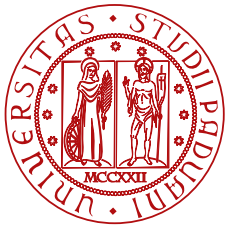


Università
degli Studi
di Padova

DIPARTIMENTO
MATEMATICA
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Tesi di Laurea in Informatica

Sviluppo di soluzioni software per la gestione commerciale su web

Laureando

Riccardo Favaron

Matricola 2042386

Relatrice

Prof.ssa Ombretta Gaggi

Università di Padova

Anno Accademico
2023/2024

Riccardo Favaron: ***Sviluppo di soluzioni software per la gestione commerciale su web****, Tesi di Laurea Triennale in Informatica, Università di Padova, Dipartimento di Matematica "Tullio Levi-Civita" ,* © *Settembre 2024.*

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine alla Professoressa Ombretta Gaggi per il costante supporto ed aiuto durante la stesura di questa tesi. La sua guida e i suoi consigli sono stati fondamentali per il completamento di questo lavoro.

Vorrei ringraziare l'azienda Ergon Informatica S.r.l. per l'opportunità che mi è stata data.

Un sentito ringraziamento va anche alla mia famiglia, che mi ha sempre sostenuto con pazienza durante tutti gli anni di studio universitario. Senza il loro incoraggiamento e sostegno questo percorso non sarebbe stato possibile.

Infine, vorrei ringraziare i miei compagni di corso con i quali ho condiviso molti momenti indimenticabili. Grazie per aver reso questo viaggio accademico un'esperienza piacevole ed arrichente.

Padova, Settembre 2024 *Riccardo Favaron*

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di *stage* dal laureando Riccardo Favaron presso l'azienda Ergon Informatica S.r.l..
Lo stage ha avuto una durata di circa 320 ore ed ha preso luogo dal 08/05/2024 al 02/07/2024.
Durante questo periodo vi è stata l'opportunità di approfondire e soprattutto di ampliare le conoscenze in ambito di sviluppo *web* e sviluppo di programmi *desktop* legati al mondo di *Windows*.

Indice

Lista delle Immagini

Lista delle Tabelle

Lista Frammenti di Codice

Lista degli Acronimi

API Application Programming Interface. 73

CSS Cascading Style Sheets. 11, 78

GUI Graphical User Interface. 17

HTML HyperText Markup Language. 11, 73, 78, 90

HTTP HyperText Transfer Protocol. 8, 12–16, 73, 74, 89, 96

IDE Integrated Development Environment. 17, 83

IIS Internet Information Services. 90

JSON JavaScript Object Notation. 73, 74

PMI Piccole Medie Imprese. 1

VPN Virtual Private Network. 5

1 Introduzione

1.1 L'azienda

Figura 1.1: Logo Ergon Informatica S.r.l.

Ergon Informatica S.r.l. è un'azienda italiana, con sede a Castelfranco Veneto (TV), fondata nel 1988 che opera nel settore IT. Sviluppa soluzioni gestionali per Piccole Medie Imprese (PMI) e conta oltre 250 clienti nel territorio nazionale.
I due software ERGDIS*[g]* ed ERGTRA*[g]*, sono degli ERP*[g]* completi, che si rivolgono in particolar modo ai due settori dell'alimentare e del trasporto.
La società si occupa di argomenti di sicuro interesse gestionale, quali il controllo di gestione, l'amministrazione e la finanza, la logistica in radiofrequenza, la *business intelligence*, le soluzioni CRM*[g]*, l'archiviazione ottica e sostitutiva, la produzione, l'automazione della forza vendite e la previsione delle vendite.
La società completa l'offerta con la vendita di prodotti *hadware*, servizi *web* e *hosting*, nonchè con progetti di *server consolidation* e virtualizzazione dei sistemi basati sulla tecnologia VMWARE*[g]*.
Il logo dell'azienda è riportato in figura 1.1.

1.2 Il progetto

1.2.1 Introduzione

Lo stage proposto prevede lo sviluppo di una *web application*, con la quale si potranno automatizzare una serie di operazioni attue a migliorare la gestione delle offerte commerciali, e dematerializzare la documentazione e le proposte ai clienti, favorendo così la comunicazione tra il cliente e la sede centrale. Il progetto si compone di tre parti principali:

- **Gestione configurazioni e domande**: consiste nella creazione di due moduli applicativi *desktop* che andranno ad integrarsi con il gestionale già presente, i quali sono necessari per la creazione e gestione delle configurazioni delle domande che verranno utilizzati dall'applicazione *web*.
- **Gestione offerta commerciale**: consiste nella creazione e nella proposta dell'offerta commerciale al cliente (listini, sconti, ecc...). È richiesto lo sviluppo di una *web app* dalla quale la risorsa commerciale (agente) dovrà compilare un questionario per il cliente e dalle risposte il *software* riuscirà a proporre in automatico le tipologie di prodotto che interessano al cliente e generale una proposta di sconti/prezzi.
- **Gestione documenti "legali"**: consiste nell'acquisizione e gestione dei dati relativi alla privacy, dei dati societari del cliente e di tutte quelle informazioni di tipo "legale". Con questi dati il *software* creerà un PDF da far firmare al cliente e trasmetterà alla sede.

1.2.2 Obiettivi

Nella tabella 1.1 sono riportati gli obiettivi previsti dallo *stage*. Sono indicati con il prefisso OB gli obiettivi obbligatori, mentre quelli desiderabili con il prefisso DE.

Codice obiettivo	Descrizione obiettivo
OB1	Sviluppo di un *web service* con tutti gli *endpoint* necessari alla lettura/scrittura dei dati utilizzati dall'applicazione *web*
OB2	Sviluppo di una *web application* dinamica per la creazione delle offerte commerciali
OB3	Sviluppo di un programma *desktop* per la configurazione delle domande necessarie alla creazione delle offerte commerciali
OB4	Sviluppo di un *web service* per la creazione di un documento PDF da restituire compilato

Tabella 1.1: Tabella degli obiettivi

1.3 Organizzazione del testo

Il secondo capitolo descrive nello specifico tutte le tecnologie e strumenti che sono stati utilizzati per la realizzazione del progetto;

Il terzo capitolo descrive nello specifico l'analisi dei requisiti che il prodotto deve soddisfare, comprensiva di diagrammi dei casi d'uso e del tracciamento dei requisiti;

Il quarto capitolo descrive nello specifico la progettazione del progetto, analizzando le scelte intraprese;

Il quinto capitolo descrive nello specifico la codifica ed illustra il prodotto *software* realizzato;

Il sesto capitolo descrive nello specifico i processi di verifica e validazione del progetto;

Il settimo capitolo riassume brevemente il lavoro svolto e presenta le conclusioni tratte dall'esperienza di *stage*.

In seguito sono descritte le convenzioni tipografiche utilizzate per la stesura di questo documento:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: $parola_{[g]}$;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere corsivo.

2

Tecnologie e strumenti

2.1 Ambiente di lavoro

Durante lo *stage* ad Ergon Informatica mi è stato messo a disposizione un *computer* portatile aziendale con all'interno installate tutte le tecnologie necessarie per lo sviluppo del progetto.
È interessante notare che i dipendenti svolgono il proprio lavoro quotidiano su macchine fisiche con sistema operativo *Windows*, oltre ad utilizzare molto frequentemente l'accesso remoto tramite *Desktop remoto*, funzionalità integrata nel sistema operativo, per connettersi ai *server* di sviluppo o alle macchine di colleghi, per recare assistenza e per il lavoro da casa, grazie anche all'utilizzo di Virtual Private Network (VPN).
Ad Ergon Informatica oltre il 90% del codice prodotto è relativo al gestionale, di conseguenza i programmatori sviluppano i singoli moduli in locale sulla propria macchina e solo successivamente li integrano assieme sul *server* di sviluppo dedicato. Di fatto, il gestionale non è altro che un grande menù che racchiude tutti i moduli richiesti e permette di organizzarli e avviarli in comodità. Per quanto riguarda lo sviluppo *web*, vengono adoperati vari *server* predisposti con le tecnologie necessarie.
Non si fa uso di sistemi per il versionamento.

La figura 2.1 riporta la schermata *home* del gestionale.

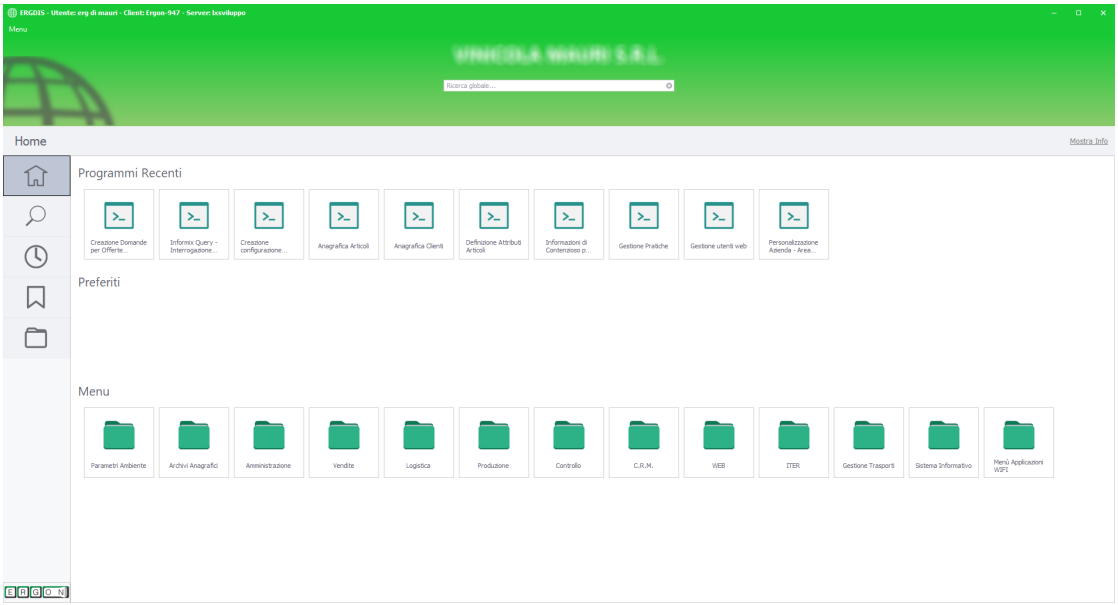


Figura 2.1: Schermata home gestionale

2.2 Introduzione al progetto

Come descritto brevemente in precedenza (sezione 1.2.1), il progetto si compone di tre parti principali, e questo ha comportato l'utilizzo di molte tecnologie, la cui maggior parte viene impiegata giornalmente dagli sviluppatori di Ergon Informatica per lo sviluppo del loro *software*. Il progetto è composto dallo sviluppo di due moduli *desktop* da integrare al gestionale già presente del cliente, dallo sviluppo di una *web app* per la gestione delle offerte commerciali e da un *web service* per la generazione di un documento per la *privacy*.

2.3 Tecnologie e strumenti

Di seguito vengono descritti tutte le tecnologie e strumenti impiegati per lo sviluppo del progetto. Buona parte di queste tecnologie non facevano parte del mio bagaglio personale e di conseguenza, durante la prima fase dello *stage*, ho dovuto realizzare uno studio approfondito necessario per portare a compimento il lavoro.

2.3.1 C#

C#[1] è un linguaggio di programmazione moderno sviluppato da *Microsoft* e strettamente legato al *framework .NET*. Il linguaggio *C#* è multi-paradigma, questo vale a dire che supporta sia la programmazione orientata agli oggetti, sia quella *event-driven*. È inoltre caratterizzato da una sintassi chiara e leggibile, mantenendo comunque una buona capacità di gestione dell'efficienza e un'alto livello di sicurezza. Il suo raggio di utilizzo è veramente ampio, ideale anche per lo sviluppo di applicazioni complesse quali *software desktop*, *web application*, nonché per lo sviluppo di giochi e soluzioni *enterprise*.

Il Linguaggio è stato adoperato per la codifica dei moduli *desktop* da integrare nel gestionale già esistente e per lo sviluppo dei due *web service* relativi all'applicazione *web* per la creazione e gestione delle offerte commerciali e per quello relativo alla generazione del documento sulla *privacy*.

La figura 2.2 riporta il logo di *C#*.

Figura 2.2: Logo C#

2.3.2 Windows Forms

Windows Forms[2] è un *framework* di interfaccia utente, parte integrante del *.NET framework* di *Microsoft* per la creazione di *app desktop* per *Windows*. Offre uno dei modi più produttivi per creare interfacce utenti interattive e visivamente attraenti, grazie all'utilizzo di una vasta gamma di componenti, controlli e funzionalità di trascinamento di essi. Questa tecnologia supporta la programmazione ad eventi e permette una gestione efficiente di quest'ultimi, rendendo molto semplice la risposta a *input* dell'utente. Inoltre è possibile realizzare applicazioni facili da distribuire, aggiornare e utilizzare sia *online*, che *offline*.

È stato impiegato per la realizzazione dei due moduli *desktop* da integrare nel gestionale già esistente.

2.3.3 ASP .NET Core

ASP .NET Core[3] è un *framework open source* ad alte prestazioni per lo sviluppo di applicazioni moderne e scalabili connesse ad *Internet*. Supporta diversi linguaggi, tra cui il *C#*, e fornisce strumenti per la gestione delle richieste Hyper-Text Transfer Protocol (HTTP), l'elaborazione delle risposte, sicurezza e tanto altro.

È stato impiegato per lo sviluppo dei due *web service* relativi all'applicazione *web* per la creazione e gestione delle offerte commerciali e per quello relativo alla generazione del documento sulla *privacy*.

La figura 2.3 riporta il logo di ASP .NET Core.

Figura 2.3: Logo ASP .NET Core

2.3.4 DevExpress

DevExpress[4] è una suite di componenti e strumenti per lo sviluppo di *software* moderni e all'avanguardia. È stata progettata per cercare di semplificare e migliorare lo sviluppo di interfacce utenti per applicazioni *desktop*, *web* e *mobile*, integrando al suo interno molte funzionalità aggiuntive quali creazione di *report* dinamici e *dashboard*. Sviluppata principalmente per il *framework .NET*, ma presenta librerie per lo sviluppo al di fuori dell'ambiente *Microsoft* quali React$_{[g]}$

, Angular[g] e Vue[g]. Inoltre offre delle librerie per la manipolazioni di file *Word*, *Excel* e *PDF*.

È stato impiegato per lo sviluppo dei moduli *desktop* e per il *web service* responsabile della generazione del documento sulla *privacy*.

La figura 2.4 riporta il logo di DevExpress.

Figura 2.4: Logo DevExpress

2.3.5 Informix

Informix[5] è un RDBMS[g] , sviluppato originariamente da *Informix Corporation*, la quale è stata successivamente acquisita da *IBM*. È un sistema noto per la sua affidabilità e la sua capacità di gestione di grandi moli di dati; ampiamente utilizzato per sistemi di gestione finanziaria, ERP e CRM.

La figura 2.5 riporta il logo di Informix.

Figura 2.5: Logo Informix

Informix non dispone di interfaccia grafica e di conseguenza i programmatori di Ergon Informatica hanno deciso di realizzare una serie di moduli grafici, facente parti del gestionale stesso, per semplificare le varie operazioni sul *database*. Il modulo in questione è chiamato *Informix Query*.
Come si può notare dalle seguenti figure, il modulo *Informix Query* è formato da varie viste tra cui quella per la gestione delle tabelle nel *database* (figura 2.6). In particolare, da questa vista è possibile ricercare una specifica tabella, visionarne i *record* e la struttura, e anche crearne una nuova, oltre ad altro ancora. È anche

presenta la vista SQL per eseguire per l'appunto comandi SQL specifici. Alla figura 2.7 è riportato uno *screenshot* della schermata.

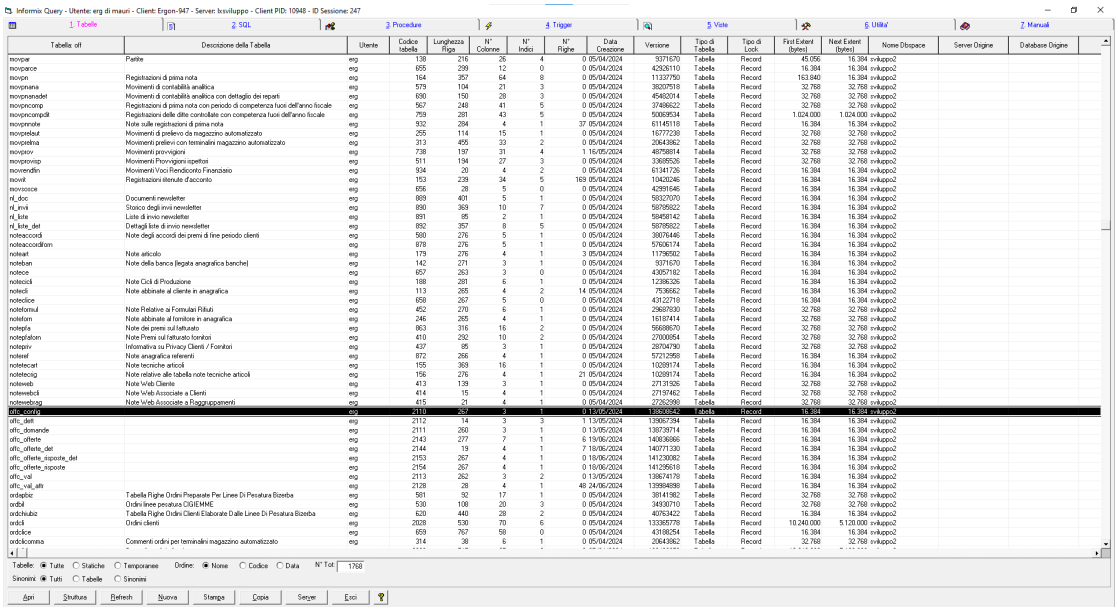


Figura 2.6: Informix Query - vista tabelle

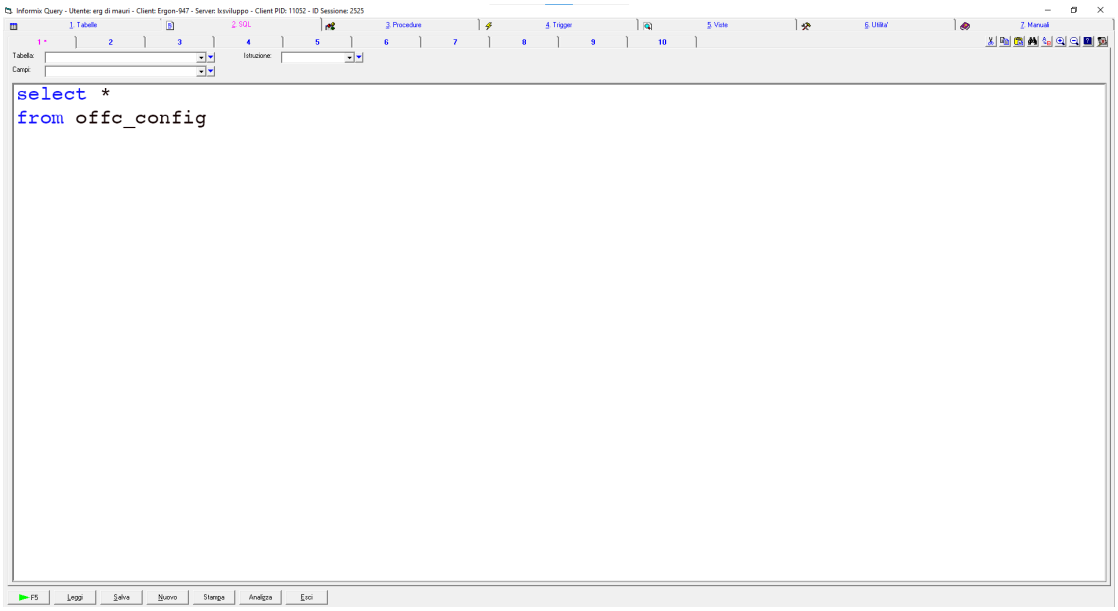


Figura 2.7: Informix Query - vista SQL

2.3.6 HTML, CSS, JavaScript

HyperText Markup Language (HTML) è il linguaggio di *markup* più utilizzato per i documento *web*, che permette di impaginare e formattare le pagine. Cascading Style Sheets (CSS) è un linguaggio di stile adoperato per descrivere la presentazione di un documento scritto, ad esempio in *HTML*, *XHTML* o *XML*. *JavaScript* è un linguaggio di programmazione multi-paradigma, ampiamente utilizzando in ambito *web* al fine, ad esempio, di variare il comportamento di una pagina.

Sono stati impiegati per la realizzazione della *web application* per la gestione delle offerte commerciali lato *front-end*.

La figura 2.8 riporta i loghi delle tecnologie impiegate per lo sviluppo del *front-end*.

Figura 2.8: Logo dello stack tecnologico del front-end

2.3.7 PHP

PHP[6] è un linguaggio di *scripting* lato *server* interpretato utilizzato soprattutto nella programmazione di pagine *web* dinamiche. La curva di apprendimento è relativamente bassa, rendendolo così un linguaggio ampiamente adoperato tutt'ora.

È stato impiegato per la realizzazione della *web application* per la gestione delle offerte commerciali lato *server*.

La figura 2.9 riporta il logi di PHP.

Figura 2.9: Logo PHP

2.3.8 cURL vs Guzzle

Introduzione

La *web application* per la gestione delle offerte commerciali necessita di dover comunicare con il *web service* realizzato ad hoc per gestire il flusso di dati tra l'applicazione e il *database*. Nasce dunque il bisogno di trovare una libreria PHP per poter effettuare delle richieste HTTP. Tra le varie opzioni si è deciso di comparare le due più utilizzate, ovvero cURL e Guzzle. Per ogni opzione si andranno ad analizzare gli aspetti principali, quali la facilità d'uso, la flessibilità, le funzionalità avanzate e le prestazioni, andando a rispondere ai seguenti punti:

- Descrizione;
- Caratteristiche;
- Utilizzo pratico;
- Altro.

