Db_Table_Abstract` 73
4.32 Definizione della classe `Application_Resource_Ordineinuscita` che estende la classe `Zend_Db_Table_Abstract` 74
4.33 Definizione della classe `Application_Resource_Prenotazione` che estende la classe `Zend_Db_Table_Abstract` 74
4.34 Definizione della classe `Application_Resource_Ruoli` che estende la classe `Zend_Db_Table_Abstract` 75
4.35 Definizione della classe `Application_Resource_Utente` che estende la classe `Zend_Db_Table_Abstract` 75
4.36 Definizione della classe `Application_Resource_Veicolo` che estende la classe `Zend_Db_Table_Abstract` 75
4.37 Definizione della classe `PublicController` che estende la classe `Zend_Controller_Action` 76
4.38 Definizione della classe `UserController` che estende la classe `Zend_Controller_Action` 81
4.39 Definizione della classe `StaffController` che estende la classe `Zend_Controller_Action` 85

Introduzione

Il dibattito relativo all'impatto dei sistemi informativi sulla gestione aziendale è iniziato negli anni cinquanta e continua tutt'oggi a destare forte interesse sia nell'ambito accademico della ricerca, sia in quello della pratica aziendale. L'attuale scenario di business, infatti, è caratterizzato da fenomeni come la globalizzazione dei mercati, la gestione del cliente, la velocità decisionale e la comunicazione, i quali richiedono una conoscenza interfunzionale dell'azienda, nonché tempestività nel reperimento e nella distribuzione di informazioni rilevanti. Appare, dunque, evidente quanto l'integrazione informativa rappresenti una condizione necessaria per una corretta comprensione e gestione dei fatti aziendali.

Prima di descrivere cos'è un sistema informativo è doveroso concettualizzare l'informatizzazione come un fenomeno sviluppatosi tra il 1980 e il 1990 che prevedeva l'introduzione e applicazione di sistemi di computer e reti informatiche ad un certo settore o attività economica. Questa nuova tendenza dell'epoca prevedeva (e prevede ancora) l'automatizzazione delle procedure d'ufficio mediante l'uso di computer e software in luogo dei vecchi supporti cartacei e delle macchine da scrivere.

Il sistema informativo, invece, può essere definito come la combinazione di risorse umane, materiali e procedure organizzate per soddisfare le seguenti funzioni:

- raccolta e acquisizione delle informazioni;
- archiviazione e conservazione delle informazioni;
- elaborazione delle informazioni;
- distribuzione e scambio delle informazioni.

Esso consente il corretto funzionamento di un sistema organizzativo di una qualsiasi struttura socio-economica, ovvero un sistema di cui sono dotate ormai tutte le aziende.

Un elemento fondamentale per un moderno sistema informativo è rappresentato dalle basi di dati. Con l'avvento di queste ultime, l'archiviazione elettronica dei dati ha fatto progressi notevoli; esse rivestono un ruolo di primaria importanza nel ciclo di vita di un sistema informativo.

La fase fondamentale di ogni sistema informativo è quella di progettazione, e negli anni si è consolidata una metodologia di progettazione oramai da tutti accettata come standard. Tale metodologia prevede in due sotto-fasi, denominate "progettazione dei dati" e "progettazione delle applicazioni".

Lo scopo di questo progetto di tesi è di analizzare ogni singola sotto-fase della progettazione e renderla funzionale per l'attività di un concessionario di automezzi denominato "Italcamion s.r.l.", per poi implementare un sito web che si interfacci con il sistema informativo implementato. Sviluppando la progettazione del sistema informativo in maniera approfondita si arriveranno ad implementare la base di dati e l'applicazione Web in maniera completa e, allo stesso tempo, funzionale all'obiettivo prefissato.

Il sistema informativo che ne deriverà è stato pensato per sopperire alle mancanze che l'azienda ha in determinate aree; queste ultime rientrano, principalmente, negli ambiti di digitalizzazione dell'azienda, gestione del magazzino e dei veicoli e, soprattutto, marketing digitale. Per questo motivo si è pensato di progettare un sistema informativo che mettesse al primo piano questi aspetti e che, compatibilmente, tenesse in considerazione e continuasse ad essere in linea con le politiche aziendali vigenti. Le due principali tecnologie che verranno prodotte saranno rappresentate dalla base di dati e dell'applicazione Web. La base di dati terrà conto della presenza di vari soggetti, quali personale dell'azienda, clienti e fornitori di varia natura, e di articoli quali materie prime e veicoli. Implementato il database, si passerà alla creazione del sito Web che si interfaccerà con esso e che terrà conto sia delle esigenze dei clienti che di quelle del personale dell'azienda. Infatti, grazie all'implementazione dei vari livelli di utenza con i quali si ha la possibilità di accedere al sito, l'utente avrà a disposizione molteplici funzioni di utilizzazione (nel caso in cui l'utente sia un cliente) e di gestione (nel caso in cui l'utente sia un membro dello staff).

Il sito Web, quindi, metterà a disposizioni diverse funzioni in base alle seguenti classi di utenza:

- Classe di utenza "public", che consente all'utente di visualizzare le informazioni generali dell'azienda, di visualizzare il catalogo dei veicoli e di potersi registrare.
- Classe di utenza "user", che consente all'utente, oltre alle funzionalità descritte in precedenza, di prenotare online il test drive gratuito del veicolo desiderato, di modificare i propri dati personali nella sezione "profilo utente" e di visualizzare il riepilogo delle prenotazioni effettuate.
- Classe di utenza "staff", che consente all'utente, oltre alle funzionalità descritte in precedenza, di effettuare operazioni CRUD su veicoli, clienti, fornitori, concessionari, ordini effettuati e ordini ricevuti e di visualizzare il prospetto delle prenotazioni ricevute.
- Classe di utenza "admin", che consente all'utente, oltre alle funzionalità descritte in precedenza, di effettuare operazioni CRUD su tutti gli utenti, a prescindere dal loro livello.

La presente tesi è strutturata come di seguito specificato:

- Nel Capitolo 1 parleremo dei sistemi informativi, di come si procederà nella loro costruzione e dei software e delle tecnologie che verranno utilizzate per il raggiungimento dell'obiettivo finale.
- Nel Capitolo 2 analizzeremo la fase preliminare della progettazione del nostro sistema informativo, delineando quali sono i requisiti funzionali e non funzionali e introducendo quale sarà il contesto di riferimento.

- Nel Capitolo 3 tratteremo la fase di progettazione del sistema informativo dividendola nelle sue due relative sotto-fasi, ovvero quella della progettazione dei dati e quella della progettazione delle applicazioni.
- Nel Capitolo 4 presenteremo l'implementazione del sistema informativo, trattando separatamente l'implementazione della base di dati e l'implementazione del sito Web.
- Nel Capitolo 5 discuteremo del sistema realizzato descrivendo, con un occhio critico, i punti di forza ed i punti di debolezza del nostro lavoro.
- Nel Capitolo 6 concluderemo il presente progetto di tesi proiettandoci nel futuro e ipotizzando a quali possano essere i punti di miglioramento del sistema realizzato.

1

Introduzione alla realizzazione dei sistemi informativi

In questo capitolo definiremo quali sono gli strumenti e le tecnologie alla base della progettazione e dell'implementazione di un sistema informativo. Verranno quindi descritte le varie fasi del ciclo di vita di un sistema informativo soffermandosi, in particolar modo, sugli step di progettazione ed implementazione del sistema

1.1 I sistemi informativi

Lo studio dei sistemi informativi automatizzati ha portato alla formulazione di un gran numero di proposte metodologiche per la loro progettazione e realizzazione. Tutte le proposte prevedono la definizione di differenti fasi e attività, in quanto uno degli obiettivi principali è quello di strutturare i processi di produzione dei nuovi sistemi informatici e di manutenzione e gestione di quelli in esercizio in maniera tale che essi possano essere controllati, migliorati e sorretti mediante adeguate metodologie, tecniche e strumenti.

Inizialmente si riteneva che lo sviluppo di un sistema potesse essere condotto mediante una approccio sequenziale, cosiddetto a cascata (waterfall), in cui, una volta stabiliti gli obiettivi ed i requisiti, si procede all'analisi del sistema nella sua interezza, quindi alla progettazione, alla realizzazione, ai test e all'avvio operativo. La complessità crescente dei sistemi da realizzare, però, ha reso impraticabile questo approccio. Infatti le attività di analisi e progettazione di un "grande sistema" si presentano di lunghezza e complessità tali da ritardare oltremodo il suo avvio e da comportare rischi di obsolescenza prima che lo stesso entri in produzione (nel tempo trascorso dalla prima definizione dei requisiti alla loro formalizzazione finale i requisiti possono cambiare). Lasciando, quindi, relegato il metodo di sviluppo a cascata ai sistemi di piccole dimensioni, si sono fatti strada i cosiddetti approcci evolutivi, ovvero quegli approcci che fanno uso di prototipi, in cui la totalità delle specifiche viene prima suddivisa in una serie di gruppi e quindi si provvede all'analisi, progettazione e realizzazione di ciascun gruppo uno dopo l'altro, rilasciando via via le funzionalità disponibili ed eventualmente rivedendole sulla base delle prime risultanze. L'utilizzo di questo tipo di approccio è stato reso possibile dal diffondersi di ambienti e strumenti di analisi, progettazione e produzione del software flessibili, ovvero capaci di accogliere, con relativa facilità, modifiche a quanto già sviluppato.

Un elenco delle fasi e/o delle attività che possono essere considerate nell'arco dell'intero ciclo di vita di un sistema informativo aziendale è il seguente:

1. Studio di fattibilità: questo si occupa di definire, in maniera per quanto possibile precisa, le priorità e le valutazioni delle componenti che portano a decidere se la realizzazione di un sistema è possibile, a quali costi e sotto quali vincoli. In generale, l'obiettivo dello studio delle fattibilità è quello di esaminare diverse alternative di realizzazione, valutarne benefici e costi e, in fine, produrre un piano di sviluppo del sistema. Quindi lo studio delle fattibilità può essere visto come un preventivo del sistema informativo in termini di benefici e costi.

2. Raccolta e analisi dei requisiti: l'obiettivo di questa fase consiste nella individuazione e nello studio delle proprietà e delle funzionalità che il sistema informativo dovrà avere per produrre poi una descrizione completa ed informale capace di rappresentare in modo soddisfacente il sistema analizzato. In questa fase gioca un ruolo importante l'interazione con gli utenti del sistema al fine di effettuare una stima dei requisiti software ed hardware del sistema.

3. Progettazione: è la fase di traduzione delle specifiche di analisi in strutture tecniche; quindi si può affermare che l'analisi stabilisce "cosa" il sistema deve fare, mentre la progettazione stabilisce il "come". La fase di progettazione si divide complessivamente in progettazione dei dati e progettazione delle applicazioni. Nella prima si individua la struttura e l'organizzazione che i dati dovranno avere; nella seconda si definiscono le qualità peculiarie dei programmi applicativi. Per la struttura dei dati, si potrà vedere più avanti che essa prenderà corpo nella fase finale della progettazione logica e farà riferimento ad un modello logico dei dati. Mentre per i programmi applicativi si è pensato di utilizzare MySQL (come RDBMS) per gestire la base di dati e il linguaggio SQL per interrogarla ed apportare modifiche. L'implementazione in PHP (PHP: Hypertext Preprocessor) permetterà invece d'interrogare il server, quindi direttamente la base di dati per mezzo di pagine web dinamiche; questo sarà lo strumento principe degli utilizzatori finali.

4. Implementazione: si tratta della realizzazione del sistema informativo secondo la struttura e le caratteristiche definite durante la fase di progettazione. Questa fase è composta dall'insieme delle attività che si occupano di costruire e sviluppare la base di dati e sviluppare il codice delle applicazioni.

5. Validazione e collaudo: consiste nel verificare il corretto funzionamento e la qualità del sistema informativo. La sperimentazione deve prevedere, per quanto possibile, tutte le condizioni operative della UX (User Experience) e, sopratutto, prevenire il più possibile la generazione di errori (in particolar modo l'assoluta prevenzione di errori nella base di dati).

6. Funzionamento: è lo step finale del ciclo di vita di un sistema informativo. In questa fase, il sistema informativo diventa operativo ed esegue i compiti per i quali era stato originariamente progettato. Da questo momento in poi verranno effettuate su di esso solo operazioni di gestione e di manutenzione.

È da precisare che questo processo non è quasi mai strettamente sequenziale in quanto, spesso, durante l'esecuzione di una delle attività citate, bisogna rivedere decisioni prese nell'attività precedente.

1.2 Database e DataBase Management System (DBMS)

I computer sono stati costruiti per migliorare il modo in cui memorizzare, recuperare ed elaborare le informazioni. Sebbene, inizialmente, fossero progettati per eseguire calcoli matematici, il loro potere e le capacità di applicazione sono cresciuti insieme con la necessità di elaborare in modo più efficiente e rapido diversi tipi di dati. Per sfruttare le funzionalità per l'archiviazione e il recupero dei dati, sono state sviluppate le basi di dati.

Una base di dati (a volte abbreviato con la sigla DB dall'inglese "data base" o "database") è definita come un archivio di dati strutturato in modo da razionalizzare la gestione e l'aggiornamento delle informazioni e, allo stesso tempo, da permettere lo svolgimento di ricerche complesse. Per fare un esempio immediatamente chiaro, un elenco telefonico è un database.

Alla base dei database ci sono i concetti di dato e di informazione. Un dato è un valore decontestualizzato. Può essere puro, ovvero adimensionale, se non è associato ad alcuna unità di misura, come ad esempio 5 o 8.6; in alternativa può essere dimensionato, come 8.6 cm, 25 px, 25/07/1987. Una informazione, invece, è un dato contestualizzato; ad esempio: Mario è alto 176cm, questo mobile pesa 12 kg.

I dati contenuti in un database possono essere collegati tra loro mediante una logica relazionale, gerarchica, reticolare, deduttiva o ad oggetti. Cronologicamente:

- i primi database si svilupparono negli anni '60 secondo una logica gerarchica ad albero;
- negli anni '70 si diffusero i database reticolari (a grafo) e quelli relazionali (in cui i dati vengono trasformati in informazione attraverso il concetto matematico di relazione);
- a metà degli anni '80 è la volta dei database deduttivi (basi di dati organizzate per inferenza);
- negli anni '90 nacquero i database ad oggetti (object oriented) e quelli semantici.

Nella Figura 1.1 vengono simbolicamente rappresentate le operazioni di salvataggio e recupero dei dati da un database.

A differenza dei vecchi archivi informatici, il database consente una gestione standardizzata delle operazioni di inserimento, aggiornamento e ricerca dei dati mediante l'utilizzo di appositi software detti DBMS (DataBase Management System). I DBMS sono sistemi per la gestione delle basi di dati che mettono a disposizione tutti gli strumenti per poter memorizzare e gestire l'informazione utilizzando un layer persistente. Dunque, un DBMS è sostanzialmente uno strato software che si frappone fra l'utente ed i dati veri e propri. Grazie a questo strato intermedio l'utente e le applicazioni non accedono ai dati così come sono memorizzati effettivamente,

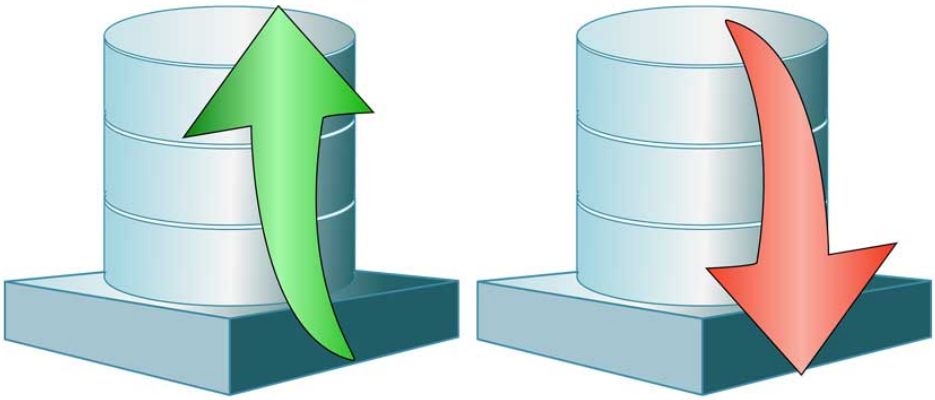


Figura 1.1. Classica rappresentazione di una base di dati; a sinistra viene simboleggiato il salvataggio dei dati mentre a destra viene rappresentato il loro recupero

cioè alla loro rappresentazione fisica, ma ne vedono solamente una rappresentazione logica. Ciò permette un elevato grado di indipendenza fra le applicazioni e la memorizzazione fisica dei dati. L'amministratore del database, se ne sente la necessità, può decidere di memorizzare i dati in maniera differente, o anche di cambiare il DBMS, senza che le applicazioni, e quindi gli utenti, ne risentano. La cosa importante è che non venga cambiata la rappresentazione logica di quei dati, che è la sola cosa che i loro utilizzatori conoscono. Questa rappresentazione logica viene chiamata "schema del database" ed è la forma di rappresentazione dei dati a cui un utente del database può accedere.

Quanto detto viene graficamente mostrato nella Figura 1.2.

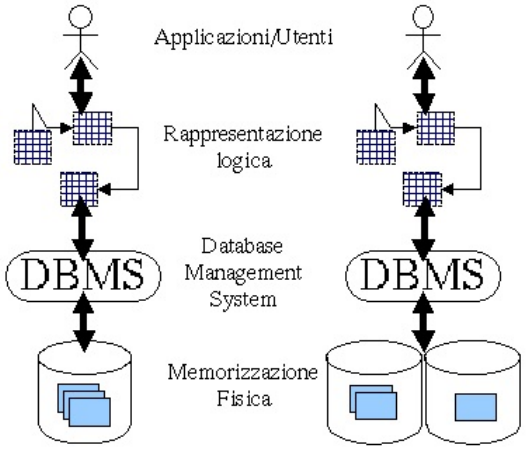


Figura 1.2. Rappresentazione di due esempi che differiscono per la struttura fisica (ma non per quella logica)

In questa figura è rappresentata una situazione in cui l'amministratore del database ha deciso che, per motivi di efficienza, era necessario cambiare il disco su cui erano memorizzati alcuni dati, partizionandoli su più dischi per permettere accessi paralleli a sottoinsiemi di dati indipendenti. Dal punto di vista dell'utente non è cambiato assolutamente nulla e, probabilmente, egli non è nemmeno a conoscenza dell'avvenuto cambiamento. Questo sta a simboleggiare il fatto che la caratteristica principale secondo cui i DBMS vengono normalmente classificati è, appunto, la rappresentazione logica dei dati che essi mostrano ai loro utilizzatori.

In questo progetto verrà utilizzato un DBMS basato su un modello relazionale (cioè un RDBMS) come modalità di gestione della rappresentazione dell'informazione. Gli RDBMS (Relational Database Management System) sono il tipo di database attualmente più diffuso.

I motivi di questo successo sono fondamentalmente due:

- forniscono sistemi semplici ed efficienti per rappresentare e manipolare i dati;
- si basano su un modello, quello relazionale, con solide basi teoriche.

Il primo a proporre il modello relazionale fu E.F. Codd, informatico britannico che, negli anni sessanta e settanta, mentre lavorava per l'IBM, creò il modello relazionale per la gestione delle basi di dati e, nel 1970, pubblicò l'articolo "A Relational Model of Data for Large Shared Data Banks" che gettò le basi teoriche del modello relazionale. Con suo grande disappunto, l'IBM tardò a sfruttare i suoi suggerimenti fino a quando la concorrenza cominciò ad implementarli. Ad esempio, Larry Ellison costruì il database Oracle sulla base delle idee di Codd. Nel 1985 Codd pubblicò il famosissimo articolo "Is Your DBMS Really Relational?" nel quale presentò "le 13 regole di Codd" per definire i requisiti che un sistema per la gestione di basi di dati deve soddisfare per essere considerato relazionale. Grazie alla sua coerenza ed usabilità, il modello è divenuto, dagli anni '80 in poi, quello più utilizzato per la produzione di DBMS, ed oggi il mercato mondiale degli RDBMS supera i 50 miliardi di dollari l'anno.

1.3 Linguaggi per DBMS ed SQL

Ad un DBMS possono essere associate diverse tipologie di linguaggi. Avremo, quindi, linguaggi appositi, che si distinguono per i rispettivi scopi, e che possiamo classificare in:

- linguaggi per la definizione dei dati (Data Definition Language - DDL): vengono utilizzati per definire lo schema della base di dati;
- linguaggi per la manipolazione dei dati (Data Manipulation Language - DML): vengono utilizzati per gestire il contenuto del DB, permettendo, cioè, all'utente di inserire, aggiornare, cancellare il contenuto degli oggetti del database;
- linguaggi d'interrogazione (Query Language o Data Query Language - QL o DQL): sono utilizzati per interrogare gli oggetti del database per ottenere le informazioni che l'utente vuole conoscere in un certo momento;
- linguaggi per il controllo dei dati (Data Control Language - DCL): vengono utilizzati per controllare la sicurezza dei dati, ovvero per gestire i permessi di accesso agli oggetti del database.

Il linguaggio di interrogazione più diffuso tra quelli usati per l'interazione con i principali DBMS, soprattutto relazionali, è il linguaggio SQL.

SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS), progettato per le seguenti operazioni:

- creare e modificare schemi di database (DDL);
- inserire, modificare e gestire dati memorizzati (DML);
- interrogare i dati memorizzati (QL);
- creare e gestire strumenti di controllo e accesso ai dati (DCL).

Quindi, possiamo dedurre che, a dispetto del nome, non si tratta soltanto di un semplice linguaggio di interrogazione.

L'SQL nasce nel 1974 ad opera di Donald Chamberlin, nei laboratori dell'IBM, come strumento per lavorare con database che seguano il modello relazionale. Su di esso si sviluppò il prototipo System R, che venne utilizzato da IBM per usi interni e per alcuni suoi clienti. Dato il suo successo, anche altre società iniziarono subito a sviluppare prodotti basati su SQL. Nel 1981 IBM iniziò a vendere alcuni prodotti relazionali e nel 1983 rilasciò DB2, il suo DBMS relazionale. Fu così che SQL divenne subito lo standard industriale per i software che utilizzano il modello relazionale. L'ANSI lo adottò come standard fin dal 1986. Nel 1987 la ISO fece lo stesso. Tale processo di standardizzazione mirava alla creazione di un linguaggio che funzionasse su tutti i DBMS relazionali; questo obiettivo, però, non fu mai raggiunto pienamente perché ogni DBMS ha sviluppato un proprio "dialetto" SQL.

1.4 Software per lo sviluppo dei database: MySQL

Esistono vari software disponibili per la realizzazione di RDBMS; i più rinomati sono:

- Software Commerciali:
  - Oracle/MySQL;
  - IBM DB2;
  - Microsoft Access.
- Software open source:
  - OpenOffice Base;
  - MariaDB;
  - PostgresSQL.

Il software che verrà utilizzato per lo sviluppo del database relativo al progetto descritto nella presente tesi rappresenta una delle tecnologie più note e diffuse nel mondo dell'IT, ovvero MySQL o Oracle MySQL. Prima di introdurre questo famosissimo RDBMS, è opportuno guardare un po' alla storia di questo DBMS. Il progetto MySQL nacque nel 1996 per opera dell'azienda svedese Tcx, basato su un DBMS relazionale preesistente, e venne distribuito in modalità open source per favorirne la crescita. Nel luglio 2007 la società svedese MySQL AB aveva 385 dipendenti in numerosi paesi. I suoi principali introiti provenivano dal supporto agli utilizzatori di MySQL tramite il pacchetto Enterprise, dalla vendita delle licenze commerciali e dall'utilizzo, da parte di terzi, del marchio MySQL.

Il 20 aprile 2009 è stata proposta l'acquisizione da parte di Oracle Corporation per 7.4 miliardi di dollari; l'acquisizione fu poi completata il 27 gennaio 2010. Anche se la Oracle garantì che il progetto sarebbe rimasto open source e freeware, c'era preoccupazione nelle comunity open source poiché Oracle era un competitor di MySQL. Una parte della community di sviluppo, non fidandosi, si staccò dal mondo MySQL e creò una fork[1] chiamata MariaDB. MySQL e MariaDB, per le funzionalità di base sono equivalenti, ed è per questo che oggi si parla di MySQL anche quando ci si riferisce a MariaDB (un po' come succede con JavaScript ed ECMA Script). A partire dal 2012 diverse distribuzioni Linux e alcuni utenti importanti, come Wikipedia, hanno iniziato a sostituire MySQL con il fork MariaDB. Bisogna sottolineare, perè, che MariaDB non è l'unico fork nato da MySQL; questo famoso RDBMS, infatti, possiede altri importanti fork come: Drizzle, Percona Server, OurDelta e Proven Scaling.

MySQL (e il suo fork MariaDB) viene attualmente utilizzato in progetti open di grandi dimensioni, come Wikipedia, ma anche in contesti come YouTube, Flickr, Facebook e Nokia.

Qui di seguito verranno elencate alcune delle principali caratteristiche di MySQL:

- è uno dei più efficaci RDBMS;
- è freeware;
- è open source;
- è robusto;
- è affidabile;
- è multi-piattaforma;
- ha ridotti requisiti hardware;
- è basato sul linguaggio SQL;
- possiede connettori e librerie verso:
  - C/C++ (libmysql);
  - Java (JDBC);
  - ODBC (Win/Unix);
  - PHP;
  - Perl;
  - Python;
  - altri linguaggi (wrapper libmysql).
- è in grado di gestire basi di dati di notevoli dimensioni;
- possiede una buona gestione di sicurezza dell'accesso.

MySQL gestisce diversi tipi di tabelle (chiamati anche storage engine). Ciascuno storage engine ha i suoi vantaggi e svantaggi.

Fra gli storage engine ufficiali di MySQL troviamo:

- MyISAM: è il motore storico di MySQL, che estende lo storage engine ISAM. Le tabelle di tipo MyISAM sono veloci e di dimensioni ridotte. La dimensione di una tabella di questa tipologia può arrivare fino a 256TB e non è transaction-safe.

[1] Sviluppo di un nuovo progetto software che parte dal codice sorgente di un altro già esistente. La community, infatti, partì con il software MySQL prima che fosse completata l'acquisizione da parte di Oracle e continuò a svilupparlo in un thread parallelo.

- InnoDB: Le tabelle InnoDB supportano completamente il modello transazionale ACID (Atomic, Consistent, Isolated, Durable), che garantisce la validità e l'integrità dei dati in una transazione. La dimensione di una tabella di tipo InnoDB può arrivare fino a 64 TB. Le tabelle InnoDB (come anche quelle MyISAM) sono portabili tra diverse piattaforme e sistemi operativi. In phpMyAdmin abbiamo adottato InnoDB come storage engine dato che quest'ultimo supporta le chiavi esterne.
- Memory (heap): le tabelle di tipo Memory vengono conservate nella memoria RAM del server e utilizzano indici hash. Sono più veloci delle tabelle MyISAM e sono adatte per chi ha la necessità di conservare le informazioni in maniera temporanea, ma con tempi di accesso molto rapidi.
- Archive: consente di memorizzare un numero elevato di record per scopi di archiviazione (a cui quindi non si accede spesso) in un formato compresso per risparmiare spazio disco. Le tabelle di tipo Archive consentono solo istruzioni di tipo INSERT e SELECT e non non supportano gli indici, per cui è necessaria una scansione completa della tabella per il recupero delle informazioni.
- CSV: lo storage engine CSV (Comma Separated Values) memorizza i dati su file di testo in cui i campi sono separati da un carattere separatore, come ad esempio la virgola o il punto e virgola. Una tabella di tipo CSV è comoda per la migrazione dei dati in applicazioni non-SQL come i fogli di calcolo che sono in grado di gestire direttamente questo formato. Non supporta il tipo di dati NULL e l'operazione di lettura richiede una scansione completa della tabella.
- NDB: si tratta del motore utilizzato in un ambiente distribuito, usato da MySQL Cluster per implementare tabelle partizionate su diversi computer.
- Merge: è un insieme di tabelle MyISAM identiche (hanno identiche informazioni sulle colonne sugli indici) che possono essere usate come un unico oggetto.
- Federated: consente di gestire i dati da un server MySQL remoto senza l'utilizzo di cluster o tecnologia di replicazione. La tabelle locali di tipo Federated non memorizzano dati e le query estraggono i dati dalle tabelle remote di tipo Federated. In questo modo MySQL offre una rappresentazione logica di tabelle differenti su diversi server, che agli occhi dell'utente sembrano un unico database.
- Blackhole: lo store engine Blackhole agisce come un "buco nero", che accetta i dati, ma li butta via e non li memorizza. Se si crea una tabella e si inseriscono dei record, una query su quella tabella restituisce sempre un resultset vuoto. Tutto ciò che finisce in questo tipo di tabella viene perso.
- Example: lo store engine Example è uno stub engine che non fa nulla; è uno scheletro che serve da esempio agli sviluppatori per scrivere un proprio engine.

Esistono anche storage engine prodotti da terze parti; alcuni di questi sono distribuiti con MariaDB o Percona Server. Citiamone alcuni:

- XtraDB: fork di InnoDB, sviluppato da Percona;
- OQGRAPH: simula strutture a grafo;
- solidDB;
- RitmarkFS: permette di accedere al filesystem in lettura e in scrittura tramite comandi SQL, supporta anche la replica del filesystem;
- Distributed Data Engine;
- SphinxSE: interfaccia MySQL con Sphinx;

- CassandraSE: interfaccia MariaDB con Apache Cassandra;
- mdbtools: permette di leggere e scrivere un file .mdb (Access);
- BrightHouse: appare all'utilizzatore come un normale MyISAM, ma struttura internamente i dati per colonne anziché per righe. Da utilizzare tipicamente in lettura per Data Warehouse: riduce da 10 a 100 volte i tempi di accesso e lo spazio per i dati.

Nella Figura 1.3 viene mostrata l'architettura di MySQL.

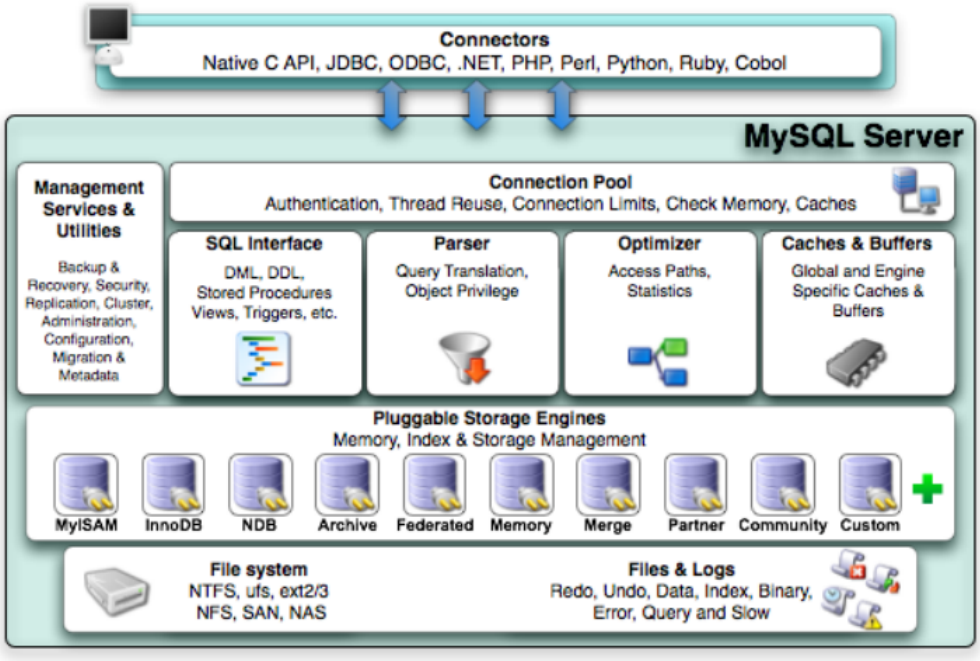


Figura 1.3. Architettura di MySQL

2

Il contesto di riferimento e analisi dei requisiti

In questo capitolo contestualizzeremo un'importante fase che introduce la progettazione di un sistema informativo: la definizione del contesto di riferimento e dell'analisi dei requisiti. Questa fase verrà prima definita e collocata all'interno del ciclo di vita di un sistema informativo, e poi implementata all'interno del sistema informativo oggetto della presente tesi.