Inoltre seguiranno degli esempi di codice per ciascuna opzione.

cURL

- **Descrizione**: libreria PHP che fornisce un'ampia gamma di funzionalità per effettuare richieste HTTP;
- **Caratteristiche**: controllo completo sulle richieste e supporto per molte funzioni avanzate;
- **Utilizzo pratico**: adatto a situazioni in cui è necessario avere un controllo dettagliato sulle richieste, come l'invio di *header* personalizzati o la gestione di autenticazione complesse;

- **Altro**: libreria molto popolare e ampiamente supportata dalla *community* PHP. Le prestazioni sono buone anche in scenari in cui si fa un intenso uso di rete; tuttavia, seppure una libreria completa rimane complessa e a volte sono necessarie molte righe di codice per effettuare delle operazioni basilari, a differenza di altre opzioni, in quanto si deve sempre inizializzare una sessione *curl* e configurare le opzioni tramite *curl_setopt()*. Il programmatore si occupa di gestire manualmente gli errori.

Ora vengono riportati degli esempi di codice per effettuare delle richieste HTTP con la libreria cURL.

Nel frammento di codice 2.1 è rappresentato un esempio di richiesta di tipo GET. La prima cosa da fare per effettuare una richiesta con la libreria cURL è quella di inizializzare una nuova sessione. Successivamente è necessario definire i vari parametri della richiesta, quali l'URL della richiesta stessa ed altri secondo le necessità. Nel nostro esempio è mostrato anche come impostare la volontà di ricevere un'eventuale risposta alla richiesta in questione. Poi si esegue la richiesta e si memorizza la risposta in una variabile. Viene anche controllata la presenza di errori durante la richiesta ed eventualmente vengono stampati a video. Infine si procede con la chiusura della sessione cURL ed è possibile utilizzare la risposta salvata in precedenza.

```php
$curl = curl_init();

$url = "https://api.example.com/resource";

curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

$response = curl_exec($curl);

if(curl_errno($curl)){
    echo 'Errore cURL: ' . curl_error($curl);
}

curl_close($curl);

echo $response;
```

Code 2.1: Richiesta GET cURL

Nel frammento di codice 2.2 è rappresentato un esempio di richiesta di tipo PUT. La prima cosa da fare per effettuare una richiesta con la libreria cURL è quella di inizializzare una nuova sessione. Successivamente è necessario definire i vari

parametri della richiesta, quali l'URL della richiesta stessa ed altri parametri secondo le necessità. Nel caso di una richiesta di tipo PUT è quasi sempre necessario definire i dati da inviare nel corpo della richiesta. Nel nostro esempio è mostrato come impostare, per l'appunto, i dati nel corpo della richiesta, la volontà di ricevere un'eventuale risposta alla richiesta in questione e come definire un *header* personalizzato. Poi si esegue la richiesta e si memorizza la risposta in una variabile. Viene anche controllata la presenza di errori durante la richiesta ed eventualmente vengono stampati a video. Infine si procede con la chiusura della sessione cURL ed è possibile utilizzare la risposta salvata in precedenza.

```php
$curl = curl_init();

$url = "https://api.example.com/resource/123";

$data = json_encode(array("name" => "Nuovo nome"));

curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl, CURLOPT_POSTFIELDS, $data);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_HTTPHEADER, array(
    'Content-Type: application/json',
    'Content-Length: ' . strlen($data)
));

$response = curl_exec($curl);

if(curl_errno($curl)){
    echo 'Errore cURL: ' . curl_error($curl);
}

curl_close($curl);

echo $response;
```

Code 2.2: Richiesta PUT cURL

Guzzle

- **Descrizione**: libreria PHP per effettuare richieste HTTP che offre un'api orientata agli oggetti e funzionalità avanzate;

- **Caratteristiche**: facilità d'uso, flessibilità, supporto per funzionalità avanzate come la gestione delle sessioni, delle automazioni e richieste asincrone;
- **Utilizzo pratico**: adatto per progetti che richiedono un'implementazione robusta e scalabile, con possibilità di gestione di scenari complessi;
- **Altro**: Gestione degli errori automatica, più leggibile rispetto a cURL; tuttavia richiede l'installazione di dipendenze aggiuntive tramite *Composer* che potrebbe aumentare la dimensione del progetto.

Ora vengono riportati degli esempi di codice per effettuare delle richieste HTTP con la libreria Guzzle.
Nel frammento di codice 2.3 è mostrato il comando per l'installazione di Guzzle.

```
composer require guzzlehttp/guzzle
```

Code 2.3: Installazione Guzzle

Nel frammento di codice 2.4 è rappresentato un esempio di richiesta di tipo GET. La prima cosa da fare per effettuare una richiesta con la libreria Guzzle è quella di importare la classe *Client* e creare una nuova istanza di quella classe. Successivamente è necessario definire i vari parametri della richiesta, quali l'URL della richiesta stessa ed altri parametri secondo le necessità. Si esegue la richiesta andando a passare i vari parametri definiti in precedenza e si memorizza la risposta in una variabile. In seguito si estrapola il contenuto del corpo della risposta ricevuta, il quale contiene al suo interno la risposta vera e propria, ed è possibile utilizzare la variabile salvata.

```
use GuzzleHttp\Client;

$client = new Client();

$url = "https://api.example.com/resource";

$response = $client->request('GET', $url);

$body = $response->getBody()->getContents();

echo $body;
```

Code 2.4: Richiesta GET Guzzle

Nel frammento di codice 2.5 è rappresentato un esempio di richiesta di tipo PUT. La prima cosa da fare per effettuare una richiesta con la libreria Guzzle è quella

di importare la classe *Client* e creare una nuova istanza di quella classe. Successivamente è necessario definire i vari parametri della richiesta, quali l'URL della richiesta stessa ed altri parametri secondo le necessità. Nel caso di una richiesta di tipo PUT è quasi sempre necessario definire i dati da inviare nel corpo della richiesta. Si esegue la richiesta andando a passare i vari parametri definiti in precedenza, tra cui i dati da inviare, e si memorizza la risposta in una variabile. In seguito si estrapola il contenuto del corpo della risposta ricevuta, il quale contiene al suo interno la risposta vera e propria, ed è possibile utilizzare la variabile salvata.

```
use GuzzleHttp\Client;

$client = new Client();

$url = "https://api.example.com/resource/123";

$data = array("name" => "Nuovo nome");

$response = $client->request('PUT', $url, [
    'json' => $data,
]);

$body = $response->getBody()->getContents();

echo $body;
```

Code 2.5: Richiesta PUT Guzzle

Conclusioni

Entrambe le opzioni sono molto valide e si sposano bene con il progetto. Guzzle si presenta come un'alternativa ad alto livello e molto semplice da utilizzare rispetto a cURL; tuttavia, necessita di un'installazione che potrebbe appesantire il progetto.
È fondamentale ricordare che Guzzle utilizza cURL sotto il proprio cofano. Per questo, gli aspetti negativi di cURL, quali la minor leggibilità e l'impiego di più righe di codice, possono passare in secondo piano, in quanto la creazione di una classe o funzione semplifica e favorisce una miglior gestione delle richieste HTTP.
Si conclude così che la scelta ricade proprio su cURL.

2.3.9 Visual Studio 2019

Visual Studio[7] è un Integrated Development Environment (IDE), ambiente di sviluppo, in italiano, sviluppato da *Microsoft* che fornisce un *set* completo di strumenti per lo sviluppo *software*. Viene impiegato per la creazioni di svariate cose, tra cui, la Graphical User Interface (GUI), applicazioni *desktop*, *mobile*, *web* e molto altro. Inoltre, non è specifico per un linguaggio in particolare, bensì supporta oltre 30 linguaggi di programmazione differenti, tra questi troviamo il *C#*.

È stato impiegato per lo sviluppo dei moduli *desktop* da integrare nel gestionale già presente e per lo sviluppo dei due *web service* relativi all'applicazione *web* per la creazione e gestione delle offerte commerciali e per quello relativo alla generazione del documento sulla *privacy*.

La figura 2.10 riporta il logo di Visual Studio.

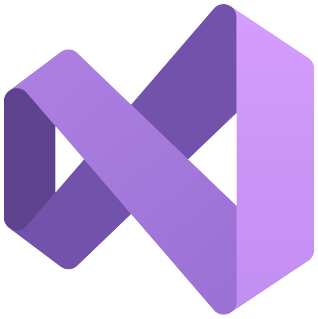


Figura 2.10: Logo Visual Studio

3

Analisi dei requisiti

3.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (anche noti con il loro termine in lingua inglese *Use Case Diagram*) sono diagrammi di tipo UML[g] , dedicati alla descrizione delle funzioni e dei servizi offerti da un sistema, in base a come tali funzioni e servizi sono state percepite e utilizzate dagli attori che interagiscono col sistema stesso. Ogni caso d'uso viene identificato con la seguente codifica:

UC<CodicePadre>.<CodiceFiglio>

dove:

- **UC**: abbreviazione di *Use Case*;
- **<CodicePadre>**: identificativo del *Use Case*; ogni *Use Case* dispone di un codice diverso;
- **<CodiceFiglio>**: identifica la relazione con un eventuale *Use Case* "padre". È importante ribadire come questo formalismo sia gerarchico, ovvero un codice figlio può essere padre di un suo eventuale codice figlio.

Per ogni caso d'uso vengono riportati i seguenti dettagli:

- **Figura**;
- **Attori principali**: gli attori che interagiscono con lo specifico caso d'uso analizzato;

- **Precondizioni**: condizioni di utente e/o sistema, necessarie affinchè si verifichi quel caso d'uso;
- **Postcondizioni**: rappresentano lo stato del sistema e/o utente, dopo che il caso d'uso è stato eseguito;
- **Scenario principale**: in questa sezione si elencano le fasi che caratterizzano il caso d'uso;
- **Estensioni**(se presenti): elenco di eventuali estensioni del caso d'uso analizzato;
- **Generalizzazioni**(se presenti): elenco di eventuali generalizzazioni tra i casi d'uso e anche tra gli attori in gioco.

3.1.1 Attori

Gli attori rappresentano gli utenti che interagiscono con il sistema.
Per il progetto sono stati individuati diversi attori, segue la loro descrizione:

- **Utente G**: rappresenta l'utente del gestionale, ovvero l'utente che può utilizzare il modulo di creazione e gestione delle configurazioni e il modulo per la creazione e gestione delle domande e valori. Per convenzione indichiamo l'*Utente G* come l'utente che dispone dei permessi necessari ad utilizzare i due moduli del gestionale realizzati in questo progetto. È interessante notare come il gestionale, in base a com'è stato richiesto, può gestire un unico utente che ha accesso a tutti i moduli, o gestire molteplici utenti con permessi diversi;
- **Utente AGE**: rappresenta l'utente agente della *web application* per la gestione delle offerte commerciali;
- **Utente CAM**: rappresenta l'utente amministratore della *web application* per la gestione delle offerte commerciali;
- **Utente W non autenticato**: rappresenta l'utente della *web application* per la gestione delle offerte commerciali non autenticato; di fatto rappresenta un utente AGE o CAM che non ha ancora effettuato il *login*.

3.1.2 Gestionale

Nella seguente sezione vengono illustrati e descritti i casi d'uso relativi ai moduli aggiunti al gestionale già esistente per la creazione e gestione delle domande e dei valori e per la creazione e gestione delle configurazioni.

Nella figura 3.1 è riportato il diagramma del modulo *offc_domande* del gestionale con tutti i casi d'uso.

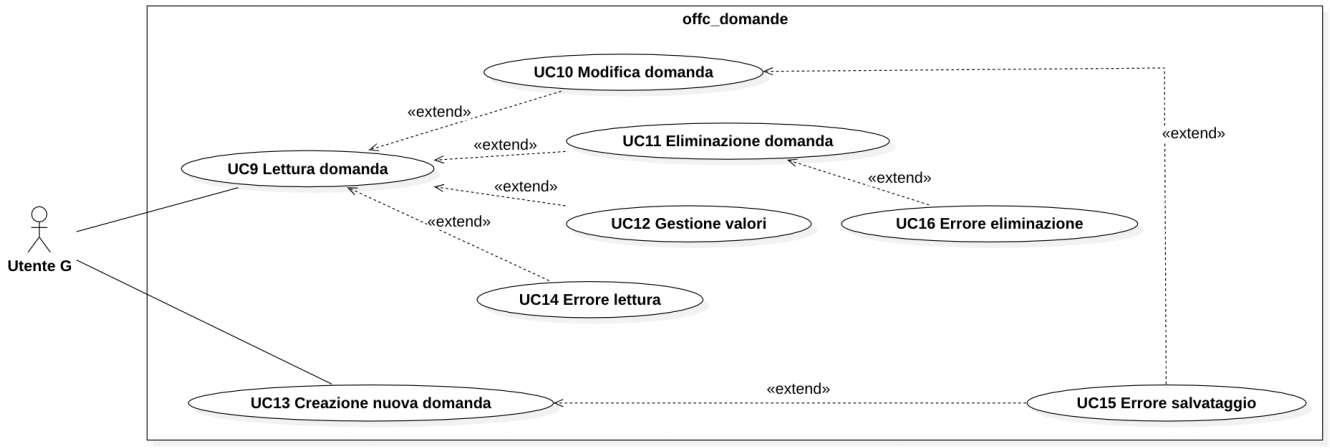


Figura 3.1: Modulo offc_config

UC1 - Lettura domanda

In figura 3.2 è rappresentato il caso d'uso tramite UML.

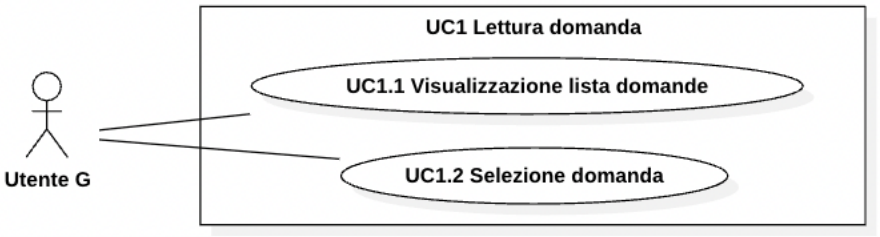


Figura 3.2: UC1 - Lettura domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha avviato il modulo *offc_domande* e vuole leggere una domanda;
- **Postcondizioni**: L'utente ha letto la domanda desiderata e visualizza la domanda;
- **Scenario principale**:
  1. L'utente clicca il bottone per la lettura di una domanda.
- **Estensioni**: UC6 - Errore lettura.

UC1.1 - Visualizzazione lista domande

In figura 3.3 è rappresentato il caso d'uso tramite UML.

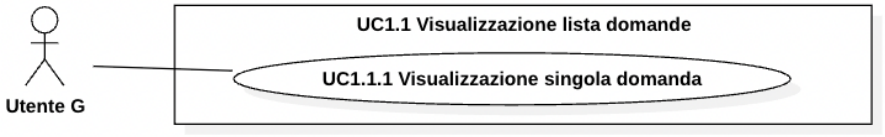


Figura 3.3: UC1.1 - Visualizzazione lista domande

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente sta effettuando la lettura di una domanda;
- **Postcondizioni**: L'utente visualizza la lista delle domande;
- **Scenario principale**:
  1. L'utente visualizza la lista delle domande.

UC1.1.1 - Visualizzazione singola domanda

In figura 3.4 è rappresentato il caso d'uso tramite UML.

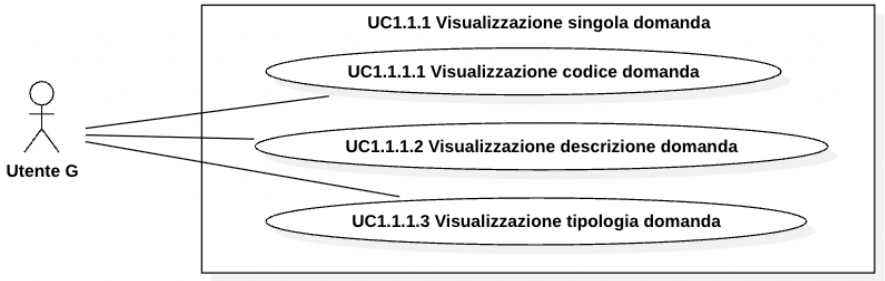


Figura 3.4: UC1.1.1 - Visualizzazione singola domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza correttamente la lista delle domande;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
  1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.

UC1.1.1.1 - Visualizzazione codice domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola domanda dalla lista;
- **Postcondizioni**: L'utente visualizza il codice della domanda che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il codice della domanda che sta osservando.

UC1.1.1.2 - Visualizzazione descrizione domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola domanda dalla lista;
- **Postcondizioni**: L'utente visualizza la descrizione della domanda che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la descrizione della domanda che sta osservando.

UC1.1.1.3 - Visualizzazione tipologia domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola domanda dalla lista;
- **Postcondizioni**: L'utente visualizza la tipologia della domanda che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la tipologia della domanda che sta osservando.

UC1.2 - Selezione domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere una domanda; l'utente clicca su una singola domanda;
- **Postcondizioni**: L'utente visualizza i dati della domanda selezionata;
- **Scenario principale**:
  1. L'utente clicca una singola domanda;
  2. L'utente visualizza i dati della domanda.

UC2 - Modifica domanda

In figura 3.5 è rappresentato il caso d'uso tramite UML.

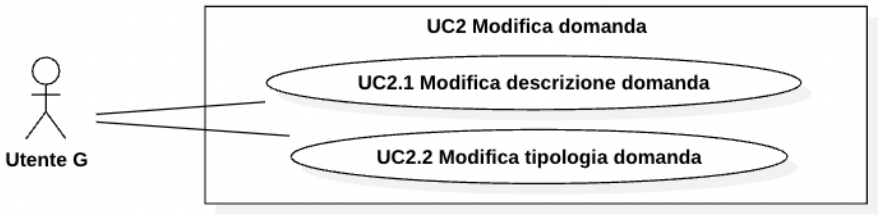


Figura 3.5: UC2 - Modifica domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la domanda letta e decide di modificarla;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare i campi modificabili, ovvero descrizione (UC2.1) e tipologia (UC2.2);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC7 - Errore salvataggio.

UC2.1 - Modifica descrizione domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la domanda letta e decide di modificare la descrizione;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare il campo descrizione;
  2. L'utente inserisce i nuovi dati.

UC2.2 - Modifica tipologia domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la domanda letta e decide di modificare la tipologia;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare il campo tipologia;
  2. L'utente inserisce i nuovi dati.

UC3 - Eliminazione domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la domanda letta e decide di eliminarla;
- **Postcondizioni**: I dati vengono eliminati dal sistema;
- **Scenario principale**:
  1. L'utente clicca il bottone per eliminare la domanda.
  2. L'utente convalida la scelta confermando l'eliminazione dei dati nel sistema;
  3. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC8 - Errore eliminazione.

UC4 - Gestione valori

In figura 3.6 è rappresentato il caso d'uso tramite UML.

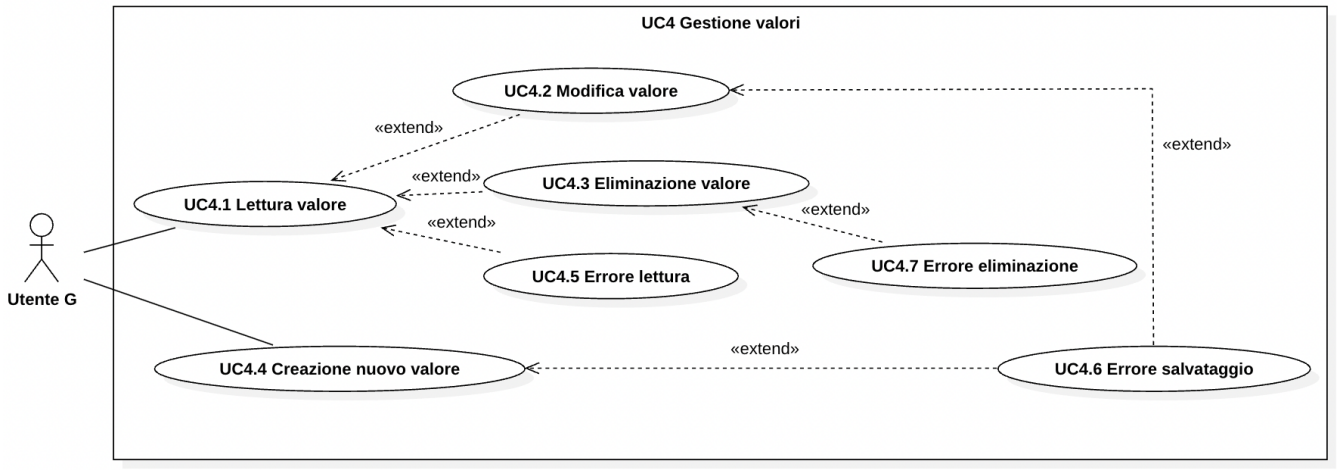


Figura 3.6: UC4 - Gestione valori

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha letto una domanda e desidera gestire i relativi valori;
- **Postcondizioni**: L'utente ha interagito con i valori della domanda letta;
- **Scenario principale**:
  1. L'utente clicca il bottone per la gestione dei valori relativi alla domanda.

UC4.1 - Lettura valori

In figura 3.7 è rappresentato il caso d'uso tramite UML.

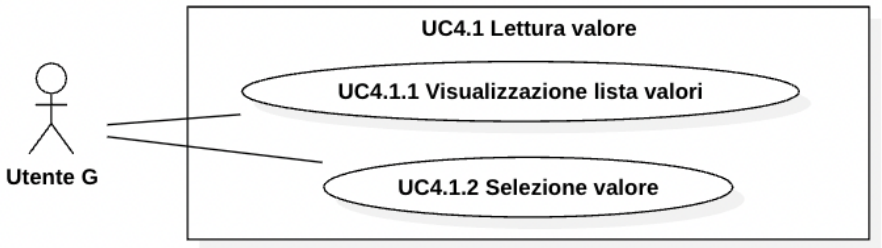


Figura 3.7: UC4.1 - Lettura valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha letto una domanda e vuole interagire con i relativi valori, desidera quindi leggere un valore;
- **Postcondizioni**: L'utente ha letto il valore desiderato e visualizza il valore;
- **Scenario principale**:
  1. L'utente clicca il bottone per la lettura di un valore;
- **Estensioni**: UC4.5 - Errore lettura.

UC4.1.1 - Visualizzazione lista valori

In figura 3.8 è rappresentato il caso d'uso tramite UML.

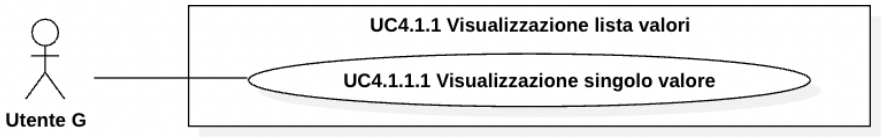


Figura 3.8: UC4.1.1 - Visualizzazione lista valori

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente sta effettuando la lettura di un valore;
- **Postcondizioni**: L'utente visualizza la lista dei valori;
- **Scenario principale**:
  1. L'utente visualizza la lista dei valori.

UC4.1.1.1 - Visualizzazione singolo valore

In figura 3.9 è rappresentato il caso d'uso tramite UML.

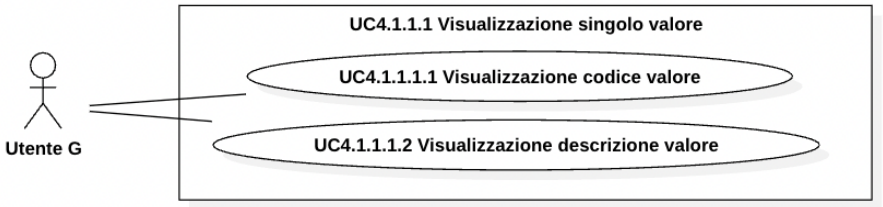


Figura 3.9: UC4.1.1.1 - Visualizzazione singolo valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza correttamente la lista dei valori;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
    1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.

UC4.1.1.1.1 - Visualizzazione codice valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza un singolo valore dalla lista;
- **Postcondizioni**: L'utente visualizza il codice del valore che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il codice del valore che sta osservando.

UC4.1.1.1.2 - Visualizzazione descrizione valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza un singolo valore dalla lista;
- **Postcondizioni**: L'utente visualizza la descrizione del valore che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la descrizione del valore che sta osservando.

UC4.1.2 - Selezione valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere un valore; l'utente clicca su un singolo valore;
- **Postcondizioni**: L'utente visualizza i dati del valore selezionato;
- **Scenario principale**:
  1. L'utente clicca un singolo valore;
  2. L'utente visualizza i dati del valore.

UC4.2 - Modifica valore

In figura 3.10 è rappresentato il caso d'uso tramite UML.

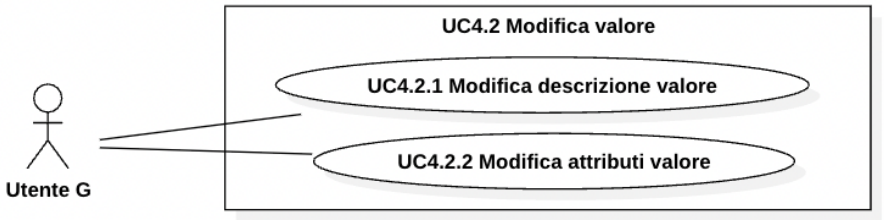


Figura 3.10: UC4.2 - Modifica valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando il valore letto e decide di modificarlo;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare i campi modificabili, ovvero descrizione (UC4.2.1) e attributi (UC4.2.2);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC4.6 - Errore salvataggio.

UC4.2.1 - Modifica descrizione valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando il valore letto e decide di modificare la descrizione;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare il campo descrizione;
  2. L'utente inserisce i nuovi dati.

UC4.2.2 - Modifica attributi

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando il valore letto e decide di modificare gli attributi;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare gli attributi;
  2. L'utente inserisce i nuovi dati.

UC4.3 - Eliminazione valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando il valore letto e decide di eliminarlo;
- **Postcondizioni**: I dati vengono eliminati dal sistema;
- **Scenario principale**:
  1. L'utente clicca il bottone per eliminare il valore.
- **Estensioni**: UC4.7 - Errore eliminazione.

UC4.4 - Creazione nuovo valore

In figura 3.11 è rappresentato il caso d'uso tramite UML.

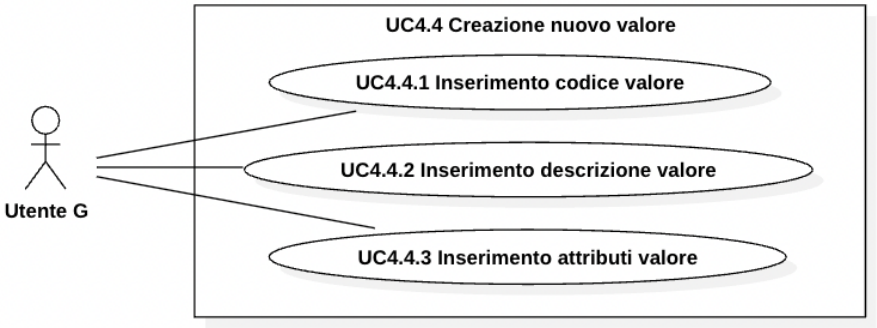


Figura 3.11: UC4.4 - Creazione nuovo valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare un nuovo valore;
- **Postcondizioni**: Il nuovo valore viene salvato nel sistema;
- **Scenario principale**:
  1. L'utente compila i campi necessari alla creazione di un nuovo valore, ovvero codice (UC4.4.1), descrizione (UC4.4.2), attributi (UC4.4.3);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC4.6 - Errore salvataggio.

UC4.4.1 - Inserimento codice valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare un nuovo valore; inserisce il nuovo codice valore;
- **Postcondizioni**: Il nuovo codice valore è visualizzato nel campo;
- **Scenario principale**:
  1. L'utente inserisce il nuovo codice nel campo.

UC4.4.2 - Inserimento descrizione valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare un nuovo valore; inserisce la nuova descrizione;
- **Postcondizioni**: La nuova descrizione è visualizzata nel campo;
- **Scenario principale**:
    1. L'utente inserisce la nuova descrizione nel campo.

UC4.4.3 - Inserimento attributi valore

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare un nuovo valore; inserisce i nuovi attributi;
- **Postcondizioni**: I nuovi attributi sono visualizzati nel campo;
- **Scenario principale**:
    1. L'utente inserisce i nuovi attributi nel campo.

UC4.5 - Errore lettura

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere un valore;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

UC4.6 - Errore salvataggio

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole creare un nuovo valore o modificarne uno esistente; l'utente inserisce dati inadeguati;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
  1. L'utente visualizza un messaggio di errore.

UC4.7 - Errore eliminazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole eliminare il valore letto;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
  1. L'utente visualizza un messaggio di errore.

UC5 - Creazione nuova domanda

In figura 3.12 è rappresentato il caso d'uso tramite UML.

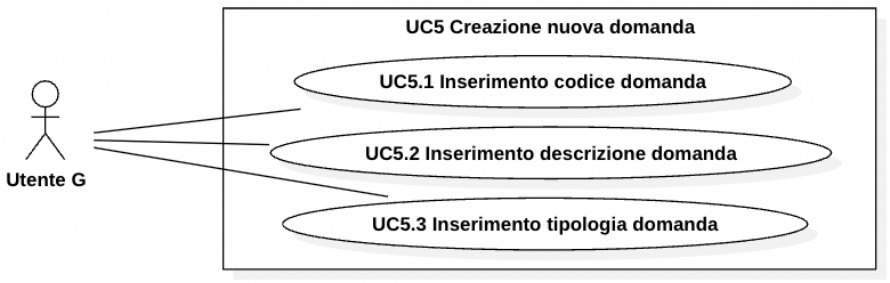


Figura 3.12: UC5 - Creazione nuova domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova domanda;
- **Postcondizioni**: La nuova domanda viene salvata nel sistema;

- **Scenario principale**:

  1. L'utente compila i campi necessari alla creazione di una nuova domanda, ovvero codice (UC5.1), descrizione (UC5.2), tipologia (UC5.3);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.

- **Estensioni**: UC7 - Errore salvataggio.

UC5.1 - Inserimento codice domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova domanda; inserisce il nuovo codice domanda;
- **Postcondizioni**: Il nuovo codice domanda è visualizzato nel campo;
- **Scenario principale**:

  1. L'utente inserisce il nuovo codice nel campo.

UC5.2 - Inserimento descrizione domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova domanda; inserisce la nuova descrizione;
- **Postcondizioni**: La nuova descrizione è visualizzata nel campo;
- **Scenario principale**:

  1. L'utente inserisce la nuova descrizione nel campo.

UC5.3 - Inserimento tipologia domanda

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova domanda; inserisce la nuova tipologia;
- **Postcondizioni**: La nuova tipologia è visualizzata nel campo;
- **Scenario principale**:
    1. L'utente inserisce la nuova tipologia nel campo.

UC6 - Errore lettura

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere una domanda;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

UC7 - Errore salvataggio

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole creare una nuova domanda o modificarne una esistente; l'utente inserisce dati inadeguati;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

UC8 - Errore eliminazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole eliminare la domanda letta;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
  1. L'utente visualizza un messaggio di errore.

Nella figura 3.13 è riportato il diagramma del modulo *offc_config* del gestionale con tutti i casi d'uso.

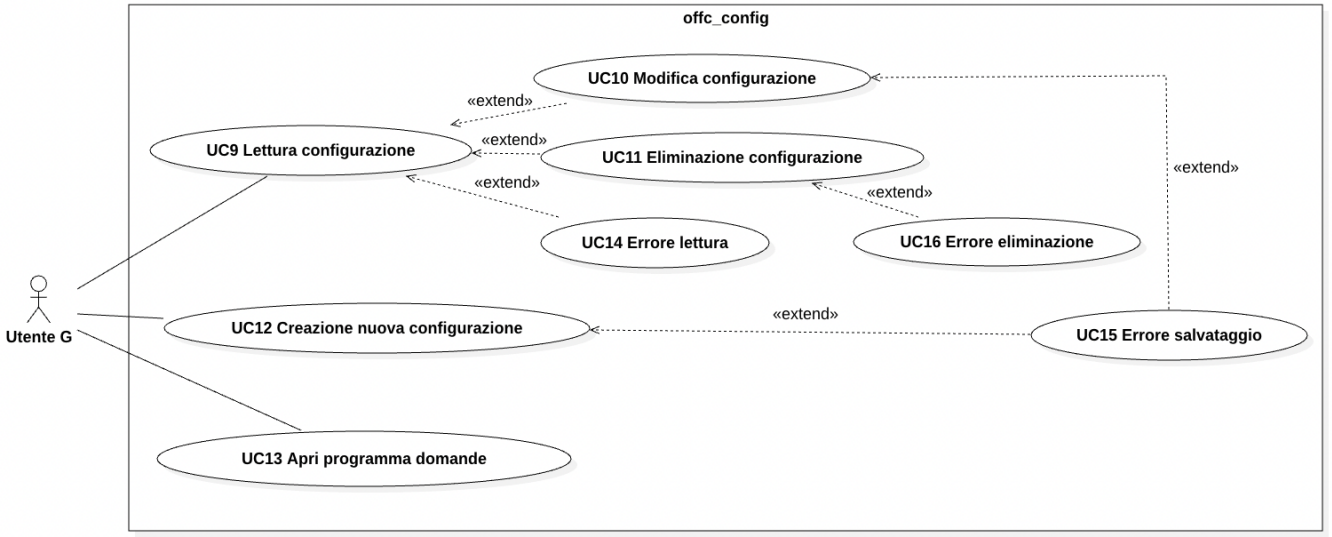


Figura 3.13: Modulo offc_config

UC9 - Lettura configurazione

In figura 3.14 è rappresentato il caso d'uso tramite UML.

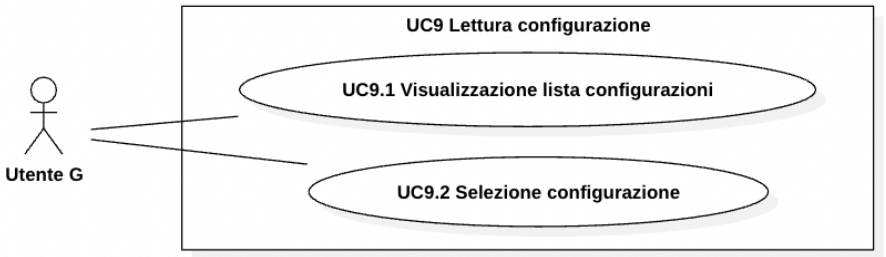


Figura 3.14: UC9 - Lettura configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha avviato il modulo *offc_config* e vuole leggere una configurazione;
- **Postcondizioni**: L'utente ha letto la configurazione desiderata e visualizza la configurazione;
- **Scenario principale**:
  1. L'utente clicca il bottone per la lettura di una configurazione;
- **Estensioni**: UC14 - Errore lettura.

UC9.1 - Visualizzazione lista configurazioni

In figura 3.15 è rappresentato il caso d'uso tramite UML.

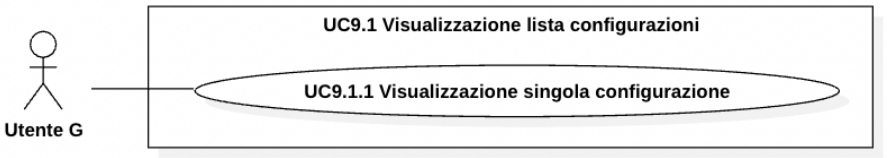


Figura 3.15: UC9.1 - Visualizzazione lista configurazioni

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente sta effettuando la lettura di una configurazione;
- **Postcondizioni**: L'utente visualizza la lista delle configurazioni;
- **Scenario principale**:
  1. L'utente visualizza la lista delle configurazioni.

UC9.1.1 - Visualizzazione singola configurazione

In figura 3.16 è rappresentato il caso d'uso tramite UML.

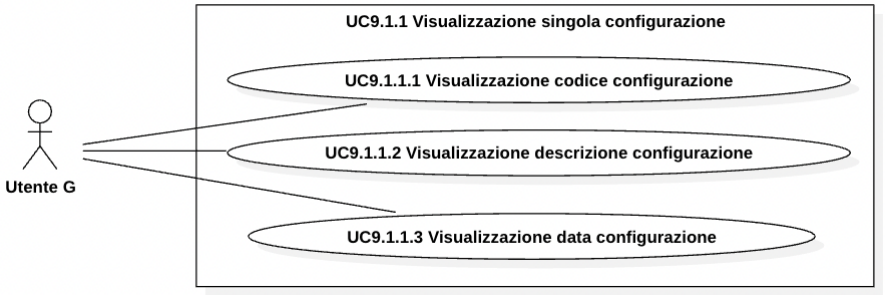


Figura 3.16: UC9.1.1 - Visualizzazione singola configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza correttamente la lista delle configurazioni;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
  1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.

UC9.1.1.1 - Visualizzazione codice configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola configurazione dalla lista;
- **Postcondizioni**: L'utente visualizza il codice della configurazione che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza il codice della configurazione che sta osservando.

UC9.1.1.2 - Visualizzazione descrizione configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola configurazione dalla lista;
- **Postcondizioni**: L'utente visualizza la descrizione della configurazione che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la descrizione della configurazione che sta osservando.

UC9.1.1.3 - Visualizzazione data configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente visualizza una singola configurazione dalla sua lista;
- **Postcondizioni**: L'utente visualizza la data della configurazione che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la data della configurazione che sta osservando.

UC9.2 - Selezione configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere una configurazione; l'utente clicca su una singola configurazione per effettuare la lettura;
- **Postcondizioni**: L'utente visualizza i dati della configurazione selezionata;
- **Scenario principale**:
    1. L'utente clicca una singola configurazione;
    2. L'utente visualizza i dati della configurazione selezionata.

UC10 - Modifica configurazione

In figura 3.17 è rappresentato il caso d'uso tramite UML.

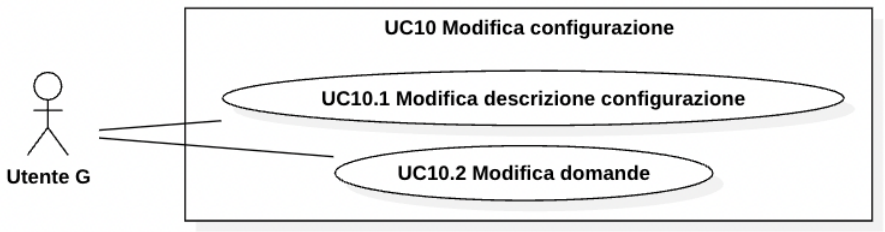


Figura 3.17: UC10 - Modifica configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la configurazione letta e decide di modificarla;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare i campi modificabili, ovvero descrizione (UC10.1) e domande (UC10.2);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC7 - Errore salvataggio.

UC10.1 - Modifica descrizione configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la configurazione letta e decide di modificare la descrizione;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare il campo descrizione;
  2. L'utente inserisce i nuovi dati.

UC10.2 - Modifica domande

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la configurazione letta e decide di modificare le domande;
- **Postcondizioni**: I nuovi dati vengono salvati nel sistema;
- **Scenario principale**:
  1. L'utente decide di modificare le domande;
  2. L'utente inserisce i nuovi dati.

UC11 - Eliminazione configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente ha effettuato correttamente una lettura, sta visualizzando la configurazione letta e decide di eliminarla;
- **Postcondizioni**: I dati vengono eliminati dal sistema;
- **Scenario principale**:
  1. L'utente clicca il bottone per eliminare la configurazione.
- **Estensioni**: UC16 - Errore eliminazione.

UC412 - Creazione nuova configurazione

In figura 3.18 è rappresentato il caso d'uso tramite UML.

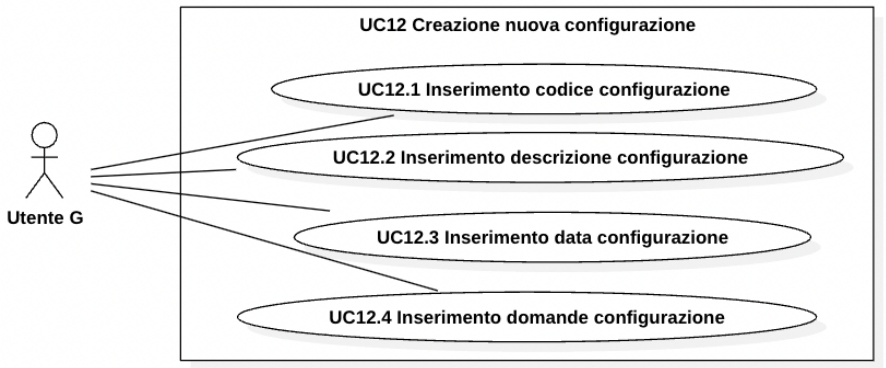


Figura 3.18: UC12 - Creazione nuova configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova configurazione;
- **Postcondizioni**: La nuova configurazione viene salvata nel sistema;
- **Scenario principale**:
  1. L'utente compila i campi necessari alla creazione di una nuova configurazione, ovvero codice (UC12.1), descrizione (UC12.2), data (UC12.3), domande (UC12.4);
  2. L'utente inserisce i nuovi dati;
  3. L'utente convalida la scelta confermando il salvataggio dei nuovi dati nel sistema;
  4. L'utente nega la scelta tramite il processo di annullamento dell'operazione.
- **Estensioni**: UC5 - Errore salvataggio.

UC12.1 - Inserimento codice configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova configurazione; inserisce il nuovo codice configurazione;
- **Postcondizioni**: Il nuovo codice configurazione è visualizzato nel campo;
- **Scenario principale**:
  1. L'utente inserisce il nuovo codice nel campo.

UC12.2 - Inserimento descrizione configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova configurazione; inserisce la nuova descrizione;
- **Postcondizioni**: La nuova descrizione è visualizzata nel campo;
- **Scenario principale**:
  1. L'utente inserisce la nuova descrizione nel campo.

UC12.3 - Inserimento data configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova configurazione; inserisce la nuova data;
- **Postcondizioni**: La nuova data è visualizzata nel campo;
- **Scenario principale**:

    1. L'utente inserisce la nuova data nel campo.

UC12.4 - Inserimento domande configurazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di creare una nuova configurazione; inserisce le nuove domande;
- **Postcondizioni**: Le nuove domande sono visualizzate nel campo;
- **Scenario principale**:

    1. L'utente inserisce le nuove domande nel campo.

UC13 - Apri programma domande

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente decide di aprire il programma per la gestione delle domande e dei relativi valori (modulo *offc_domande*);
- **Postcondizioni**: Il programma per la gestione delle domande e dei relativi valori viene aperto;
- **Scenario principale**:

    1. L'utente clicca il bottone per l'apertura del programma per la gestione delle domande e dei relativi valori.

UC14 - Errore lettura

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole leggere una configurazione;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

UC15 - Errore salvataggio

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole creare una nuova configurazione o modificarne una esistente; l'utente inserisce dati inadeguati;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

UC16 - Errore eliminazione

- **Attori principali**: Utente G;
- **Precondizioni**: L'utente vuole eliminare la configurazione letta;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
    1. L'utente visualizza un messaggio di errore.

3.1.3 Web application

Nella seguente sezione vengono illustrati e descritti i casi d'uso relativi alla *web application* per la gestione delle offerte commerciali.
Nella figura 3.19 è riportato il diagramma della *web application* per la gestione delle offerte commerciali con tutti i casi d'uso.