2.1 Contesto di riferimento

Lo sviluppo di un moderno Sistema Informativo non può oggi prescindere dal contesto nel quale si trova ad operare. Distinguere l'informazione utile dal "rumore informativo", cioè da ciò che non è rilevante per il problema applicativo del momento, non è facile: un oggetto, una qualsiasi entità informativa, o anche reale, possono essere considerati in modo diverso, anche dallo stesso utente, quando questo si trova in diverse situazioni, e cioè in contesti differenti.

In generale, il contesto può essere visto come un insieme di variabili i cui valori possono interessare un attore e influenzarne le azioni; variabili legate al contesto sono, ad esempio, il luogo nel quale ci si trova, l'ora del giorno, il tempo atmosferico, la situazione lavorativa o la compagnia presente; un cambiamento del contesto può, infatti, causare un cambiamento nella rappresentazione mentale che l'attore si crea della realtà, anche se quest'ultima non è cambiata.

L'azienda di riferimento per il nostro progetto è la Italcamion s.r.l. e si occupa della vendita di veicoli industriali (autocarri e gru) usati e nuovi, degli allestimenti, dei relativi servizi finanziari e delle pratiche amministrative, consentendo al cliente di avere un unico referente totalmente responsabile di ogni fase della preparazione del veicolo. Essa è situata in Abruzzo ed è concessionaria dei prestigiosi marchi, quali, ad esempio, Scania, Nissan, Effer e Benelli.

La policy di quest'azienda è quella di avere sempre disponibile un certo numero di autocarri di varie tipologie in modo tale da consentire ai clienti di avere un'ampia e soddisfacente scelta di veicoli confacenti alle proprie necessità. L'attività di allestimento degli autoveicoli consiste nella progettazione e realizzazione di cassoni fissi o ribaltabili, nella costruzione di centine e piani di carico, nell'installazione di

gru oleodinamiche per autocarri di fabbricazione Effer S.p.A. di Castel Maggiore (BO) e di piattaforme per lavoro aereo di fabbricazione Benelli di Reggio Emilia.

Le figure "interne" all'azienda sono:

- i due titolari, che si occupano della direzione dei seguenti settori: tecnico, commerciale, amministrativo e finanziario;
- un ragioniere, che si occupa di contabilità ed amministrazione;
- i membri dello staff (dipendenti);
- gli operai.

Le figure "esterne" all'azienda sono:

- case produttrici e concessionarie, che forniscono i veicoli (nuovi o usati);
- fornitori, che forniscono le materie prime necessarie per l'allestimento;
- clienti, che possono essere privati o aziende.

Il sistema è pensato per essere usato dai titolari dell'azienda, dallo staff e dai clienti; ogni livello di utenza ha diritti differenti e differenti possibilità di accesso alle varie funzionalità fornite dal sistema. Grazie ad un sito web, i clienti hanno la possibilità di visionare i prodotti offerti dall'azienda attraverso dispositivi digitali (Pc, tablet e smartphone); i dipendenti e i titolari dell'azienda (che quindi possiedono un livello "staff") hanno la completa gestione delle funzionalità del sito relative all'amministrazione on-line dell'azienda; l'amministratore è quello che possiede tutti i privilegi e tutti i diritti. Non è prevista la vendita on-line ed il ritiro del veicolo si effettua in sede; tuttavia è possibile prenotare sul sito la prova gratuita del mezzo al quale il cliente è interessato.

2.2 Analisi dei requisiti

Il reperimento e l'analisi dei requisiti sono attività difficilmente standardizzabili perché dipendono notevolmente dall'applicazione da realizzare. Per raccolta dei requisiti si intende la completa individuazione dei problemi che il sistema da realizzare deve risolvere e delle caratteristiche che tale sistema dovrà avere. I requisiti vengono inizialmente raccolti in specifiche, espresse generalmente in linguaggio naturale, e, per questo motivo, spesso ambigue e disorganizzate. L'analisi dei requisiti, invece, consiste nel chiarimento e nell'organizzazione delle specifiche sui requisiti. Si tratta, ovviamente, di attività fortemente interconnesse: l'attività di raccolta inizia con i primi requisiti ottenuti per poi procedere di pari passo con l'attività di analisi dei requisiti raccolti.

I requisiti di un'applicazione provengono, nella maggior parte dei casi, da fonti diverse. Le principali fonti di informazione sono, in genere, le seguenti:

- Gli utenti dell'applicazione, contattati mediante opportune interviste, anche ripetute, oppure attraverso una documentazione scritta predisposta appositamente per questo scopo. Infatti, durante la fase di acquisizione delle specifiche, gioca un ruolo importante l'interazione con gli utenti del sistema informativo.
- La documentazione esistente, che ha qualche attinenza con il problema in esame come, ad esempio, moduli, regolamenti interni, procedure aziendali, normative.

- Eventuali realizzazioni pre-esistenti, ovvero applicazioni che si devono rimpiazzare o che devono interagire in qualche modo con il sistema da realizzare.

Ecco alcune regole generali per ottenere una specifica dei requisiti più precisa e senza ambiguità:

- Scegliere il corretto livello di astrazione. È bene evitare di utilizzare termini troppo generici o troppo specifici che rendono poco chiaro un concetto.
- Standardizzare la struttura delle frasi. Nella specifica dei requisiti è preferibile utilizzare sempre lo stesso stile sintattico.
- Evitare frasi contorte. Le definizioni devono essere semplici e chiare.
- Individuare sinonimi/omonimi e unificare i termini
- Rendere esplicito il riferimento tra termini. Può succedere che l'assenza di un contesto di riferimento renda alcuni concetti ambigui; in questi casi è necessario esplicitare il riferimento tra termini.
- Costruire un glossario dei termini. È molto utile definire un glossario che, per ciascun termine, contenga una breve descrizione, possibili sinonimi e altri termini ivi contenuti con i quali esiste un legame logico.

Due importanti caratteristiche delle specifiche dei requisiti sono la completezza e la coerenza. Un documento di specifica dei requisiti è completo se tutti i requisiti richiesti dagli utenti sono definiti; esso è consistente quando non vi sono requisiti in conflitto tra loro. Sebbene per sistemi di piccole dimensioni questi risultati siano facilmente raggiungibili, per i sistemi caratterizzati da un numero elevato di requisiti è richiesto uno sforzo maggiore. Infatti, in questo secondo caso, la difficoltà è dovuta alla maggiore occorrenza di errori ed omissioni, nonché alla presenza di un numero elevato di stakeholder con esigenze differenti, a volte contrastanti, che possono non essere valutate nella prima fase di scrittura delle specifiche. Dunque, la presenza di inconsistenze potrebbe essere scorta addirittura dopo aver consegnato il prodotto al cliente, dando luogo a costi correzione e modifica esorbitanti.

Nelle aziende odierne vi sono figure specializzate nella raccolta ed analisi dei requisiti, a testimonianza del fatto che oggi tali attività sono di vitale importanza per la nascita di un sistema informativo robusto ed efficiente. Tali figure si chiamano Analisti.

Quando parliamo di requisiti è utile separare i requisiti funzionali da quelli non funzionali; nelle omonime sotto-sezioni di questo capitolo vedremo le differenze fra le due tipologie di requisiti e la definizione degli stessi per ciò che concerne il nostro sistema.

2.2.1 Requisiti funzionali e diagrammi dei casi d'uso

l requisiti funzionali sono requisiti che definiscono una o più funzioni di un sistema (oppure i requisiti di uno o più dei suoi componenti), specificando la tipologia degli ingressi e delle uscite, nonché il comportamento. Essi si presentano come elenchi di funzionalità o servizi che il sistema deve fornire; essi descrivono, inoltre, il comportamento del sistema a fronte di particolari input e come esso dovrebbe reagire in determinate situazioni.

I requisiti funzionali del sistema informativo relativo all'azienda Italcamion s.r.l. sono i seguenti:

- RF1: il sistema dovrà gestire le attività CRUD sui Titolari;
- RF2: il sistema dovrà gestire le attività CRUD sui Dipendenti;
- RF3: il sistema dovrà gestire le attività CRUD sui Clienti;
- RF4: il sistema dovrà gestire le attività CRUD su Case produttrici e Concessionarie;
- RF5: il sistema dovrà gestire le attività CRUD sui Fornitori;
- RF6: il sistema dovrà gestire le attività CRUD sugli ordini;
- RF7: il sistema dovrà gestire le attività CRUD sui veicoli;
- RF8: il sistema dovrà gestire le attività CRUD sulle materie prime;
- RF9: il sistema dovrà consentire ai clienti di gestire solo i propri dati;
- RF10: il sistema dovrà consentire agli amministratori il pieno controllo del sistema informativo, senza restrizioni;
- RF11: il sistema dovrà prevedere un meccanismo di accesso all'area riservata distinto per clienti, ragionieri (che rientrano nei membri dello staff) e amministratori;
- RF12: il sistema dovrà consentire agli amministratori di rimuovere un account relativo ad un cliente o ad un membro dello staff;
- RF13: il sistema dovrà consentire ai membri dello staff di creare, modificare o rimuovere un account relativo ad un cliente;
- RF14: il sistema dovrà consentire ai membri dello staff di visualizzare il prospetto delle prenotazioni effettuate dai clienti;
- RF15: il sistema dovrà consentire agli utenti di prenotare una prova gratuita del veicolo da loro selezionato.

Nella costruzione di un sistema informativo, la prima cosa da fare è stabilire quali siano i suoi confini. Infatti, in molti progetti, sono sorti problemi proprio a causa di un confine del sistema non ben definito. Questi confini vengono delimitati dai casi d'uso. Questi ultimi sono definiti come la specifica di una sequenza di azioni, incluse eventuali sequenze alternative e sequenze di errore, che un sistema, un sottosistema o una classe può eseguire interagendo con attori esterni. L'attore identifica il ruolo che un'entità esterna assume quando interagisce direttamente con il sistema (quindi può rappresentare il ruolo di un utente o di un altro sistema che in qualche modo interessa il confine del sistema). Dunque possiamo definire il caso d'uso come qualcosa che un attore vuole che il sistema faccia.

In UML, gli Use Case Diagram (diagrammi dei casi d'uso) sono diagrammi dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono con esso. Gli Use Case Diagram vengono utilizzati soprattutto nel contesto della Use Case View (vista dei casi d'uso) di un modello, e si possono considerare come uno strumento di rappresentazione dei requisiti funzionali di un sistema. I casi d'uso vengono rappresentati con un attore che usa una funzione del sistema (possiamo vederne una semplice rappresentazione grafica nella Figura 2.1).

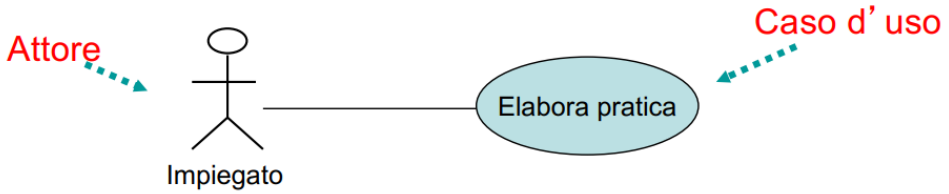


Figura 2.1. Rappresentazione di un generico caso d'uso

Nelle Figure 2.2, 2.3, e 2.4 vengono mostrati i diagrammi relativi ai casi d'uso dei 3 attori del sistema informativo che stiamo prendendo in esame: gli amministratori, i dipendenti e i clienti.

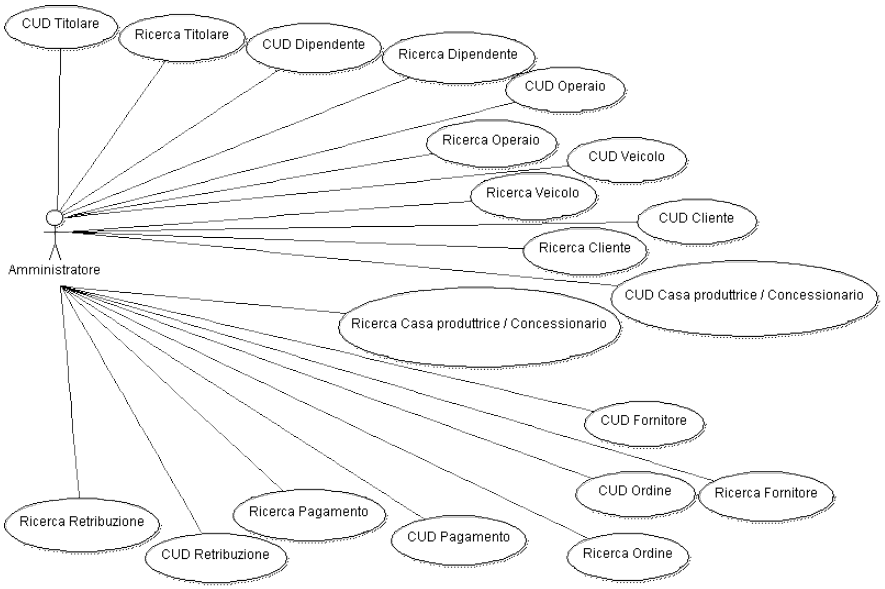


Figura 2.2. Diagramma dei casi d'uso relativi all'attore "Amministratore"

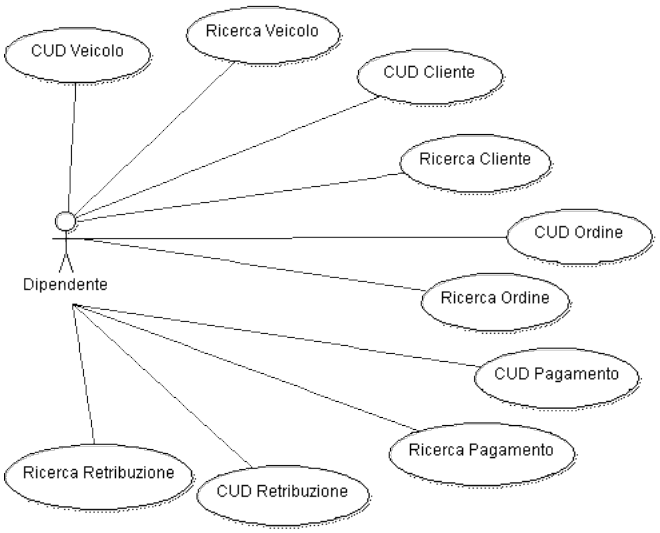


Figura 2.3. Diagramma dei casi d'uso relativi all'attore "Dipendente"

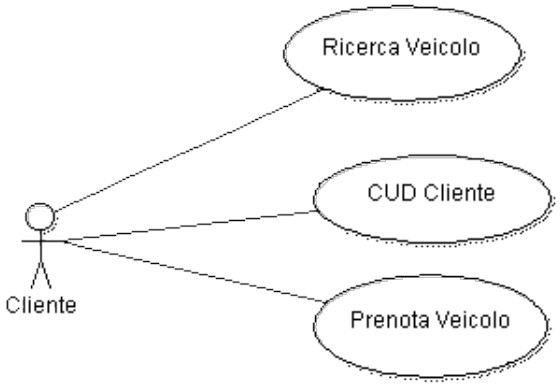


Figura 2.4. Diagramma dei casi d'uso relativi all'attore "Cliente"

2.2.2 Requisiti non funzionali

I requisiti non funzionali descrivono vincoli sui servizi offerti dal sistema, nonché sul suo processo di sviluppo. Tipicamente, tali requisiti non si applicano a singole funzioni o servizi, bensì all'intero sistema. Essi non riguardano direttamente le specifiche funzioni fornite dal sistema, ma possono riferirsi sia a caratteristiche che si desiderano per il sistema (come l'affidabilità, i tempi di risposta, l'occupazione in memoria), sia ai vincoli ai quali il sistema deve sottostare (come la capacità dei dispositivi di I/O e la rappresentazione dei dati utilizzata nelle interfacce del sistema). I requisiti non funzionali sono strettamente vincolati alle esigenze degli utenti, alle politiche organizzative adottate, agli standard adoperati e alla necessaria modalità di interazione del relativo sistema con altri componenti.

Pertanto, essi possono essere classificati in:

- *requisiti di prodotto*, che descrivono le caratteristiche del prodotto, in termini di usabilità, efficienza, affidabilità e portabilità;
- *requisiti organizzativi (o di processo)*, che derivano dalle politiche e procedure organizzative relative al cliente ed allo sviluppatore;
- *requisiti esterni*, che si riferiscono a fattori estranei al sistema ed al relativo processo di sviluppo, come requisiti legislativi, requisiti di interoperabilità.

In Figura 2.5 viene rappresentata una classificazione dei requisiti non funzionali in linea con quanto detto sopra.

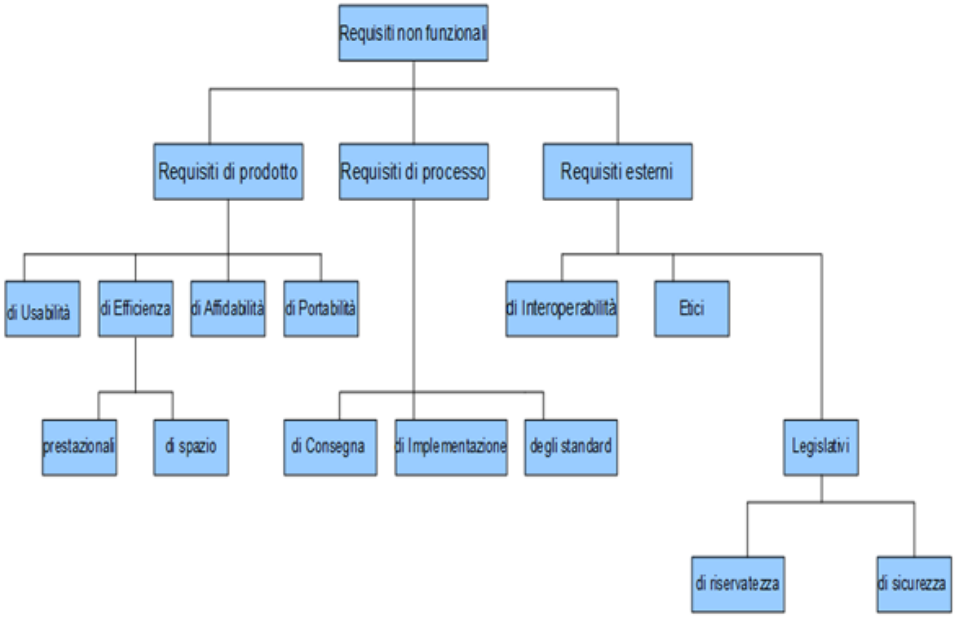


Figura 2.5. Classificazione gerarchica dei requisiti non funzionali

I requisiti non funzionali del sistema informativo relativo all'azienda Italcamion s.r.l. sono i seguenti:

- il sistema dovrà richiedere ad ogni utilizzatore uno username di almeno 8 caratteri, che dovrà essere diverso da quello degli altri utenti, e una password di almeno 8 caratteri;
- il sistema dovrà essere implementato sfruttando esclusivamente tecnologia open-source;
- il linguaggio che si utilizzerà per i diagrammi dei casi d'uso e delle attività è UML;
- il software che si utilizzerà per lo sviluppo dei diagrammi dei casi d'uso e delle attività è ArgoUML;
- il linguaggio che si utilizzerà per l'implementazione sarà PHP;
- il framework che verrà utilizzato per l'implementazione è Zend Framework, Versione 1.12.20;
- il software che si utilizzerà per la gestione del database è MySQL;
- l'applicazione Web che si utilizzerà per gestire l'amministrazione di MySQL sarà phpMyAdmin;
- il sistema di codifica che verrà utilizzato è UNICODE;
- lo standard di codifica UNICODE che verrà utilizzata per la codifica dei caratteri è UTF-8.

3

Progettazione del sistema

La progettazione è la fase più importante nel ciclo di vita di un sistema informativo ed è indispensabile per la successiva implementazione di quest'ultimo. In questo capitolo entreremo nel vivo della progettazione analizzando ogni aspetto delle due macro-fasi in cui si divide la progettazione di un sistema informativo, ovvero la progettazione dei dati e la progettazione delle applicazioni.

3.1 Progettazione della componente dati

La progettazione della componente dati si articola nelle tre seguenti fasi:

- progettazione concettuale;
- progettazione logica;
- progettazione fisica.

Per effettuare le varie attività di progettazione, abbiamo bisogno di opportuni modelli. Ma cos'è un modello? Un modello dei dati è un formalismo matematico composto da una notazione per descrivere i dati e da un insieme di operazioni per poterli manipolare.

I modelli si suddividono in:

- Modelli concettuali, che vengono utilizzati per descrivere i dati in maniera completamente indipendente dalla struttura del DBMS sottostante e tendono a descrivere i concetti nel mondo reale, piuttosto che i dati utili a rappresentarli. Questi modelli vengono utilizzati durante la fase preliminare del processo di progettazione di una base di dati per analizzare nel modo migliore la realtà di interesse, senza preoccuparsi del DBMS con cui questa viene successivamente rappresentata. Il più importante modello concettuale è il modello Entità/Relazione (che è quello che verrà adottato per la fase di progettazione concettuale della presente tesi).
- Modelli logici, che rappresentano gli effettivi modelli di riferimento per i vari DBMS. Essi, infatti, descrivono la realtà avendo come riferimento una strutturazione concreta dei dati.
  I più comuni modelli logici usati in questa fase sono i seguenti:

- il modello gerarchico, che è basato sull'utilizzo di strutture ad albero, e quindi di gerarchie;
- il modello reticolare, che è basato sull'uso dei grafi;
- il modello relazionale, che rappresenta la realtà di interesse per mezzo di relazioni le cui righe rappresentano specifiche istanze e le cui colonne corrispondono a specifiche proprietà (e che sarà, come abbiamo già sottolineato, il modello logico che si adotterà per il presente progetto di tesi);
- il modello ad oggetti, che estende alle basi di dati il paradigma di programmazione orientata agli oggetti.

L'obiettivo della progettazione della componente dati è quello di realizzare un modello dei dati ben strutturato e, soprattutto, disponibile per essere gestito dal DBMS e dalle piattaforme software che si hanno a disposizione e che contribuiranno alla gestione e all'implementazione del sistema informativo. Per fare questo bisogna partire dal contesto di riferimento e dall'analisi dei requisiti effettuate in precedenza, ed iniziare a realizzare la progettazione della componente dati partendo proprio dal primo step: la progettazione concettuale.

3.1.1 Progettazione concettuale

La progettazione concettuale di una base di dati consiste nella costruzione di uno schema concettuale in grado di descrivere al meglio la realtà di interesse. Il modello concettuale che utilizzeremo, come abbiamo già detto, è il modello Entità/Relazione: esso fornisce una serie di strutture atte a descrivere la realtà di interesse in modo facilmente comprensibile e prescindendo dai criteri di organizzazione dei dati.

Le tre componenti principali del modello E/R sono:

- Le entità, che rappresentano la classe di oggetti che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse; un'istanza di entità è un oggetto della classe rappresentata dall'entità. Nei diagrammi E/R un'entità viene rappresentata con un rettangolo che contiene al proprio interno il nome dell'entità stessa.
- Le relazioni (o associazioni), che rappresentano legami logici tra due o più entità; un'istanza di relazione è un'ennupla (coppia nel caso più frequente di relazione binaria) costituita da istanze di entità, una per ciascuna delle entità coinvolte. In uno schema E/R una relazione è rappresentata da un rombo che contiene al proprio interno il nome della relazione stessa.
- Gli attributi, che descrivono le proprietà elementari di entità o relazioni; un attributo associa a ciascuna istanza di entità (o di relazione) un valore appartenente ad un insieme detto dominio.

I costrutti messi a disposizione dal modello E/R vengono utilizzati per definire schemi che consentono la rappresentazione grafica della progettazione concettuale; la rappresentazione grafica del modello E/R è chiamata diagramma E/R. La Figura 3.1 rappresenta il diagramma E/R relativo al sistema informativo dell'azienda di interesse per la presente tesi.

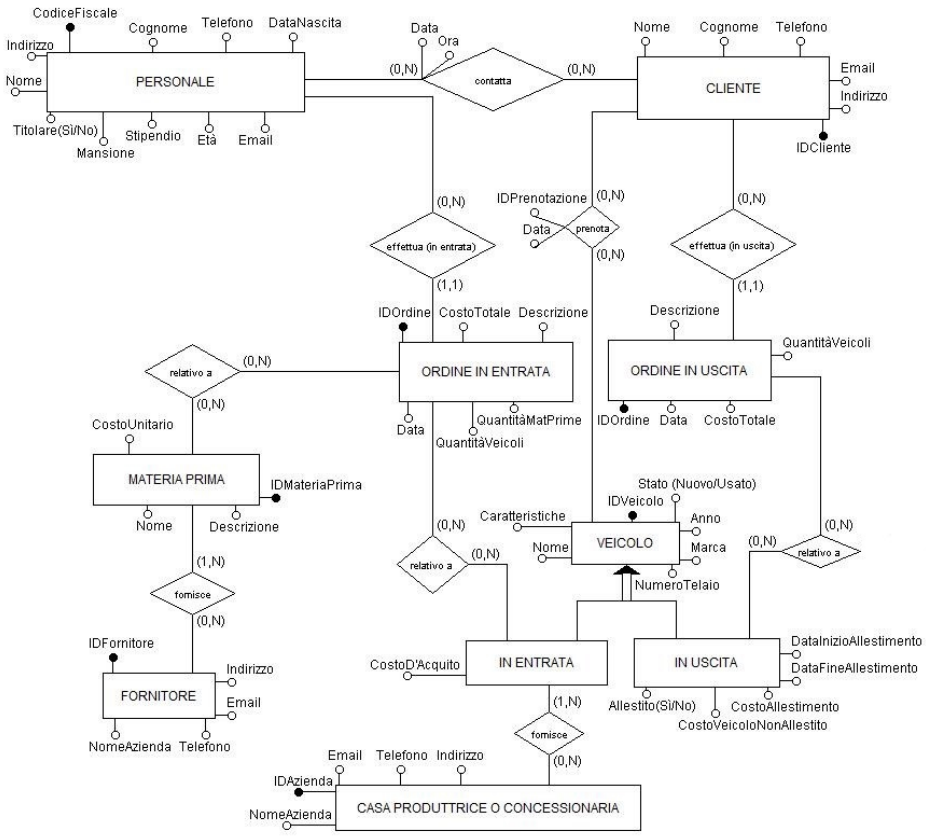


Figura 3.1. Diagramma E/R

Le Figure 3.2, 3.3 e 3.4, invece, raffigurano il Dizionario delle Entità, il Dizionario delle Relazioni e l'insieme dei vincoli associati allo schema della Figura 3.1.

ENTITA'	DESCRIZIONE	ATTRIBUTI	CHIAVE
PERSONALE	Può essere un titolare o un dipendente; con la parola "personale" indichiamo qualunque persona che lavora all'interno dell'azienda	CodiceFiscale, Nome, Cognome, Indirizzo, Telefono, DataNascita, Mansione, Titolare, Età, Stipendio, Email	CodiceFiscale
CLIENTE	Colui che richiede il prodotto	IDCliente, Nome, Cognome, Telefono, Email, Indirizzo	IDCliente
ORDINE IN ENTRATA	Ordine che viene effettuato dall'azienda ad un fornitore o ad una casa produttrice/concessionaria al fine di acquistare, rispettivamente, materie prime o veicoli	IDOrdine, CostoTotale, Descrizione, Data, QuantitàVeicoli, QuantitàMatPrime	IDOrdine
ORDINE IN USCITA	Ordine relativo ad un veicolo, che viene effettuato dal cliente	IDOrdine, CostoTotale, Descrizione, Data, QuantitàVeicoli	IDOrdine
VEICOLO	Veicolo presente in azienda	IDVeicolo, Nome, Caratteristiche, NumeroTelaio, Anno, Marca, Stato	IDVeicolo
(VEICOLO) IN ENTRATA	Veicolo che è stato acquistato dall'azienda, quindi relativo ad un ordine effettuato	CostoD'acquisto	VEICOLO.IDVeicolo
(VEICOLO) IN USCITA	Veicolo dell'azienda che deve essere venduto al cliente	DataInizioAllestimento, DataFineAllestimento, CostoAllestimento, CostoVeicoloNonAllestito, Allestito(Sì/No)	VEICOLO.IDVeicolo
MATERIA PRIMA	Prodotto che viene ordinato per poi essere usato durante la fase di allestimento	IDMateriaPrima, Nome, Descrizione, CostoUnitario	IDMateriaPrima
FORNITORE	Colui che fornisce la materia prima all'azienda	IDFornitore, NomeAzienda, Telefono, Email, Indirizzo	IDFornitore
CASA PRODUTTRICE O CONCESSIONARIA	Soggetto dalla quale l'azienda acquista i veicoli (nuovi o usati)	IDAzienda, NomeAzienda, Telefono, Email, Indirizzo	IDAzienda

Figura 3.2. Dizionario delle Entità per lo schema E/R di Figura 3.1

RELAZIONE	DESCRIZIONE	ENTITA' COINVOLTE	ATTRIBUTI
contatta	Descrive il momento in cui il cliente contatta una persona dell'azienda per acquistare un veicolo	PERSONALE (0,N) - CLIENTE (0,N)	data, ora
prenota	Descrive il momento in cui il cliente effettua una prenotazione per una prova gratuita del veicolo desiderato, in una data selezionata	CLIENTE(0,N) – VEICOLO(0,N)	IDPrenotazione, data
effettua (in entata)	Descrive l'atto dell'ordine che l'azienda effettua, con lo scopo di acquistare materie prime o veicoli (nuovi o usati)	PERSONALE (0,N) - ORDINE IN ENTRATA (1,1)	
effettua (in uscita)	Descrive l'atto dell'ordine che il cliente effettua nei confronti dell'azienda, al fine di acquistare da essa un veicolo	CLIENTE (0,N) - ORDINE IN USCITA (1,1)	
relativo a (materia prima)	Descrive cosa ordina l'azienda: in questo caso l'ordine che effettua l'azienda si riferisce ad un veicolo	ORDINE IN ENTRATA (0,N) - MATERIA PRIMA (0,N)	
relativo a (veicolo in entrata)	Descrive cosa ordina l'azienda: in questo caso l'ordine che effettua l'azienda si riferisce alle materie prime	ORDINE IN ENTRATA (0,N) - (VEICOLO) IN ENTRATA (0,N)	
relativo a (veicolo in uscita)	Descrive cosa ordina un cliente alla nostra azienda: l'ordine effettuato dal cliente si riferisce ad un veicolo (quindi per la nostra azienda sarà un veicolo in uscita)	ORDINE IN USCITA (0,N) - (VEICOLO) IN USCITA (0,N)	
fornisce (materia prima)	Descrive la fornitura di materia prima	FORNITORE (0,N) - MATERIA PRIMA (1,N)	
fornisce (veicolo)	Descrive la fornitura di un veicolo	CASA PRODUTTRICE O CONCESSIONARIA (0,N) - (VEICOLO) IN ENTRATA (1,N)	

Figura 3.3. Dizionario delle Relazioni per lo schema E/R di Figura 3.1

VINCOLO	DESCRIZIONE
V1	L'attributo "Titolare" dell'entità "PERSONALE" può assumere solo i valori "Sì" o "No"
V2	Se il valore dell'attributo "Titolare" appartenente all'entità "PERSONALE" è posto a "Sì", allora gli attributi "Stipendio" e "Mansione", appartenenti sempre all'entità "PERSONALE", non devono essere presenti
V3	L'attributo "Età" dell'entità "PERSONALE" è un campo calcolato dall'attributo "DataNascita" che appartiene sempre all'entità "PERSONALE"
V4	L'attributo "Allestito" dell'entità figlia denominata "IN USCITA" relativa all'entità padre "VEICOLO" può assumere solo i valori "Sì" o "No"
V5	Se un veicolo ha il valore "No" sull'attributo "Allestito" allora gli attributi "DataInizioAllestimento" e "DataFineAllestimento" non devono essere presenti, mentre l'attributo "CostoAllestimento" deve avere valore uguale a zero.
V6	L'attributo "Stato" dell'entità "VEICOLO" può assumere solo i valori "Nuovo" o "Usato"
V7	L'attributo "Indirizzo", ogniqualvolta si presenta, è un attributo composto
V8	L'attributo "CostoTotale" dell'entità "ORDINE IN ENTRATA" è un campo calcolato dall'operazione: (QuantitàVeicoli * Veicolo_In_Entrata.CostoD'acquisto) + (QuantitàMatPrima * MateriaPrima.CostoUnitario)
V9	L'attributo "CostoTotale" dell'entità "ORDINE IN USCITA" è un campo calcolato dall'operazione: QuantitàVeicoli * (Veicolo_In_Uscita.CostoVeicoloNonAllestito + Veicolo_In_Uscita.CostoAllestimento)

Figura 3.4. Vincoli relativi allo schema E/R di Figura 3.1

3.1.2 Progettazione logica

La fase di progettazione logica consiste nella traduzione dello schema concettuale (realizzato nella fase della progettazione concettuale) in termini delle strutture di rappresentazione proprie del tipo di DBMS a disposizione. Il prodotto di questa fase viene denominato schema logico della base di dati e fa riferimento al modello logico dei dati descritto all'inizio di questo capitolo. Durante questa fase le scelte progettuali tengono conto anche dei criteri di ottimizzazione delle rappresentazioni e della qualità dello schema logico ottenuto.

La progettazione logica si suddivide in due fasi:

- ristrutturazione dello schema E/R;
- traduzione verso il modello logico.

Iniziamo, ora, ad analizzare com'è stata realizzata la progettazione logica di questo progetto partendo dallo schema E/R che ereditiamo dalla progettazione concettuale.

La fase di ristrutturazione di uno schema E/R si può suddividere in una serie di passi da effettuare in sequenza. Tali passi sono:

1. *Analisi delle ridondanze*: l'unica ridondanza riscontrata è rappresentata dagli attributi "Età" e "Data Nascita" dell'entità "PERSONALE", dal momento che il primo attributo è derivabile dal secondo. Abbiamo, quindi, scelto di eliminare l'attributo "Età" mantenendo, invece, l'attributo "DataNascita".

2. *Eliminazione delle generalizzazioni*: come possiamo vedere nella Figura 3.5, abbiamo scelto di mantenere sia le entità figlie che l'entità padre perché ritenute importanti nella funzionalità del progetto.

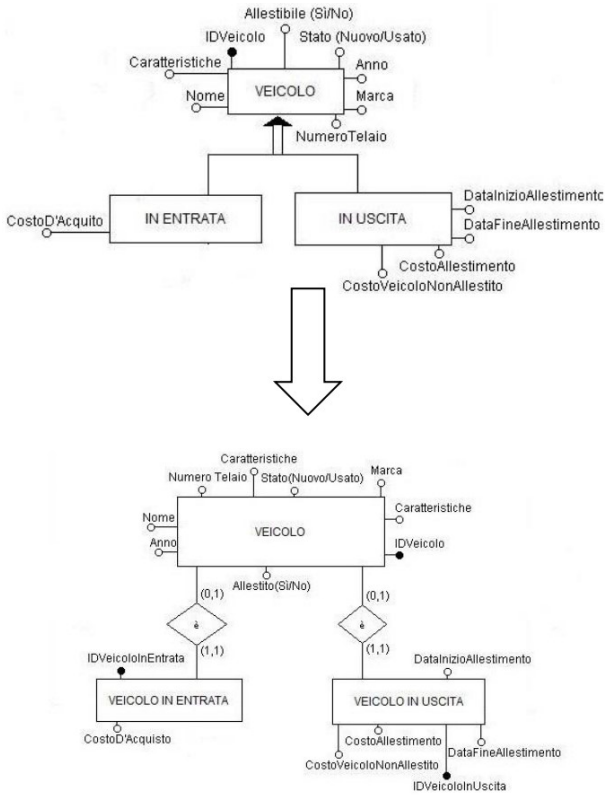


Figura 3.5. Ristrutturazione della porzione dello schema E/R relativa ai vincoli

3. *Partizionamento/accorpamento di entità e associazioni*: nel nostro caso questo passo non è necessario.
4. *Eliminazione degli attributi multi-valore*: nel nostro caso questa attività non è necessaria.
5. *Eliminazione degli attributi composti*: è stato eliminato l'attributo composto "Indirizzo", ogniqualvolta esso figurava, ed è stato sostituito con gli attributi semplici che lo componevano: Via, N.Civico, CAP e Città.
6. *Scelta degli identificatori primari*: in realtà tali indicatori erano già stati scelti nello scema E/R.

Si è arrivati, quindi, a poter rappresentare lo schema E/R della realtà di interesse dopo la ristrutturazione; tale schema prende il nome di schema E/R ristrutturato e viene mostrato nella Figura 3.6.

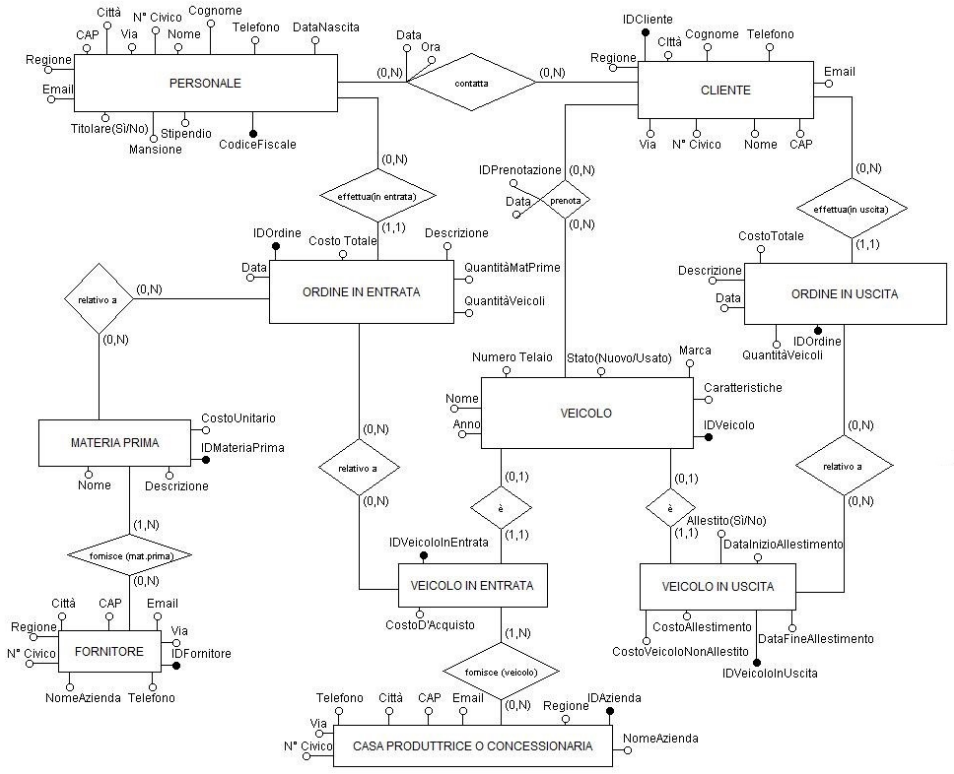


Figura 3.6. Schema E/R ristrutturato

Occupiamoci, ora, della fase di traduzione verso il modello logico analizzando singolarmente ogni relazione dello schema E/R ristrutturato:

- Relazione "contatta": la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.7.

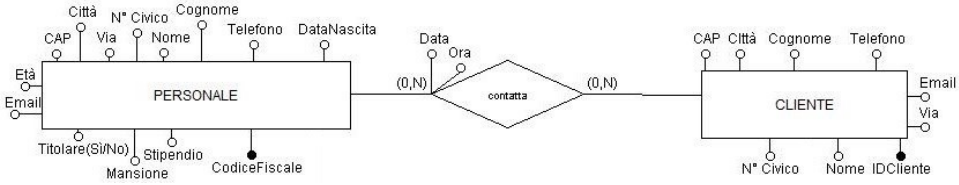


Figura 3.7. Porzione dello schema E/R di Figura 3.1 relativo alla relazione "contatta"

Si tratta di un'associazione molti-a-molti. Dobbiamo però sottolineare il fatto che lo stesso cliente potrebbe contattare lo stesso addetto del personale ma in date e ore diverse. Quindi non è sufficiente avere come chiavi quelle di PERSONALE e CLIENTE, ma è necessario aggiungere anche la data e l'ora. La

traduzione di questa porzione di schema è, pertanto, la seguente:

- PERSONALE(CodiceFiscale, Nome, Cognome, Via, N.Civico, CAP, Città, Telefono, Email, DataNascita, Titolare(Sì/No), Mansione, Stipendio)
- CLIENTE(IDCliente, Nome, Cognome, Via, N.Civico, CAP, Città, Telefono, Email)
- contatta(CodiceFiscale, IDCliente, Data, Ora)

Infine, sempre in quanto alla traduzione, emergono dei vincoli di integrità referenziale fra gli attributi "CodiceFiscale" e "IDCliente" della relazione "contatta" e le rispettive relazioni "PERSONALE" e "CLIENTE".

- Relazione "prenota": la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.8.

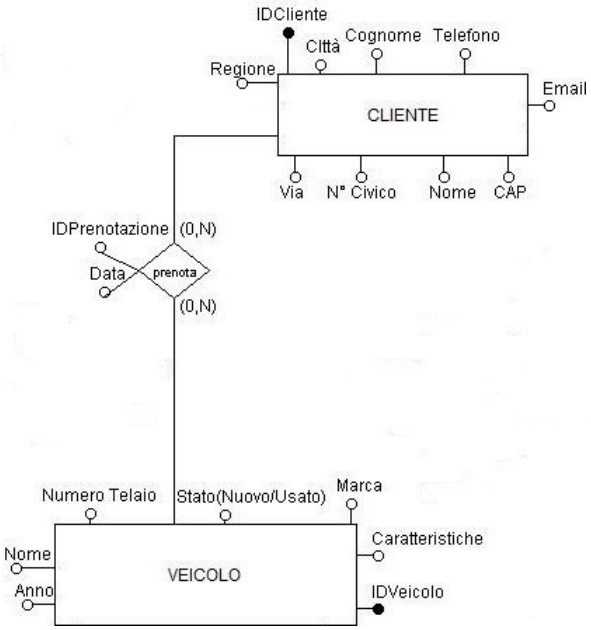


Figura 3.8. Porzione dello schema E/R di Figura 3.1 relativo alla relazione "prenota"

La traduzione di questa porzione di schema è, pertanto, la seguente:

- CLIENTE(IDCliente, Nome, Cognome, Via, N.Civico, CAP, Città, Telefono, Email)
- VEICOLO(IDVeicolo, NumeroTelaio, Stato, Caratteristiche, Anno, Marca, Nome)

– prenota (IDPrenotazione, IDCliente, IDVeicolo, Data)

In seguito alla traduzione emergono dei vincoli di integrità referenziale fra gli attributi “IDCliente” e “IDVeicolo” della relazione “prenota” e le rispettive relazioni “CLIENTE” e “VEICOLO”.

- Relazione “effettua (in entrata)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.9.

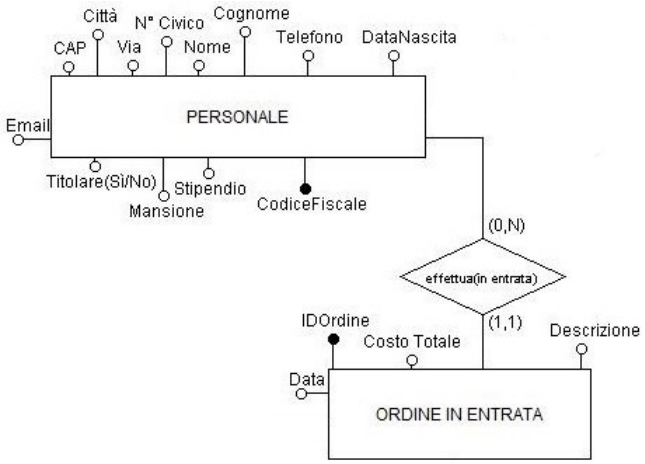


Figura 3.9. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “effettua (in entrata)”

Si tratta di una relazione uno-a-molti. È sufficiente quindi aggiungere il campo “CodiceFiscale” alla relazione “ORDINE IN ENTRATA”. Anche i campi della relazione, nel caso in cui essa li avesse, verranno aggiunti nella relazione “ORDINE IN ENTRATA”. Per maggiore chiarezza rinominiamo l’attributo “CodiceFiscale” di “ORDINE IN ENTRATA” con “CodiceFiscaleAss”.
La traduzione di questa porzione di schema è, pertanto, la seguente:

– PERSONALE (CodiceFiscale, Nome, Cognome, Via, N.Civico, CAP, Città, Telefono, Email, DataNascita, Titolare(Sì/No), Mansione, Stipendio)
– ORDINE IN ENTRATA (IDOrdine, CostoTotale, Data, Descrizione, CodiceFiscaleAss)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra l’attributo “CodiceFiscaleAss” della relazione “ORDINE IN ENTRATA” e la relazione “PERSONALE”.

- Relazione “effettua (in uscita)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.10.

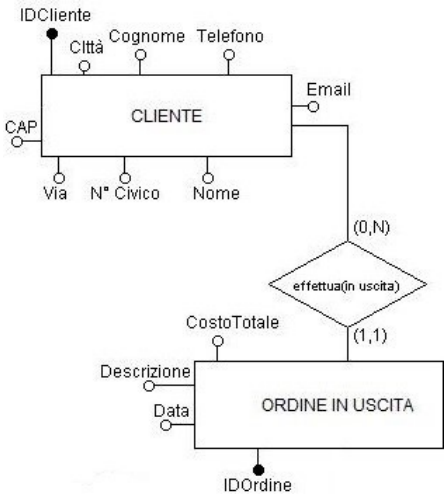


Figura 3.10. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “effettua (in uscita)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- CLIENTE (IDCliente, Nome, Cognome, Via, N.Civico, CAP, Città, Telefono, Email)
- ORDINE IN USCITA (IDOrdine, CostoTotale, Data, Descrizione, IDClienteAss)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra l’attributo “IDClienteAss” della relazione “ORDINE IN USCITA” e la relazione “CLIENTE”.

- Relazione “relativo a (materia prima)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.11.

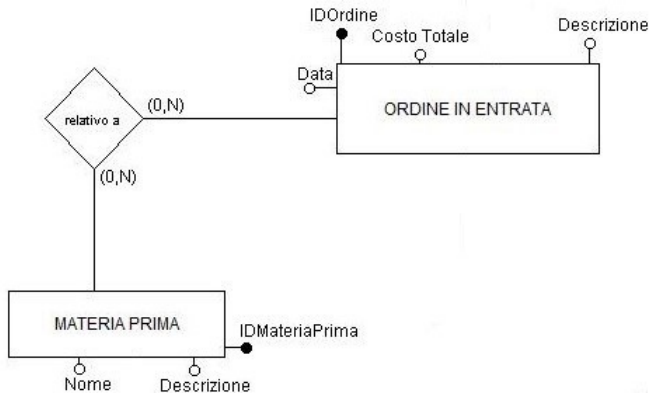


Figura 3.11. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “relativo a (materia prima)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- ORDINE IN ENTRATA (IDOrdine, CostoTotale, Data, Descrizione)
- MATERIA PRIMA (IDMateriaPrima, Nome, Descrizione)
- relativo a (IDOrdine, IDMateriaPrima)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra gli attributi “IDOrdine” e “IDMateriaPrima” della relazione “relativo a” e le rispettive relazioni “ORDINE IN ENTRATA” e “MATERIA PRIMA”.

- Relazione “relativo a (veicolo in entrata)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.12.

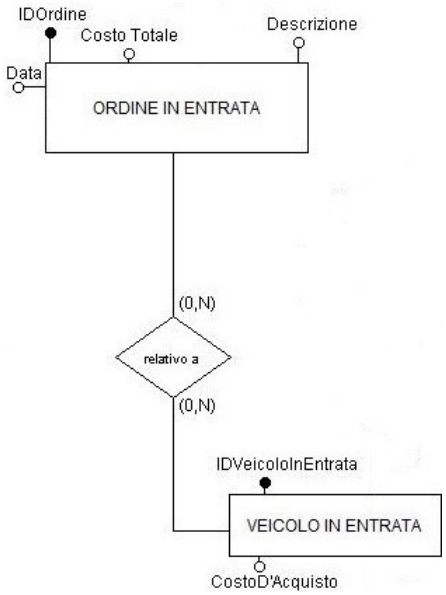


Figura 3.12. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “relativo a (veicolo in entrata)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- ORDINE IN ENTRATA (IDOrdine, CostoTotale, Data, Descrizione)
- VEICOLO IN ENTRATA (IDVeicoloInEntrata, CostoD’Acquisto)
- relativo a (IDOrdine, IDVeicoloInEntrata)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra gli attributi “IDOrdine” e “IDVeicoloInEntrata” della relazione “relativo a” e le rispettive relazioni “ORDINE IN ENTRATA” e “VEICOLO IN ENTRATA”.

- Relazione “relativo a (veicolo in uscita)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.13.

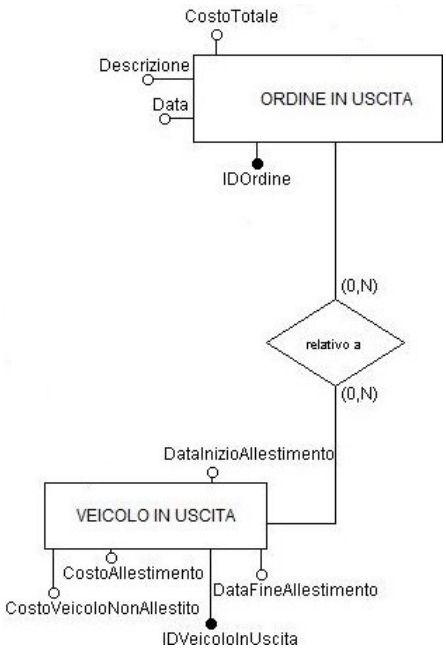


Figura 3.13. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “relativo a (veicolo in uscita)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- ORDINE IN USCITA (IDOrdine, Data, Descrizione, CostoTotale)
- VEICOLO IN USCITA (IDVeicoloInUscita, DataInizioAllestimento, DataFineAllestimento, CostoAllestimento, CostoVeicoloNonAllestito)
- relativo a (IDOrdine, IDVeicoloInUscita)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra gli attributi “IDOrdine” e “IDVeicoloInUscita” della relazione “relativo a” e le rispettive relazioni “ORDINE IN USCITA” e “VEICOLO IN USCITA”.

- Relazione “è (veicolo in entrata)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.14.

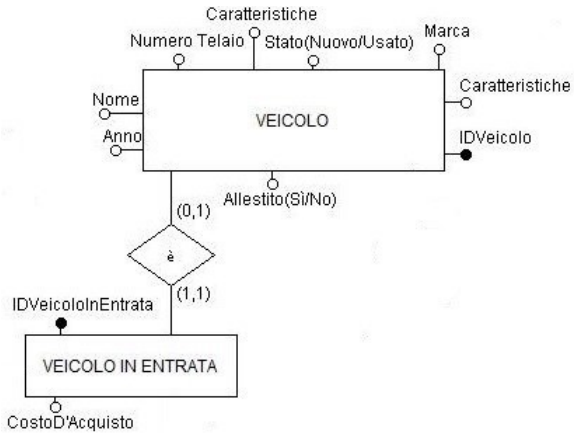


Figura 3.14. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “è (veicolo in entrata)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- VEICOLO (IDVeicolo, Nome, Anno, NumeroTelaio, Caratteristiche, Stato(Nuovo/Usato), Allestito(Sì/No), Marca)
- VEICOLO IN ENTRATA (IDVeicoloInEntrata, CostoD’Acquisto, IDGeneraleVeicolo)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra l’attributo “IDGeneraleVeicolo” della relazione “VEICOLO IN ENTRATA” e l’attributo “IDVeicolo” della relazione “VEICOLO”.

- Relazione “è (veicolo in uscita)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.15.

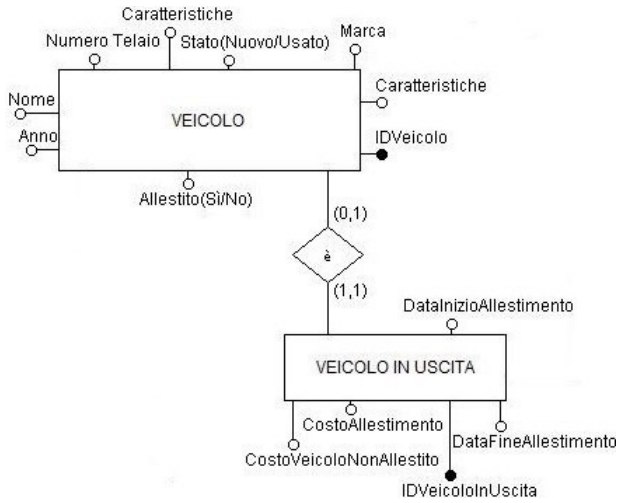


Figura 3.15. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “è (veicolo in uscita)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- VEICOLO (IDVeicolo, Nome, Anno, NumeroTelaio, Caratteristiche, Stato(Nuovo/Usato), Allestito(Sì/No), Marca)
- VEICOLO IN USCITA (IDVeicoloInUscita, DataInizioAllestimento, DataFineAllestimento, CostoAllestimento, CostoVeicoloNonAllestito, IDGeneraleVeicolo)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra l’attributo “IDGeneraleVeicolo” della relazione “VEICOLO IN USCITA” e l’attributo “IDVeicolo” della relazione “VEICOLO”.

- Relazione “fornisce (veicolo)”: la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.16.

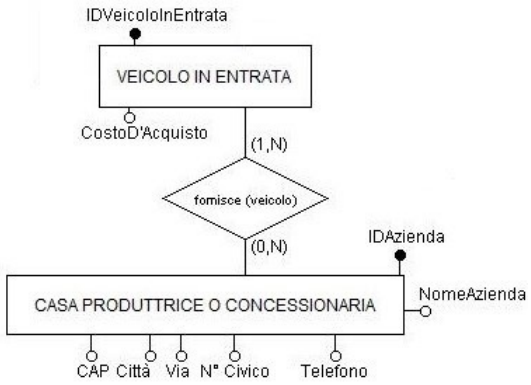


Figura 3.16. Porzione dello schema E/R di Figura 3.1 relativo alla relazione “fornisce (veicolo)”

La traduzione di questa porzione di schema è, pertanto, la seguente:

- VEICOLO IN ENTRATA (IDVeicoloInEntrata, CostoD’Acquisto)
- CASA PRODUTTRICE O CONCESSIONARIA (IDAzienda, NomeAzienda, Telefono, Via, N.Civico, CAP, Città)
- fornisce (veicolo) (IDVeicoloInEntrata, IDAzienda)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra gli attributi “IDVeicoloInEntrata” e “IDAzienda” della relazione “fornisce(veicolo)” e le rispettive relazioni “VEICOLO IN ENTRATA” e “CASA PRODUTTRICE O CONCESSIONARIA”.

- Relazione "fornisce (materia prima)": la porzione dello schema E/R di Figura 3.1 associata a tale relazione viene mostrata nella Figura 3.17.

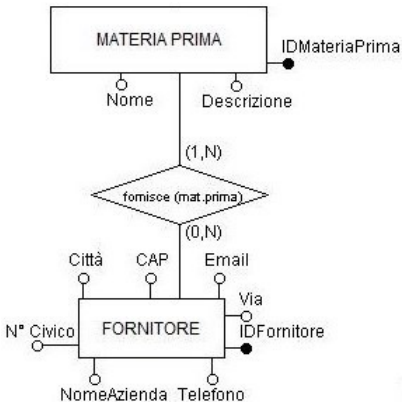


Figura 3.17. Porzione dello schema E/R di Figura 3.1 relativo alla relazione "fornisce (materia prima)"

La traduzione di questa porzione di schema è, pertanto, la seguente:

- MATERIA PRIMA (IDMateriaPrima, Nome, Descrizione)
- FORNITORE (IDFornitore, NomeAzienda, Via, N.Civico, CAP, Città, Email, Telefono)
- fornisce(materia prima) (IDMateriaPrima, IDFornitore)

In seguito alla traduzione emerge un vincolo di integrità referenziale fra gli attributi "IDMateriaPrima" e "IDFornitore" della relazione "fornisce(materia prima)" e le rispettive relazioni "MATERIA PRIMA" e "FORNITORE".

Combinando insieme tutte le traduzioni viste in precedenza si ottiene il seguente schema relazionale completo:

- PERSONALE (CodiceFiscale, Nome, Cognome, Via, N.Civico, CAP, Città, Regione, Telefono, Email, DataNascita, Titolare(Sì/No), Mansione, Stipendio)
- CLIENTE (IDCliente, Nome, Cognome, Via, N.Civico, CAP, Città, Regione, Telefono, Email)
- ORDINE IN ENTRATA (IDOrdine, CostoTotale, Data, Descrizione, QuantitàVeicoli, QuantitàMatPrime, CodiceFiscaleAss)
- ORDINE IN USCITA (IDOrdine, CostoTotale, Data, Descrizione, QuantitàVeicoli, IDClienteAss)
- MATERIA PRIMA (IDMateriaPrima, Nome, Descrizione, CostoUnitario)
- VEICOLO (IDVeicolo, Nome, Anno, NumeroTelaio, Caratteristiche, Stato(Nuovo/Usato), Marca)

- VEICOLO IN ENTRATA (IDVeicoloInEntrata, CostoD'Acquisto, IDGeneraleVeicolo)
- VEICOLO IN USCITA (IDVeicoloInUscita, Allestito(Sì/No), DataInizioAllestimento, DataFineAllestimento, CostoAllestimento, CostoVeicoloNonAllestito, IDGeneraleVeicolo)
- CASA PRODUTTRICE O CONCESSIONARIA (IDAzienda, NomeAzienda, Telefono, Email, Via, N.Civico, CAP, Città, Regione)
- FORNITORE (IDFornitore, NomeAzienda, Via, N.Civico, CAP, Città, Regione, Email, Telefono)
- contatta (CodiceFiscale, IDCliente, Data, Ora)
- prenota(IDPrenotazione, IDCliente, IDVeicolo, Data)
- relativo a(materia prima) (IDOrdine, IDMateriaPrima)
- relativo a(veicolo in entrata) (IDOrdine, IDVeicoloInEntrata)
- relativo a(veicolo in uscita) (IDOrdine, IDVeicoloInUscita)
- fornisce (veicolo) (IDVeicoloInEntrata, IDAzienda)
- fornisce(materia prima) (IDMateriaPrima, IDFornitore)

3.1.3 Progettazione fisica

La fase della progettazione fisica non è un'attività isolata in quanto influenza, ed è a sua volta influenzata, dalla progettazione concettuale, dalla progettazione logica, nonché dalla progettazione delle applicazioni. Infatti, nella progettazione fisica, si fa uso dello schema logico e delle specifiche sulle operazioni per ottimizzare le prestazioni del sistema e renderlo disponibile per l'implementazione.

L'obiettivo di questa fase, dunque, è quello di partire dallo schema logico, generato durante i passi precedenti, e completarlo con la specifica dei parametri fisici di memorizzazione dei dati. Lo schema prodotto da questa fase viene denominato schema fisico e fa riferimento a un modello fisico dei dati che dipende dal DBMS scelto.

Per quanto riguarda l'applicazione oggetto della presente tesi, l'unico aspetto sul quale si è deciso di puntare i riflettori è senza dubbio la scelta degli indici. Il DBMS usato creerà automaticamente, per ogni tabella, gli indici primari sulle chiavi. Nella Figura 3.18 è possibile visualizzare tutti gli indici primari di ogni tabella del nostro DBMS.

Analizzando gli indici presenti, abbiamo ritenuto opportuno creare un vincolo di unicità sul campo "NumeroTelaio" della tabella "VEICOLO". Il vincolo di unicità fa sì che questo campo rispetti la proprietà di univocità pur non essendo chiave primaria.

Oltre a questi indici, però, potrebbe essere utile creare altri indici al fine di aumentare la velocità di accesso ai record mediante selezioni che non riguardano la chiave primaria. In particolare, abbiamo ritenuto opportuno operare le seguenti scelte in merito alla creazione di nuovi indici:

- Sulla relazione PERSONALE non si reputa di dover definire nessun indice ulteriore in quanto, essendo l'azienda di piccole dimensioni, la tabella sarà molto piccola e gli accessi in lettura saranno pochi.

TABELLA	INDICI PRIMARI
PERSONALE	CodiceFiscale
CLIENTE	IDCliente
ORDINE IN ENTRATA	IDOrdine
ORDINE IN USCITA	IDOrdine
MATERIA PRIMA	IDMateriaPrima
VEICOLO	IDVeicolo
VEICOLO IN ENTRATA	IDVeicoloInEntrata
VEICOLO IN USCITA	IDVeicoloInUscita
CASA PRODUTTRICE O CONCESSIONARIA	IDAzienda
FORNITORE	IDFornitore
contatta	COdiceFiscale, IDCliente, Data, Ora
relativo a (mat.prima)	IDOrdine, IDMateriaPrima
relativo a (veicolo in entrata)	IDOrdine, IDVeicoloInEntrata
relativo a (veicolo in uscita)	IDOrdine, IDVeicoloInUscita
fornisce (veicolo)	IDVeicoloInEntrata, IDAzienda
fornisce (mat.prima)	IDMateriaPrima, IDFornitore

Figura 3.18. Elenco degli indici primari relativi ad ogni tabella

- Sulle relazioni FORNITORE e CASA PRODUTTRICE O CONCESSIONARIA riteniamo necessario creare indici secondari sui campi “Regione” appartenenti ad entrambe le tabelle: questo perché l’amministratore potrà spesso avere la necessità di effettuare una ricerca in base alla regione in cui ha sede l’azienda fornitrice di materie prime o veicoli. Riteniamo indispensabile la creazione di questo indice perché, presupponendo che durante l’anno non verranno aggiunti frequentemente nuovi fornitori, gli accessi in lettura saranno molto più frequenti di quelli in scrittura.
- Sulla relazione VEICOLO non definiamo nessun altro indice. Questa scelta è stata dettata da due fattori:
  - questa tabella sarà sicuramente quella su cui faremo più accessi in scrittura;
  - su questa tabella non abbiamo un campo particolare su cui si concentreranno la maggior parte delle nostre ricerche, ma ogni campo ha più o meno la stessa probabilità degli altri di essere oggetto di ricerca.
- Sulle relazioni VEICOLO IN ENTRATA e VEICOLO IN USCITA non definiamo nessun altro indice dal momento che gli accessi saranno effettuati principalmente in base all’attributo definito come chiave primaria.
- Sulla relazione MATERIA PRIMA può rivelarsi molto utile la creazione di un indice costituito da una chiave secondaria formata dal campo “Nome”; questo perché ricorre spesso la ricerca di un articolo per nome.
- Per quanto riguarda gli ordini, sia per la relazione ORDINE IN ENTRATA che per quella ORDINE IN USCITA, oltre all’indice sulla chiave primaria, abbiamo ritenuto utile un indice su più campi; in particolare, per favorire la contabilità è utile definire un indice sugli attributi “CostoTotale” e “Data”. Dato, però, il consistente numero di accessi in lettura ed in scrittura, risulta conveniente, ai

fini dell'ottimizzazione del sistema, cancellare l'indice prima delle operazioni di scrittura e reinserirlo successivamente.
- Per la tabella CLIENTE sceglieremo un indice costituito da una chiave secondaria composta dal campo "Cognome"; poniamo la nostra attenzione su questo campo perché è molto spesso quello prescelto per la ricerca dei clienti da parte degli amministratori.

3.2 Progettazione della componente applicativa

Per progettazione delle applicazioni si intende quella fase della progettazione di un sistema informativo che si occupa di strutturare tutti i programmi che dovranno operare sui dati relativi al sistema informativo stesso. Tale fase della progettazione è, senza dubbio, meno standardizzata rispetto a quella precedente poiché dipende fortemente dalla realtà di riferimento e dal sistema da realizzare. Per condurre tale attività si usano, generalmente, degli approcci poco formali.