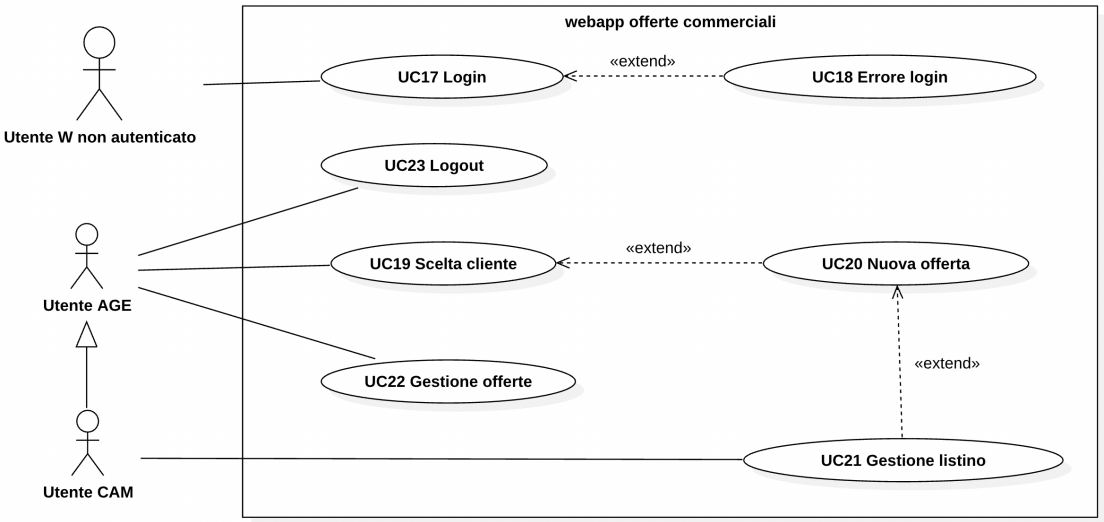


Figura 3.19: Web application

UC17 - Login

In figura 3.20 è rappresentato il caso d'uso tramite UML.

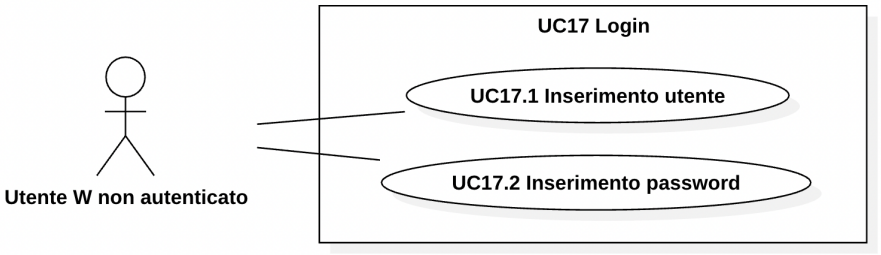


Figura 3.20: UC17 - Login

- **Attori principali**: Utente W non autenticato;
- **Precondizioni**: L'utente possiede un account valido, non ha effettuato l'accesso ed intende farlo;
- **Postcondizioni**: L'utente ha effettuato l'accesso ed è stato autenticato dal sistema;

- **Scenario principale**:
  1. L'utente compila i campi necessari per potersi autenticare nel sistema, ovvero utente (UC17.1) e password (UC17.2).
- **Estensioni**: UC18 - Errore login.

UC17.1 - Inserimento utente

- **Attori principali**: Utente W non autenticato;
- **Precondizioni**: L'utente sta facendo il login;
- **Postcondizioni**: L'utente inserito è visualizzato nel campo;
- **Scenario principale**:
  1. L'utente inserisce il proprio utente nel campo.

UC17.2 - Inserimento password

- **Attori principali**: Utente W non autenticato;
- **Precondizioni**: L'utente sta facendo il login;
- **Postcondizioni**: La password inserita è visualizzata nel campo;
- **Scenario principale**:
  1. L'utente inserisce la propria password nel campo.

UC18 - Errore login

- **Attori principali**: Utente W non autenticato;
- **Precondizioni**: L'utente vuole autenticarsi nel sistema; l'utente ha inserito una combinazione di utente e password errata, oppure ha inserito utente e/o password in un formato sbagliato, o ancora ha lasciato vuoto qualche campo durante il processo di login;
- **Postcondizioni**: L'utente visualizza un messaggio di errore;
- **Scenario principale**:
  1. L'utente visualizza un messaggio di errore.

UC19 - Scelta cliente

In figura 3.21 è rappresentato il caso d'uso tramite UML.

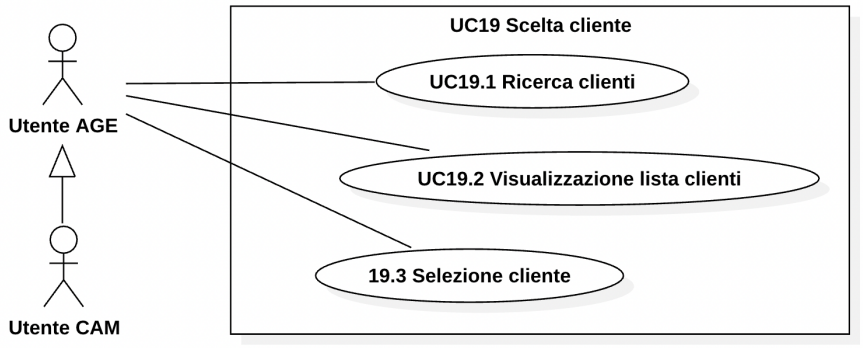


Figura 3.21: UC19 - Scelta cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e ha selezionato la funzionalità "scelta cliente" dal menù;
- **Postcondizioni**: L'utente ha selezionato il cliente desiderato;
- **Scenario principale**:
  1. L'utente seleziona la funzionalità "scelta cliente" dal menù della *web application*.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.1 - Ricerca clienti

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina scelta cliente; l'utente decide di ricercare un cliente;
- **Postcondizioni**: L'utente visualizza i risultati della ricerca (UC19.2);
- **Scenario principale**:
  1. L'utente compila il capo di ricerca con una stringa contenente quello che vuole ricercare;
  2. L'utente visualizza i risultati della ricerca (UC19.2).
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2 - Visualizzazione lista clienti

In figura 3.22 è rappresentato il caso d'uso tramite UML.

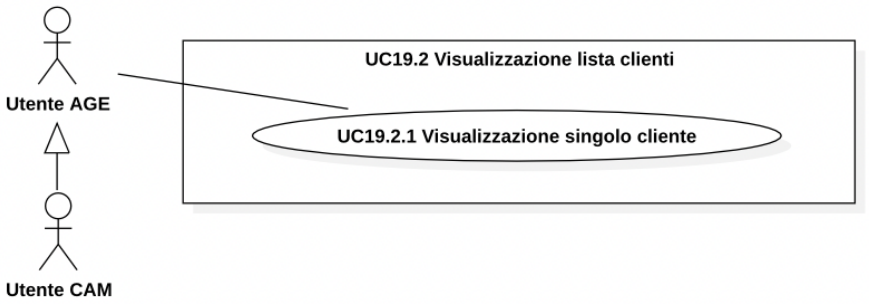


Figura 3.22: UC19.2 - Visualizzazione lista clienti

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina scelta cliente;
- **Postcondizioni**: L'utente visualizza la lista dei clienti e, nel caso abbia effettuato una ricerca (UC19.1), ne visualizza i risultati;
- **Scenario principale**:
  1. L'utente visualizza la lista dei clienti;
  2. L'utente visualizza i risultati della ricerca effettuata.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1 - Visualizzazione singolo cliente

In figura 3.23 è rappresentato il caso d'uso tramite UML.

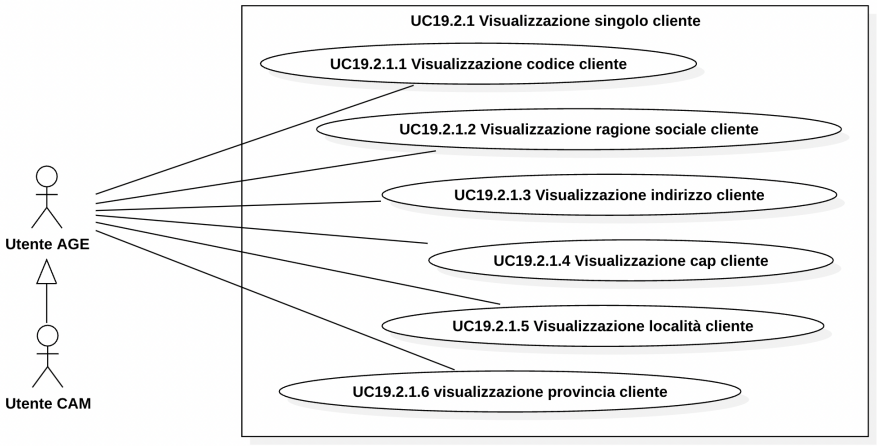


Figura 3.23: UC19.2.1 - Visualizzazione singolo cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza correttamente la lista dei clienti;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
    1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.1 - Visualizzazione codice cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza il codice del cliente che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il codice del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.2 - Visualizzazione ragione sociale cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza la ragione sociale del cliente che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la ragione sociale del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.3 - Visualizzazione indirizzo cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza l'indirizzo del cliente che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza l'indirizzo del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.4 - Visualizzazione cap cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza il cap del cliente che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il cap del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.5 - Visualizzazione località cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza la località del cliente che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza la località del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.2.1.6 - Visualizzazione provincia cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo cliente della sua ricerca;
- **Postcondizioni**: L'utente visualizza la provincia del cliente che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza la provincia del cliente che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC19.3 - Selezione cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente sta visualizzanto la pagina "scelta cliente" e intende appunto selezionare un cliente;
- **Postcondizioni**: L'utente visualizza i dati del cliente selezionato;
- **Scenario principale**:
  1. L'utente clicca un singolo cliente;
  2. L'utente visualizza i dati del cliente.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC20 - Nuova offerta

In figura 3.24 è rappresentato il caso d'uso tramite UML.

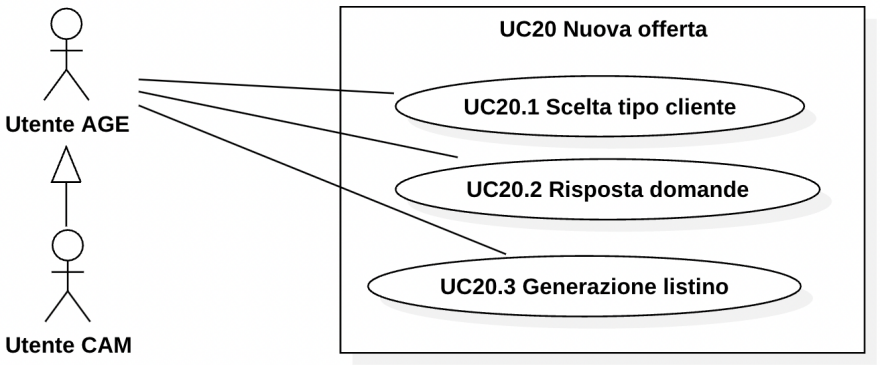


Figura 3.24: UC20 - Nuova offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema, l'utente ha già selezionato un cliente e ha anche selezionato la funzionalità "nuova offerta" dal menù;
- **Postcondizioni**: L'utente ha generato una nuova offerta;
- **Scenario principale**:
  1. L'utente seleziona la funzionalità "nuova offerta" dal menù della *web application*.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC20.1 - Scelta tipo cliente

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente sta visualizzando la pagina "nuova offerta"; l'utente sta compilando il form per creare una nuova offerta;
- **Postcondizioni**: Il campo tipo cliente mostra l'opzione scelta dall'utente;
- **Scenario principale**:
  1. L'utente visualizza l'opzione di *default* per l'input;
  2. L'utente seleziona l'opzione di *default* per modificarla;
  3. L'utente visualizza le opzioni possibili per l'input;
  4. L'utente sceglie un'opzione.

- **Generalizzazioni**: Utente CAM → Utente AGE.

UC20.2 - Risposta domande

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente sta visualizzando la pagina "nuova offerta"; l'utente sta compilando il form per creare una nuova offerta;
- **Postcondizioni**: I campi domande mostrano le risposte inserite dall'utente;
- **Scenario principale**:
  1. L'utente compila i vari input con le risposte alle domande;
  2. L'utente visualizza le risposte nei vari input;
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC20.3 - Generazione listino

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente sta visualizzanto la pagina "nuova offerta" e vuole generare il listino per la nuova offerta;
- **Postcondizioni**: L'offerta viene salvata e viene generato il relativo listino;
- **Scenario principale**:
  1. L'utente clicca il bottone per generare il listino.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC21 - Gestione listino

In figura 3.25 è rappresentato il caso d'uso tramite UML.

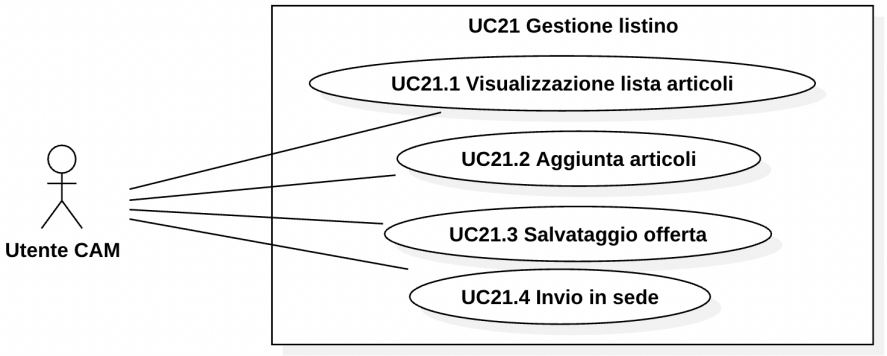


Figura 3.25: UC21 - Gestione listino

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema; l'utente ha generato il listino dalla pagina nuovo listino;
- **Postcondizioni**: L'utente gestisce il listino;
- **Scenario principale**:
  1. L'utente ha generato un listino ed ha la possibilita di gestirlo.

UC21.1 - Visualizzazione lista articoli

In figura 3.26 è rappresentato il caso d'uso tramite UML.

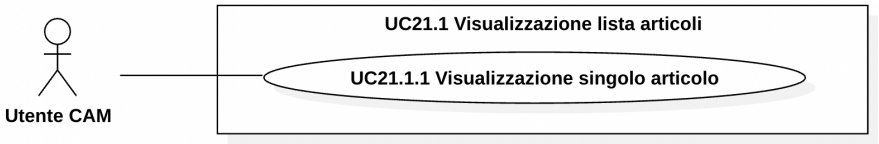


Figura 3.26: UC21.1 - Visualizzazione lista articoli

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina di gestione listino;
- **Postcondizioni**: L'utente visualizza la lista degli articoli presenti nel listino generato;

- **Scenario principale**:
  1. L'utente visualizza la lista degli articoli.

UC21.1.1 - Visualizzazione singolo articolo

In figura 3.27 è rappresentato il caso d'uso tramite UML.

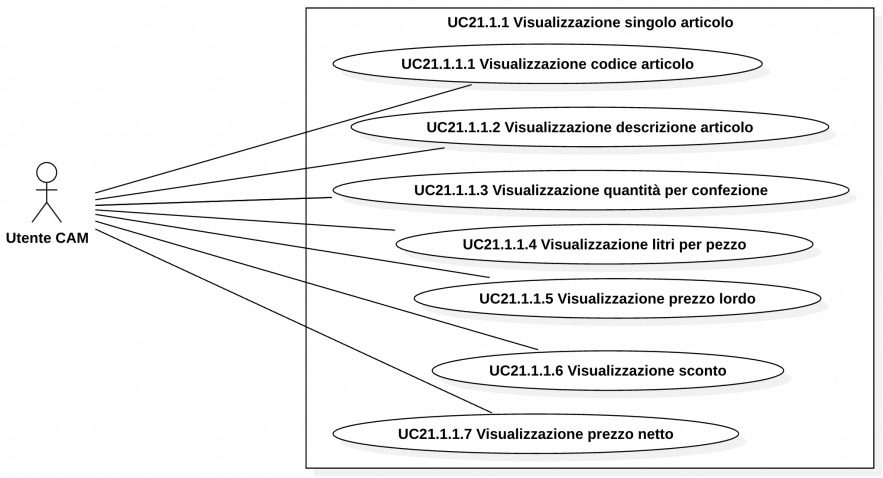


Figura 3.27: UC21.1.1 - Visualizzazione singolo articolo

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza correttamente la lista degli articoli;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
  1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.

UC21.1.1.1 - Visualizzazione codice articolo

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza il codice dell'articolo che sta osservando;

- **Scenario principale**:
    1. L'utente visualizza il codice dell'articolo che sta osservando.

UC21.1.1.2 - Visualizzazione descrizione articolo

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza la descrizione dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la descrizione dell'articolo che sta osservando.

UC21.1.1.3 - Visualizzazione quantità per confezione

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza la quantità per confezione dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la quantità per confezione dell'articolo che sta osservando.

UC21.1.1.4 - Visualizzazione litri per pezzo

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza i litri per pezzo dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza i litri per pezzo dell'articolo che sta osservando.

UC21.1.1.5 - Visualizzazione prezzo lordo

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza il prezzo lordo dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il prezzo lordo dell'articolo che sta osservando.

UC21.1.1.6 - Visualizzazione sconto

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza lo sconto dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza lo sconto dell'articolo che sta osservando.

UC21.1.1.7 - Visualizzazione prezzo netto

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente visualizza un singolo articolo della sua lista;
- **Postcondizioni**: L'utente visualizza il prezzo netto dell'articolo che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza il prezzo netto dell'articolo che sta osservando.

UC21.2 - Aggiunta articoli

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina di gestione listino; l'utente decide di inserire un nuovo articolo al listino;
- **Postcondizioni**: L'utente ha aggiunto un nuovo articolo al listino;
- **Scenario principale**:
  1. L'utente aggiunge un nuovo articolo al listino;
  2. Il nuovo articolo è stato aggiunto al listino.

UC21.3 - Salvataggio offerta

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina di gestione listino; l'utente decide di salvare l'offerta;
- **Postcondizioni**: L'utente ha salvato l'offerta;
- **Scenario principale**:
  1. L'utente clicca il bottone per salvare l'offerta.

UC21.4 - Invio in sede

- **Attori principali**: Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina di gestione listino; l'utente decide di inviare in sede l'offerta;
- **Postcondizioni**: L'utente ha inviato in sede l'offerta, l'offerta non è più modificabile;
- **Scenario principale**:
  1. L'utente clicca il bottone per inviare in sede l'offerta;
  2. L'offerta cambia di stato ed ora non è più modificabile.

UC22 - Gestione offerte

In figura 3.28 è rappresentato il caso d'uso tramite UML.

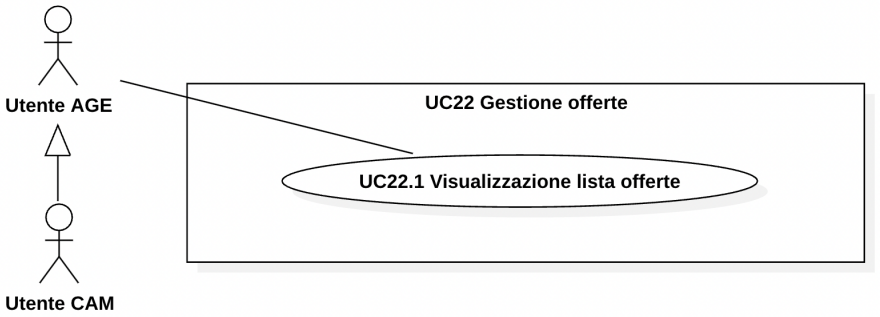


Figura 3.28: UC22 - Gestione offerte

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema; l'utente ha selezionato la funzionalità "gestione offerte" dal menù;
- **Postcondizioni**: L'utente visualizza correttamente la pagina;
- **Scenario principale**:
  1. L'utente seleziona la funzionalità "gestione offerte" dal menù della *web application*.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1 - Visualizzazione lista articoli

In figura 3.29 è rappresentato il caso d'uso tramite UML.

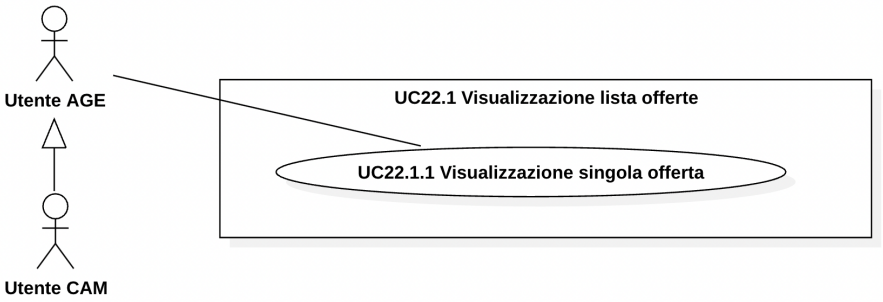


Figura 3.29: UC22.1 - Visualizzazione lista offerte

- **Attori principali**: Utente AGE, Utente CAM;

- **Precondizioni**: L'utente è autenticato nel sistema e sta visualizzando la pagina di gestione offerte;
- **Postcondizioni**: L'utente visualizza la lista delle offerte;
- **Scenario principale**:
  1. L'utente visualizza la lista delle offerte.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1 - Visualizzazione singola offerta

In figura 3.30 è rappresentato il caso d'uso tramite UML.