Generalmente, in tali approcci, si possono riscontrare le seguenti tre macro-fasi:

- definizione dell'architettura del sistema;
- definizione dei mock-up;
- progettazione dei process flow.

3.2.1 Definizione dell'architettura del sistema

Durante questa fase si cerca di individuare uno schema ad albero, o a grafo, che rappresenti quali sono i moduli che compongono un determinato sistema e come essi interagiscono tra loro. Lo schema che ne deriva ricorda molto da vicino le cosiddette "mappe dei siti" che spesso si trovano in molti portali. Le mappe dei siti possono essere profonde o superficiali, larghe o piccole, ad albero, combinate, a stella o a tab, ma devono essere autoesplicative! Nella Figura 3.19 possiamo vedere una rappresentazione superficiale, ma molto chiara, che descrive la strutturazione del nostro sito.

Nelle Figure 3.20-3.24, invece, abbiamo gli schemi dei link[2] distinti nei quattro livelli di utenza in cui è suddivisa la nostra applicazione web: "public", "user", "staff" e "admin". Questi schemi rappresentano un livello profondo di mappe dei siti poiché viene indicato il path[3] di ogni pagina del nostro sito (è doveroso sottolineare che le pagine corrispondenti a funzioni comuni a diversi livelli di utenza non sono state ripetute).

[2] Lo schema dei link è un componente fondamentale nella progettazione di un sito web e contiene tutti i collegamenti (con i relativi percorsi) alle pagine del sito partendo da una pagina di partenza. Nei nostri schemi di link si è scelto di prendere come pagina "root" di ogni schema di link la home page di ogni livello di utenza.

[3] Come si può vedere, negli schemi dei link è riportata solo l'ultima parte di ogni path, cioè quella formata da <controller>/<action>/<nome parametro>/<valore parametro>. Questi particolari path sono dovuti all'uso del Framework Zend che, infatti, non utilizza lo stesso meccanismo di routing del PHP, ma utilizza un meccanismo di routing su base URL.

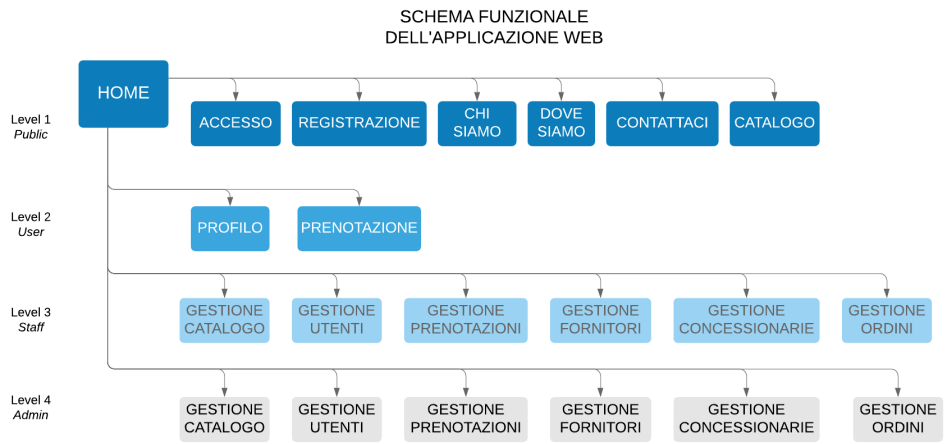


Figura 3.19. Schema funzionale che rappresenta la mappa dell'applicazione web

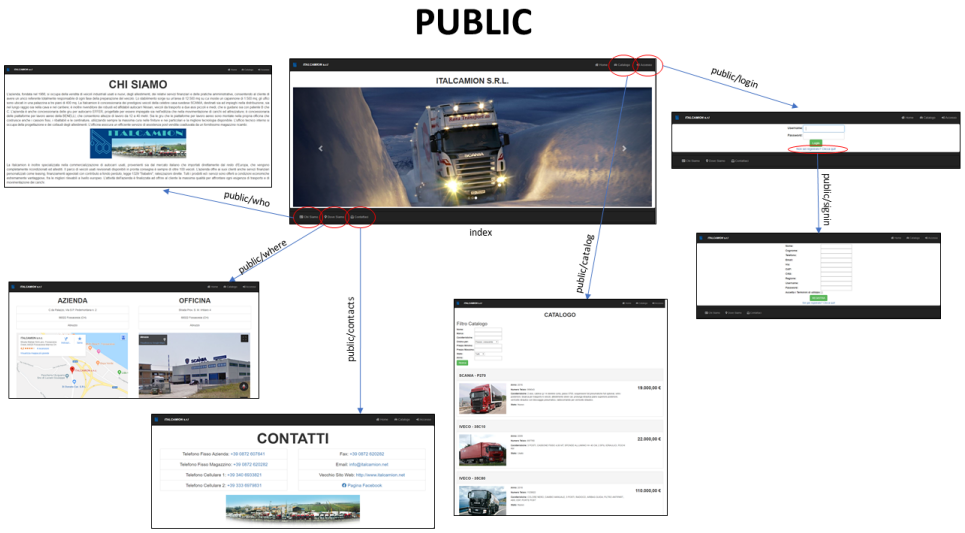


Figura 3.20. Schema dei link relativo alla sezione "pubblica" dell'applicazione web

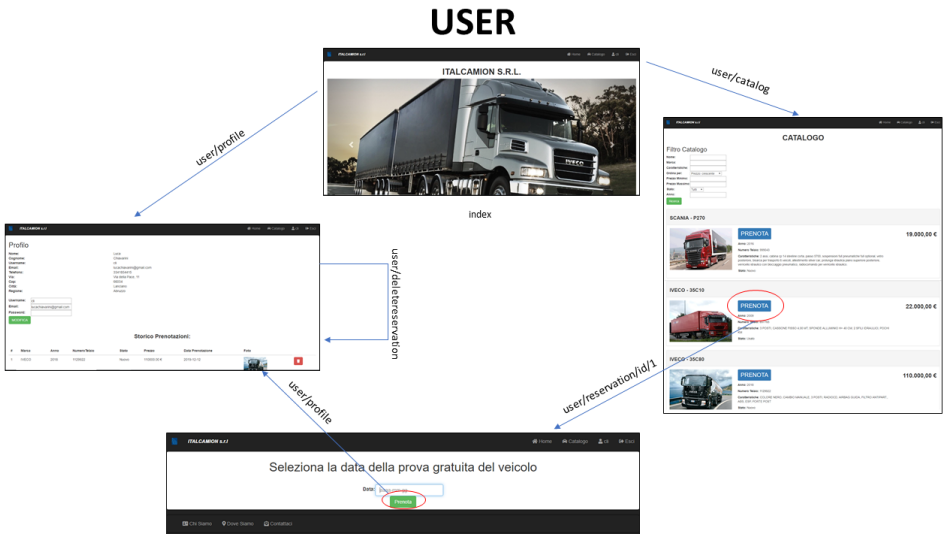


Figura 3.21. Schema dei link relativo alla sezione "utente" dell'applicazione web

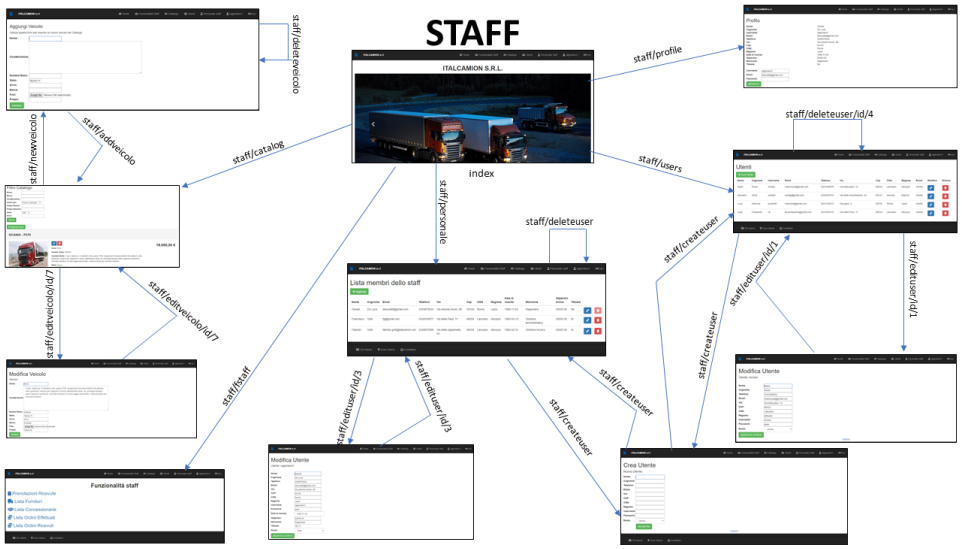


Figura 3.22. Schema dei link relativo alla sezione "staff" dell'applicazione web

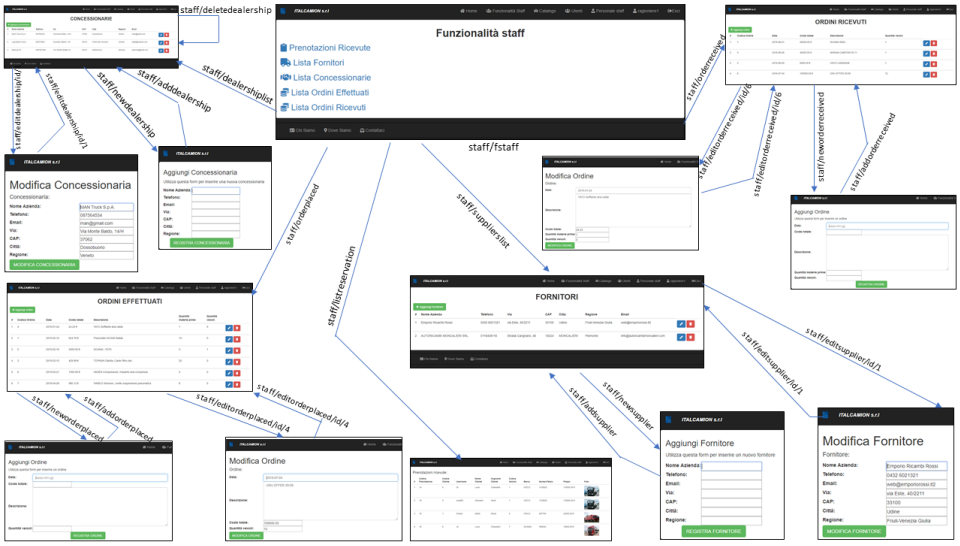


Figura 3.23. Schema dei link relativo alla restante sezione "staff" dell'applicazione web (non specificata nella figura precedente)

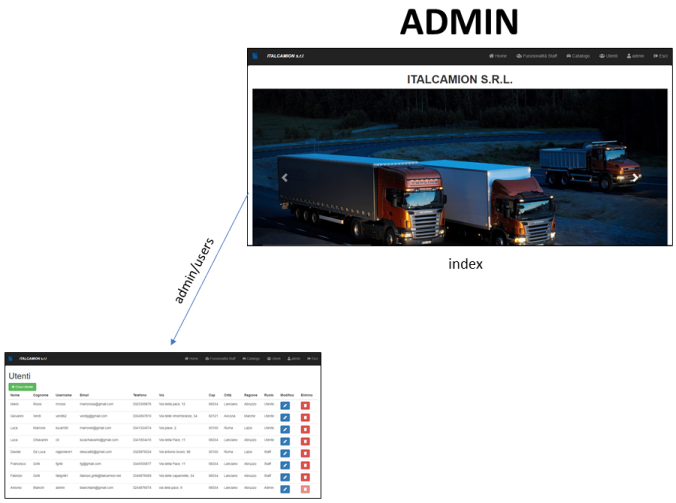


Figura 3.24. Schema dei link relativo alla sezione "admin" dell'applicazione web

3.2.2 Definizione dei mock-up

I mock-up possono essere definiti come illustrazioni schematiche del contenuto delle pagine del sito, composti da linee pulite e senza grafica. Essi risultano semplici da realizzare, ma complessi nella sostanza, perché rappresentano il legame tra l'idea, le

intenzioni e gli obiettivi da una parte, e il display finale dall'altra. Dunque possiamo identificare i mock-up come un modello dimostrativo dell'oggetto originale che verrà realizzato, con l'obiettivo di dare un'idea dell'oggetto finito.

I mock-up hanno la funzione di:

- comunicare l'idea iniziale del progetto;
- focalizzare l'attenzione solo su architettura e contenuti, senza le "distrazioni" della grafica;
- comunicare cosa si vedrà;
- essere la base del prototipo.

Per ciò che riguarda l'applicazione web oggetto della presente tesi:

- Nelle Figure 3.25-3.31 vengono mostrati i mock-up del livello di utenza "public".
- Nelle Figure 3.32-3.34 vengono mostrati i mock-up del livello di utenza "user".
- Nelle Figure 3.35-3.44 vengono mostrati i mock-up del livello di utenza "staff".

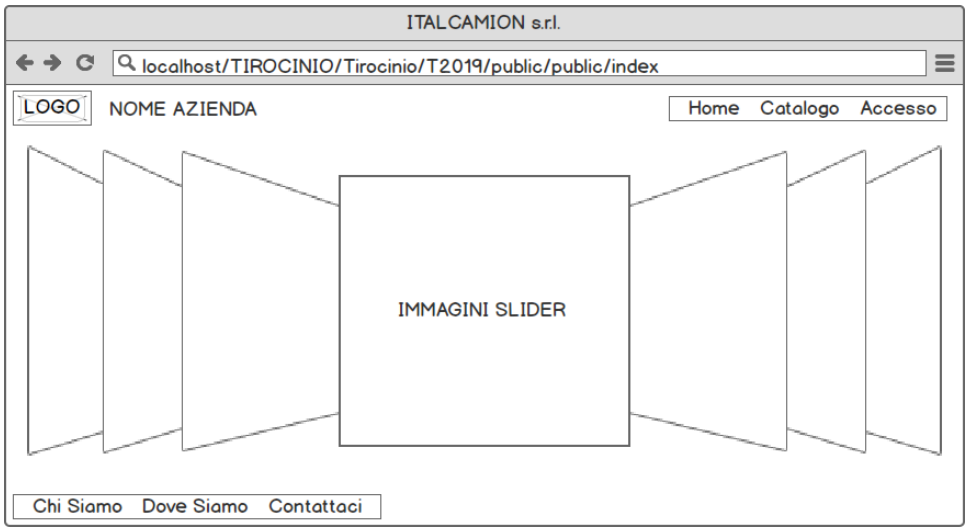


Figura 3.25. Mock-up relativo alla home page della sezione "public" del sito

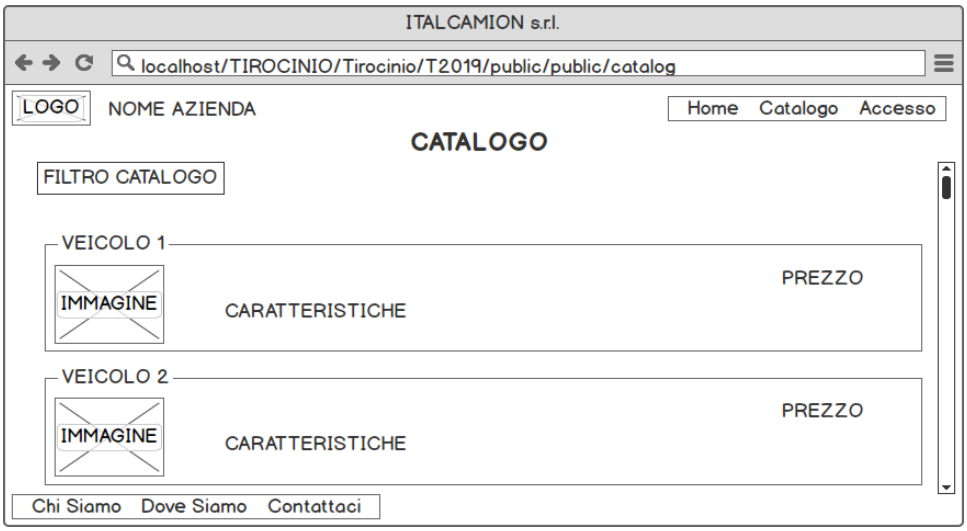


Figura 3.26. Mock-up relativo alla pagina "catalog" della sezione "public"; in questa pagina è possibile visualizzare i veicoli e/o filtrarli secondo molteplici parametri

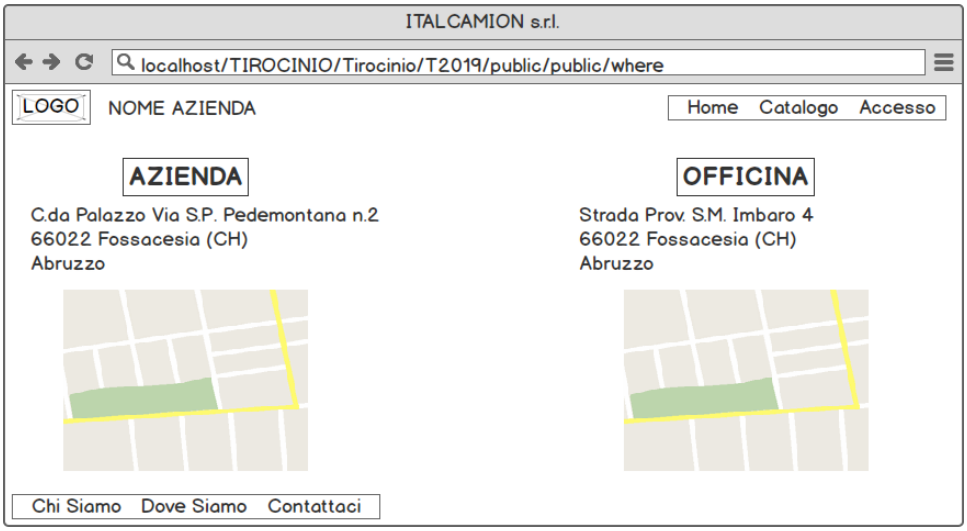


Figura 3.27. Mock-up relativo alla pagina "where" della sezione "public"; in questa pagina troviamo tutte le informazioni relative al luogo in cui è dislocata l'azienda

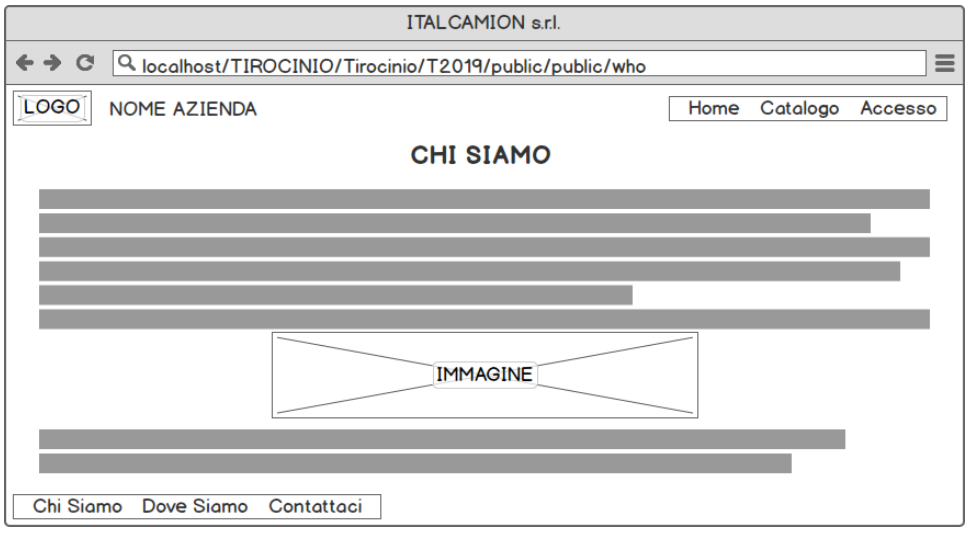


Figura 3.28. Mock-up relativo alla pagina "who" della sezione "public"; in questa pagina troviamo tutte le informazioni relative alla storia, all'organizzazione e all'obiettivo dell'azienda

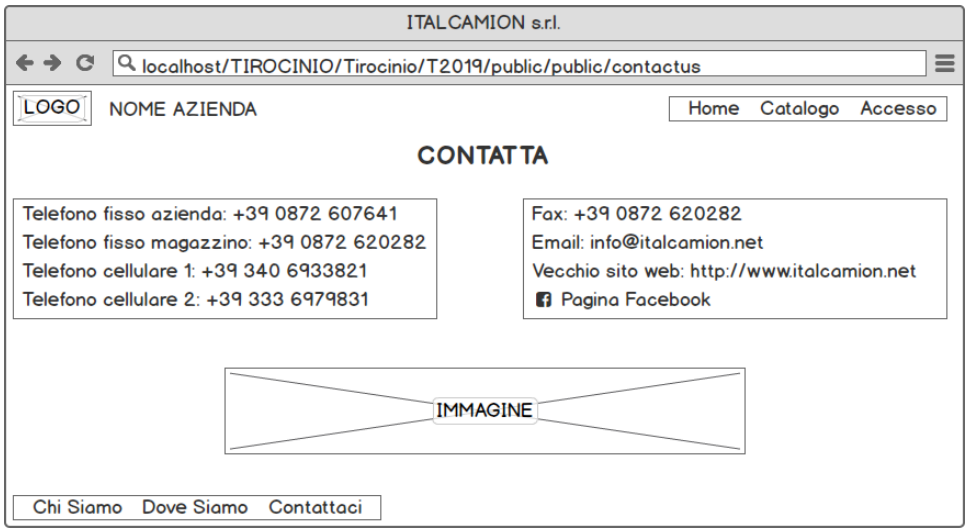


Figura 3.29. Mock-up relativo alla pagina "contactus" della sezione "public"; in questa pagina troviamo tutti i contatti dell'azienda

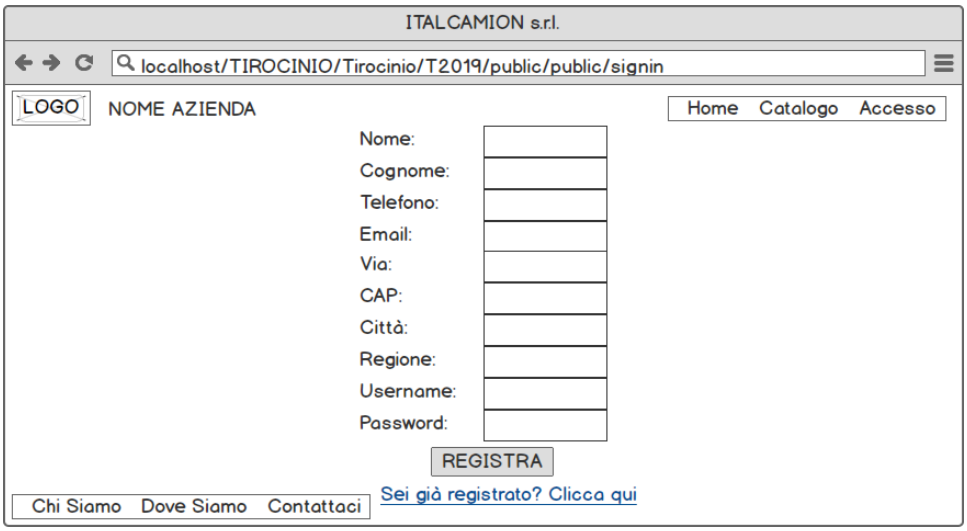


Figura 3.30. Mock-up relativo alla pagina "signin" della sezione "public"; questa pagina consente ad un utente di registrarsi al sito

Figura 3.31. Mock-up relativo alla pagina "login" della sezione "public"; questa pagina consente ad un utente di accedere al sito

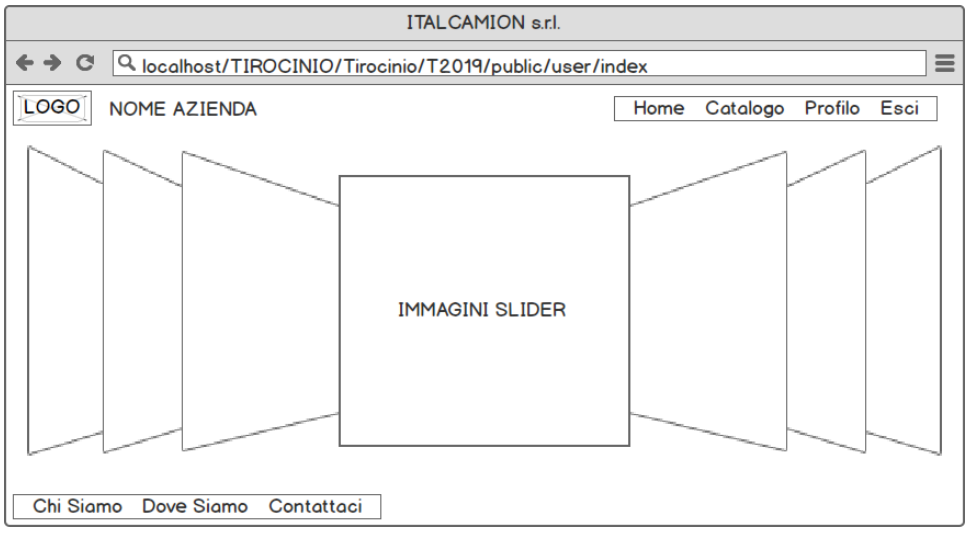


Figura 3.32. Mock-up relativo alla home page della sezione "user" del sito

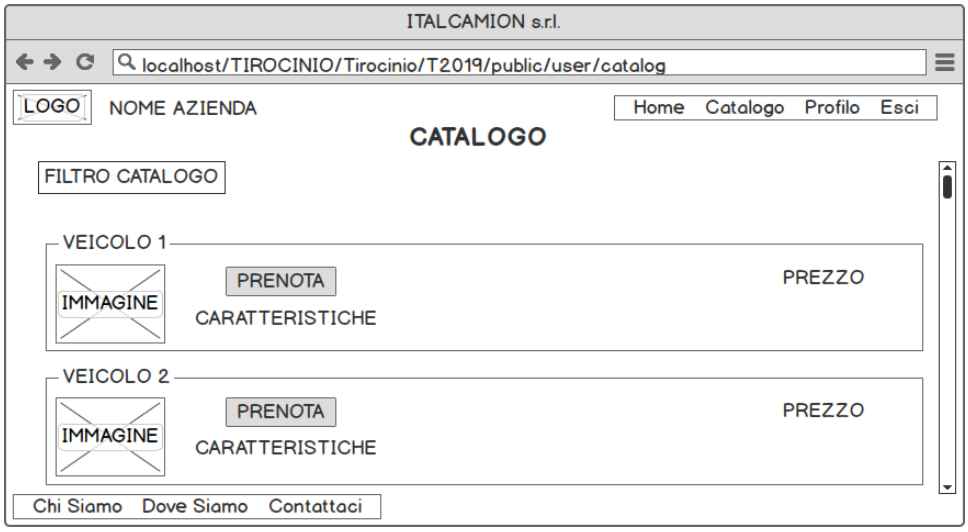


Figura 3.33. Mock-up relativo alla pagina "catalog" della sezione "user"; da questa pagina il cliente, oltre alle funzionalità già descritte per il livello di utenza "public", può prenotare la prova gratuita del veicolo selezionato

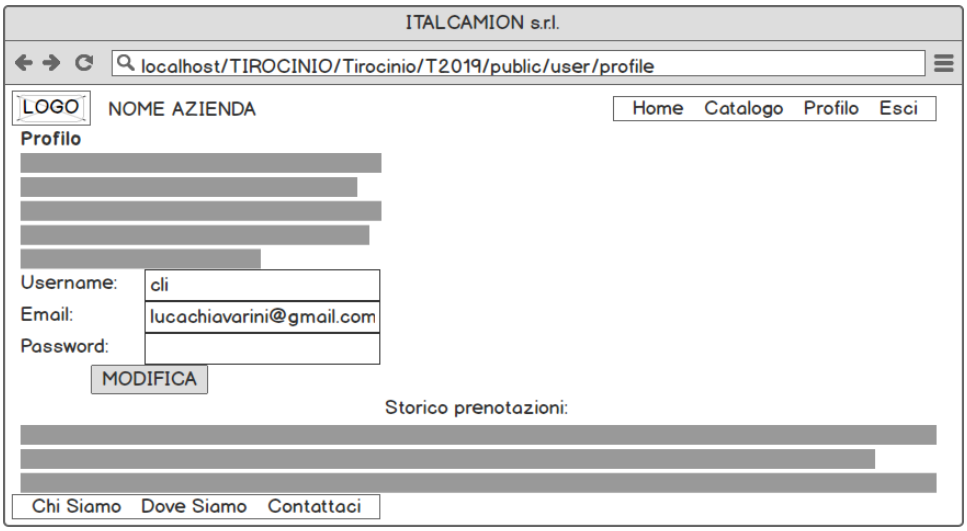


Figura 3.34. Mock-up relativo alla pagina "profile" della sezione "user"; da questa pagina l'utente, dopo essersi autenticato, può visualizzare e modificare le proprie informazioni personali. In essa, inoltre, è possibile visualizzare il riepilogo delle prenotazioni effettuate

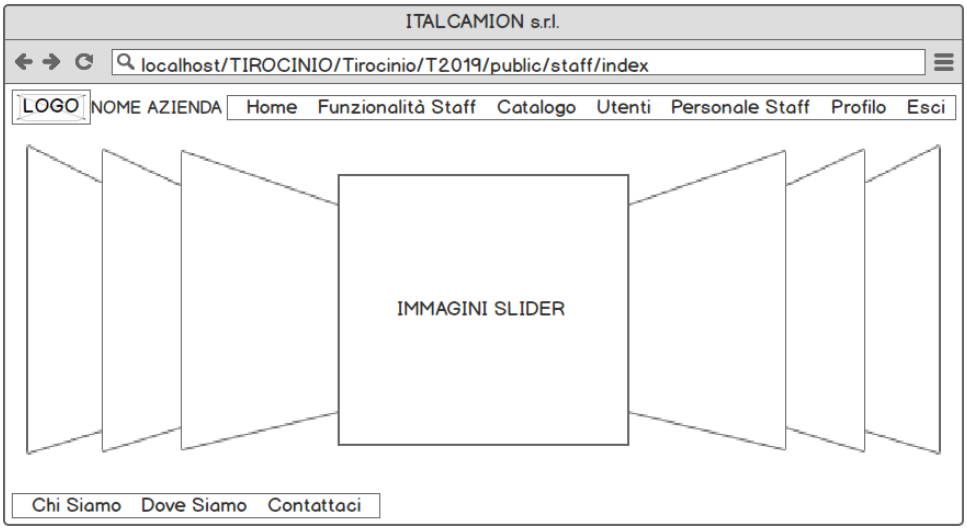


Figura 3.35. Mock-up relativo alla home page della sezione "staff" del sito

Figura 3.36. Mock-up relativo alla pagina "catalog" della sezione "staff"; tramite questa pagina un componente dello staff può aggiungere, modificare o eliminare un veicolo

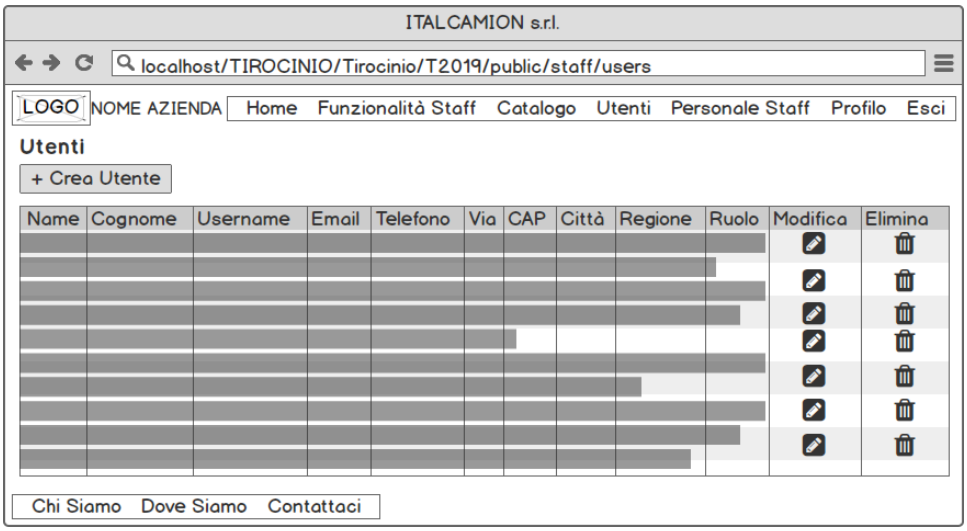


Figura 3.37. Mock-up relativo alla pagina "users" della sezione "staff"; tramite questa pagina un componente dello staff ha la possibilità di visualizzare, aggiungere, modificare o eliminare clienti iscritti al sito

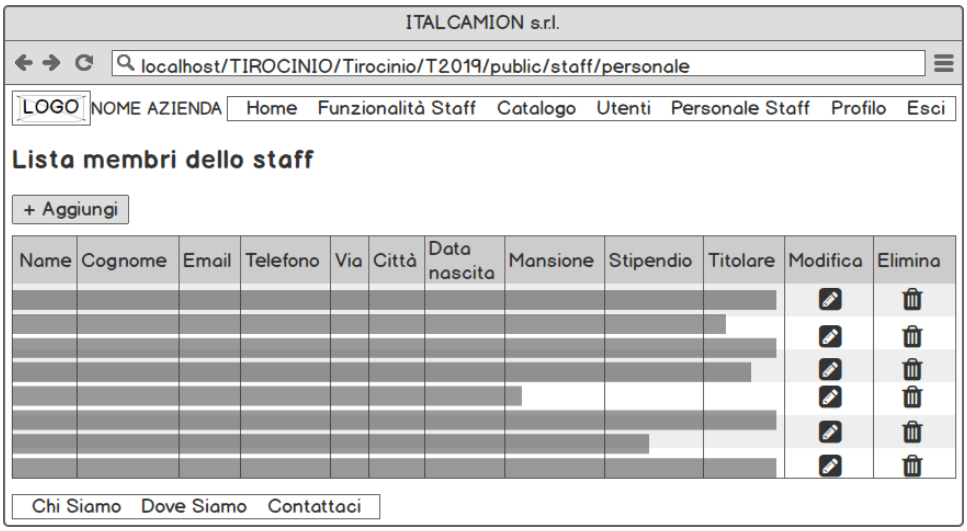


Figura 3.38. Mock-up relativo alla pagina "personale" della sezione "staff"; questa pagina contiene le informazioni relative a tutti i componenti dello staff (quindi di tutto il personale dell'azienda). Tramite essa è possibile, inoltre, aggiungere, modificare o eliminare un componente dello staff

Figura 3.39. Mock-up relativo alla pagina "fstaff" della sezione "staff"; questa pagina racchiude i link per le funzionalità più importanti di un utente di livello "staff"

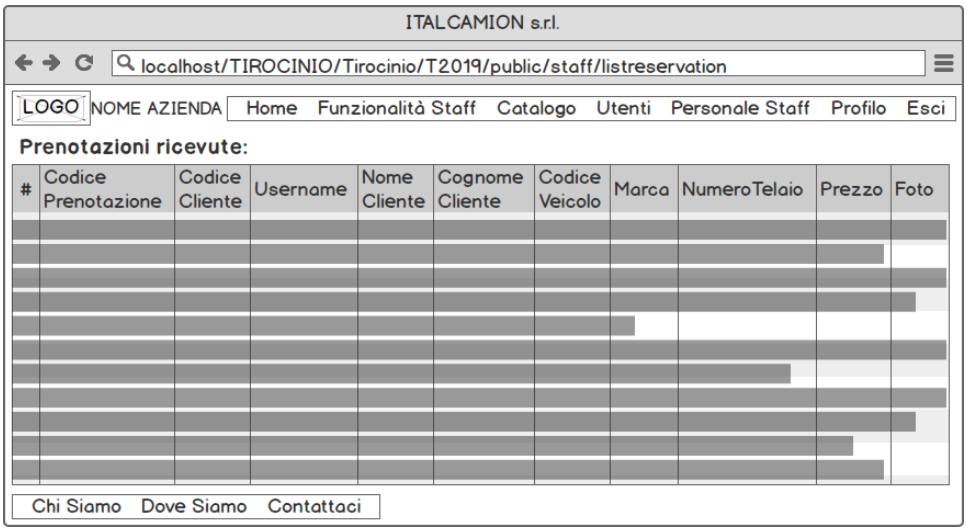


Figura 3.40. Mock-up relativo alla pagina "listreservation" della sezione "staff"; questa pagina contiene la lista di tutte le prenotazioni e, per ciascuna di essa, vengono mostrate le informazioni sul cliente che l'ha effettuata, sul veicolo prenotato e sulla prenotazione ricevuta

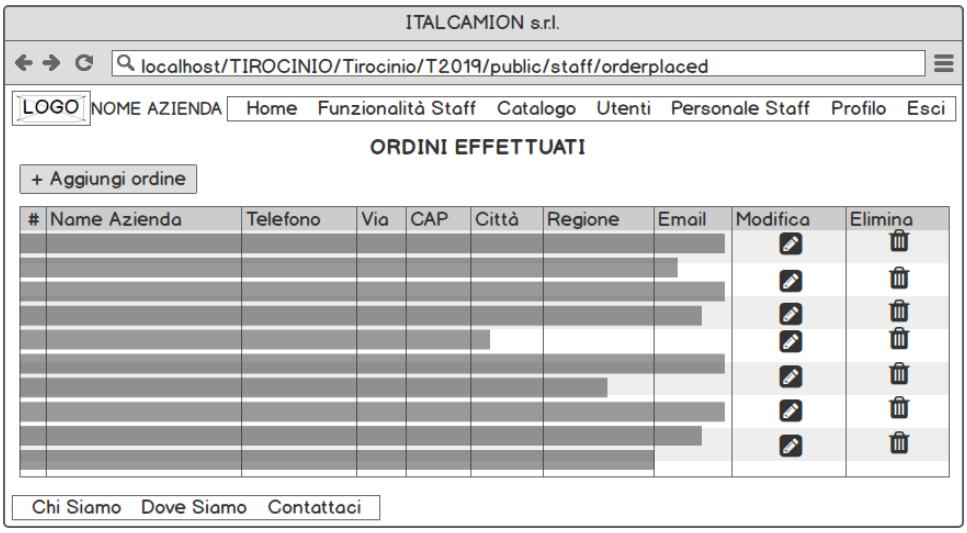


Figura 3.41. Mock-up relativo alla pagina "orderplaced" della sezione "staff"; questa pagina contiene la lista di tutti gli ordini effettuati dall'azienda e dà la possibilità di aggiungere, modificare o eliminare un ordine

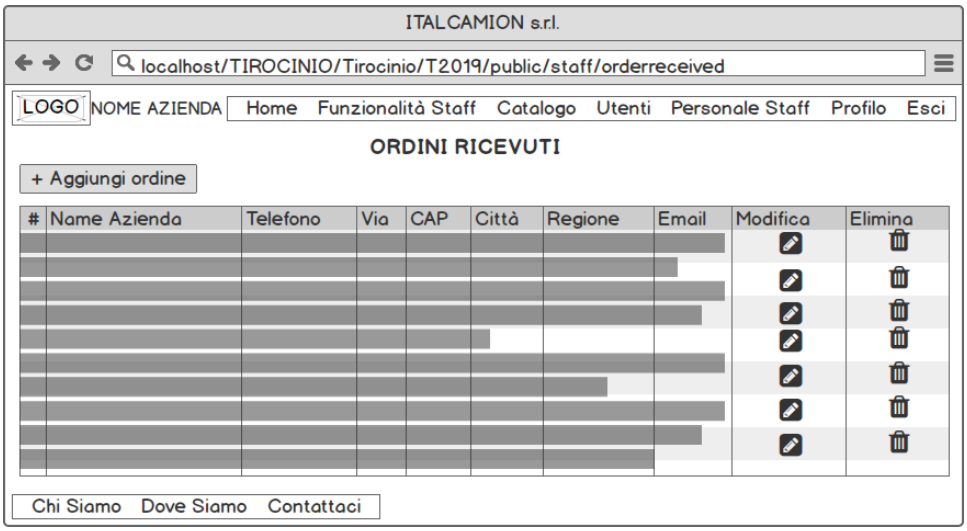


Figura 3.42. Mock-up relativo alla pagina "orderreceived" della sezione "staff"; questa pagina contiene la lista di tutti gli ordini ricevuti dall'azienda (che, quindi, sono stati effettuati dal cliente) e dà la possibilità di aggiungere, modificare o eliminare un ordine

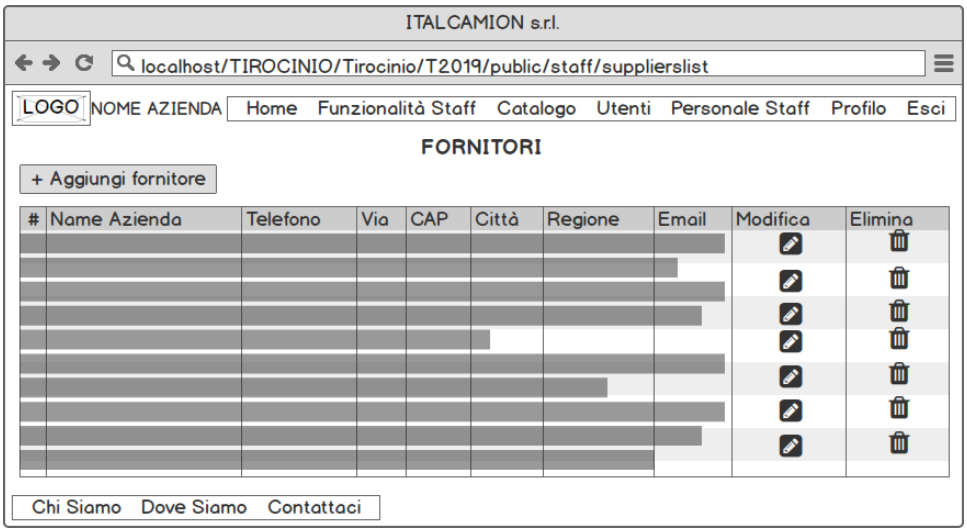


Figura 3.43. Mock-up relativo alla pagina "supplierslist" della sezione "staff"; questa pagina contiene la lista di tutti i fornitori da cui l'azienda acquista le materie prime utili all'allestimento degli automezzi. Da qui, inoltre, è possibile aggiungere, modificare o eliminare un fornitore

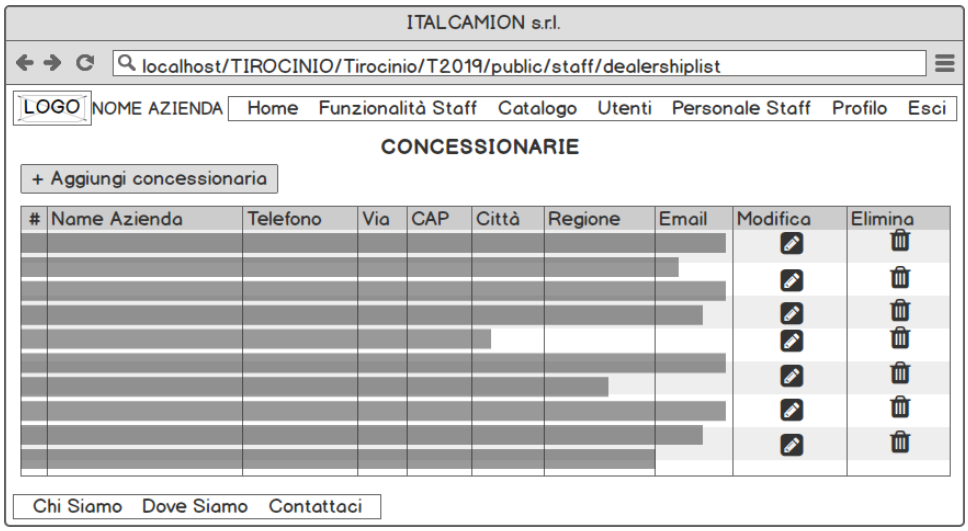


Figura 3.44. Mock-up relativo alla pagina "dealershiplist" della sezione "staff"; questa pagina contiene la lista di tutte le concessionarie e di tutte le case produttrici da cui l'azienda acquista i veicoli (nuovi o usati). Da qui, inoltre, è possibile aggiungere, modificare o eliminare una concessionaria o una casa produttrice

È importante sottolineare che per ogni livello di utenza alcuni mock-up vengono "ereditati" dal livello di utenza precedente e, per questo motivo, non sono stati riportati (ad esempio, il mock-up della pagina "where" è lo stesso per ogni livello di utenza e, quindi, l'abbiamo inserito una volta sola).

In modo analogo si è proceduto per i mock-up del livello di utenza "admin" che non sono stati inseriti poiché sono strutturalmente uguali a quelli del livello "staff"; le sottili differenze che ci sono fra le due classi di utenza si evinceranno solo nella fase di esecuzione dell'applicazione web.

3.2.3 Progettazione dei Process Flow

Il Process Flow è un documento altamente dinamico, che ferma la relazione tra utenti e sito Web attraverso il movimento e l'interazione. La differenza essenziale con le mappe è negli step, che non sono strutturali, gerarchici o multidimensionali, ma possono essere solo sequenziali. Infatti, mentre le mappe possono escludere la user experience dalle fasi di costruzione, nei process flow tale aspetto diventa dominante e accompagna il processo dall'inizio alla fine. I process flow descrivono come gli utenti raggiungono (o non raggiungono) l'obiettivo attraverso un percorso dettagliato ne dettano il funzionamento. Ogni utente agisce in maniera diversa e presenta motivazioni ed esperienze differenti; compito dei process flow è normalizzare tale processo.

Quando lavoriamo su un process flow dobbiamo stabilire a priori uno scenario, un utente tipo e un obiettivo. Ad esempio: Anna (utente tipo) deve cambiare il proprio contratto di telefonia mobile (obiettivo) ed è preoccupata perché è la prima volta che decide di farlo online tramite il sito (scenario).

Nelle Figure 3.45-3.53 vengono raffigurati i vari process flow dell'applicazione web realizzata.

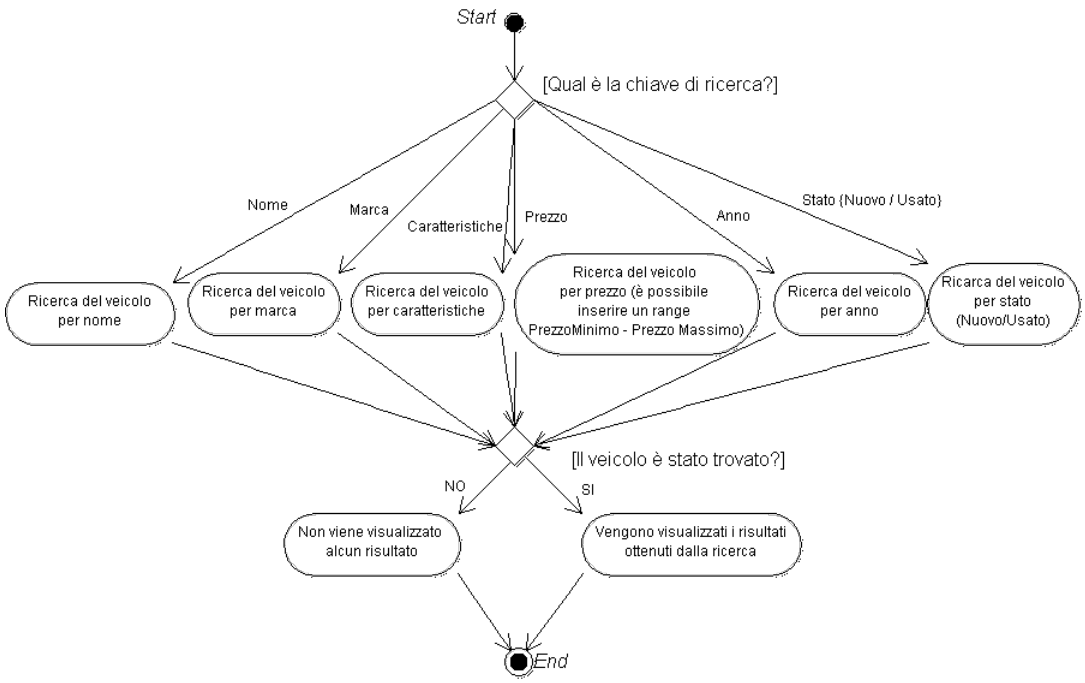


Figura 3.45. Process Flow relativo all'attività di ricerca di un veicolo da parte di un utente appartenente a qualsiasi livello di utenza

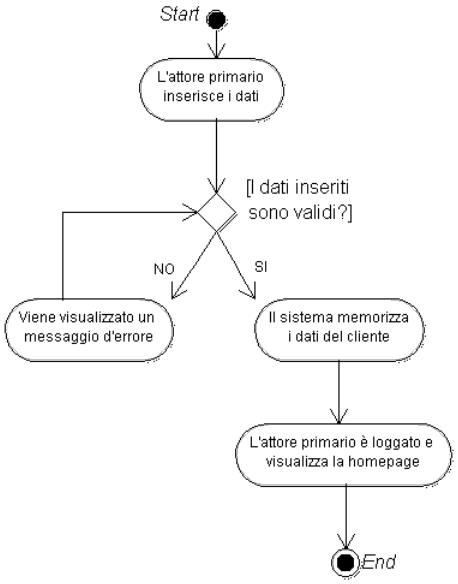


Figura 3.46. Process Flow relativo all'attività di registrazione da parte di un nuovo cliente

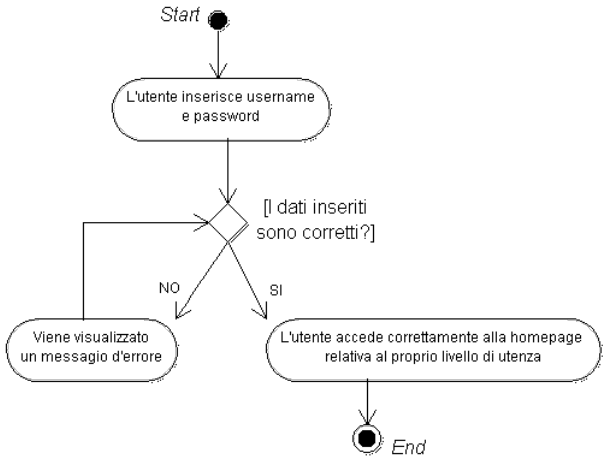


Figura 3.47. Process Flow relativo all'attività di login di un utente

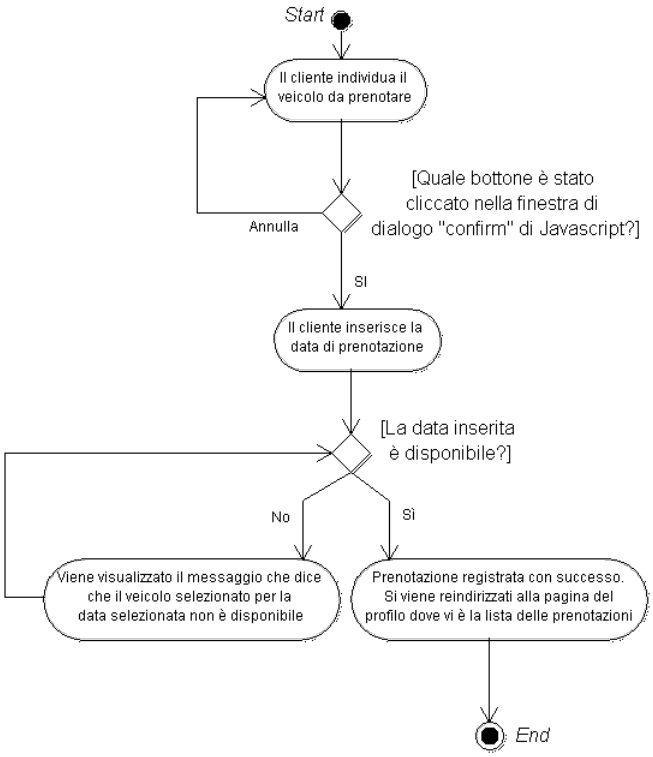


Figura 3.48. Process Flow relativo all'attività di prenotazione della prova gratuita di un veicolo da parte di un cliente già registrato

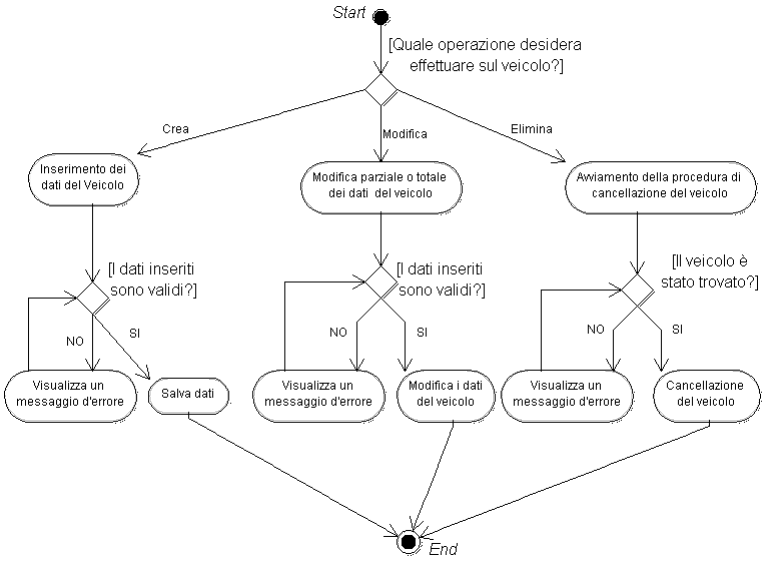


Figura 3.49. Process Flow relativo alle attività di creazione, modifica e cancellazione di un veicolo da parte di un componente dello staff

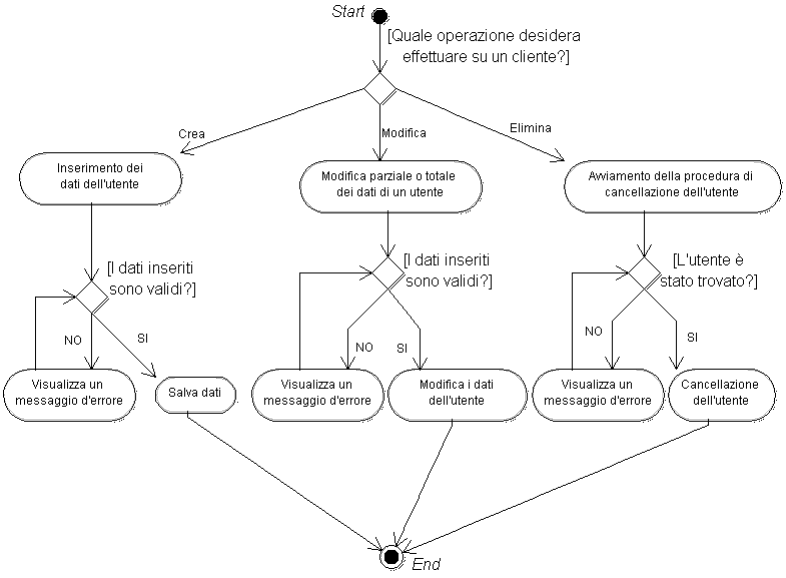


Figura 3.50. Process Flow relativo alle attività di creazione, modifica e cancellazione di un utente da parte di un componente dello staff

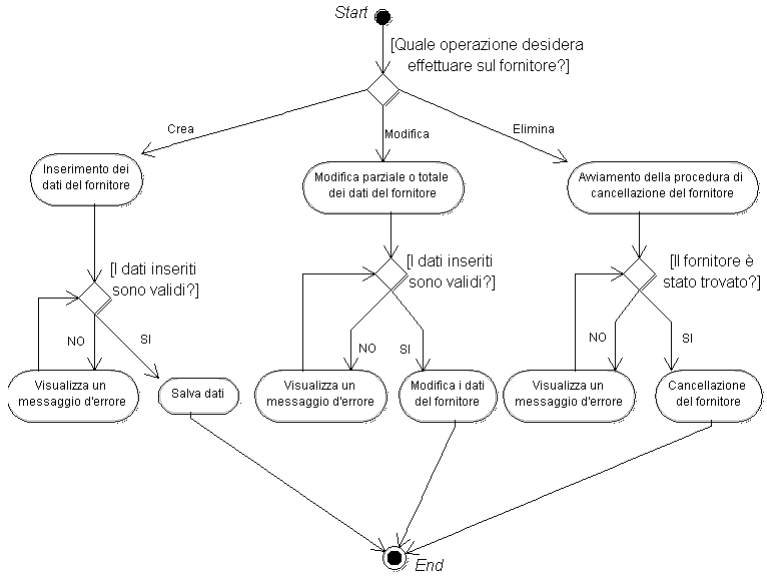


Figura 3.51. Process Flow relativo alle attività di creazione, modifica e cancellazione di un fornitore da parte di un componente dello staff

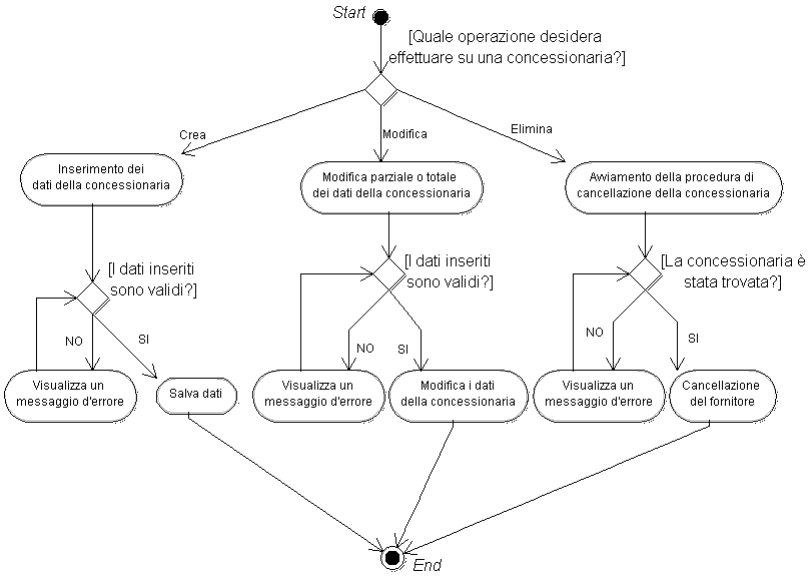


Figura 3.52. Process Flow relativo alle attività di creazione, modifica e cancellazione di una concessionaria da parte di un componente dello staff

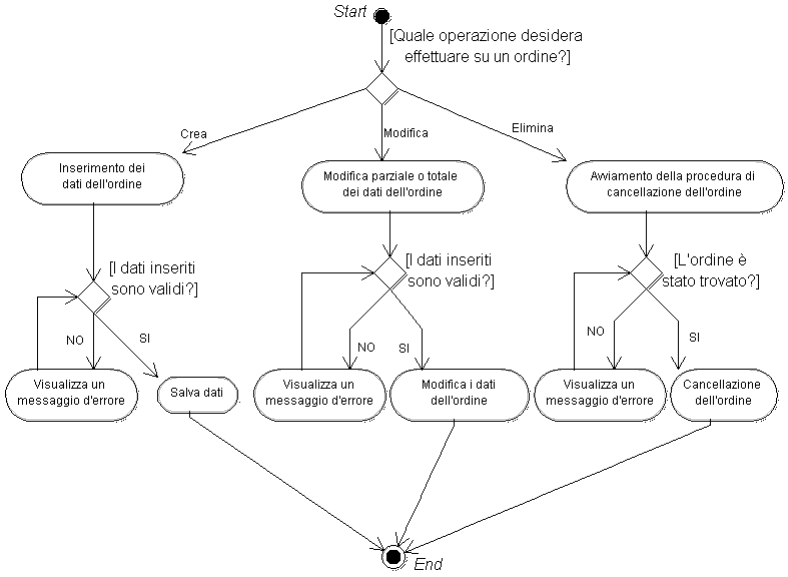


Figura 3.53. Process Flow relativo alle attività di creazione, modifica e cancellazione di un ordine in entrata (o in uscita), da parte di un componente dello staff

4

Implementazione del sistema

In questo capitolo procederemo con l'implementazione il sistema informativo secondo la struttura e le caratteristiche definite durante la fase di progettazione realizzata nei capitoli precedenti. Il capitolo verrà diviso in due macro-aree, ovvero l'implementazione del database e l'implementazione dell'applicazione Web.