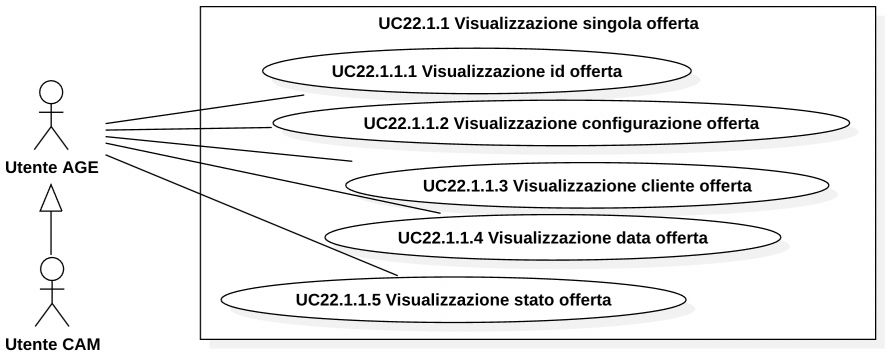


Figura 3.30: UC22.1.1 - Visualizzazione singola offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza correttamente la lista delle offerte;
- **Postcondizioni**: L'utente visualizza un singolo elemento della lista;
- **Scenario principale**:
  1. L'utente visualizza un singolo elemento della lista tra i vari mostrati.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1.1 - Visualizzazione id offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza una singola offerta della sua lista;
- **Postcondizioni**: L'utente visualizza l'id dell'offerta che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza l'id dell'offerta che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1.2 - Visualizzazione configurazione offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza una singola offerta della sua lista;
- **Postcondizioni**: L'utente visualizza la configurazione dell'offerta che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza la configurazione dell'offerta che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1.3 - Visualizzazione cliente offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza una singola offerta della sua lista;
- **Postcondizioni**: L'utente visualizza il cliente dell'offerta che sta osservando;
- **Scenario principale**:
  1. L'utente visualizza il cliente dell'offerta che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1.4 - Visualizzazione id offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza una singola offerta della sua lista;
- **Postcondizioni**: L'utente visualizza la data dell'offerta che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza la data dell'offerta che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC22.1.1.5 - Visualizzazione stato offerta

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente visualizza una singola offerta della sua lista;
- **Postcondizioni**: L'utente visualizza lo stato dell'offerta che sta osservando;
- **Scenario principale**:
    1. L'utente visualizza lo stato dell'offerta che sta osservando.
- **Generalizzazioni**: Utente CAM → Utente AGE.

UC23 - Logout

- **Attori principali**: Utente AGE, Utente CAM;
- **Precondizioni**: L'utente è autenticato nel sistema e desidera uscire;
- **Postcondizioni**: L'utente è reindirizzato alla pagina di *login*;
- **Scenario principale**:
    1. L'utente decide di uscire dal sistema tramite l'opzione di *logout*.
- **Generalizzazioni**: Utente CAM → Utente AGE.

3.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli *use case* effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto ai casi d'uso.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Ogni requisito verrà identificato con la seguente codifica:

R<Tipologia><Importanza><codice>

dove:

- **Tipologia**: può assumere i seguenti valori:
  - **F**: Funzionale;
  - **Q**: di Qualità;
  - **V**: di Vincolo.
- **Importanza**: può assumere i seguenti valori:
  - **O**: Obbligatorio;
  - **D**: Desiderabile;
  - **F**: Facoltativo;
- **Codice**: identifica in modo univoco ogni requisito.

3.2.1 Requisiti funzionali

Nella tabella 3.1 sono elencati i requisiti funzionali delineati in fase di analisi.

Requisito	**Descrizione**	**Use case**
RFO1	L'utente deve poter leggere una domanda esistente	UC1
RFO2	L'utente deve poter visualizzare la lista delle domande e deve poter visualizzare il Codice, la Descrizione e la Tipologia di una domanda	UC1.1, UC1.1.1, UC1.1.1.1, UC1.1.1.2, UC1.1.1.3

Requisito	Descrizione	Use case
RFO3	L'utente che sta leggendo deve poter selezionare una domanda per completare la lettura	UC1.2
RFO4	L'utente che ha letto una domanda deve poter modificare i campi	UC2
RFO5	L'utente che ha scelto di modificare i campi della domanda deve poter modificare la Descrizione e la Tipologia	UC2.1, UC2.2
RFO6	L'utente che ha letto una domanda deve poterla eliminare	UC3
RFO7	L'utente che ha letto una domanda deve poter gestire i relativi valori	UC4
RFO8	L'utente deve poter leggere un valore esistente relativo alla domanda	UC4.1
RFO9	L'utente deve poter visualizzare la lista dei valori e deve poter visualizzare il Codice e la Descrizione di un valore	UC4.1.1, UC4.1.1.1, UC4.1.1.1.1, UC4.1.1.1.2
RFO10	L'utente che sta leggendo deve poter selezionare un valore per completare la lettura	UC4.1.2
RFO11	L'utente che ha letto un valore deve poter modificare i campi	UC4.2
RFO12	L'utente che ha scelto di modificare i campi del valore deve poter modificare la Descrizione e gli Attributi	UC4.2.1, UC4.2.2
RFO13	L'utente che ha letto un valore deve poterlo eliminare	UC4.3
RFO14	L'utente deve poter creare un nuovo valore relativo alla domanda	UC4.4
RFO15	L'utente che sta creando un nuovo valore deve poter inserire i campi Codice, Descrizione e Attributi	UC4.4.1, UC4.4.2, UC4.4.3
RFO16	L'utente deve poter visualizzare un eventuale messaggio di errore relativo alla lettura di un valore	UC4.5

Requisito	Descrizione	Use case
RFO17	L'utente deve poter visualizzare un eventuale messaggio di errore relativo al salvataggio di un valore	UC4.6
RFO18	L'utente deve poter visualizzare un eventuale messaggio di errore relativo all'eliminazione di un valore	UC4.7
RFO19	L'utente deve poter creare una nuova domanda	UC5
RFO20	L'utente che sta creando una nuova domanda deve poter inserire i campi Codice, Descrizione e Tipologia	UC5.1, UC5.2, UC5.3
RFO21	L'utente deve poter visualizzare un eventuale messaggio di errore relativo alla lettura di una domanda	UC6
RFO22	L'utente deve poter visualizzare un eventuale messaggio di errore relativo al salvataggio di una domanda	UC7
RFO23	L'utente deve poter visualizzare un eventuale messaggio di errore relativo all'eliminazione di una domanda	UC8
RFO24	L'utente deve poter leggere una configurazione esistente	UC9
RFO25	L'utente deve poter visualizzare la lista delle configurazioni e deve poter visualizzare il Codice, la Descrizione e la Data di una configurazione	UC9.1, UC9.1.1, UC9.1.1.1, UC9.1.1.2, UC9.1.1.3
RFO26	L'utente che sta leggendo deve poter selezionare una configurazione per completare la lettura	UC9.2
RFO27	L'utente che ha letto una configurazione deve poter modificare i campi	UC10
RFO28	L'utente che ha scelto di modificare i campi della configurazione deve poter modificare la Descrizione e le Domande	UC10.1, UC10.2
RFO29	L'utente che ha letto una configurazione deve poterla eliminare	UC11

Requisito	Descrizione	Use case
RFO30	L'utente deve poter creare una nuova configurazione	UC12
RFO31	L'utente che sta creando una nuova configurazione deve poter inserire i campi Codice, Descrizione, Data e Domande	UC12.1, UC12.2, UC12.3, UC12.4
RFO32	L'utente deve poter aprire il programma per la gestione delle domande e relativi valori	UC13
RFO33	L'utente deve poter visualizzare un eventuale messaggio di errore relativo alla lettura di una configurazione	UC14
RFO34	L'utente deve poter visualizzare un eventuale messaggio di errore relativo al salvataggio di una configurazione	UC15
RFO35	L'utente deve poter visualizzare un eventuale messaggio di errore relativo all'eliminazione di una configurazione	UC16
RFO36	L'utente deve autenticarsi all'interno del sistema per poter accedere all'applicazione *web*	UC17
RFO37	L'utente deve inserire il proprio utente e password per procedere all'autenticazione	UC17.1, UC17.2
RFO38	L'utente deve poter visualizzare un eventuale messaggio di errore ralativo al login sull'applicazione *web*	UC18
RFO39	L'utente deve poter visualizzare la pagina di scelta del cliente	UC19
RFO40	L'utente deve poter ricercare un cliente e visualizzare i risultati	UC19.1, UC19.2
RFO41	L'utente che ha effettuato una ricerca o non deve poter visualizzare il Codice, la Ragione Sociale, l'Indirizzo, il Cap, la Località e la Provincia di un cliente	UC19.2.1, UC19.2.1.1, UC19.2.1.2, UC19.2.1.3, UC19.2.1.4, UC19.2.1.5, UC19.2.1.6

Requisito	Descrizione	Use case
RFO42	L'utente deve poter selezionare un cliente	UC19.3
RFD43	L'utente deve essere avvisato se il cliente selezionato ha offerte in sospeso	-
RFO44	L'utente che ha selezionato un cliente deve poter visualizzare la pagina di nuova offerta	UC20
RFO45	L'utente deve poter selezionare il tipo cliente una volta all'interno della pagina per la nuova offerta	UC20.1
RFO46	L'utente deve poter rispondere alle domande una volta all'interno della pagina per la nuova offerta	UC20.2
RFO47	L'utente deve poter generare il listino una volta all'interno della pagina per la nuova offerta	UC20.3
RFO48	L'utente amministratore dopo aver generato il listino deve poter visualizzare la pagina di gestione listino	UC21
RFO49	L'utente amministratore all'interno della pagina di gestione listino deve poter visualizzare gli articoli	UC21.1
RFO50	L'utente amministratore all'interno della pagina di gestione listino deve poter visualizzare tutti i campi di un articolo	UC21.1.1, UC21.1.1.1, UC21.1.1.2, UC21.1.1.3, UC21.1.1.4, UC21.1.1.5, UC21.1.1.6, UC21.1.1.7
RFO51	L'utente amministratore all'interno della pagina di gestione listino deve poter aggiungere un nuovo articolo al listino	UC21.2
RFO52	L'utente amministratore all'interno della pagina di gestione listino deve poter salvare l'offerta	UC21.3
RFO53	L'utente amministratore all'interno della pagina di gestione listino deve poter inviare l'offerta in sede	UC21.4

Requisito	Descrizione	Use case
RFO54	L'utente deve poter vedere la pagina di gestione offerte	UC22
RFO55	L'utente all'interno della pagina di gestione offerte deve poter visualizzare le offerte	UC22.1
RFO56	L'utente all'interno della pagina di gestione offerte deve poter visualizzare i campi di un'offerta	UC22.1.1, UC22.1.1.1, UC22.1.1.2, UC22.1.1.3, UC22.1.1.4, UC22.1.1.5
RFO57	L'utente autenticato deve poter effetturare il logout	UC23

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

3.2.2 Requisiti di qualità

Nella tabella 3.2 sono elencati i requisiti di qualità delineati in fase di analisi.

Requisito	Descrizione	Use case
RQO1	Deve essere redatto un documento che descrive l'architettura dell'applicazione	-
RQO2	Il codice deve essere documentato tramite commenti	-
RQO3	L'applicazione *web* deve essere testata con dei *tablet*	-
RQO4	L'applicazione *web* deve essere testata con i *browser Google Chrome*, *Mozilla Firefox* e *Microsoft Edge*	-

Tabella 3.2: Tabella del tracciamento dei requisiti di qualità

3.2.3 Requisiti di vincolo

Nella tabella 3.3 sono elencati i requisiti di vincolo delineati in fase di analisi.

Requisito	**Descrizione**	**Use case**
RVO1	I moduli *desktop* devono essere scritti con il linguaggio *C#*	-
RVO2	I moduli *desktop* devono essere sviluppati con il *framework Windows Forms* e con i componenti grafici *Devexpress*	-
RVO3	I *web services* devono essere sviluppati con il *framework ASP .NET Core*	-
RVO4	Il *web service* per la privacy deve essere sviluppato con l'utilizzo delle *Office API* di *Devexpress*	-

Tabella 3.3: Tabella del tracciamento dei requisiti di vincolo

4 Progettazione

4.1 Architettura generale

Nella figura 4.1 che segue è riportato lo schema ad alto livello che descrive le componenti che comprendono il sistema per la gestione delle offerte commerciali e le loro interazioni.

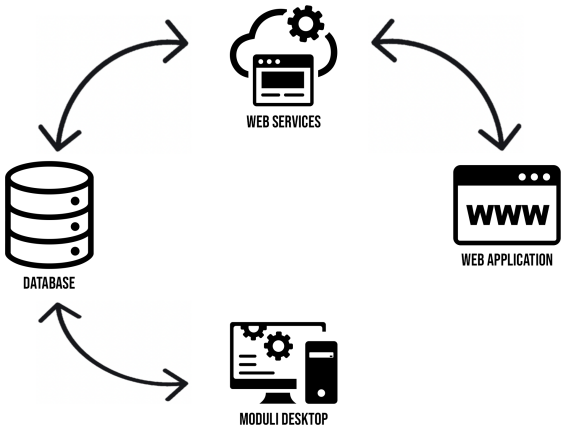


Figura 4.1: Architettura generale del sistema di gestione delle offerte commerciali

Come si può osservare dalla figura 4.1, il sistema è formato da 4 componenti principali:

- I due moduli *desktop offc_domande* e *offc_config* integrati con il gestionale per la completa gestione delle configurazioni delle domande e dei relativi valori utilizzati dalla *web application*. I moduli comunicano direttamente con il *db*, performando operazioni CRUD$_{[g]}$;

- *Database Informix*;
- *Web services* per la gestione del flusso dei dati da *db* e la *web application*;
- *Web application* per la gestione delle offerte commerciali, la quale interagisce con il *web services* per effettuare operazioni CRUD sul *db*.

Per quanto riguarda l'architettura relativa al sistema per la gestione dei documenti definiti "legali", nella figura 4.2 che segue è riportato lo schema ad alto livello che descrive le parti che lo compongono.

Figura 4.2: Architettura generale del sistema di gestione dei documenti "legali"

Come si può osservare dalla figura 4.2, il sistema per la gestione dei documenti "legali" è formato da un unico componente:

- *Web services* che dispone degli *endpoint* per la generazione del documento PDF richiesto.

4.2 Moduli desktop

Per la progettazione dei due moduli *offc_domande* e *desktop offc_config* da integrare al gestionale, è stato deciso di utilizzare il *framework Windows Forms* insieme ai componenti grafici di *Devexpress* e Entity Framework[g] .
È molto complicato realizzare un diagramma preciso delle classi che illustra tutti gli aspetti di una applicazione *Windows Forms* e per questo motivo ogni modulo viene rappresentato con un diagramma semplificato che ne descrive le componenti principali.

4.2.1 Modulo offc_domande

Il modulo *offc_domande* permette di creare e gestire delle domande. Nella figura 4.3 è rappresentato il diagramma delle classi che ne rappresenta le principali componenti.

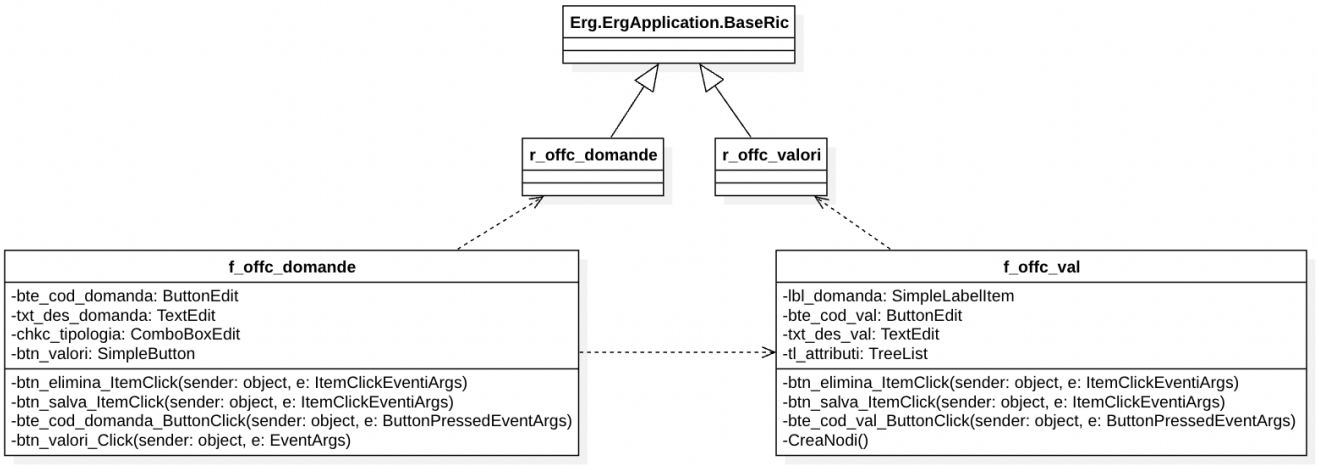


Figura 4.3: Diagramma delle classi modulo offc_domande

Come si può evincere dalla figura 4.3, il modulo per la gestione delle domande e dei relativi valori è composto dal *form f_offc_domande*, il quale contiene una serie di componenti *Devexpress*, tra cui:

- bte_cod_domanda: un *ButtonEdit* per l'inserimento e per la lettura / ricerca di un codice domanda;
- txt_des_domanda: un *TextEdit* per l'inserimento della descrizione di una domanda;
- chkc_tipologia: un *ComboBoxEdit* per la selezione della tipologia di una domanda;
- btn_valori: un *SimpleButton* per l'apertura del *form f_offc_val* per la gestione dei valori relativi alla domanda.

Tra le funzioni principali ci sono:

- btn_elimina_ItemClick: funzione richiamata al *click* del bottone di eliminazione, provvede dunque ad eliminare la domanda;
- btn_salva_ItemClick: funzione richiamata al *click* del bottone di salvataggio, provvede dunque a salvare la domanda;
- bte_cod_domanda_ButtonClick: funzione richiamata al *click* del bottone *bte_cod_domanda*, provvede dunque ad aprire il form *r_offc_domande* per la lettura di una domanda;

- btn_valori_Click: funzione richiamata al *click* del bottone *btn_valori*, provvede dunque ad aprire il *form f_offc_val* per la gestione dei valori relativi alla domanda.

Il modulo per la gestione delle domande e dei relativi valori è composto dal *form f_offc_val*, il quale contiene una serie di componenti *Devexpress*, tra cui:

- lbl_domanda: un *SimpleLabelItem* per indicare la domanda di riferimento ai valori;
- bte_cod_val: un *ButtonEdit* per l'inserimento e per la lettura / ricerca di un codice valore;
- txt_des_val: un *TextEdit* per l'inserimento della descrizione di un valore;
- tl_attributi: un *TreeList* per l'assegnazione degli attributi relativi alla risposta (valore).

Tra le funzioni principali ci sono:

- btn_elimina_ItemClick: funzione richiamata al *click* del bottone di eliminazione, provvede dunque ad eliminare il valore;
- btn_salva_ItemClick: funzione richiamata al *click* del bottone di salvataggio, provvede dunque a salvare il valore;
- bte_cod_val_ButtonClick: funzione richiamata al *click* del bottone *bte_cod_val*, provvede dunque ad aprire il form *r_offc_valori* per la lettura di un valore;
- CreaNodi(): funzione che va a creare i nodi di *tl_attributi*.

4.2.2 Modulo offc_config

Il modulo *offc_config* permette di creare e gestire delle configurazioni di domande per la gestione commerciale. Nella figura 4.4 è rappresentato il diagramma delle classi che ne rappresenta le principali componenti.

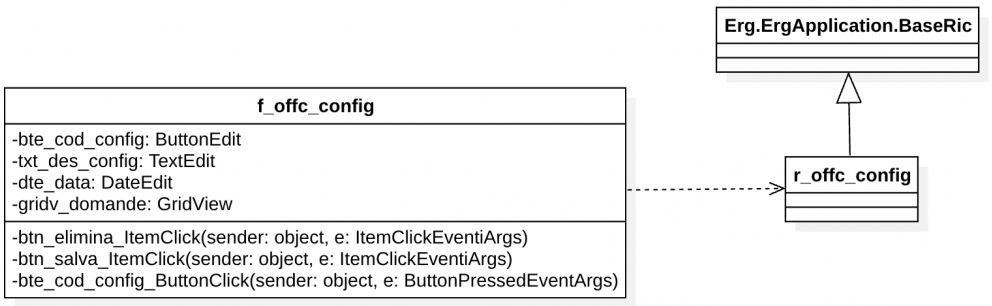


Figura 4.4: Diagramma delle classi modulo offc_config

Come si può evincere dalla figura 4.4, il modulo per la gestione delle configurazioni è composto dal *form f_offc_config*, il quale contiene una serie di componenti *Devexpress*, tra cui:

- bte_cod_config: un *ButtonEdit* per l'inserimento e per la lettura / ricerca di un codice configurazione;
- txt_des_config: un *TextEdit* per l'inserimento della descrizione di una configurazione;
- dte_data: una *DateEdit* per l'inserimento della data di una configurazione;
- grid_view: una *GridView* per l'inserimento delle domande legate alla configurazione.

Tra le funzioni principali ci sono:

- btn_elimina_ItemClick: funzione richiamata al *click* del bottone di eliminazione, provvede dunque ad eliminare la configurazione;
- btn_salva_ItemClick: funzione richiamata al *click* del bottone di salvataggio, provvede dunque a salvare la configurazione;
- bte_cod_config_ButtonClick: funzione richiamata al *click* del bottone *bte_cod_config*, provvede dunque ad aprire il form *r_offc_config* per la lettura di una configurazione.

4.3 Web services

Per la progettazione dei due *web services* per la gestione commerciale e per la gestione dei documenti "legali", è stato deciso di utilizzare il *framework ASP .NET Core* insieme al Entity Framework per la realizzazione di *REST API*.
REST API[8] è un'Application Programming Interface (API) che è conforme ai principi di progettazione di stile architetturale *REST*. Le *REST API* comunicano tramite delle richieste HTTP per eseguire delle operazioni standard su un *database*, come la creazione, la lettura, l'aggiornamento o l'eliminazione di un *record* (CRUD *operation*). Ad esempio, una *REST API* utilizza una richiesta GET per recuperare i campi di un *record*; una richiesta di tipo POST viene impiegata per la creazione di un nuovo *record*; una richiesta PUT aggiorna un *record* e una DELETE ne elimina un altro. Sempre tramite richiesta HTTP, le informazioni possono essere inviate o ricevute in diversi formati, come JavaScript Object Notation (JSON), HTML, XLT, Python, PHP o testo semplice. JSON è il formato

attualmente più popolare, in quanto è leggibile sia dagli esseri umani che dalle macchine ed è indipendente dal linguaggio di programmazione utilizzato. Altri aspetti da tenere in considerazione se si vuole rispettare i criteri di una *REST API* sono: un'architettura *client-server stateless*, che non prevede la memorizzazione delle informazioni del *client* da parte del *server*, e l'ottimizzazione delle interazioni con utilizzo della *cache*.