4.1 Implementazione della base di dati

In questa sezione, attraverso il linguaggio SQL, verrà mostrato il codice relativo all'implementazione della base di dati. In particolare, verranno esposti i listati relativi alla creazione del database (Listato 4.1), alla creazione delle varie tabelle (Listati 4.2-4.18), all'implementazione dei vincoli di unicità sui campi (Listato 4.19) e all'implementazione di ulteriori indici sui campi (Listati 4.20-4.25).

```
1          CREATE DATABASE db_italcamion;
```

Listato 4.1. Creazione del database

```
1    CREATE TABLE personale(
2    CodiceFiscale VARCHAR(16) NOT NULL PRIMARY KEY,
3    Nome VARCHAR(50) NOT NULL,
4    Cognome VARCHAR(50) NOT NULL,
5    Via VARCHAR(50) NOT NULL,
6    NCivico VARCHAR(10) NOT NULL,
7    CAP CHAR(5) NOT NULL,
8    Citta VARCHAR(50) NOT NULL,
9    Regione VARCHAR(50) NOT NULL,
10   Telefono VARCHAR(15) NOT NULL,
11   Email VARCHAR(100) NOT NULL,
12   DataNascita DATE NOT NULL,
13   Titolare CHAR(2) NOT NULL,
14   Mansione VARCHAR(100),
15   Stipendio FLOAT
16   );
```

Listato 4.2. Creazione della tabella `PERSONALE`

```
1    CREATE TABLE Cliente(
2    IDCliente INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3    Nome VARCHAR(50) NOT NULL,
4    Cognome VARCHAR(50) NOT NULL,
5    Via VARCHAR(50) NOT NULL,
```

```
6   NCivico VARCHAR(10) NOT NULL,
7   CAP CHAR(5) NOT NULL,
8   Citta VARCHAR(50) NOT NULL,
9   Regione VARCHAR(50) NOT NULL,
10  Telefono VARCHAR(15) NOT NULL,
11  Email VARCHAR(100) NOT NULL
12  );
```

Listato 4.3. Creazione della tabella `UTENTE`

```
1   CREATE TABLE Ordine_In_Entrata(
2   IDOrdine INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   CostoTotale FLOAT UNSIGNED NOT NULL,
4   Data date NOT NULL,
5   Descrizione VARCHAR(100) NOT NULL,
6   QuantitaVeicoli INT UNSIGNED NOT NULL,
7   QuantitaMatPrime INT UNSIGNED NOT NULL,
8   CodiceFiscaleAss VARCHAR(16) NOT NULL references personale(codicefiscale)
9   );
```

Listato 4.4. Creazione della tabella `ORDINE_IN_ENTRATA`

```
1   CREATE TABLE Ordine_In_Uscita(
2   IDOrdine int auto_increment NOT NULL primary key,
3   CostoTotale FLOAT UNSIGNED NOT NULL,
4   Data date NOT NULL,
5   Descrizione VARCHAR(100) NOT NULL,
6   QuantitaVeicoli INT UNSIGNED NOT NULL,
7   IDClienteAss int NOT NULL REFERENCES cliente(IDCLiente)
8   );
```

Listato 4.5. Creazione della tabella `ORDINE_IN_USCITA`

```
1   CREATE TABLE Materia_prima(
2   IDMateriaPrima INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   Nome VARCHAR(50) NOT NULL,
4   Descrizione VARCHAR (100) NOT NULL,
5   CostoUnitario FLOAT UNSIGNED NOT NULL
6   );
```

Listato 4.6. Creazione della tabella `MATERIA_PRIMA`

```
1   CREATE TABLE Veicolo(
2   IDVeicolo INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   Nome VARCHAR(50) NOT NULL,
4   Anno CHAR(4) NOT NULL,
5   NumeroTelaio CHAR(17) NOT NULL,
6   Caratteristiche VARCHAR(100) NOT NULL,
7   Stato CHAR(5) NOT NULL,
8   Marca VARCHAR(25) NOT NULL
9   );
```

Listato 4.7. Creazione della tabella `VEICOLO`

```
1   CREATE TABLE Veicolo_In_Entrata(
2   IDVeicoloInEntrata INT AUTO_INCREMENT NOT NULL primary key,
3   CostoDAcquisto FLOAT UNSIGNED NOT NULL,
4   IDGeneraleVeicolo INT NOT NULL REFERENCES Veicolo(IDVeicolo)
5   );
```

Listato 4.8. Creazione della tabella `VEICOLO_IN_ENTRATA`

```
1   CREATE TABLE dati_Veicolo_In_Uscita(
2   IDVeicoloInUscita INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   DataInizioAllestimento DATE,
4   DataFineAllestimento DATE,
```

```
5   CostoAllestimento FLOAT UNSIGNED DEFAULT 0,
6   CostoVeicoloNonAllestito FLOAT UNSIGNED NOT NULL,
7   Allestito CHAR(2) DEFAULT 'No',
8   IDGeneraleVeicolo int NOT NULL REFERENCES Veicolo(IDVeicolo)
9   );
```

Listato 4.9. Creazione della tabella VEICOLO_IN_USCITA

```
1   CREATE TABLE Casa_Produttrice_Concessionaria(
2   IDAzienda INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   NomeAzienda VARCHAR (50) NOT NULL,
4   Via VARCHAR(50) NOT NULL,
5   NCivico VARCHAR(10) NOT NULL,
6   CAP CHAR(5) NOT NULL,
7   Citta VARCHAR(50) NOT NULL,
8   Regione VARCHAR(50) NOT NULL,
9   Telefono VARCHAR(15) NOT NULL,
10  Email VARCHAR(100) NOT NULL
11  );
```

Listato 4.10. Creazione della tabella CASA_PRODUTTRICE_CONCESSIONARIA

```
1   CREATE TABLE fornitore(
2   IDFornitore INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   NomeAzienda VARCHAR (50) NOT NULL,
4   Via VARCHAR(50) NOT NULL,
5   NCivico VARCHAR(10) NOT NULL,
6   CAP CHAR(5) NOT NULL,
7   Citta VARCHAR(50) NOT NULL,
8   Regione VARCHAR(50) NOT NULL,
9   Telefono VARCHAR(15) NOT NULL,
10  Email VARCHAR(100) NOT NULL
11  );
```

Listato 4.11. Creazione della tabella FORNITORE

```
1   CREATE TABLE contatta(
2   CodiceFiscale VARCHAR(16) NOT NULL REFERENCES personale(CodiceFiscale),
3   IDCliente INT NOT NULL REFERENCES cliente(IDCliente),
4   Data DATE NOT NULL,
5   Ora TIME NOT NULL,
6   PRIMARY KEY (CodiceFiscale, IDCliente, Data, Ora)
7   );
```

Listato 4.12. Creazione della tabella CONTATTA

```
1   CREATE TABLE prenota(
2   IDPrenotazione INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   IDCliente INT NOT NULL REFERENCES cliente(IDCliente),
4   IDVeicolo INT NOT NULL REFERENCES veicolo(IDVeicolo),
5   Data DATE NOT NULL,
6   );
```

Listato 4.13. Creazione della tabella PRENOTA

```
1   CREATE TABLE relativo_a_mat_prima(
2   IDOrdine INT NOT NULL REFERENCES Ordine_In_Entrata(IDOrdine),
3   IDMateriaPrima INT NOT NULL REFERENCES MateriaPrima(IDMateriaPrima),
4   PRIMARY KEY (IDOrdine, IDMateriaPrima)
5   );
```

Listato 4.14. Creazione della tabella RELATIVO_A(MAT. PRIMA)

```
1   CREATE TABLE relativo_a_veicolo_in_entrata(
2   IDOrdine INT NOT NULL REFERENCE Ordine_In_Entrata(IDOrdine),
3   IDVeicoloInEntrata INT NOT NULL REFERENCE Veicolo_In_Entrata(IDVeicoloInEntrata),
```

```
4   PRIMARY KEY (IDOrdine, IDVeicoloInEntrata)
5   );
```

Listato 4.15. Creazione della tabella `RELATIVO_A(VEICOLO_IN_ENTRATA)`

```
1   CREATE TABLE relativo_a_veicolo_in_uscita(
2   IDOrdine INT NOT NULL REFERENCE Ordine_In_Uscita(IDOrdine),
3   IDVeicoloInUscita INT NOT NULL REFERENCE Veicolo_In_Uscita(IDVeicoloInUscita),
4   PRIMARY KEY (IDOrdine, IDVeicoloInUscita)
5   );
```

Listato 4.16. Creazione della tabella `RELATIVO_A(VEICOLO_IN_USCITA)`

```
1   CREATE TABLE fornisce_veicolo(
2   IDVeicoloInEntrata INT NOT NULL REFERENCES Veicolo_In_Entrata(IDVeicoloInEntrata),
3   IDAzienda INT NOT NULL REFERENCES Casa_Produttrice_Concessionaria(IDAzienda),
4   PRIMARY KEY (IDVeicoloInEntrata, IDAzienda)
5   );
```

Listato 4.17. Creazione della tabella `FORNISCE(VEICOLO)`

```
1   CREATE TABLE fornisce_mat_prima(
2   IDMateriaPrima INT NOT NULL REFERENCES Materia_Prima(IDMateriaPrima),
3   IDFornitore INT NOT NULL REFERENCES Fornitore(IDFornitore),
4   PRIMARY KEY (IDMateriaPrima, IDFornitore)
5   );
```

Listato 4.18. Creazione della tabella `FORNISCE(MAT. PRIMA)`

```
1   CREATE UNIQUE INDEX Indice_Numero_Telaio ON Veicolo(NumeroTelaio);
```

Listato 4.19. Implementazione del vincolo di unicità sul campo `NumeroTelaio` della tabella `VEICOLO`

```
1   ALTER TABLE Fornitore ADD INDEX Regione_Fornitore (Regione);
```

Listato 4.20. Implementazione dell'indice sul campo `Regione` della tabella `FORNITORE`

```
1   ALTER TABLE Casa_Produttrice_Concessionaria ADD INDEX Regione_Venditore (Regione);
```

Listato 4.21. Implementazione dell'indice sul campo `Regione` della tabella `CASA PRODUTTRICE O CONCESSIONARIA`

```
1   ALTER TABLE Materia_Prima ADD INDEX Nome_MatPrima(Nome);
```

Listato 4.22. Implementazione dell'indice sul campo `Nome` della tabella `MATERIA PRIMA`

```
1   ALTER TABLE Ordine_In_Entrata ADD INDEX Indice_Ordine_In_Entrata(CostoTotale, Data);
```

Listato 4.23. Implementazione dell'indice sui campi `CostoTotale` e `Data` della tabella `ORDINE IN ENTRATA`

```
1    ALTER TABLE Ordine_In_Uscita ADD INDEX Indice_Ordine_In_Uscita(CostoTotale, Data);
```

Listato 4.24. Implementazione dell'indice sui campi `CostoTotale` e `Data` della tabella `ORDINE IN USCITA`

```
1    ALTER TABLE cliente ADD INDEX Indice_Cognome (Cognome);
```

Listato 4.25. Implementazione dell'indice sul campo `Cognome` della tabella `CLIENTE`

4.2 Implementazione del sito Web

Il classico supporto per un'interfaccia Web ad una base di dati è l'utilizzo di codice PHP. Il PHP (acronimo ricorsivo di "PHP: Hypertext Preprocessor") è un linguaggio di programmazione che permette la generazione di pagine dal contenuto dinamico operando lato server. Questo popolare linguaggio di scripting interpretato, però, ha il difetto di fornire troppe soluzioni alternative per attuare l'interfacciamento con i server di dati e, a causa di questa quantità veramente enorme di classi e funzioni, per il presente progetto di tesi si è scelto di utilizzare uno dei vari framework PHP chiamato "Zend Framework".

Tale framework fornisce un supporto notevole nella generazione di progetti Web dinamici con l'impiego di un design pattern architetturale denominato Pattern MVC (acronimo di "Model View Controller"). Questo particolare design pattern prevede la presenza di tre componenti:

- la componente denominata "Model" si occupa di gestire tutti i dati dell'applicazione Web;
- la componente denominata "View" si occupa di implementare l'interfaccia utente dell'applicazione;
- la componente denominata "Controller", invece, ha il compito di gestire il tutto, dialogando con le altre due componenti.

Nella Figura 4.1 viene mostrata l'architettura del Pattern MVC all'interno di un classico ciclo di funzionamento di un'applicazione Internet. Come si evince dall'immagine (Figura 4.1), l'utente, nelle vesti di client, interfacciandosi con un Web browser, invia la richiesta al server; il Web server (rappresentato in figura dalle tre componenti del Pattern MVC) elabora la richiesta, se ne ha bisogno interagisce con la base di dati, predispone la risposta e la invia al client.

Ma, quindi, come è possibile gestire una base di dati attraverso il framework Zend?

Il modulo ZendDb rappresenta il layer di astrazione, basato su PDO (acronimo di "PHP Data Object"), utilizzato dal framework Zend per accedere ai database. L'utilizzo di PDO come libreria nativa permette al layer di astrazione di fornire l'accesso a tutti i sistemi di database supportati dalla libreria stessa (tra i più usati ci sono MySQL, PostgreSQL, Microsoft SQL Server, SQLite, ed altri). Per connettere, quindi, la base di dati all'applicazione Web è necessario creare un modello dei dati al quale far riferimento ogni volta che si vorrà accedere ai dati.

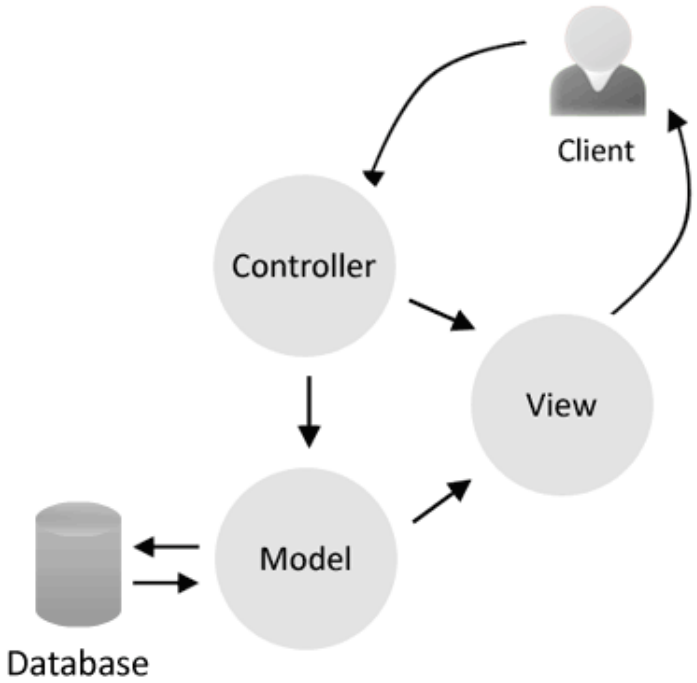


Figura 4.1. Architettura del pattern MVC

Innanzitutto, è necessario costruire un ORM("Object Relational Mapping"), cioè un substrato tra la rappresentazione ad oggetti e la sorgente dei dati in modo tale da consentire la conversione dal modello relazionale al modello orientato agli oggetti e viceversa (come mostrato nella Figura 4.2).

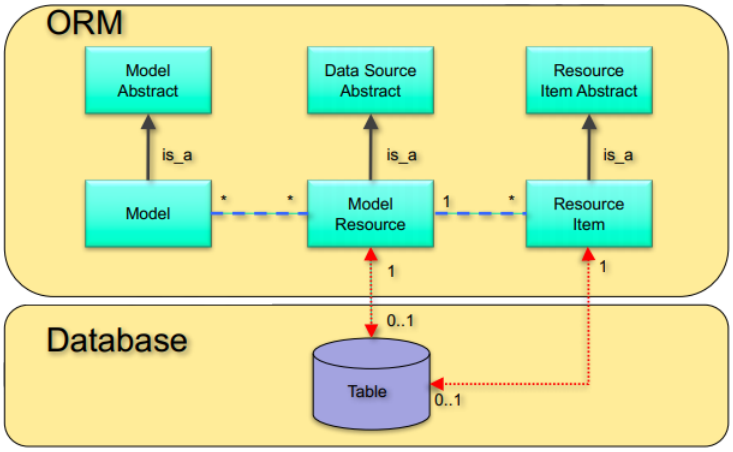


Figura 4.2. Rappresentazione del funzionamento dell'ORM di Zend Framework

La prima operazione da fare è stabilire la connessione tramite il file `include/connect.php` (Listato 4.26) e, una volta fatto ciò, inizializzare il database tramite la funzione `_initDbParms()`, contenuta nella classe `Bootstrap` definita all'interno del file `ZendProject/application/bootstrap.php` (Listato 4.27).

```
1  <?php
2   $USER="root";
3   $DB="db_italcamion";
4   $HOST="localhost";
5   $PASSWORD="";
6  ?>
```

Listato 4.26. Codice PHP contenuto nel file `include/connect.php` che consente la connessione con il database

```
1
2       protected function _initDbParms() {
3           include_once (APPLICATION_PATH . '/../../include/connect.php');
4                   $this->_db = new Zend_Db_Adapter_Pdo_Mysql(array(
5               'host'     => $HOST,
6               'username' => $USER,
7               'password' => $PASSWORD,
8               'dbname'   => $DB
9           ));
10                  Zend_Db_Table_Abstract::setDefaultAdapter($this->_db);
11          $this->_dbcontext = Application_Model_DBContext::Instance();
12      }
```

Listato 4.27. Definizione della funzione `_initDbParms()` contenuta nella classe `Bootstrap`

Fatto questo, bisogna creare le classi dalle quali, poi, verranno istanziati gli oggetti che rappresenteranno i dati:

- La classe `Application_Model_Abstract` (Listato 4.28), che deriva dal componente `Model Abstract` della Figura 4.2, avrà il ruolo di collettore tra la componente Model e la componente Controller. Questa classe, nel nostro caso, avrà il compito di smistare tutte le azioni che derivano dal Controller alle tabelle corrispondenti.
- La classe `Zend_DB_Table_Abstract` (Listati 4.29-4.36), che deriva dal componente `Data Source Abstract` della Figura 4.2, consentirà di manipolare la tabella.
- La classe `Zend_Db_Table_Raw_Abstract`, che deriva dal componente `Resource Item` della Figura 4.2, consentirà di manipolare le singole tuple della tabella (questa classe, nel nostro caso, è stata solo creata ma non sfruttata, dal momento che non si è avuta la necessità di operare sulle singole tuple).

Per scrivere una query atta ad interrogare il database, se non avessimo utilizzato Zend Framework, avremmo usufruito della funzione `mysqli_query()` del PHP che permette di scrivere una query MySQL in PHP. Lavorando con Zend, invece, non possiamo operare nella stessa maniera! Per risolvere questo problema, infatti, si utilizza uno dei tanti metodi che la classe `Zend_DB_Table_Abstract` mette a disposizione, denominato `select`; questa funzione definisce l'accesso ad un oggetto che ci consente di specificare i parametri del comando `select` che verrà inviato a MySQL. Quindi, come viene mostrato nei vari listati, per comporre una query metteremo, nella variabile `$select`, la stringa SQL del comando desiderato scritta seguendo la sintassi che Zend mette a disposizione. Per quanto riguarda l'estrazione dei dati, invece, viene usato il metodo predefinito di Zend chiamato `fetchAll` che

prende come parametro il risultato di una select ed estrae tutte le tuple che soddisfano la query. Successivamente, restituiremo il risultato della `fetchAll` alla classe `Application_Model_Abstract` che, come abbiamo detto, restituisce il risultato al controller.

```
<?php

class Application_Model_DBContext extends Application_Model_Abstract {

    //METODI TABELLA UTENTI
    public function getUserByUsername($username){ return $this->getResource('Utente')->getByUsername($username); }
    public function getUserById($id){ return $this->getResource('Utente')->getById($id); }
    public function insertUser($user){ return $this->getResource('Utente')->insert($user); }
    public function updateUser($user){ return $this->getResource('Utente')->updateUser($user); }
    public function deleteUser($id){ return $this->getResource('Utente')->delete('IDUtente = ' . intval($id)); }
    public function getAllUsers(){ return $this->getResource('Utente')->getAll();}
    public function getOnlyStaff(){ return $this->getResource('Utente')->getStaff();}
    public function getOnlyCustomers(){return $this->getResource('Utente')->getCustomers();}

    //METODI TABELLA CATALOGO
    public function getCatalog($values){return $this->getResource('Veicolo')->getCatalog($values);}
    public function getTruckById($id){return $this->getResource('Veicolo')->getById($id);}
    public function deleteTruck($id){return $this->getResource('Veicolo')->delete('IDVeicolo =' . intval($id));}
    public function saveTruck($values){ return $this->getResource('Veicolo')->insertTruck($values); }
    public function updateTruck($truck){ return $this->getResource('Veicolo')->updateTruck($truck); }

    //METODI TABELLA PRENOTAZIONE
    public function reserveTruck($reserve){return $this->getResource('Prenotazione')->insertReservation($reserve);}
    public function getStoricoPrenotazioniUtente($userId){ return $this->getResource('Prenotazione')->
    getStoricoUtente($userId); }
    public function deleteReservtion($id){return $this->getResource('Prenotazione')->
    delete('IDPrenotazione =' . intval($id));}
    public function getReservationById($id){return $this->getResource('Prenotazione')->getById($id);}
    public function getPrenotazioni(){return $this->getResource('Prenotazione')->getAll();}
    public function getGiornoByData($data){return $this->getResource('Prenotazione')->getByData($data);}

    //METODI TABELLA FORNITORE
    public function getAllSuppliers(){return $this->getResource('Fornitore')->getAll();}
    public function getSupplierById($id){return $this->getResource('Fornitore')->getById($id);}
    public function deleteSupplier($id){return $this->getResource('Fornitore')->
    delete('IDFornitore =' . intval($id));}
    public function saveSupplier($values){ return $this->getResource('Fornitore')->
    insertSupplier($values); }
    public function updateSupplier($supplier){ return $this->getResource('Fornitore')->updateFornitore($supplier); }

    //METODI TABELLA CONCESSIONARIA
    public function getAllDealerships(){return $this->getResource('Concessionaria')->getAll();}
    public function getDealershipById($id){return $this->getResource('Concessionaria')->getById($id);}
    public function deleteDealership($id){return $this->getResource('Concessionaria')
    ->delete('IDAzienda =' . intval($id));}
    public function saveDealership($values){ return $this->getResource('Concessionaria')
    ->insertDealership($values); }
    public function updateDealership($supplier){ return $this->getResource('Concessionaria')
    ->updateConcessionaria($supplier); }

    //METODI TABELLA ORDINI IN ENTRATA
    public function getAllOrderPlaced(){return $this->getResource('Ordineinentrata')->getAll();}
    public function getOrderPlacedById($id){return $this->getResource('Ordineinentrata')->getById($id);}
    public function deleteOrderPlaced($id){return $this->getResource('Ordineinentrata')
    ->delete('IDOrdine =' . intval($id));}
    public function saveOrderPlaced($values){ return $this->getResource('Ordineinentrata')
    ->insertOrderPlaced($values); }
    public function updateOrderPlaced($orderplaced){ return $this->getResource('Ordineinentrata')
    ->updateOrderPlaced($orderplaced); }

    //METODI TABELLA ORDINI IN USCITA
    public function getAllOrderReceived(){return $this->getResource('Ordineinuscita')->getAll();}
    public function getOrderReceivedById($id){return $this->getResource('Ordineinuscita')->getById($id);}
    public function deleteOrderReceived($id){return $this->getResource('Ordineinuscita')
    ->delete('IDOrdine =' . intval($id));}
    public function saveOrderReceived($values){ return $this->getResource('Ordineinuscita'
    )->insertOrderReceived($values); }
    public function updateOrderReceived($orderreceived){ return $this->getResource('Ordineinuscita')
    ->updateOrderReceived($orderreceived); }

    //METODI TABELLA RUOLI
    public function getRoles(){ return $this->getResource('Ruoli')->getAll(); }

    public static function Instance(){
        static $inst = null;
        if ($inst === null) { $inst = new Application_Model_DBContext(); }
        return $inst;
        }}
```

Listato 4.28. Definizione della classe `Application_Model_DBContext` che estende la classe `Application_Model_Abstract`

```php
<?php

class Application_Resource_Concessionaria extends Zend_Db_Table_Abstract {
    protected $_name  = 'concessionaria';
    protected $_primary = 'IDAzienda';
    protected $_rowClass = 'Application_Resource_Fornitore_Item';

    public function init() { }

    public function getAll(){
        $select = $this->select();
        return $this->fetchAll($select);
    }

    public function getById($id){
        $select=$this->select()->where('IDAzienda IN(?)', $id);
        $row = $this->fetchAll($select);
        if(count($row)>0){return $row[0];}
        else return null;
    }

    public function insertDealership($values){ $this->insert($values); }

    public function updateConcessionaria($dealership){
        $where = $this->getAdapter()->quoteInto('IDAzienda = ?', $dealership['IDAzienda']);
        return $this->update((array)$dealership, $where);
    }
}
```

Listato 4.29. Definizione della classe `Application_Resource_Concessionaria` che estende la classe `Zend_Db_Table_Abstract`

```php
<?php

class Application_Resource_Fornitore extends Zend_Db_Table_Abstract {
    protected $_name  = 'fornitore';
    protected $_primary = 'IDFornitore';
    protected $_rowClass = 'Application_Resource_Fornitore_Item';

    public function init() { }

    public function getAll(){
        $select = $this->select();
        return $this->fetchAll($select);
    }

    public function getById($id){
        $select=$this->select()->where('IDFornitore IN(?)', $id);
        $row = $this->fetchAll($select);
        if(count($row)>0){return $row[0];}
        else return null;
    }

    public function insertSupplier($values){ $this->insert($values); }

    public function updateFornitore($supplier){
        $where = $this->getAdapter()->quoteInto('IDFornitore = ?', $supplier['IDFornitore']);
        return $this->update((array)$supplier, $where);
    }
}
```

Listato 4.30. Definizione della classe `Application_Resource_Fornitore` che estende la classe `Zend_Db_Table_Abstract`

```php
<?php

class Application_Resource_Ordineinentrata extends Zend_Db_Table_Abstract {
    protected $_name  = 'ordine_in_entrata';
    protected $_primary = 'IDOrdine';
    protected $_rowClass = 'Application_Resource_Ordineinentrata_Item';

    public function init() { }

    public function getAll(){
        $select = $this->select()->order('Data ASC');
        return $this->fetchAll($select);
    }

    public function getById($id){
        $select=$this->select()->where('IDOrdine IN(?)', $id);
        $row = $this->fetchAll($select);
        if(count($row)>0){return $row[0];}
        else return null;
```

```
20      }
21
22      public function insertOrderPlaced($values){ $this->insert($values); }
23
24      public function updateOrderPlaced($orderplaced){
25          $where = $this->getAdapter()->quoteInto('IDOrdine = ?', $orderplaced['IDOrdine']);
26          return $this->update((array)$orderplaced, $where);
27      }
28  }
```

Listato 4.31. Definizione della classe `Application_Resource_Ordineinentrata` che estende la classe `Zend_Db_Table_Abstract`

```
1   <?php
2
3   class Application_Resource_Ordineinuscita extends Zend_Db_Table_Abstract {
4       protected $_name  = 'ordine_in_uscita';
5       protected $_primary = 'IDOrdine';
6       protected $_rowClass = 'Application_Resource_Ordineinuscita_Item';
7
8       public function init() { }
9
10      public function getAll(){
11          $select = $this->select()->order('Data ASC');
12          return $this->fetchAll($select);
13      }
14
15      public function getById($id){
16          $select=$this->select()->where('IDOrdine IN(?)', $id);
17          $row = $this->fetchAll($select);
18          if(count($row)>0){return $row[0];}
19          else return null;
20      }
21
22      public function insertOrderReceived($values){ $this->insert($values); }
23
24      public function updateOrderReceived($orderreceived){
25          $where = $this->getAdapter()->quoteInto('IDOrdine = ?', $orderreceived['IDOrdine']);
26          return $this->update((array)$orderreceived, $where);
27      }
28  }
```

Listato 4.32. Definizione della classe `Application_Resource_Ordineinuscita` che estende la classe `Zend_Db_Table_Abstract`

```
1   <?php
2
3   class Application_Resource_Prenotazione extends Zend_Db_Table_Abstract {
4       protected $_name  = 'prenotazione';
5       protected $_primary = 'IDPrenotazione';
6       protected $_rowClass = 'Application_Resource_Prenotazione_Item';
7
8       public function init() { }
9
10      public function insertReservation($reserve){
11          $this->insert($reserve);
12      }
13
14      public function getStoricoUtente($userId){
15          return $this->fetchAll($this->select()
16              ->from (array('p'=>'prenotazione', 'v' => 'veicolo', 'c' => 'utente'))
17              ->join(array('v' => 'veicolo') , 'p.Veicolo = v.IDVeicolo')
18              ->join(array('c' => 'utente') , 'p.Prenotatore = c.IDUtente')
19              ->where("p.Prenotatore = ? ", $userId)
20              ->setIntegrityCheck(false)
21              );
22      }
23
24      public function getById($id){
25          $select=$this->select()->where('IDPrenotazione IN(?)', $id);
26          $row = $this->fetchAll($select);
27          if(count($row)>0){return $row[0];}
28          else return null;
29      }
30
31      public function getAll(){
32          return $this->fetchAll($this->select()
33              ->from (array('p'=>'prenotazione', 'v' => 'veicolo', 'c' => 'utente'))
34              ->join(array('v' => 'veicolo') , 'p.Veicolo = v.IDVeicolo')
35              ->join(array('c' => 'utente') , 'p.Prenotatore = c.IDUtente')
36              ->setIntegrityCheck(false)
37              );
38      }
```

```
39
40      public function getByData($data){
41          $where = $this->getAdapter()->quoteInto('Data = ?', $data);
42          $select = $this->select()->where($where);
43          $result = $this->fetchAll($select);
44          return count($result) > 0 ? $result[0] : null;
45      }
46  }
```

Listato 4.33. Definizione della classe `Application_Resource_Prenotazione` che estende la classe `Zend_Db_Table_Abstract`

```
1   <?php
2
3   class Application_Resource_Ruoli extends Zend_Db_Table_Abstract {
4       protected $_name  = 'ruoli';
5       protected $_primary = 'ID';
6       protected $_rowClass = 'Application_Resource_Ruoli_Item';
7
8       public function init() { }
9
10      public function getAll(){ return $this->fetchAll($this->select()->order('Livello')); }
11  }
```

Listato 4.34. Definizione della classe `Application_Resource_Ruoli` che estende la classe `Zend_Db_Table_Abstract`

```
1   <?php
2
3   class Application_Resource_Utente extends Zend_Db_Table_Abstract {
4       protected $_name  = 'utente';
5       protected $_primary = 'IDUtente';
6       protected $_rowClass = 'Application_Resource_Utente_Item';
7
8       public function init() { }
9
10      public function getByUsername($username){
11          $where = $this->getAdapter()->quoteInto('Username = ?', $username);
12          $select = $this->select()->where($where);
13          $result = $this->fetchAll($select);
14          return count($result) > 0 ? $result[0] : null;
15      }
16
17      public function getById($id){
18          $select = $this->select()->where('IDUtente = ?', $id);
19          $result = $this->fetchAll($select);
20          return count($result) > 0 ? $result[0] : null;
21      }
22
23      public function updateUser($user){
24          $where = $this->getAdapter()->quoteInto('IDUtente = ?', $user['IDUtente']);
25          return $this->update((array)$user, $where);
26      }
27
28      public function getAll(){ return $this->fetchAll($this->select()->order('Ruolo ASC')); }
29      public function getStaff(){ return $this->fetchAll($this->select()->where('Ruolo = (?)', 3)); }
30      public function getCustomers(){ return $this->fetchAll($this->select()->where('Ruolo = (?)', 2)); }
31  }
```

Listato 4.35. Definizione della classe `Application_Resource_Utente` che estende la classe `Zend_Db_Table_Abstract`

```
1   <?php
2
3   class Application_Resource_Veicolo extends Zend_Db_Table_Abstract {
4       protected $_name  = 'veicolo';
5       protected $_primary = 'IDVeicolo';
6       protected $_rowClass = 'Application_Resource_Veicolo_Item';
7
8       public function init() { }
9
10      public function getCatalog($values=null){
11          $select = $this->select();
12
13          if($values != null){
14              $prezzoMin = $values['prezzoMin'] != '' ? floatval($values['prezzoMin']) : null;
15              $prezzoMax = $values['prezzoMax'] != '' ? floatval($values['prezzoMax']) : null;
16              $nome=$values['Nome'];
```

```
17              $marca=$values['Marca'];
18              $caract=$values['Caratteristiche'];
19              $ordinator=$values['OrderBy'];
20              $anno = $values['Anno'] != '' ? intval($values['Anno']) : null;
21              $stato=$values['Stato'];
22
23              if($nome!=null){
24                  $select = $select->where('Nome LIKE ?', '%'.$nome.'%');
25              }
26
27              if($marca!=null){
28                  $select = $select->where('Marca LIKE ?', '%'.$marca.'%');
29              }
30
31              if($caract!=null){
32                  $select = $select->where('Caratteristiche LIKE ?', '%'.$caract.'%');
33              }
34
35              if($anno!=null){
36                  $select=$select->where('Anno =?',$anno);
37              }
38
39              switch($stato){
40                  case '0': $select=$select->where('Stato IN (?)','Nuovo');break;
41                  case '1': $select=$select->where('Stato IN (?)','Usato');break;
42                  default;
43              }
44
45              if($prezzoMin<=$prezzoMax){
46                  if($prezzoMin != null ){$select = $select->where('Prezzo >= ?', $prezzoMin);}
47                  if($prezzoMax != null ){ $select = $select->where('Prezzo <= ?', $prezzoMax);}
48              }
49              else{$select=$select->where('IDVeicolo =?','0');}
50
51              switch($ordinator){
52                  case 'ASC_A': $select = $select->order('Anno ASC');
53                  case 'DESC_A': $select = $select->order('Anno DESC');
54                  case 'DESC_P': $select = $select->order('Prezzo DESC');
55                  default: case 'ASC_P': $select = $select->order('Prezzo ASC');
56              }
57
58      return $this->fetchAll($select);
59          }
60      }
61
62      public function getById($id){
63          $select=$this->select()->where('IDVeicolo IN(?)', $id);
64          $row = $this->fetchAll($select);
65          if(count($row)>0){return $row[0];}
66          else return null;
67      }
68
69      public function insertTruck($values){ $this->insert($values); }
70
71      public function updateTruck($truck){
72          $where = $this->getAdapter()->quoteInto('IDVeicolo = ?', $truck['IDVeicolo']);
73          return $this->update((array)$truck, $where);
74      }
75  }
```

Listato 4.36. Definizione della classe `Application_Resource_Veicolo` che estende la classe `Zend_Db_Table_Abstract`

4.2.1 Sezione "public" del sito

La sezione "public" è quella porzione di sito Web accessibile a tutti coloro che entrano, indipendentemente dal fatto che si siano registrati o meno. Dalla home page, o da qualunque altra pagina di questa sezione, se l'utente si è precedentemente registrato, è possibile effettuare il login accedendo, così, al livello di utenza di appartenenza ("user", "staff" o "admin"). Di seguito verrà riportato il codice del Controller relativo alla classe di utenza "public" (Listato 4.37).

Nelle Figure 4.3-4.9, invece, vengono mostrate le schermate relative al livello di utenza "public".

```
1   <?php
2
3   class PublicController extends Zend_Controller_Action
```

```
4   {
5       protected $_database;
6       protected $_redirector;
7       protected $_form;
8
9       public function init()
10      {
11          $this->_database = Application_Model_DBContext::Instance();
12          $this->_redirector = $this->_helper->getHelper('Redirector');
13          $this->view->layout = 'public';
14      }
15
16      public function indexAction(){}
17      public function contactsAction(){}
18      public function whoAction(){}
19      public function whereAction(){}
20      public function catalogAction(){
21          $filterForm = new Application_Form_Public_Catalogo_Filter();
22          $this->view->filterForm = $filterForm;
23          if (!$filterForm->isValid($_POST)) { return $this->render('catalog'); }
24          $values = $filterForm->getValues();
25          $this->view->assign(array('catalog' => $this->_database->getCatalog($values)));
26      }
27
28      protected function _getAuthAdapter(){
29          $dbAdapter = Zend_Db_Table::getDefaultAdapter();
30          $authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
31          $authAdapter->setTableName('utente')
32                      ->setIdentityColumn('Username')
33                      ->setCredentialColumn('Password');
34          return $authAdapter;
35      }
36
37      protected function _login($values){
38          $adapter = $this->_getAuthAdapter();
39          $adapter->setIdentity($values['username']);
40          $adapter->setCredential($values['password']);
41          $auth = Zend_Auth::getInstance();
42          $result = $auth->authenticate($adapter);
43          if ($result->isValid()) {
44              $user = $adapter->getResultRowObject();
45              $auth->getStorage()->write($user);
46              return true;
47          }
48          return false;
49      }
50
51      protected function _logout(){
52          Zend_Auth::getInstance()->clearIdentity();
53          return true;
54      }
55
56      public function loginAction(){
57          $loginForm = new Application_Form_Public_Utenti_Login();
58          if(count($_POST) > 0 && $loginForm->isValid($_POST)){
59              if($this->_login($loginForm->getValues())){ $this->_redirector->gotoSimple('index', 'user'); }
60              else { $this->view->loginError = true; }
61          }
62          $this->view->loginForm = $loginForm;
63      }
64
65      public function logoutAction(){
66          $this->_logout();
67          $this->_redirector->gotoSimple('index', 'public');
68      }
69
70      public function signinAction(){
71          $signinForm = new Application_Form_Public_Utenti_Signin();
72          if(count($_POST) > 0 && $signinForm->isValid($_POST)){
73              $values = $signinForm->getValues();
74              $usr = $this->_database->getUserByUsername($values['username']);
75              if(!$values['condizioni']){
76                  $this->view->error = 'Devi accettare i termini di utilizzo!';
77              }
78              else
79                  if($usr == null){
80                  //VALIDATORE LATO SERVER
81                  $values['ruolo'] = 2;
82                  unset($values['condizioni']);
83                  $this->_database->insertUser($values);
84                  $this->_login($values);
85                  $this->_redirector->gotoSimple('index', 'user');
86              }
87          }
88          $this->view->signinForm = $signinForm;
89      }
90  }
```

Listato 4.37. Definizione della classe `PublicController` che estende la classe `Zend_Controller_Action`

Figura 4.3. Schermata relativa alla home page della sezione "public" del sito

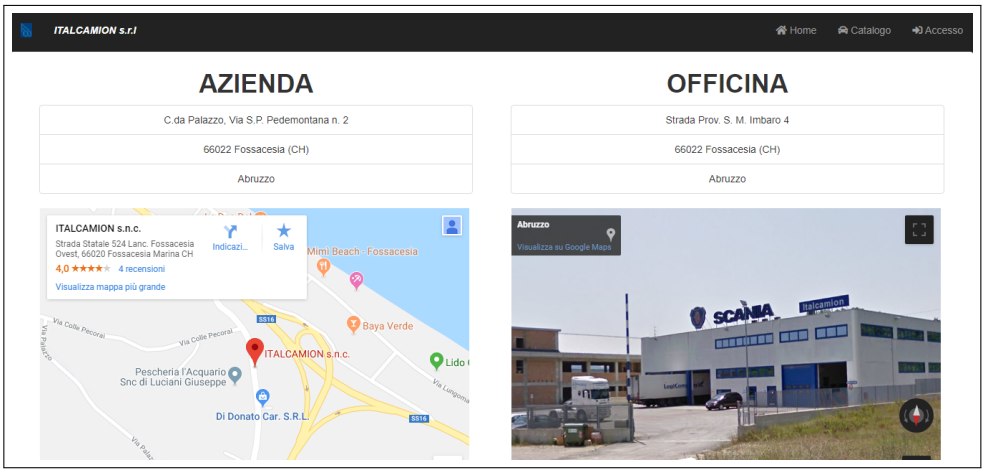


Figura 4.4. Schermata relativa alla pagina "where" della sezione "public"; in questa pagina troviamo tutte le informazioni relative al luogo in cui è dislocata l'azienda

Figura 4.5. Schermata relativa alla pagina "who" della sezione "public"; in questa pagina troviamo tutte le informazioni relative alla storia, all'organizzazione e all'obiettivo dell'azienda

Figura 4.6. Schermata relativa alla pagina "contactus" della sezione "public"; in questa pagina troviamo tutti i contatti dell'azienda

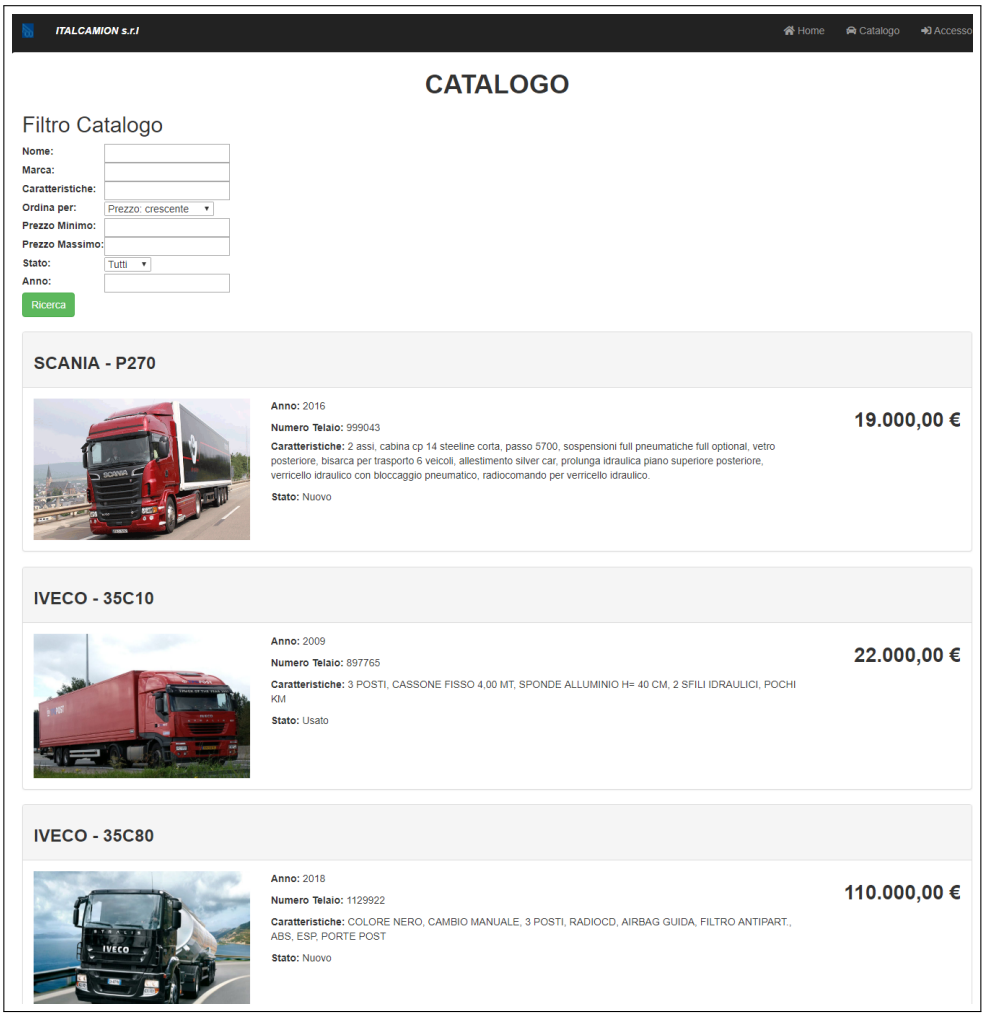


Figura 4.7. Schermata relativa alla pagina “catalog” della sezione “public”; in questa pagina è possibile visualizzare i veicoli e/o filtrarli secondo molteplici parametri

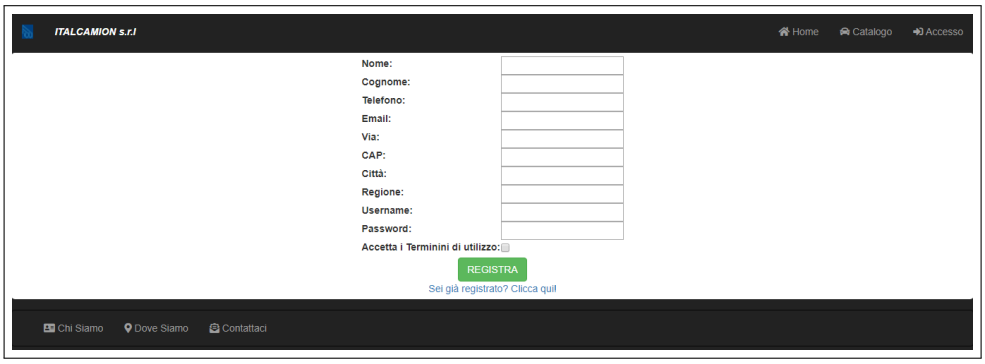


Figura 4.8. Schermata relativa alla pagina "signin" della sezione "public"; questa pagina consente ad un utente di registrarsi al sito

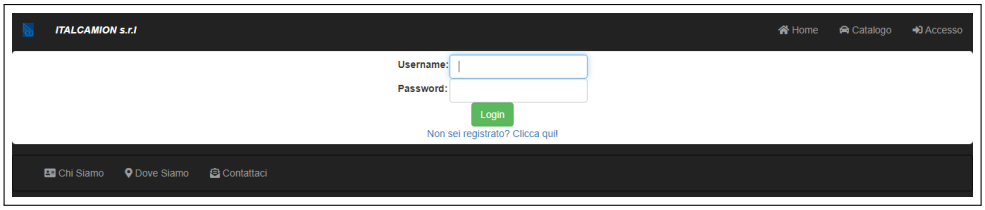


Figura 4.9. Schermata relativa alla pagina "login" della sezione "public"; questa pagina consente ad un utente di accedere al sito

4.2.2 Sezione "user" del sito

La sezione "user" dell'applicazione Web è dedicata a tutti i clienti che si sono registrati sul portale tramite la form di registrazione compilabile nell'apposita pagina. Di seguito verrà riportato il codice del Controller relativo alla classe di utenza "user" (Listato 4.38).

Nelle Figure 4.10-4.12, invece, vengono mostrate le schermate relative al livello di utenza "user".

```
1   <?php
2
3   class UserController extends Zend_Controller_Action
4   {
5       protected $_database;
6       protected $_redirector;
7
8       public function init()
9       {
10          $this->_database = Application_Model_DBContext::Instance();
11          $this->_redirector = $this->_helper->getHelper('Redirector');
12          //QUI CONTROLLA CHE L'UTENTE ABBIA IL RUOLO NECESSARIO PER ACCEDERE AL CONTROLLER
13          if(!$this->view->acl->isAllowed($this->view->currentRole, null, 'Utente')){
14              $this->_redirector->gotoSimple('auth', 'error');
15          }
16          $this->view->layout = 'user';
17      }
18
19      public function indexAction() {
20      if($this->view->currentRoleLevel >= 2){ $this->_redirector->gotoSimple('index', 'staff'); }
21      $this->_helper->viewRenderer->renderBySpec('index', array('controller' => 'public'));
22      }
```

```
23
24      public function contactsAction(){
25          $this->_helper->viewRenderer->renderBySpec('contacts', array('controller' => 'public'));
26      }
27
28      public function whoAction(){
29          $this->_helper->viewRenderer->renderBySpec('who', array('controller' => 'public'));
30      }
31
32      public function whereAction(){
33          $this->_helper->viewRenderer->renderBySpec('where', array('controller' => 'public'));
34      }
35
36      public function catalogAction(){
37          $filterForm = new Application_Form_Public_Catalogo_Filter();
38          $this->view->filterForm = $filterForm;
39          if (!$filterForm->isValid($_POST)) { return $this->render('catalog'); }
40          $values = $filterForm->getValues();
41          $this->view->assign(array('catalog' => $this->_database->getCatalog($values)));
42      }
43
44      public function reservationAction(){
45          $truckid = intval($this->_getParam('id', 0));
46          $truck = $this->_database->getTruckById($truckid);
47          if($truck == null){ $this->view->error = 'Veicolo non trovato'; }
48          else{
49          $resForm = new Application_Form_User_Prenotazioni_Prenotazione();
50          if(count($_POST) > 0 && $resForm->isValid($_POST)){
51              $values = $resForm->getValues();
52              $giorno = $this->_database->getGiornoByData($values['data']);
53              if($giorno == null){
54              $reserve = array('Veicolo'=>$truckid,'Prenotatore'=>$this->view->user->IDUtente,
55              'Data'=>$values['data']);
56              $this->_database->reserveTruck($reserve);
57              $this->_redirector->gotoSimple('profile', 'user');
58              }
59          }
60          $this->view->resForm = $resForm;
61          }
62  }
63
64      public function profileAction(){
65          $this->view->prenotazioniList = $this->_database->getStoricoPrenotazioniUtente($this->view->user->IDUtente);
66          $profileForm = new Application_Form_User_Utenti_Profile($this->view->user);
67          if(count($_POST) > 0 && $profileForm->isValid($_POST)){
68              $values = $profileForm->getValues();
69              $update = array();
70              $update['IDUtente'] = $this->view->user->IDUtente;
71              $update['Username'] = $values['username'];
72              $update['Email'] = $values['email'];
73              $this->view->user->Username = $values['username'];
74              $this->view->user->Email = $values['email'];
75              if($values['password']){
76                  $update['Password'] = $values['password'];
77                  $this->view->user->Password = $values['password'];
78              }
79              $this->_database->updateUser($update);
80              $this->view->success = 'Modifiche apportate con successo!';
81          }
82          $this->view->profileForm = $profileForm;
83      }
84
85      public function deletereservationAction(){
86          $reservationid = intval($this->_getParam('id', 0));
87          $reservation = $this->_database->getReservationById($reservationid);
88          if($reservation == null){ $this->view->error = 'Prenotazione non trovata'; }
89          else{
90              $this->_database->deleteReservtion($reservation['IDPrenotazione']);
91              $this->_redirector->goToSimple('profile', 'user');
92          }
93      }
94  }
```

Listato 4.38. Definizione della classe `UserController` che estende la classe `Zend_Controller_Action`

Figura 4.10. Schermata relativa alla home page della sezione “user” del sito

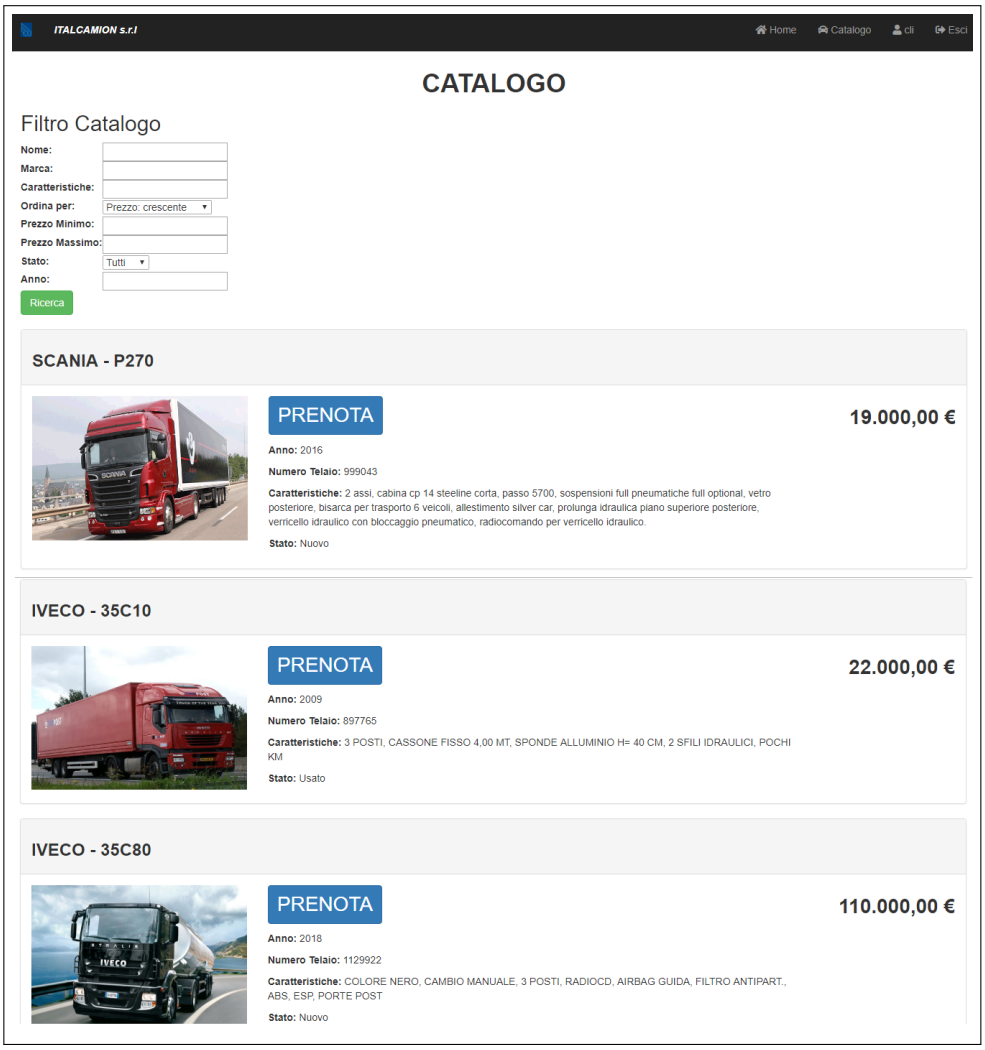


Figura 4.11. Schermata relativa alla pagina “catalog” della sezione “user”; da questa pagina il cliente, oltre alle funzionalità già descritte per il livello di utenza “public”, può prenotare la prova gratuita del veicolo selezionato

Figura 4.12. Schermata relativa alla pagina "profile" della sezione "user"; da questa pagina l'utente, dopo essersi autenticato, può visualizzare e modificare le proprie informazioni personali. In essa, inoltre, è possibile visualizzare il riepilogo delle prenotazioni effettuate

4.2.3 Sezione "staff" del sito

La sezione "staff" dell'applicazione Web è dedicata a tutti i membri dello staff dell'azienda. Questa sezione è la più corposa dal momento che il portale è pensato non solo per il cliente ma anche per i dipendenti dell'azienda che possono mettere in atto le rispettive mansioni di gestione direttamente tramite le funzionalità fornite dall'applicazione web. Di seguito verrà riportato il codice del Controller relativo alla classe di utenza "staff" (Listato 4.39).

Nelle Figure 4.13-4.22, invece, vengono mostrate le schermate relative al livello di utenza "staff".

```
<?php

class StaffController extends Zend_Controller_Action
{
    protected $_database;
    protected $_redirector;
    protected $_addForm;
    protected $_editForm;
    protected $_form;

    public function init()
    {
        $this->_database = Application_Model_DBContext::Instance();
        $this->_redirector = $this->_helper->getHelper('Redirector');
        if(!$this->view->acl->isAllowed($this->view->currentRole, null, 'Staff'))
        { $this->_redirector->gotoSimple('auth', 'error'); }
        $this->view->layout = 'staff';
        $this->view->veicoloForm = $this->getVeicoloForm();
        $this->view->fornitoreForm = $this->getFornitoreForm();
        $this->view->concessionariaForm=$this->getConcessionariaForm();
        $this->view->ordineeffettuatoForm=$this->getOrdineEffettuatoForm();
        $this->view->ordinericevutoForm=$this->getOrdineRicevutoForm();
    }

    public function fstaffAction(){}

    public function indexAction() {
    if($this->view->currentRoleLevel >= 3){ $this->_redirector->gotoSimple('index', 'admin'); }
    $this->_helper->viewRenderer->renderBySpec('index', array('controller' => 'public'));
    }

    public function contactsAction(){
        $this->_helper->viewRenderer->renderBySpec('contacts', array('controller' => 'public'));
```

```
}

public function whoAction(){
    $this->_helper->viewRenderer->renderBySpec('who', array('controller' => 'public'));
}

public function whereAction(){
    $this->_helper->viewRenderer->renderBySpec('where', array('controller' => 'public'));
}

public function profileAction(){
    $this->view->prenotazioniList = $this->_database->getStoricoPrenotazioniUtente($this->view->user->IDUtente);
    $profileForm = new Application_Form_User_Utenti_Profile($this->view->user);
    if(count($_POST) > 0 && $profileForm->isValid($_POST)){
        $values = $profileForm->getValues();
        $update = array();
        $update['IDUtente'] = $this->view->user->IDUtente;
        $update['Username'] = $values['username'];
        $update['Email'] = $values['email'];
        $this->view->user->Username = $values['username'];
        $this->view->user->Email = $values['email'];
        if($values['password']){
            $update['Password'] = $values['password'];
            $this->view->user->Password = $values['password'];
        }
        $this->_database->updateUser($update);
        $this->view->success = 'Modifiche apportate con successo!';
    }
    $this->view->profileForm = $profileForm;
}

public function catalogAction(){
    $filterForm = new Application_Form_Public_Catalogo_Filter();
    $this->view->filterForm = $filterForm;
    if (!$filterForm->isValid($_POST)) { return $this->render('catalog'); }
    $values = $filterForm->getValues();
    $this->view->assign(array('catalog' => $this->_database->getCatalog($values)));
}

public function personaleAction(){
    $this->view->users = $this->_database->getOnlyStaff()->toArray();
    $roles = $this->_database->getRoles()->toArray();
    for($i = 0; $i < count($this->view->users); $i++){
        $this->view->users[$i]['RuoloTesto'] = $roles[$this->view->users[$i]['Ruolo'] - 1]['Nome'];
    }
}

public function usersAction(){
    $this->view->users = $this->_database->getOnlyCustomers()->toArray();
    $roles = $this->_database->getRoles()->toArray();
    for($i = 0; $i < count($this->view->users); $i++){
        $this->view->users[$i]['RuoloTesto'] = $roles[$this->view->users[$i]['Ruolo'] - 1]['Nome'];
    }
}

public function createuserAction(){
    $roleNames = array();
    foreach($this->view->allRoles as $role){ if($role->Livello > 0){ $roleNames[$role->ID] = $role->Nome; } }
    $createForm = new Application_Form_Public_Utenti_Signin(false, $roleNames);
    if(count($_POST) > 0 && $createForm->isValid($_POST)){
        $values = $createForm->getValues();
        $usr = $this->_database->getUserByUsername($values['username']);
        if($usr == null){
            $this->_database->insertUser($values);
            $this->_redirector->gotoSimple('users', 'staff');
        }
    }
    $this->view->createForm = $createForm;
}

public function deleteuserAction(){
    $userid = intval($this->_getParam('id', 0));
    $user = $this->_database->getUserById($userid);
    if($user == null){ $this->view->error = 'Utente non trovato'; }
    else
        if($user['IDUtente'] == $this->view->user->IDUtente)
        { $this->view->error = 'Impossibile cancellare il proprio utente'; }
    else{
        $this->_database->deleteUser($user['IDUtente']);
        $this->_redirector->goToSimple('users', 'staff');
    }
}

public function edituserAction(){
    $userid = intval($this->_getParam('id', 0));
    $user = $this->_database->getUserById($userid);
    if($user == null){ $this->view->error = 'Utente non trovato'; }
    else{
        $roleNames = array();
        foreach($this->view->allRoles as $role){ if($role->Livello > 0){ $roleNames[$role->ID] = $role->Nome; } }
        $this->view->editUser = $user;
        if($user->Ruolo == 3)
            {$editForm = new Application_Form_Admin_Staff_Modify($roleNames,$user);}
        else
```

```
            {$editForm = new Application_Form_Admin_Utenti_Modify($roleNames, $user);}
        if(count($_POST) > 0 && $editForm->isValid($_POST)){
            $values = $editForm->getValues();
            $values['IDUtente'] = $user->IDUtente;
            $this->_database->updateUser($values);
            $this->_redirector->goToSimple('users', 'staff');
        }
        $this->view->editForm = $editForm;
    }
}

public function newveicoloAction(){}

public function addveicoloAction()
    {
          if (!$this->getRequest()->isPost()) {
                $this->_redirector->goToSimple('catalog', 'staff');
          }
          $form=$this->_form;
          if (!$form->isValid($_POST)) {
                return $this->render('newveicolo');
          }
          $values = $form->getValues();
          $this->_database->saveTruck($values);
    $this->_redirector->goToSimple('catalog', 'staff');
    }
private function getVeicoloForm()
    {
          $urlHelper = $this->_helper->getHelper('url');
          $this->_form = new Application_Form_Staff_Veicoli_Add();
          $this->_form->setAction($urlHelper->url(array(
                        'controller' => 'staff',
                        'action' => 'addveicolo'),
                        'default'
                        ));
          return $this->_form;
    }

public function editveicoloAction(){
    $truckid = intval($this->_getParam('id', 0));
    $truck = $this->_database->getTruckById($truckid);
    if($truck == null){ $this->view->error = 'Veicolo non trovato'; }
    else{
        $this->view->editVeicolo = $truck;
     $editForm = new Application_Form_Staff_Veicoli_Modify($truck);
        if(count($_POST) > 0 && $editForm->isValid($_POST)){
            $values = $editForm->getValues();
            $values['IDVeicolo'] = $truck->IDVeicolo;
            if(!$values['Foto']){ unset($values['Foto']); }
            $this->_database->updateTruck($values);
            $this->_redirector->goToSimple('catalog', 'staff');
        }
        $this->view->editForm = $editForm;
    }
}

public function deleteveicoloAction(){
    $truckid = intval($this->_getParam('id', 0));
    $truck = $this->_database->getTruckById($truckid);
    if($truck == null){ $this->view->error = 'Veicolo non trovato'; }
    else{
        $this->_database->deleteTruck($truck['IDVeicolo']);
        $this->_redirector->goToSimple('catalog', 'staff');
    }
}

public function listreservationAction(){
    $this->view->assign(array('list' => $this->_database->getPrenotazioni()));
}

public function supplierslistAction(){
    $this->view->assign(array('allSuppliers' => $this->_database->getAllSuppliers()));
}

public function deletesupplierAction(){
    $supplierid = intval($this->_getParam('id', 0));
    $supplier = $this->_database->getSupplierById($supplierid);
    if($supplier == null){ $this->view->error = 'Fornitore non trovato'; }
    else{
        $this->_database->deleteSupplier($supplier['IDFornitore']);
        $this->_redirector->goToSimple('supplierslist', 'staff');
    }
}

public function newsupplierAction(){}

public function addsupplierAction()
    {
          if (!$this->getRequest()->isPost()) {
                $this->_redirector->goToSimple('supplierslist', 'staff');
          }
          $form=$this->_form;
          if (!$form->isValid($_POST)) {
                return $this->render('newsupplier');
```

```
            }
            $values = $form->getValues();
            $this->_database->saveSupplier($values);
    $this->_redirector->goToSimple('supplierslist', 'staff');
    }
private function getFornitoreForm()
    {
            $urlHelper = $this->_helper->getHelper('url');
            $this->_form = new Application_Form_Staff_Fornitori_Add();
            $this->_form->setAction($urlHelper->url(array(
                        'controller' => 'staff',
                        'action' => 'addsupplier'),
                        'default'
                        ));
            return $this->_form;
    }

public function editsupplierAction(){
    $supplierid = intval($this->_getParam('id', 0));
    $supplier = $this->_database->getSupplierById($supplierid);
    if($supplier == null){ $this->view->error = 'Fornitore non trovato'; }
    else{
        $this->view->editSupplier = $supplier;
        $editForm = new Application_Form_Staff_Fornitori_Modify($supplier);
        if(count($_POST) > 0 && $editForm->isValid($_POST)){
            $values = $editForm->getValues();
            $values['IDFornitore'] = $supplier->IDFornitore;
            $this->_database->updateSupplier($values);
            $this->_redirector->goToSimple('supplierslist', 'staff');
        }
        $this->view->editForm = $editForm;
    }
}

public function dealershiplistAction(){
    $this->view->assign(array('allDealerships' => $this->_database->getAllDealerships()));
}

public function deletedealershipAction(){
    $dealershipid = intval($this->_getParam('id', 0));
    $dealership = $this->_database->getDealershipById($dealershipid);
    if($dealership == null){ $this->view->error = 'Concessionaria non trovata'; }
    else{
        $this->_database->deleteDealership($dealership['IDAzienda']);
        $this->_redirector->goToSimple('dealershiplist', 'staff');
    }
}

public function newdealershipAction(){}
public function adddealershipAction()
    {
            if (!$this->getRequest()->isPost()) {
                    $this->_redirector->goToSimple('dealershiplist', 'staff');
            }
            $form=$this->_form;
            if (!$form->isValid($_POST)) {
                    return $this->render('newdealership');
            }
            $values = $form->getValues();
            $this->_database->saveDealership($values);
    $this->_redirector->goToSimple('dealershiplist', 'staff');
    }
private function getConcessionariaForm()
    {
            $urlHelper = $this->_helper->getHelper('url');
            $this->_form = new Application_Form_Staff_Concessionaria_Add();
            $this->_form->setAction($urlHelper->url(array(
                        'controller' => 'staff',
                        'action' => 'adddealership'),
                        'default'
                        ));
            return $this->_form;
    }

public function editdealershipAction(){
    $dealershipid = intval($this->_getParam('id', 0));
    $dealership = $this->_database->getDealershipById($dealershipid);
    if($dealership == null){ $this->view->error = 'Concessionaria non trovata'; }
    else{
        $this->view->editDealership = $dealership;
        $editForm = new Application_Form_Staff_Concessionaria_Modify($dealership);
        if(count($_POST) > 0 && $editForm->isValid($_POST)){
            $values = $editForm->getValues();
            $values['IDAzienda'] = $dealership->IDAzienda;
            $this->_database->updateDealership($values);
            $this->_redirector->goToSimple('dealershiplist', 'staff');
        }
        $this->view->editForm = $editForm;
    }
}

public function orderplacedAction(){
    $this->view->assign(array('allOrdersPlaced' => $this->_database->getAllOrderPlaced()));
}
```

```
public function deleteorderplacedAction(){
    $orderplacedid = intval($this->_getParam('id', 0));
    $orderplaced = $this->_database->getOrderPlacedById($orderplacedid);
    if($orderplaced == null){ $this->view->error = 'Ordine non trovato'; }
    else{
        $this->_database->deleteOrderPlaced($orderplaced['IDOrdine']);
        $this->_redirector->goToSimple('orderplaced', 'staff');
    }
}

public function neworderplacedAction(){}
public function addorderplacedAction()
    {
          if (!$this->getRequest()->isPost()) {
                $this->_redirector->goToSimple('orderplaced', 'staff');
          }
          $form=$this->_form;
          if (!$form->isValid($_POST)) {
                return $this->render('neworderplaced');
          }
          $values = $form->getValues();
          $this->_database->saveOrderPlaced($values);
    $this->_redirector->goToSimple('orderplaced', 'staff');
    }
private function getOrdineEffettuatoForm()
    {
          $urlHelper = $this->_helper->getHelper('url');
          $this->_form = new Application_Form_Staff_Ordineeffettuato_Add();
          $this->_form->setAction($urlHelper->url(array(
                        'controller' => 'staff',
                        'action' => 'addorderplaced'),
                        'default'
                        ));
          return $this->_form;
    }

public function editorderplacedAction(){
    $orderplacedid = intval($this->_getParam('id', 0));
    $orderplaced = $this->_database->getOrderPlacedById($orderplacedid);
    if($orderplaced == null){ $this->view->error = 'Ordine non trovato'; }
    else{
        $this->view->editOrderPlaced = $orderplaced;
        $editForm = new Application_Form_Staff_Ordineeffettuato_Modify($orderplaced);
        if(count($_POST) > 0 && $editForm->isValid($_POST)){
            $values = $editForm->getValues();
            $values['IDOrdine'] = $orderplaced->IDOrdine;
            $this->_database->updateOrderPlaced($values);
            $this->_redirector->goToSimple('orderplaced', 'staff');
        }
        $this->view->editForm = $editForm;
    }
}

public function orderreceivedAction(){
    $this->view->assign(array('allOrdersReceived' => $this->_database->getAllOrderReceived()));
}

public function deleteorderreceivedAction(){
    $orderreceivedid = intval($this->_getParam('id', 0));
    $orderreceived = $this->_database->getOrderReceivedById($orderreceivedid);
    if($orderreceived == null){ $this->view->error = 'Ordine non trovato'; }
    else{
        $this->_database->deleteOrderReceived($orderreceived['IDOrdine']);
        $this->_redirector->goToSimple('orderreceived', 'staff');
    }
}

public function neworderreceivedAction(){}
public function addorderreceivedAction()
    {
          if (!$this->getRequest()->isPost()) {
                $this->_redirector->goToSimple('orderreceived', 'staff');
          }
          $form=$this->_form;
          if (!$form->isValid($_POST)) {
                return $this->render('neworderreceived');
          }
          $values = $form->getValues();
          $this->_database->saveOrderReceived($values);
    $this->_redirector->goToSimple('orderreceived', 'staff');
    }

private function getOrdineRicevutoForm()
    {
          $urlHelper = $this->_helper->getHelper('url');
          $this->_form = new Application_Form_Staff_Ordinericevuto_Add();
          $this->_form->setAction($urlHelper->url(array(
                        'controller' => 'staff',
                        'action' => 'addorderreceived'),
                        'default'
                        ));
          return $this->_form;
    }
```

```
    public function editorderreceivedAction(){
        $orderreceivedid = intval($this->_getParam('id', 0));
        $orderreceived = $this->_database->getOrderReceivedById($orderreceivedid);
        if($orderreceived == null){ $this->view->error = 'Ordine non trovato'; }
        else{
            $this->view->editOrderReceived = $orderreceived;
            $editForm = new Application_Form_Staff_Ordinericevuto_Modify($orderreceived);
            if(count($_POST) > 0 && $editForm->isValid($_POST)){
                $values = $editForm->getValues();
                $values['IDOrdine'] = $orderreceived->IDOrdine;
                $this->_database->updateOrderReceived($values);
                $this->_redirector->goToSimple('orderreceived', 'staff');
            }
            $this->view->editForm = $editForm;
        }
    }
}
```