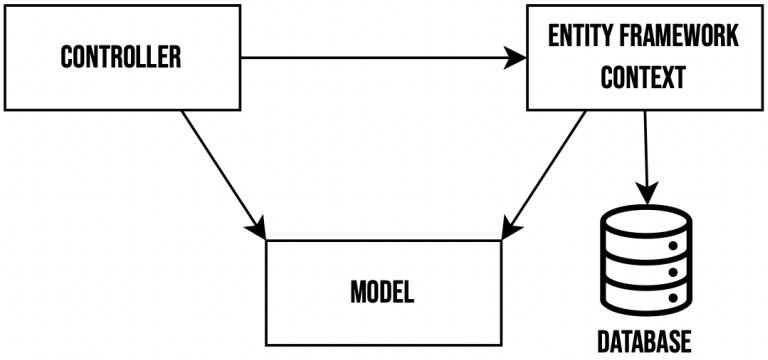


Figura 4.5: Schema generale web services

Come si può notare dalla figura 4.5, in entrambi i *web services* fanno parte le seguenti componenti:

- Controller: hanno il compito di gestire le richieste HTTP provenienti dall'esterno, le quali vengono mappate ai vari metodi dei *controller* grazie ad un sistema di *routing* in base all'*url* e al metodo della richiesta stessa. I *controller* implementano anche la *business logic* e interagiscono con il *database* fornendo una risposta alla richiesta;
- Model: rappresentano le entità necessarie per la gestione dei dati;
- Context: ha il compito di astrarre i dettagli di accesso al *database* e fornire i metodi per effettuare le operazioni CRUD sulle entità.

Ogni risposta da parte del *web services* restituisce il codice di stato HTTP corrispondente e una stringa JSON contenente la risposta vera e propria. I possibili codici di stato per le risposte sono:

- **200**: richiesta andata a buon fine;
- **400**: errore nella richiesta;
- **401**: utente non autenticato;
- **500**: errore interno al *server*.

4.3.1 Endpoint per gestione commerciale

Di seguito vengono elencati gli *endpoint* progettati per il *web services* per la gestione delle offerte commerciali:

TokenController

- **POST** /api/token
  Parametri *request body*:
  - Credential credential.

  Descrizione: Restituisce il *token* necessario per l'autenticazione alle altre richieste e info relative all'utente.

ConfigurazioniController

- **GET** /api/configurazioni
  Descrizione: Restituisce la lista delle configurazioni.
- **GET** /api/configurazioni/{cod_config}
  Parametri *query*:
  - string cod_config.

  Descrizione: Restituisce i dati e le rispettive domande e valori di una specifica configurazione.

ArticoliController

- **POST** /api/articoli
  Parametri *request body*:
  - List<Models.Articolo> art.

  Parametri *query*:
  - string tipoCliente.

  Descrizione: Restituisce la lista di articoli per l'aggiunta di articoli al listino.

ClientiController

- **GET** /api/clienti
  Parametri *query*:
  - string cod_agente;
  - int pageSize;
  - int pageNumber;
  - string filter.

Descrizione: Restituisce la lista di clienti di uno specifico agente con supporto per paginazione e filtro opzionale.

- **GET** /api/clienti/{cod_cli}
  Parametri *path*:
  - int cod_cli.

  Descrizione: Restituisce le info del cliente richiesto.

GenerazioneListiniController

- **POST** /api/generalistini
  Parametri *request body*:
  - Models.GROfferta ric.

  Parametri *query*:
  - string tipoCliente.

  Descrizione: Crea la nuova offerta e restituisce un dizionario contenente l'id dell'offerta e la lista di articoli che forma il listino.
- **PUT** /api/generalistini/{id}
  Parametri *request body*:
  - Models.GROfferta ric.

  Parametri *path*:
  - int id.

  Parametri *query*:
  - string tipoCliente.

  Descrizione: Modifica l'offerta e restituisce un dizionario contenente l'id dell'offerta e la lista di articoli che forma il listino.

OfferteController

- **GET** /api/offerte
  Parametri *query*:
  - string cod_config;
  - int cod_cli.

  Descrizione: Restituisce, in base alla valorizzazione dei parametri, una lista di offerte.
- **GET** /api/offerte/{id}
  Parametri *path*:
  - int id.

Descrizione: Restituisce le info, comprese di risposte, dell'offerta richiesta.

- **GET** /api/offerte/insospeso/{cod_cli}
  Parametri *path*:
  - int cod_cli.

  Descrizione: Restituisce il numero di offerte in sospeso per il cliente richiesto.

OfferteDetController

- **PUT** /api/offertedet/{id_offerta}
  Parametri *request body*:
  - List<Models.OffertaDet> listDet.

  Parametri *path*:
  - int id_offerta.

  Descrizione: Modifica il dettaglio offerta richiesto.
- **PUT** /api/offertedet/inviainsede/{id_offerta}
  Parametri *request body*:
  - List<Models.OffertaDet> listDet.

  Parametri *path*:
  - int id_offerta.

  Descrizione: Modifica il dettaglio offerta richiesto ed inivia l'offerta in sede.

4.3.2 Endpoint per gestione documenti

Di seguito vengono elencati gli *endpoint* progettati per il *web services* per la gestione dei documenti "legali":

TokenController

- **POST** /api/token
  Descrizione: Restituisce il *token* necessario per l'autenticazione alle altre richieste.

GeneraPDFController

- **POST** /api/generaPDF
  Parametri *request body*:
  - Models.InfoAzienda az.

  Descrizione: Genera il documento della *privacy* in formato PDF ed espone il link al *file*.

4.4 Web application

Per la progettazione della *web application* lato *front-end* si è utilizzato il classico *stack* formato da *HTML*, *CSS* e *JavaScript*, mentre per il lato *back-end* si è utilizzato il linguaggio *PHP* con la libreria *cURL*.
Sono stati realizzati dei semplici *mockup* in modalità *tablet*, in quanto l'applicazione viene utilizzata per il 99% delle volte da questi dispositivi. I vari *mockup* sono stati realizzati in seguito a uno studio di altre *web application*, realizzate dall'azienda in cui ho svolto il mio *stage*. Di conseguenza, questa deve avere un *design* in linea con gli altri prodotti dell'azienda.
Successivamente seguiranno i *mockup* realizzati, rappresentanti le pagine della *web application*.

La pagina di login deve avere un *input* per l'utente e uno per la password, così da favorire l'accesso dell'utilizzatore. Alla figura 4.6 viene rappresentato a grandi linee l'aspetto della pagina.

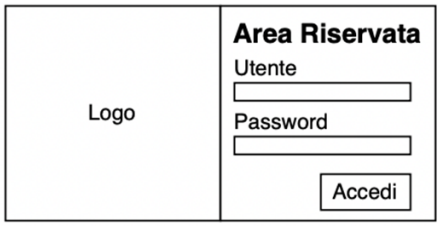


Figura 4.6: Mockup pagina login

La figura 4.7 rappresenta la pagina del menù. Le varie opzioni sono rappresentate sotto forma di quadratoni per favorire il tocco tramite *touch screen*.

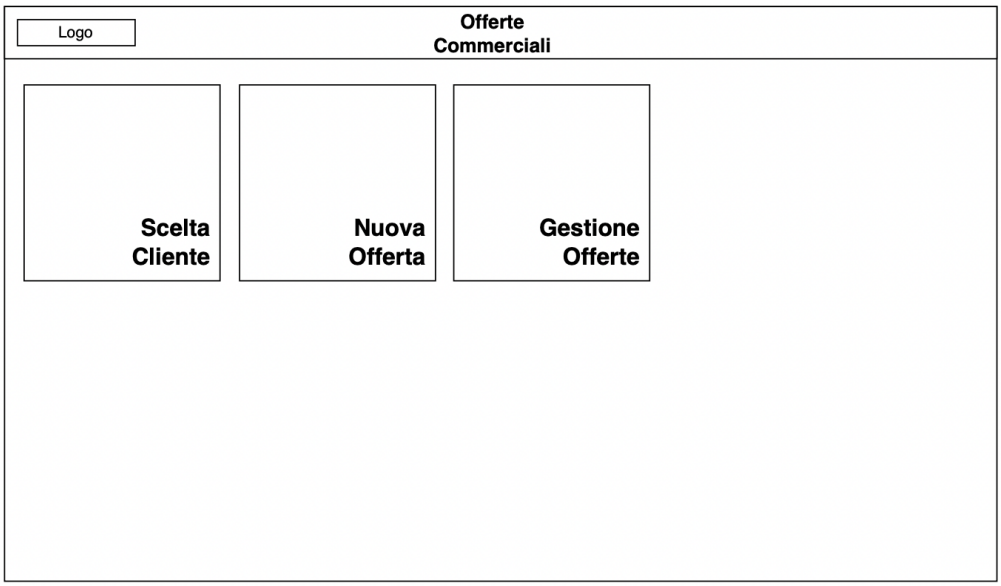


Figura 4.7: Mockup pagina menù

La pagina "scelta cliente" (figura 4.8) è dedicata alla visualizzazione e selezione del cliente. Ogni cliente è rappresentato sotto forma di quadratone per favorire il tocco tramite *touch screen*, proprio come le opzioni del menù.

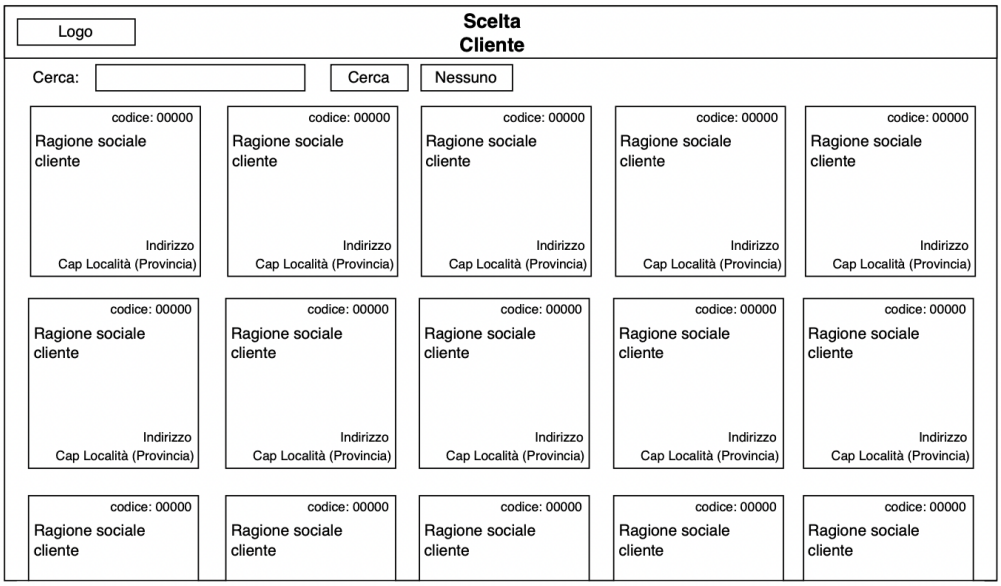


Figura 4.8: Mockup pagina scelta cliente

La figura 4.9 rappresenta la pagina di nuova offerta, la quale è caratterizzata da una parte sempre presente, ovvero la scelta della tipologia cliente, e da una parte dinamica che si popola in base alla configurazione del utente. In base

alla tipologia della domanda vengono create delle *checkbox* (multipla) o dei *radio* (singola) o ancora *input* testuali (testo) o numerici (intero).

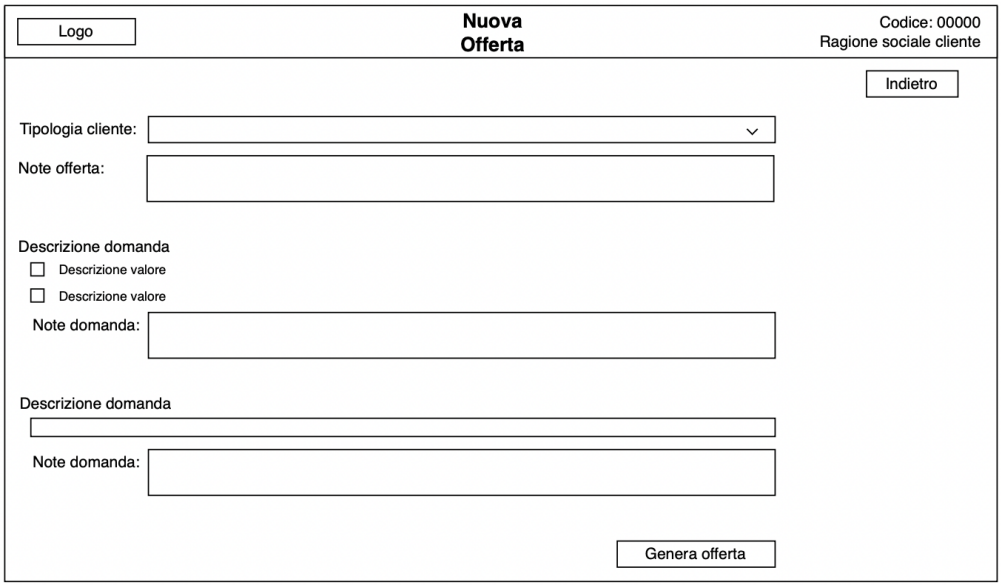
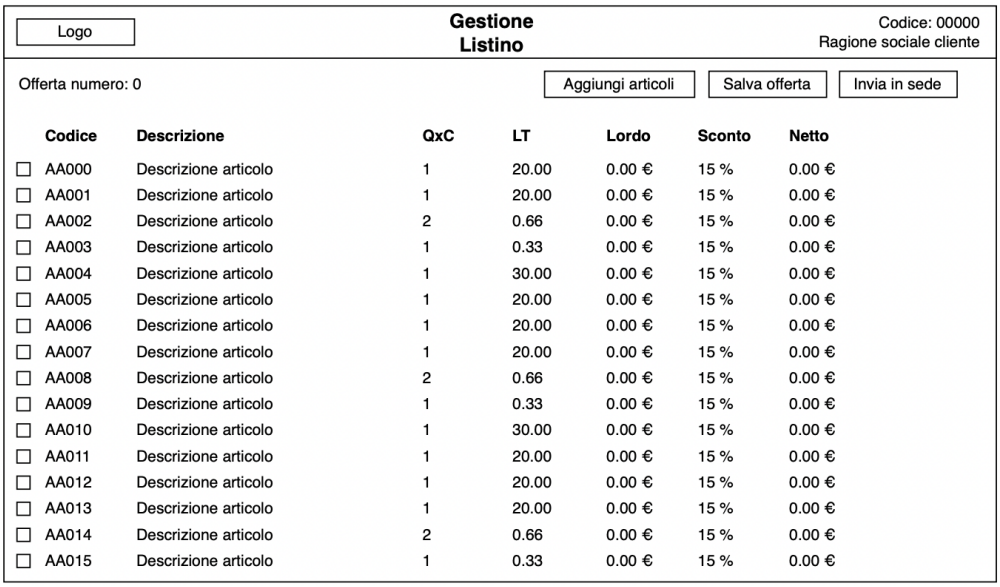


Figura 4.9: Mockup pagina nuova offerta

La pagina di gestione listino (figura 4.10) rappresenta gli articoli generati dal sistema sotto forma tabellare. È presente un *input checkbox* in ogni articolo, per esprimere la volontà di volerlo o meno nel listino.



Codice	Descrizione	QxC	LT	Lordo	Sconto	Netto
AA000	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA001	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA002	Descrizione articolo	2	0.66	0.00 €	15 %	0.00 €
AA003	Descrizione articolo	1	0.33	0.00 €	15 %	0.00 €
AA004	Descrizione articolo	1	30.00	0.00 €	15 %	0.00 €
AA005	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA006	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA007	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA008	Descrizione articolo	2	0.66	0.00 €	15 %	0.00 €
AA009	Descrizione articolo	1	0.33	0.00 €	15 %	0.00 €
AA010	Descrizione articolo	1	30.00	0.00 €	15 %	0.00 €
AA011	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA012	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA013	Descrizione articolo	1	20.00	0.00 €	15 %	0.00 €
AA014	Descrizione articolo	2	0.66	0.00 €	15 %	0.00 €
AA015	Descrizione articolo	1	0.33	0.00 €	15 %	0.00 €

Figura 4.10: Mockup pagina gestione listino

La pagina di gestione offerte, rappresentata alla figura 4.11, stampa le varie offerte in una struttura tabellare.

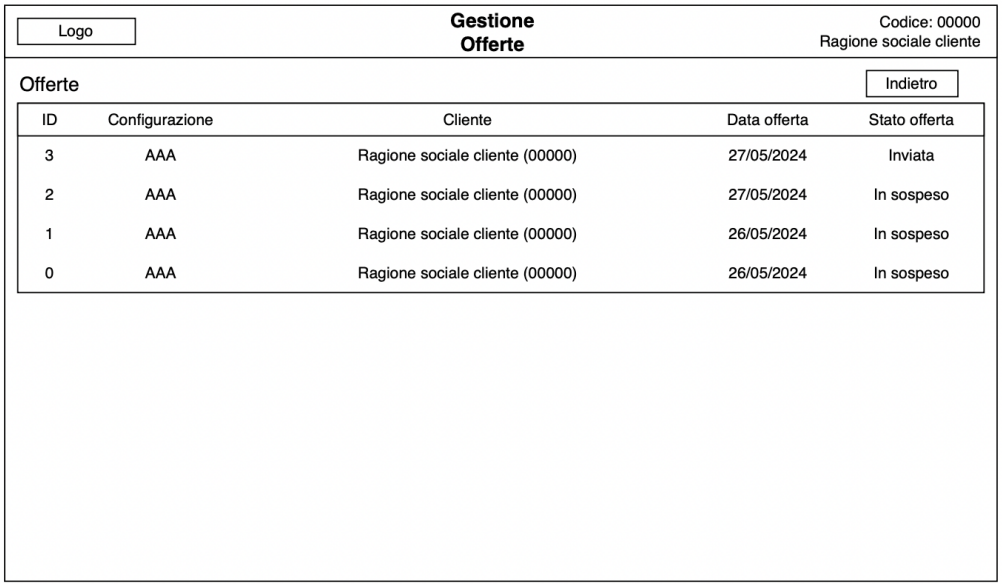


Figura 4.11: Mockup pagina gestione offerte

5

Codifica e prodotto finale

5.1 Organizzazione generale

La fase di codifica per la gestione delle offerte commerciali è divisa, data la natura del progetto, in tre parti fondamentali:

- Sviluppo moduli *desktop*;
- Sviluppo *web services*;
- Sviluppo *web application*.

Inoltre, si è sviluppato il *web services* per la gestione dei documenti "legali".
Per motivi puramente di design del documento in questione, si è deciso di descrivere la codifica del *web services* per la gestione di documenti "legali" all'interno della sezione 5.3 *Web services*. La scelta di procedere in questo preciso ordine di sviluppo è scaturita dal fatto che i moduli *desktop*, come abbiamo visto dallo schema che descrive l'architettura generale del sistema di gestione delle offerte commerciali (4.1), leggono e scrivono dati interagendo direttamente con il *database*, mentre la *web application* necessita, appunto, del *web services* realizzato a tale scopo.

5.2 Moduli desktop

La codifica dei moduli *desktop* comprende il *setup* del progetto, la composizione puramente grafica dei vari *form* in linea con il resto del gestionale e la

gestione dei vari eventi scatenati.

Come prima cosa ho creato un nuovo progetto *Windows forms* in locale per ogni modulo. Successivamente, ho allineato tutti i nuovi progetti come quelli già esistenti, quindi ho installato le dipendenze necessarie, quali l'*Entity framework 6.4.0*, il *driver* per la connessione al *database Informix* e le librerie dei vari componenti *Devexpress*. In seguito, ho creato i *form* veri e propri, popolandoli con i vari componenti e definendo anche le proprietà necessarie di essi, come l'etichetta di un *button*, il testo di una *label*, la dimensione di un campo e molte altre. Queste proprietà sono modificabili tramite il codice vero e proprio, quindi facendo il *set* della proprietà stessa, oppure utilizzando la scheda "Proprietà", messa a disposizione dall'IDE *Visual Studio*, nella quale sono presenti vari *input* con cui interagire; questo secondo metodo facilita e riduce notevolmente il tempo di impiego dell'operazione. Infine ho definito gli eventi sui componenti che ne richiedono l'impiego; questi, quando invocati, eseguono il codice definito. Un esempio per comprendere quest'ultimo passaggio lo si può vedere tramite l'utilizzo del bottone "Salva", il cui evento è associato al *click* del *mouse*, che estrapola così il contenuto dai vari campi del *form* e lo salva sul *database*, utilizzando Entity Framework. Simile è la logica applicata ad altri componenti dei moduli.

5.2.1 Modulo offc_domande

Per quanto riguarda il modulo *offc_domande*, seguiranno delle figure accompagnate da una breve descrizione per spiegare le funzionalità principali sviluppate.

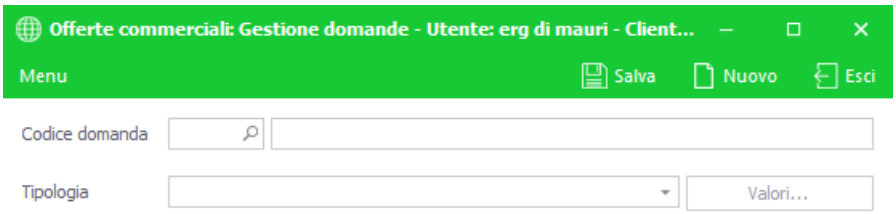


Figura 5.1: Modulo offc_domande

Il modulo *offc_domande* si presenta come mostrato in figura 5.1. Andando a compilare i campi è possibile creare una nuova domanda, mentre cliccando il bottone a forma di lente di ingrandimento presente sul campo relativo a codice

domanda è possibile aprire un altro *form* per leggere una domanda.
La figura 5.2 mostra come si presenta il *form* per la ricerca / lettura di una domanda. Cliccando su una riga della tabella è possibile completare la lettura. Una volta completata correttamente la lettura, la domanda, con tutti i suoi campi, sarà visualizzata sugli input in precedenza vuoti (vedi figura 5.3). Se la tipologia della domanda letta è SINGOLA o MULTIPLA, il bottone "Valori..." sarà cliccabile e con quello sarà possibile gestire i valori relativi alla domanda letta.

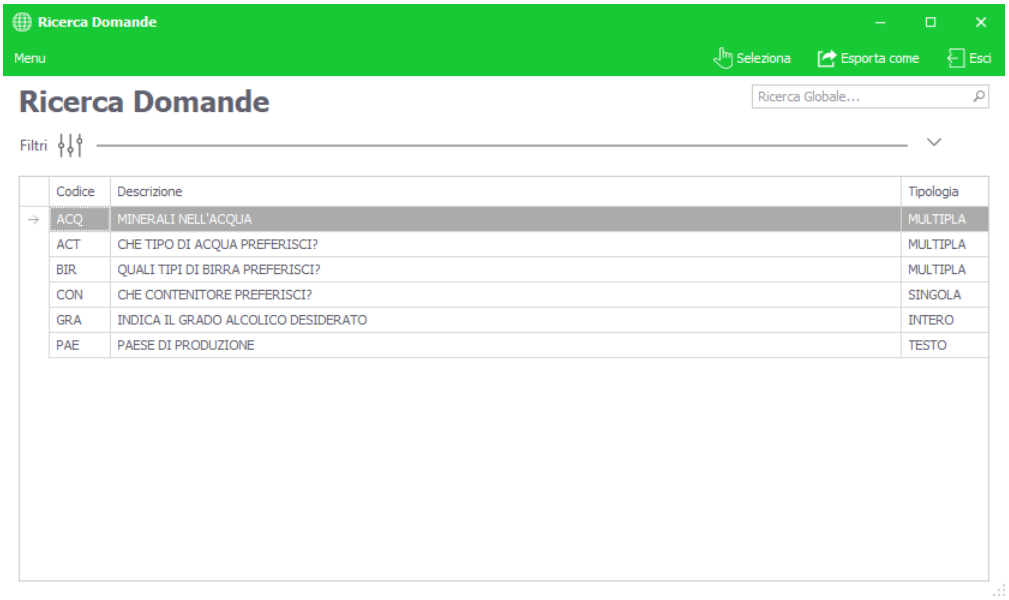


Figura 5.2: Lettura domanda

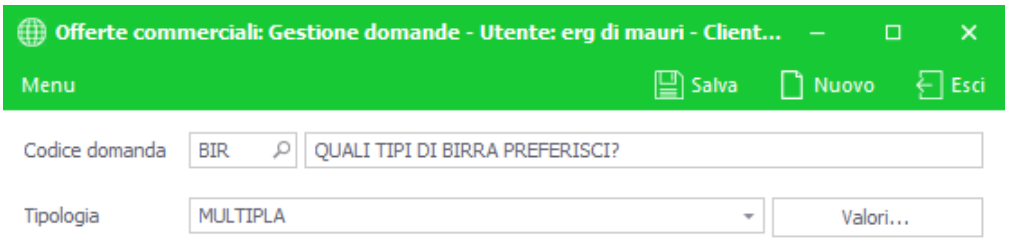


Figura 5.3: Domanda letta correttamente

Il *form* relativo alla gestione valori si presenta come mostrato in figura 5.4. Andando a compilare i campi è possibile creare un nuovo valore da associare alla domanda letta in precedenza, mentre cliccando il bottone a forma di lente di ingrandimento presente sul campo relativo al codice valore è possibile aprire un

altro *form* per leggere un valore.
La figura 5.5 mostra come si presenta il *form* per la ricerca / lettura di un valore.
Cliccando su una riga della tabella è possibile completare la lettura.
Una volta completata correttamente la lettura, il valore sarà visualizzato con tutti i suoi campi sugli input in precedenza vuoti (vedi figura 5.6).

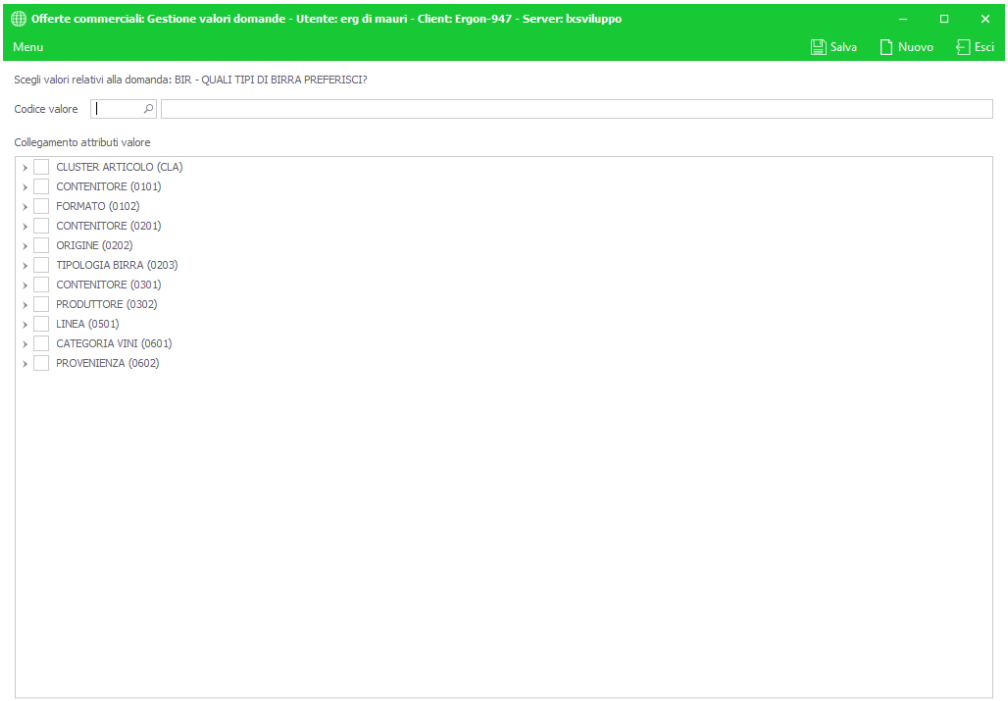


Figura 5.4: Gestione valori relativi alla domanda letta

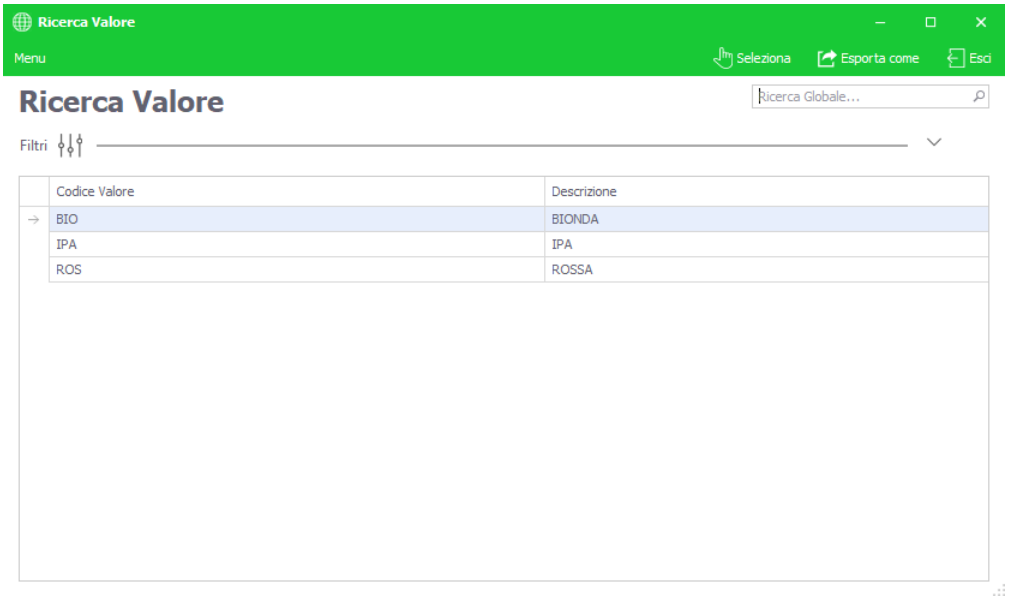


Figura 5.5: Lettura valore

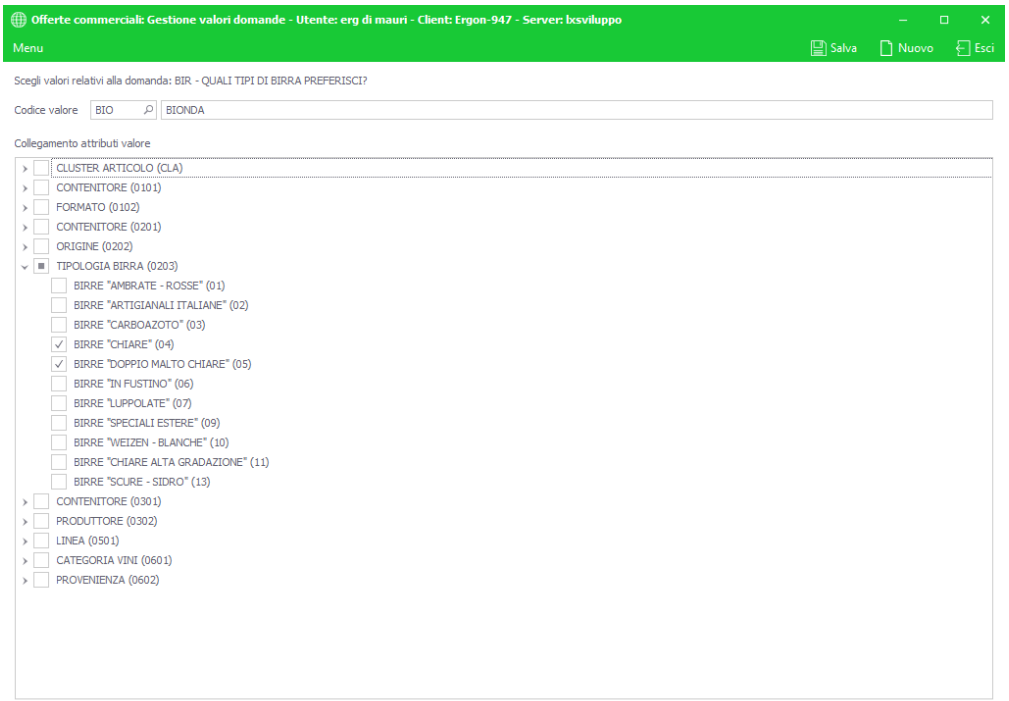


Figura 5.6: Valore letto correttamente

Un aspetto implementativo interessante è quello del metodo *CreaNodi()*. Questo metodo permette di popolare i vari nodi della *treelist* con le varie voci estrapolate dal *database*.

Come prima cosa è necessario estrarre la lista dei *tipi attributo*, i quali definiscono il primo livello della gerarchia (ad esempio la voce "TIPOLOGIA BIRRA (0203)" in figura 5.6). Viene poi eseguito un ciclo della lista, quindi per ogni elemento viene creato il nuovo nodo, e successivamente si ricava anche la corrispondente lista dei *dettagli*, i quali definiscono di conseguenza il secondo livello della gerarchia (ad esempio "BIRRE AMBRATE - ROSSE (01)" in figura 5.6). Infine per ogni dettaglio viene creato il nuovo nodo, a cui viene assegnato il corrispondente *tipo attributo* come genitore per realizzare correttamente la gerarchia ad albero.

5.2.2 Modulo offc_config

Per quanto riguarda il modulo *offc_config*, seguiranno delle figure accompagnate da una breve descrizione per spiegare le funzionalità principali sviluppate.

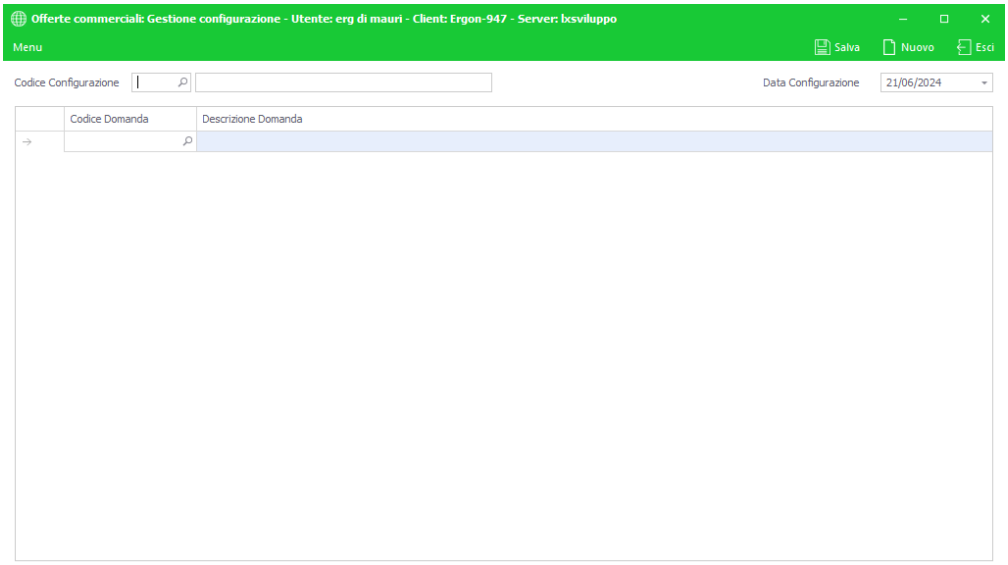


Figura 5.7: Modulo offc_config

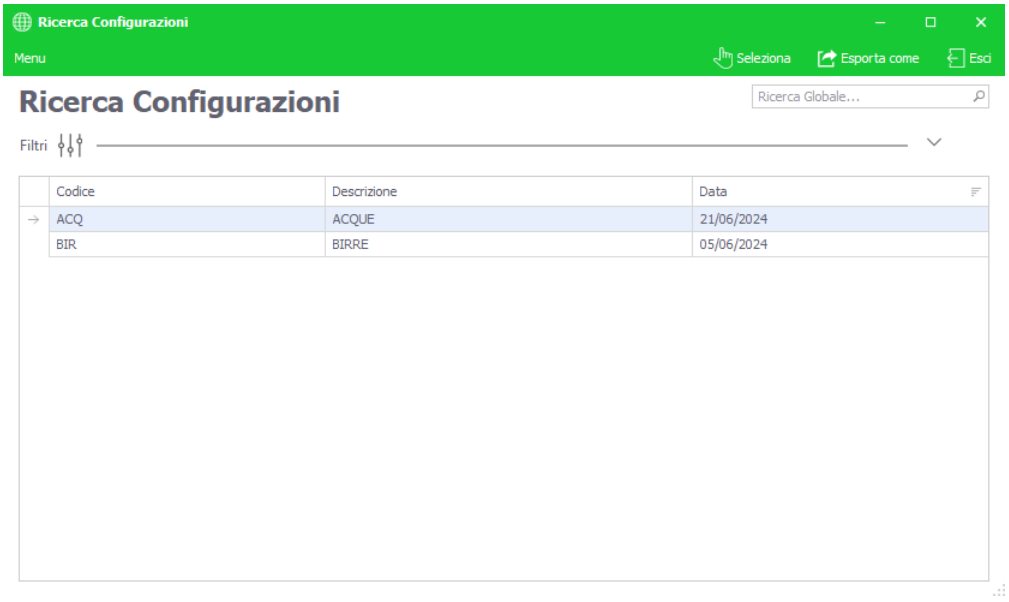


Figura 5.8: Lettura configurazione

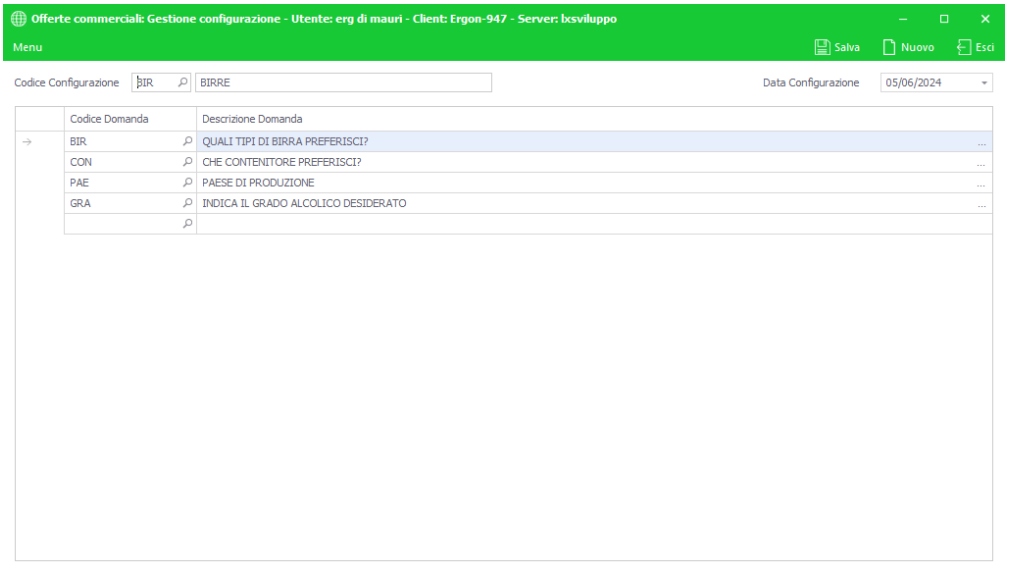


Figura 5.9: Configurazione letto correttamente

Il *form* relativo alla gestione configurazioni si presenta come mostrato in figura 5.7. Andando a compilare i campi è possibile creare una nuova configurazione, mentre cliccando il bottone a forma di lente di ingrandimento presente sul campo relativo al codice configurazione è possibile aprire un altro *form* per leggere una configurazione.
La figura 5.8 mostra come si presenta il *form* per la ricerca / lettura di una configurazione. Cliccando su una riga della tabella è possibile completare la lettura.

Una volta completata correttamente la lettura, la configurazione sarà visualizzata con tutti i suoi campi sugli input in precedenza vuoti (vedi figura 5.9). È possibile aggiungere eventuali domande alla configurazione, cliccando il bottone a forma di lente di ingrandimento presente sul campo relativo al codice domanda a riga vuota della tabella.

5.3 Web services

Per lo sviluppo di entrambi i *web services* è stato impiegato un approccio incrementale, così da avere sempre un prodotto funzionante. Sono stati definiti i modelli necessari e successivamente è stato codificato il metodo del controllo che mappa la richiesta HTTP in questione.
La codifica dei modelli è un'operazione semplice, che richiede la creazione di una classe con gli attributi necessari. Questo significa che durante l'implementazione di un modello per una tabella del *databse*, non bisogna per forza definire tutti gli attributi presenti. Ad esempio, nel modello che ho realizzato per la tabella *clienti* non ho definito l'attributo *telefono*, anche se presente nell'entità in questione, in quanto non mi era necessario. Inoltre, ho realizzato un modello personalizzato volto a gestire la struttura dati più complessa, che viene ricevuta come *input* dagli *endpoint* di *GenerazioneListiniController*. Per quanto riguarda i *controller*, invece, è consigliato definire un prefisso di *route* per la classe e solo successivamente definire i vari metodi. I metodi della classe implementano la logica dell'*endpoint* richiesto. È necessario anche stabilire un eventuale sotto *route* e il metodo HTTP a cui esso risponde.

```
1    [RoutePrefix("api/configurazioni")]
2    public class ConfigurazioniController : ApiController
3    {
4        [Route("")]
5        [JwtAuthentication]
6        [HttpGet]
7        public IHttpActionResult GetTutte() {...}
8
9        [Route("{cod_config:string}")]
10       [JwtAuthentication]
11       [HttpGet]
```

```
12          public IHttpActionResult GetPiuRecente(string cod_config)
    {...}
13      }
```

Code 5.1: Esempio controller

Una volta terminata la codifica di un *endpoint* o di molteplici, e dopo averli testati, si procede con la pubblicazione del servizio nel *web server* di sviluppo. I *web services* sono dunque pubblicati su Internet Information Services (IIS) per *Windows Server*. IIS è un *web server* flessibile, sicuro e gestibile per ospitare qualsiasi tipo di contenuto sul *web*, dallo *streaming* multimediale alle applicazioni *web* e ai servizi di *API*, il tutto grazie alla sua scalabilità e architettura pronta a gestire operazioni dispendiose e impegnative.

5.4 Web application

Per quanto riguarda la *web application*, la codifica si è divisa tra le varie pagine. Tutte le pagine sono *responsive* e quindi si adattano a molteplici dispositivi, anche se l'applicazione verrà utilizzata pressochè tutte le volte da dei *tablet*. Lo stile delle pagine segue il *design* indicatomi dall'azienda e in linea con gli altri prodotti.

Ogni pagina della *web application* presenta una specifica logica di funzionamento. I vari elementi con cui è possibile interagire, ad esempio un semplice bottone di un *form*, richiamano i relativi *script Javascript*; questi a loro volta effettuano una chiamata *Ajax* al *backend* della pagina stessa, indicando però nella richiesta un comando, che non è altro che una variabile nell'*url*. La pagina *PHP*, dato il comando, esegue la porzione di codice specifica ed eventualmente esegue richieste con la libreria *cURL* al *web services* per effettuare operazioni sul *database*. Al termine di ciò, vengono restituiti i risultati, indicati gli eventuali errori, oppure l'utente viene reindirizzato a una nuova pagina. Il comando è gestito intuitivamente dalla struttura di controllo *switch*, nella quale ogni *case* identifica il comando in questione. Quando viene richiesta per la prima volta una pagina, viene eseguito il *case* di *default*, che, nella pagina "gestione offerte", ad esempio, esegue una chiamata al *web services* per ricavare le offerte e restituire al *browser* il codice HTML per la corretta visualizzazione.