Listato 4.39. Definizione della classe `StaffController` che estende la classe `Zend_Controller_Action`

Figura 4.13. Schermata relativa alla home page della sezione "staff" del sito

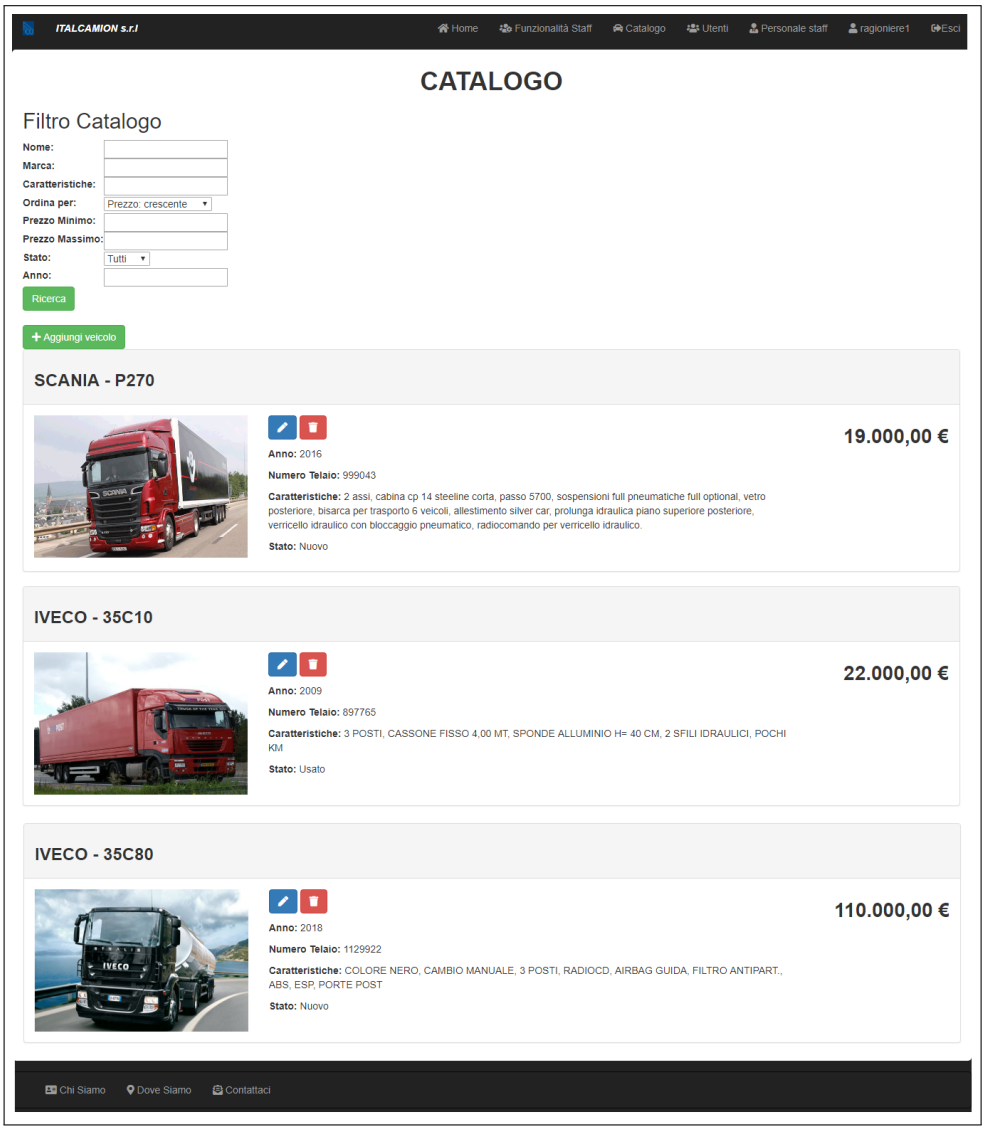


Figura 4.14. Schermata relativa alla pagina “catalog” della sezione “staff”; tramite questa pagina un componente dello staff può aggiungere, modificare o eliminare un veicolo

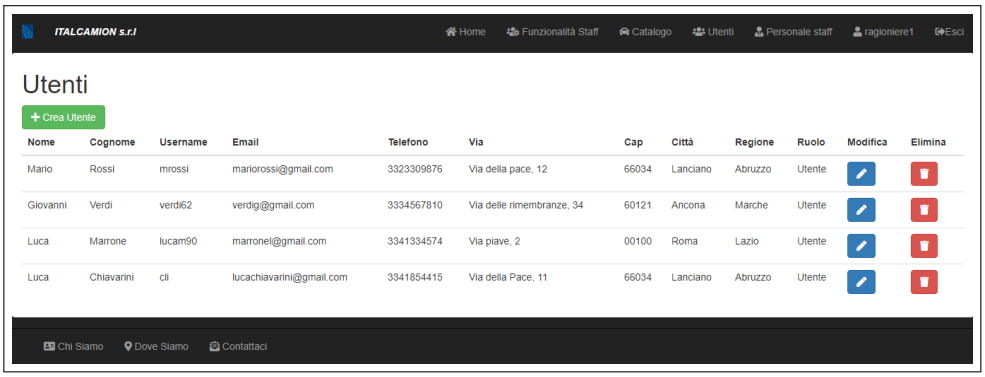


Figura 4.15. Schermata relativa alla pagina "users" della sezione "staff"; tramite questa pagina un componente dello staff ha la possibilità di visualizzare, aggiungere, modificare o eliminare clienti iscritti al sito

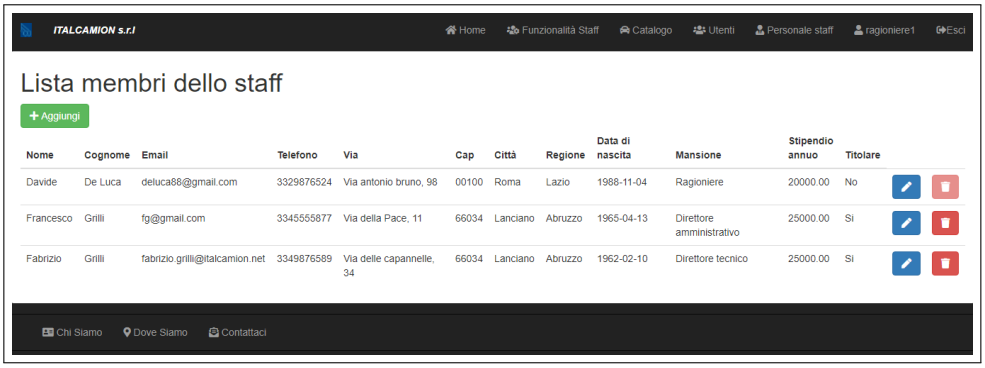


Figura 4.16. Schermata relativa alla pagina "personale" della sezione "staff"; questa pagina contiene le informazioni relative a tutti i componenti dello staff (quindi di tutto il personale dell'azienda). Essa, inoltre, consente di aggiungere, modificare o eliminare un componente dello staff

Figura 4.17. Schermata relativa alla pagina "fstaff" della sezione "staff"; questa pagina racchiude i link per le funzionalità più importanti di un utente di livello "staff"

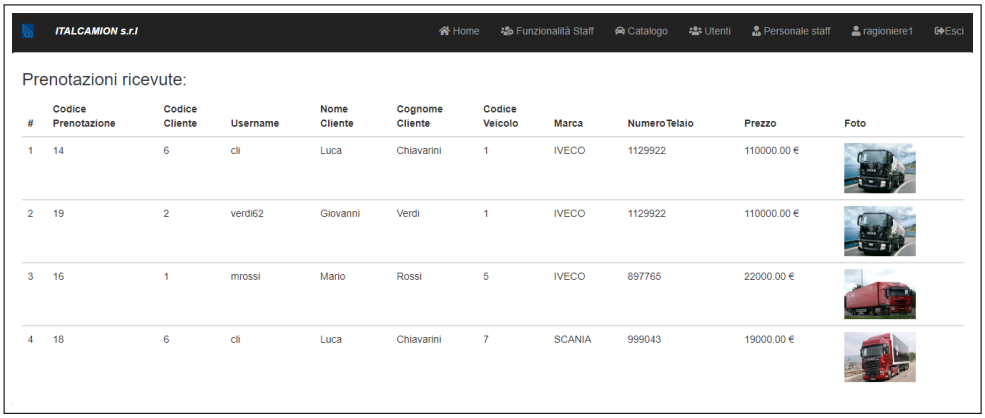


Figura 4.18. Schermata relativa alla pagina "listreservation" della sezione "staff"; questa pagina contiene la lista di tutte le prenotazioni e, per ciascuna di essa, vengono mostrate le informazioni sul cliente che l'ha effettuata, sul veicolo prenotato e sulla prenotazione ricevuta

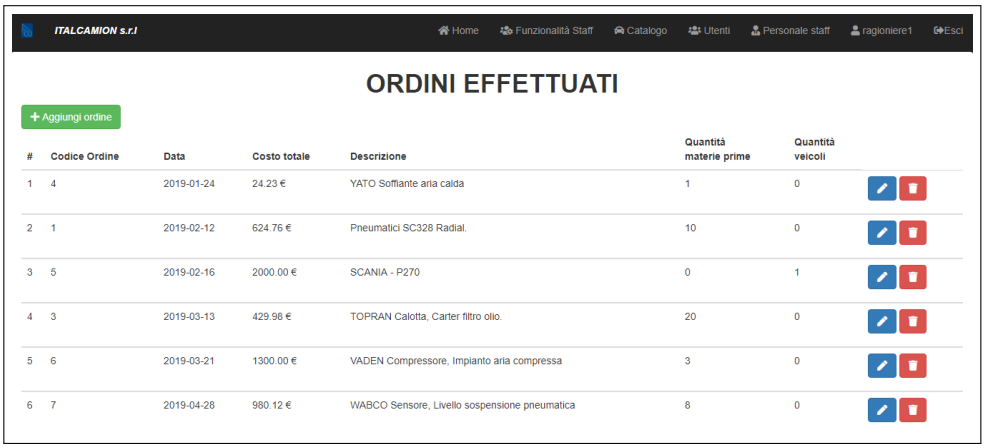


Figura 4.19. Schermata relativa alla pagina "orderplaced" della sezione "staff"; questa pagina contiene la lista di tutti gli ordini effettuati dall'azienda e dà la possibilità di aggiungere, modificare o eliminare un ordine

Figura 4.20. Schermata relativa alla pagina "orderreceived" della sezione "staff"; questa pagina contiene la lista di tutti gli ordini ricevuti dall'azienda (che, quindi, sono stati effettuati dal cliente) e dà la possibilità di aggiungere, modificare o eliminare un ordine

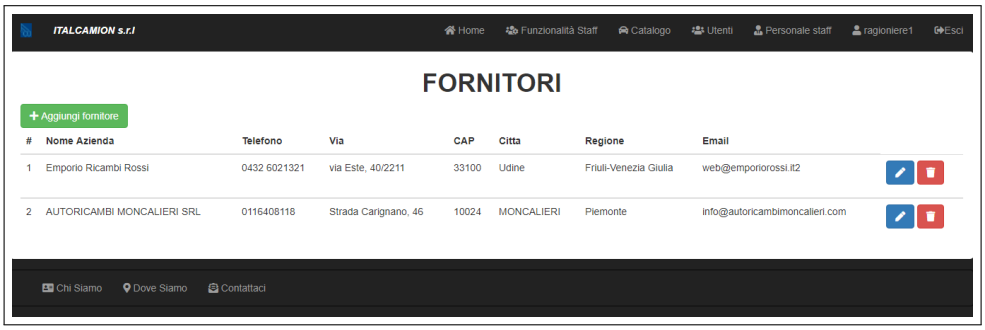


Figura 4.21. Schermata relativa alla pagina "supplierslist" della sezione "staff"; questa pagina contiene la lista di tutti i fornitori da cui l'azienda acquista le materie prime utili all'allestimento degli automezzi. Da qui, inoltre, è possibile aggiungere, modificare o eliminare un fornitore

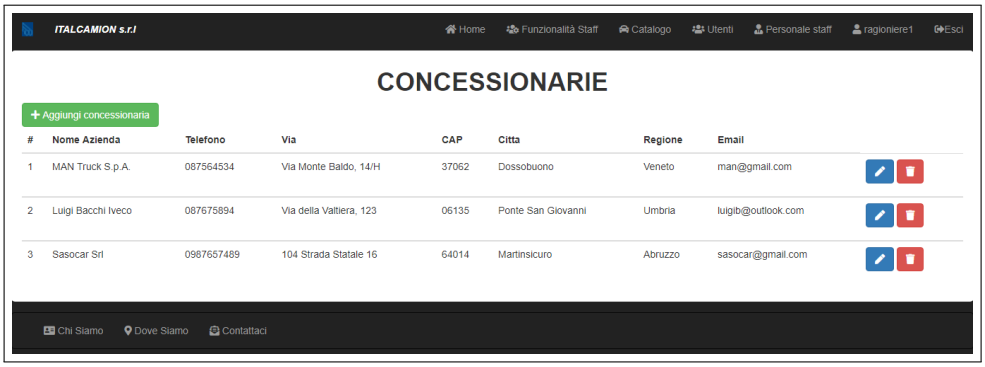
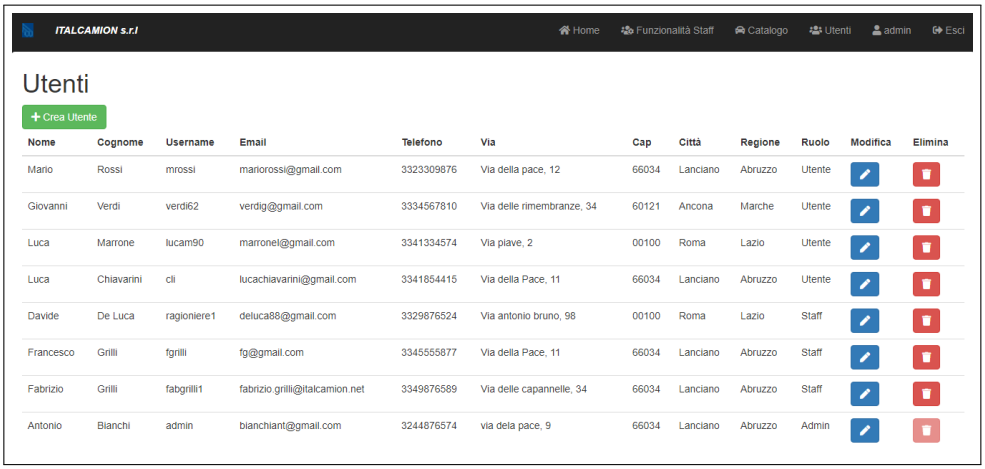


Figura 4.22. Schermata relativa alla pagina "dealershiplist" della sezione "staff"; questa pagina contiene la lista di tutte le concessionarie e di tutte le case produttrici da cui l'azienda acquista i veicoli (nuovi o usati). Da qui, inoltre, è possibile aggiungere, modificare o eliminare una concessionaria o una casa produttrice

4.2.4 Sezione "admin" del sito

Il codice del Controller del livello di utenza "admin" non verrà riportato dal momento che segue la falsariga di quello del Controller di livello "admin". Nelle Figure 4.23-4.24 vengono mostrate le schermate relative al livello di utenza "admin".

Figura 4.23. Schermata relativa alla home page della sezione "admin" del sito



Nome	Cognome	Username	Email	Telefono	Via	Cap	Città	Regione	Ruolo	Modifica	Elimina
Mario	Rossi	mrossi	mariorossi@gmail.com	3323309876	Via della pace, 12	66034	Lanciano	Abruzzo	Utente		
Giovanni	Verdi	verdi62	verdig@gmail.com	3334567810	Via delle rimembranze, 34	60121	Ancona	Marche	Utente		
Luca	Marrone	lucam90	marronel@gmail.com	3341334574	Via piave, 2	00100	Roma	Lazio	Utente		
Luca	Chiavarini	cli	lucachiavarini@gmail.com	3341854415	Via della Pace, 11	66034	Lanciano	Abruzzo	Utente		
Davide	De Luca	ragioniere1	deluca88@gmail.com	3329876524	Via antonio bruno, 98	00100	Roma	Lazio	Staff		
Francesco	Grilli	fgrilli	fg@gmail.com	3345555877	Via della Pace, 11	66034	Lanciano	Abruzzo	Staff		
Fabrizio	Grilli	fabgrilli1	fabrizio.grilli@italcamion.net	3349876589	Via delle capannelle, 34	66034	Lanciano	Abruzzo	Staff		
Antonio	Bianchi	admin	bianchiant@gmail.com	3244876574	via dela pace, 9	66034	Lanciano	Abruzzo	Admin		

Figura 4.24. Schermata relativa alla pagina "users" della sezione "admin"; tramite questa pagina l'amministratore del sito è in grado di creare, modificare od eliminare qualsiasi utente (a prescindere dalla classe di utenza alla quale quest'ultimo appartiene)

5

Discussione in merito al sistema realizzato

Il capitolo che segue analizzerà, in maniera critica, il progetto della presente tesi, andando a riconoscere le varie aree che rivestono un ruolo fondamentale nel progetto; fatto ciò, si analizzerà anche il rovescio della medaglia, quindi quelle aree che presentano alcune carenze, o che sono state curate di meno rispetto ad altre, e che, quindi, possono essere migliorate in futuro.

5.1 Punti di forza del sistema realizzato

Come si può chiaramente constatare dalle fasi di progettazione ed implementazione elaborate nei capitoli precedenti, le due principali tecnologie prodotte dal presente progetto di tesi sono rappresentate dalla base di dati e dall'applicazione Web. In entrambi, infatti, è possibile distinguere alcune aree che sono state sviluppate con maggior meticolosità e che sono poi diventate i punti di forza del sistema realizzato.

Di seguito verranno elencati i principali punti di forza su cui si basa il presente progetto di tesi:

- Aver adottato un design pattern come il Pattern MVC ha offerto la possibilità di separare la logica applicativa (in questo contesto spesso chiamata "logica di business"), a carico delle componenti denominate Controller e Model, e l'interfaccia utente, a carico della componente View. Questa distinzione, infatti, permette di cambiare il look-and-feel del sito agendo solo sulla componente View (quindi senza dover modificare le altre parti del codice).
- L'utilizzo del Framework Zend ha sicuramente favorito la modularità dell'applicazione Web che, quindi, non è composta da un unico file avente oltre 10000 righe di codice, ma è strutturata secondo un'architettura ben precisa (che è quella di Zend) che, ovviamente, consente molto più agevolmente la lettura del listato.
- È stato creato un file di configurazione generale per salvare le varie informazioni statiche del sito, come, ad esempio, i link delle immagini. In questo modo, in futuro, è possibile cambiare un'immagine in tutto il sito semplicemente modificando tale file (senza andare, quindi, a toccare il codice).
- L'utilizzo, nell'applicazione Web, di validatori (sia lato client che lato server) che regolano l'inserimento dei dati all'interno delle form. Consente di evitare

l'inserimento nel database di dati che possano generare errori nel momento in cui non rispettano i formati previsti per il database.
- Aver utilizzato il software MySQL come RDBMS assicura robustezza, sicurezza e affidabilità.
- Aver predisposto una progettazione completa e dettagliata del sistema informativo realizzato fa sì che chiunque andrà a lavorarci in futuro sarà agevolato dalla documentazione che è stata redatta.

5.2 Punti di debolezza del sistema realizzato

Il sistema realizzato nel presente progetto di tesi, oltre ai punti di forza sopra citati, che rappresentano i capisaldi del lavoro realizzato, presenta degli aspetti migliorabili, o che comunque sono stati trattati in maniera meno meticolosa dal momento che non si sono rivelati indispensabili alla causa. Queste aree, che possono essere definite come punti di debolezza del sistema realizzato, sono le seguenti:

- La mancanza di un'app per smartphone o di un sito web mobile; nonostante il sito Web abbia un layout scalabile e adattabile al dispositivo che la esegue, potrebbe essere un'idea implementare il sito web mobile o, ancora meglio, un'applicazione pensata appositamente per gli smartphone (essendo oggi il mezzo tecnologico più diffuso per accedere ad Internet).
- La mancanza di un sistema che permetta ad utenti ipovedenti di potersi interfacciare alla piattaforma. È possibile abilitare questa funzionalità iniziando dall'inserimento e dalla corretta valorizzazione del tag HTML denominata tt alt nei contenuti multimediali; agendo in questa maniera, attraverso un software di screen reader (come WebbiE), verrà letto il contenuto dell'alt text e compreso il visual, cosicché il browser abilitato interpreterà e interfaccerà l'immagine per l'utente.
- La mancanza di un sistema di internazionalizzazione dell'applicazione Web sia per quanto riguarda la lingua che per quanto riguarda la valuta; ora, infatti, l'unica lingua è l'italiano e l'unica valuta è l'euro.
- La mancanza di ulteriore sicurezza sulle password del database, dato che ora sono in chiaro.
- La cura scrupolosa della funzionalità di prenotazione online della prova gratuita di un automezzo.
- Il testing del sistema informativo realizzato; questa fase, infatti, poteva essere eseguita con maggior meticolosità. Questo è uno step importante nel ciclo di vita di un sistema informativo e viene utilizzato per individuare le carenze di correttezza, completezza e affidabilità delle componenti software in corso di sviluppo.

5.3 Lezioni apprese

Sia che le lezioni apprese vengano utilizzate per preparare i progetti in corso, sia che vengano utilizzate per identificare eventuali miglioramenti ai progetti su cui già si

sta lavorando, apprendere dai fallimenti o dai successi dei progetti passati diventa un elemento importantissimo. D'altronde, non prendere in considerazione ciò che è andato male durante un progetto è una pratica poco intelligente che potrebbe avere come conseguenza la ripetizione degli stessi errori. Al contrario, anche non sottolineare i successi di un progetto, e come questi sono stati raggiunti, è un errore poiché si perde l'opportunità di implementare buoni processi e best practice che possano contribuire al positivo completamento di un lavoro esistente o futuro. In breve, le "lesson learned" sono le informazioni documentate che riflettono sia le esperienze positive che quelle negative di un progetto. Esse rappresentano l'impegno dell'organizzazione verso l'eccellenza nella gestione dei progetti.

Gli insuccessi, in particolare, possono dire molto se si ha tempo e determinazione nell'analizzarli, ma può essere difficile tenere traccia di tutto questo senza un sistema per formalizzare e gestire le lezioni apprese come base di conoscenza utile per i progetti futuri. Tutto ciò richiede un investimento di tempo che, però, nel lungo periodo, si traduce in un risparmio di tempo determinato proprio dalla capacità di non ripetere gli errori e di utilizzare ciò che ha dimostrato di produrre risultati positivi.

Documentare le lezioni apprese significa registrare in corso d'opera i successi e gli insuccessi di un progetto per poi, alla fine dello stesso, costruire un rapporto che dettagli:

- i motivi delle difficoltà incontrate;
- le contromisure che sono state adottate;
- il livello di efficacia delle contromisure;
- le raccomandazioni per future iniziative.

Il "Lessons Learned Register" è un documento di progetto che viene creato la prima volta che si esegue il processo e che viene aggiornato in continuazione dal processo stesso e da tanti altri processi. All'interno di questo documento viene illustrato tutto ciò che è accaduto nel progetto, nel bene e nel male, per fare esperienza sia nel futuro a breve termine che in quello a lungo termine.

Le lezioni che sono state apprese durante la progettazione e l'implementazione del sistema informativo preso in oggetto per la presente tesi sono state le seguenti:

- La progettazione è la fase più importante di tutte. Abbiamo, infatti, potuto constatare che da una solida e completa progettazione ne deriva un'implementazione molto più agevole e molto meno improvvisata e lasciata al caso.
- Non bisogna cambiare le cose in corso d'opera ma bisogna prevedere tutto anticipatamente; è stato riscontrato, nel corso del progetto, che da un minimo cambiamento possono derivare cambiamenti concatenati molto dispendiosi in termini di tempo.
- Rispettare rigorosamente i vari step della progettazione del sistema informativo è fondamentale per la successiva implementazione e la buona riuscita del progetto.
- Documentarsi il più possibile circa l'azienda a cui è rivolto il sistema informativo e, soprattutto, recarsi sul posto è risultato molto importante per riuscire a capire gli aspetti fondamentali su cui porre maggior attenzione.
- Basarsi su documenti e materiale dell'azienda già prodotto da altri può essere un ottimo spunto da cui prendere ispirazione.

- È conveniente prendersi più tempo per occuparsi della fase di testing del software per individuare le carenze di correttezza, completezza e affidabilità presenti.
- È opportuno utilizzare sempre uno strumento che ci consenta di effettuare il backup periodico del nostro progetto; ad esempio, per il presente progetto, è stato utilizzato Git, un software di controllo del versioning distribuito utilizzabile da interfaccia a riga di comando.
- È opportuno cercare di utilizzare sempre software open source e freeware per evitare di avere problemi legali di qualsiasi natura.

6

Conclusioni e sguardo al futuro

Nella presente tesi abbiamo trattato le varie fasi che compongono la progettazione e l'implementazione di un sistema informativo prendendo in esame le attività dell'azienda "Italcamion s.r.l.", concessionaria di automezzi situata nel sulla costa del territorio abruzzese. Quanto trattato può essere ovviamente sfruttato dall'azienda presa in esame, ma potrebbe essere anche un buon punto di partenza per qualunque altra azienda che abbia caratteristiche simili e opera nello stesso settore della società "Italcamion". Guardando con un occhio critico il sistema informativo che ne è venuto fuori, è possibile senz'altro individuare aspetti migliorabili o funzionalità necessarie non presenti; alcune idee per il futuro potrebbero essere le seguenti:

- rendere accessibile il portale anche per smartphone attraverso un'app per smartphone o un sito web mobile;
- abilitare il sito Web anche per browser pensati per utenti ipovedenti;
- migliorare la funzionalità di prenotazione online della prova gratuita inserendo lo scaglionamento orario dei giorni;
- abilitare la possibilità di acquistare online, quindi abilitare i pagamenti online;
- internazionalizzare il sito web sia per quanto riguarda la lingua che per quanto riguarda la valuta;
- criptare le password utilizzando, ad esempio, metodi come l'hasing (o altri algoritmi crittografici), magari aggiungendo anche il salt nel database;
- aumentare la visibilità del sito Web su Google con la SEO o attraverso Google AdWords;
- inserire la validazione delle form tramite la tecnologia AJAX[4] per permettere di presentare all'utente un messaggio di errore non appena il campo contenente un valore non valido viene abbandonato per passare al campo successivo;
- aumentare, sull'applicazione Web, le funzionalità disponibili per il cliente, come ad esempio:
  - l'inserimento dell'assistenza clienti (che comprenda anche un servizio di live chat);

[4] La tecnologia basata su JavaScript che permette l'interazione asincrona tra Web client e Web server è chiamata AJAX (acronimo di "Asynchronous Javascript and XML") e, grazie ad essa, si ha la possibilità di fare una richiesta al server ed acquisire dati che possono essere iniettati nel documento visualizzato senza doverlo ricaricare.

  - l'inserimento della prenotazione online anche per il tagliando;
  - l'inserimento di una pagina dedicata alle FAQ;
  - l'inserimento di un blog all'interno del sito.
- aggiornare e manutenere costantemente il sistema informativo (un sito Web deve cambiare interfaccia almeno a cadenza annuale per evitare l'obsolescenza[5]).

[5] Avendo usato Zend siamo agevolati nell'aggiornare annualmente l'interfaccia (quindi la componente View) dal momento che questo framework struttura l'applicazione Web in maniera modulare.

Riferimenti bibliografici

1. Aumentare le visite di un sito web: 5 strategie che funzionano. `https://www.studiosamo.it/guide/aumentare-le-visite-del-sito/`, 2017.
2. The difference between Encryption, Hashing and Salting. `https://www.thesslstore.com/blog/difference-encryption-hashing-salting/`, 2018.
3. Articoli — HTML.it. `https://www.html.it/articoli/`, 2019.
4. Documentation - Zend Framework. `https://framework.zend.com/learn`, 2019.
5. T. Butler and K. Yank. *Sviluppare applicazioni con PHP e MySQL*. Apogeo, 2018.
6. P. Camagni and R. Nikolassy. *Database SQL e PHP*. Hoepli, 2018.
7. C. Cappiello, M. Fugini, P. Grefen, B. Pernici, and P. Plebani. *Fondamenti di Sistemi Informativi*. Createspace Independent Pub, 2018.
8. R. Cervelli. *Database e linguaggio SQL*. Createspace Independent Pub, 2015.
9. J. Duckett. *HTML e CSS. Progettare e costruire siti web*. Apogeo, 2017.
10. J. Duckett. *JavaScript e JQuery. Sviluppare interfacce web interattive*. Apogeo, 2017.
11. M. Fowler, L. Baresi, and S. Gaburri. *UML distilled*. Pearson, 2018.
12. A. Ghu. *Programmare con PHP*. FAG, 2006.
13. V. Giacchina. *Web design con Bootstrap*. Apogeo, 2017.
14. K. Lioy. *SVILUPPO WEB: Server Side Web Development*. Independently published, 2019.
15. C. Lucchese. *HTML e CSS: Corso di programmazione per principianti*. Independently published, 2019.
16. C. Lucchese. *Sviluppo Web: Web Development Lato Client per Principianti*. Independently published, 2019.
17. M. Pighin and A. Marzona. *Sistemi informativi aziendali*. Pearson, 2018.
18. J. Valacich and C. Schneider. *Sistemi informativi e trend digitali*. Pearson, 2019.
19. V. Vaswani. *Zend Framework, A Beginner's Guide*. McGraw-Hill Education, 2010.