Di seguito viene descritta ogni pagina con anche la relativa figura.

Login
La pagina di *login* permette agli utenti di autenticarsi nel sistema. Se le credenziali sono corrette si verrà reindirizzati alla pagina del menù.

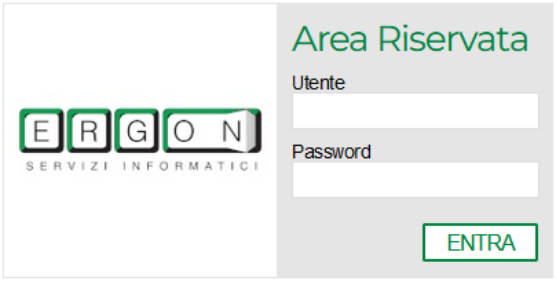


Figura 5.10: Pagina login

Menù
La pagina menù permette agli utenti di selezionare la funzionalità desiderata. Si è deciso di utilizzare dei "quadratoni" per favorire l'utilizzo del *touch* dei *tablet*. La funzionalità "nuova offerta" è visibile solo dopo aver selezionato un cliente.

Figura 5.11: Pagina menù

Scelta cliente
La pagina "scelta cliente" permette all'utente di selezionare un cliente per poi

andare a generare un listino a lui dedicato. La pagina utilizza la stessa idea dei "quadratoni" impiegata nel menù. È inoltre possibile effettuare ricerche.
Una volta selezionato il cliente desiderato si viene reindirizzati al menù, dove è possibile scegliere la funzionalità "nuova offerta".

Figura 5.12: Pagina scelta cliente

Nuova offerta

La pagina "nuova offerta" è la pagina più importante della *web application*. Qui l'utente ha modo di compilare le risposte alle domande, per poi andare a generare il listino personalizzato al cliente. Una volta generata l'offerta dall'apposito bottone, l'utente viene reindirizzato al menù in caso si tratti di un agente commerciale, mentre se si tratta di un utente amministratore viene reindirizzato alla pagina di gestione del listino generato.

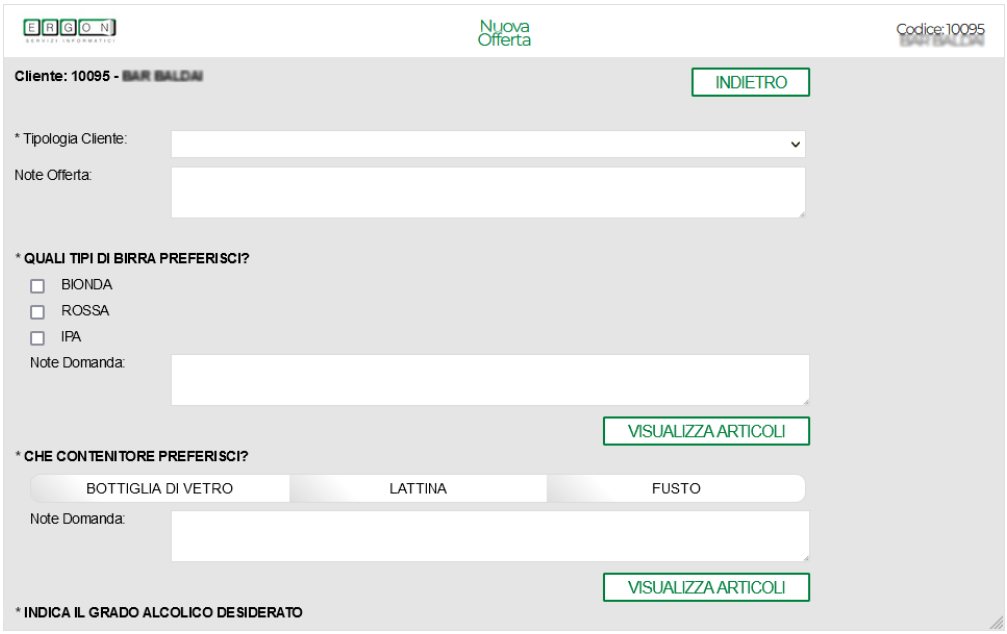
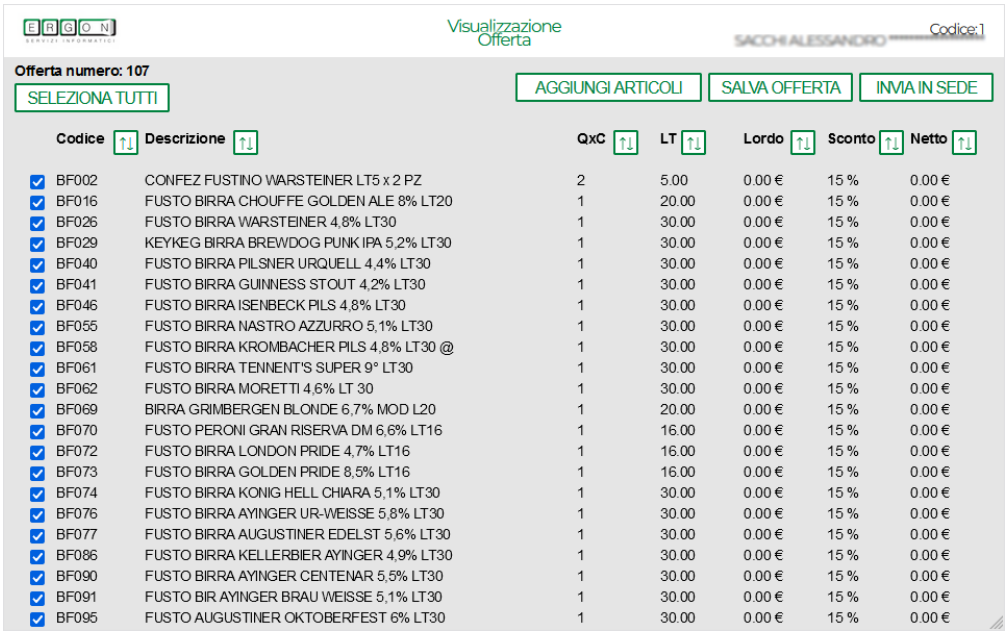


Figura 5.13: Pagina nuova offerta

Gestione listino

La pagina gestione listino permette la fruizione e la gestione del listino appena generato. È possibile aggiungere uno o molteplici nuovi articoli all'offerta ed inoltre è possibile salvare l'offerta o inviarla in sede, rendendola così non più modificabile.



Codice	Descrizione	QxC	LT	Lordo	Sconto	Netto
BF002	CONFEZ FUSTINO WARSTEINER LT5 x 2 PZ	2	5.00	0.00 €	15 %	0.00 €
BF016	FUSTO BIRRA CHOUFFE GOLDEN ALE 8% LT20	1	20.00	0.00 €	15 %	0.00 €
BF026	FUSTO BIRRA WARSTEINER 4,8% LT30	1	30.00	0.00 €	15 %	0.00 €
BF029	KEYKEG BIRRA BREWDOG PUNK IPA 5,2% LT30	1	30.00	0.00 €	15 %	0.00 €
BF040	FUSTO BIRRA PILSNER URQUELL 4,4% LT30	1	30.00	0.00 €	15 %	0.00 €
BF041	FUSTO BIRRA GUINNESS STOUT 4,2% LT30	1	30.00	0.00 €	15 %	0.00 €
BF046	FUSTO BIRRA ISENBECK PILS 4,8% LT30	1	30.00	0.00 €	15 %	0.00 €
BF055	FUSTO BIRRA NASTRO AZZURRO 5,1% LT30	1	30.00	0.00 €	15 %	0.00 €
BF058	FUSTO BIRRA KROMBACHER PILS 4,8% LT30 @	1	30.00	0.00 €	15 %	0.00 €
BF061	FUSTO BIRRA TENNENT'S SUPER 9° LT30	1	30.00	0.00 €	15 %	0.00 €
BF062	FUSTO BIRRA MORETTI 4,6% LT 30	1	30.00	0.00 €	15 %	0.00 €
BF069	BIRRA GRIMBERGEN BLONDE 6,7% MOD L20	1	20.00	0.00 €	15 %	0.00 €
BF070	FUSTO PERONI GRAN RISERVA DM 6,6% LT16	1	16.00	0.00 €	15 %	0.00 €
BF072	FUSTO BIRRA LONDON PRIDE 4,7% LT16	1	16.00	0.00 €	15 %	0.00 €
BF073	FUSTO BIRRA GOLDEN PRIDE 8,5% LT16	1	16.00	0.00 €	15 %	0.00 €
BF074	FUSTO BIRRA KONIG HELL CHIARA 5,1% LT30	1	30.00	0.00 €	15 %	0.00 €
BF076	FUSTO BIRRA AYINGER UR-WEISSE 5,8% LT30	1	30.00	0.00 €	15 %	0.00 €
BF077	FUSTO BIRRA AUGUSTINER EDELST 5,6% LT30	1	30.00	0.00 €	15 %	0.00 €
BF086	FUSTO BIRRA KELLERBIER AYINGER 4,9% LT30	1	30.00	0.00 €	15 %	0.00 €
BF090	FUSTO BIRRA AYINGER CENTENAR 5,5% LT30	1	30.00	0.00 €	15 %	0.00 €
BF091	FUSTO BIR AYINGER BRAU WEISSE 5,1% LT30	1	30.00	0.00 €	15 %	0.00 €
BF095	FUSTO AUGUSTINER OKTOBERFEST 6% LT30	1	30.00	0.00 €	15 %	0.00 €

Figura 5.14: Pagina gestione listino

Gestione offerte

La pagina "gestione offerte" permette la visualizzazioni delle offerte create. Un utente agente visualizza solamente le offerte create dalla sua configurazione personale, mentre un utente amministratore ha la possibilità di visualizzare tutte le offerte. Cliccando su una specifica offerta si viene indirizzati alla pagina "variante" (così definita dall'azienda Ergon) di nuova offerta, che dinamicamente compila gli input con le risposte salvate in fase precedente, permettendo così all'utente di cambiare le risposte per poi rigenerare l'offerta.

Figura 5.15: Pagina gestione offerte

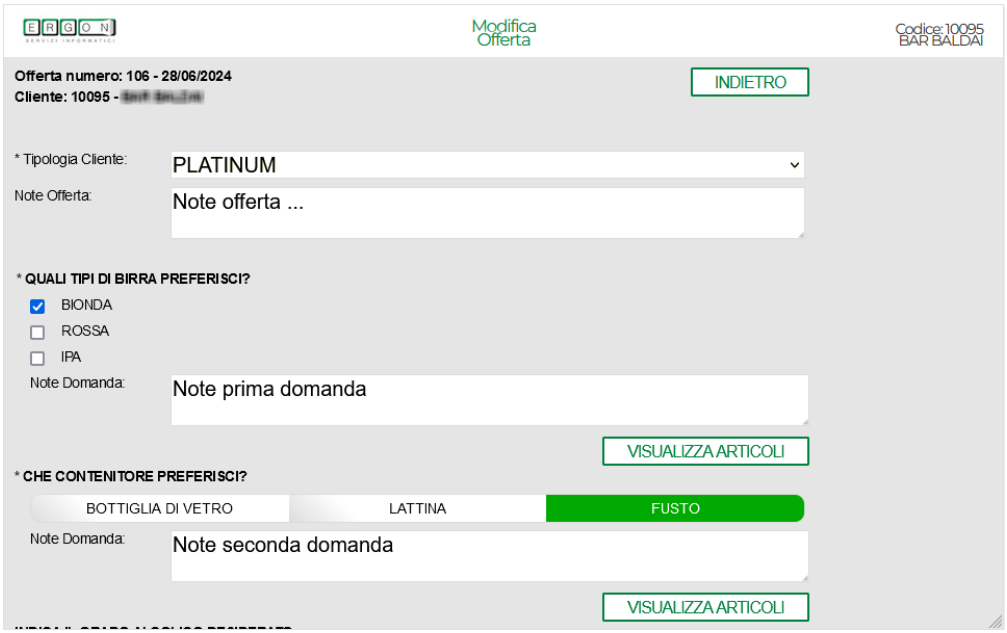


Figura 5.16: Pagina "variante" nuova offerta

6 Verifica e validazione

6.1 Verifica

Il progetto non prevede lo sviluppo di *test* in forma di codice sorgente. Si tratta di un tipo di operazione veramente complessa e, dato che quasi l'intero *software* richiede l'interazione fisica da parte di un utente, si è optato con dei test di utilizzo reale basati sull'esecuzione di porzioni di progetto, che vanno dunque a verificare il corretto funzionamento. Il mio *tutor* aziendale, insieme ad altri colleghi, ha verificato quanto sviluppato mano a mano che le varie funzionalità venivano codificate.

6.1.1 Moduli desktop e web application

Per quanto riguarda entrambi i moduli *desktop* e la *web application* sono stati eseguiti dei *test* manuali di normale utilizzo dei vari sistemi. Durante l'esecuzione dei *form* e la visualizzazione delle varie pagine, si è valutato il corretto funzionamento del sistema, sia analizzando gli scenari considerati "corretti", sia analizzando le situazioni eccezionali. Questo tipo di operazione è stata eseguita da me durante lo sviluppo delle funzionalità richieste e anche da parte del *tutor* aziendale e degli altri colleghi.

6.1.2 Web services

Per quanto riguarda entrambi i *web services*, quello per la gestione commerciale e quello per la gestione dei documenti "legali", sono stati testati tutti gli *endpoint* realizzati con *Postman*. *Postman* è uno strumento che permette di effettuare richieste HTTP, specificando il metodo, eventuali parametri ed altre impostazioni, favorendo così un ambiente completo per testare le proprie *API*. Nella figura 6.1 è riportato un esempio di *test* di un specifico *endpoint* utilizzando lo strumento appena descritto.

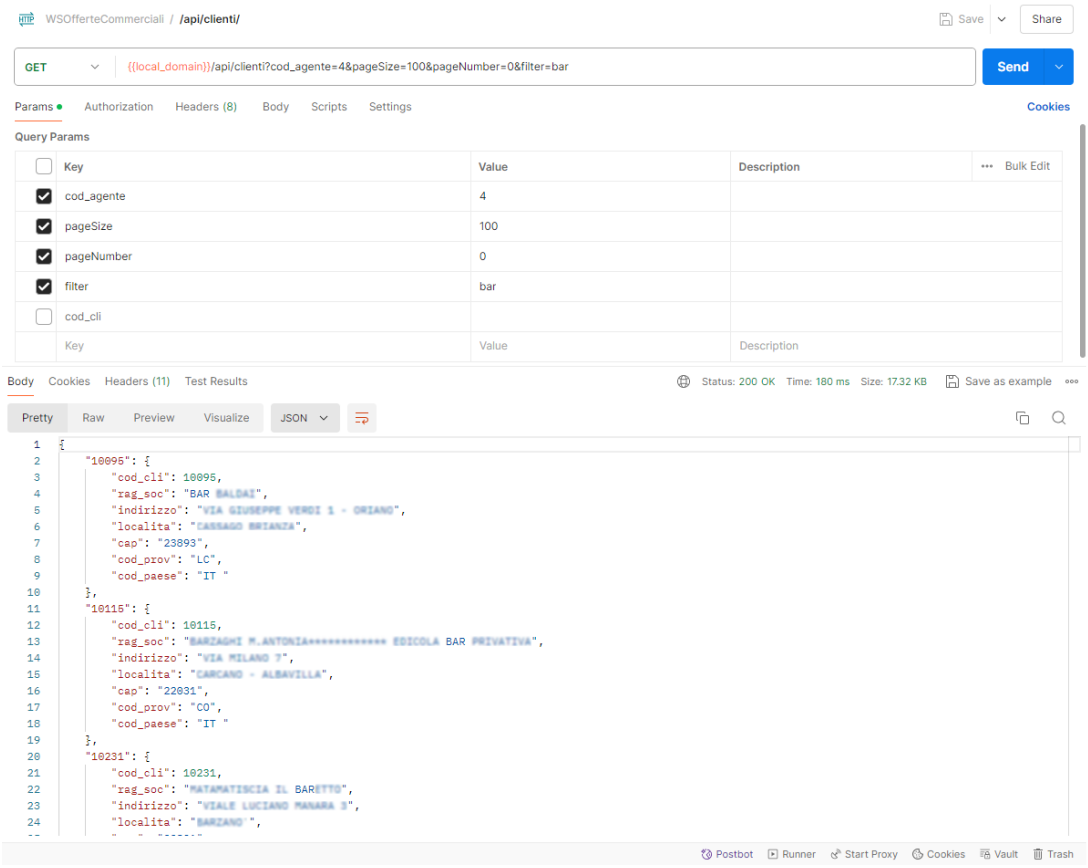


Figura 6.1: Test con Postman

6.2 Validazione

6.2.1 Codice

La validazione del codice prodotto per il progetto è stata fatta dal *tutor* aziendale. Sono stati effettuati controlli periodici del codice, verificando che quello

che è stato fatto rispettasse le linee guida desiderate in ambito di correttezza e leggibilità.

6.2.2 Requisiti

Seguono le tabelle dello stato dei requisiti definiti in fase di analisi. Da notare che tutti i requisiti obbligatori sono stati soddisfatti.

Requisito	**Stato**
RFO1	Soddisfatto
RFO2	Soddisfatto
RFO3	Soddisfatto
RFO4	Soddisfatto
RFO5	Soddisfatto
RFO6	Soddisfatto
RFO7	Soddisfatto
RFO8	Soddisfatto
RFO9	Soddisfatto
RFO10	Soddisfatto
RFO11	Soddisfatto
RFO12	Soddisfatto
RFO13	Soddisfatto
RFO14	Soddisfatto
RFO15	Soddisfatto
RFO16	Soddisfatto
RFO17	Soddisfatto
RFO18	Soddisfatto
RFO19	Soddisfatto

Requisito	**Stato**
RFO20	Soddisfatto
RFO21	Soddisfatto
RFO22	Soddisfatto
RFO23	Soddisfatto
RFO24	Soddisfatto
RFO25	Soddisfatto
RFO26	Soddisfatto
RFO27	Soddisfatto
RFO28	Soddisfatto
RFO29	Soddisfatto
RFO30	Soddisfatto
RFO31	Soddisfatto
RFO32	Soddisfatto
RFO33	Soddisfatto
RFO34	Soddisfatto
RFO35	Soddisfatto
RFO36	Soddisfatto
RFO37	Soddisfatto
RFO38	Soddisfatto
RFO39	Soddisfatto
RFO40	Soddisfatto
RFO41	Soddisfatto
RFO42	Soddisfatto
RFD43	Soddisfatto

Requisito	Stato
RFO44	Soddisfatto
RFO45	Soddisfatto
RFO46	Soddisfatto
RFO47	Soddisfatto
RFO48	Soddisfatto
RFO49	Soddisfatto
RFO50	Soddisfatto
RFO51	Soddisfatto
RFO52	Soddisfatto
RFO53	Soddisfatto
RFO54	Soddisfatto
RFO55	Soddisfatto
RFO56	Soddisfatto
RFO57	Soddisfatto

Tabella 6.1: Tabella validazione requisiti funzionali

Requisito	Stato
RQO1	Soddisfatto
RQO2	Soddisfatto
RQO3	Soddisfatto
RQO4	Soddisfatto

Tabella 6.2: Tabella validazione requisiti di qualità

Requisito	**Stato**
RVO1	Soddisfatto
RVO2	Soddisfatto
RVO3	Soddisfatto
RVO4	Soddisfatto

Tabella 6.3: Tabella validazione requisiti di vincolo

7 Conclusioni

7.1 Obiettivi prefissati

Come descritto nei capitoli precedenti, lo scopo del progetto era quello di realizzare un sistema completo per la gestione delle offerte commerciali e un sistema completo per la generazione di un documento PDF precompilato, il tutto utilizzando le tecnologie che vengono quotidianamente impiegate dagli sviluppatori di Ergon Informatica S.r.l..
Il prodotto finale è composto da varie parti che si interfacciano tra di loro e la parte più importante è, senz'ombra di dubbio, la *web application* per la creazione e gestione delle offerte.
Nel primo capitolo di questo documento (1) erano stati definiti degli obiettivi da portare a compimento per una buona riuscita del progetto di *stage*; nella tabella 7.1 che segue viene illustrato il soddisfacimento di questi obiettivi.

Codice obiettivo	Descrizione obiettivo	Stato
OB1	Sviluppo di un *web service* con tutti gli *end-point* necessari alla lettura/scrittura dei dati utilizzati dall'applicazione *web*	Soddisfatto
OB2	Sviluppo di una *web application* dinamica per la creazione delle offerte commerciali	Soddisfatto
OB3	Sviluppo di un programma *desktop* per la configurazione delle domande necessarie alla creazione delle offerte commerciali	Soddisfatto
OB4	Sviluppo di un *web service* per la creazione di un documento PDF da restituire compilato	Soddisfatto

Tabella 7.1: Tabella con lo stato degli obiettivi

7.2 Conoscenze acquisite

Durante l'esperienza in Ergon Informatica S.r.l., sono stati affrontati diversi temi legati al mondo dell'informatica, dalla programmazione di oggetti allo sviluppo *web*.
Lo sviluppo della *web application* è stato il tema principale dello *stage* e anche la parte che mi ha occupato più tempo. Le nozioni apprese durante il corso di studio si sono dimostrate fondamentali per la realizzazione del sistema per la gestione delle offerte commerciali, oltre al fatto che si sono arricchite grazie all'approfondimento della libreria *PHP cURL*, utilizzata appunto in questo progetto, ed anche delle tecniche e logiche impiegate dai programmatori dell'azienda.
Un altro aspetto fondamentale di questa esperienza è stato la conoscenza acquisita del linguaggio *C#* e dei *framework*, quali *Windows Forms* e *ASP .NET Core*, legati al mondo *Microsoft* per lo sviluppo di applicazioni *desktop* e legati al *web*. Sono tutte nozioni e tecnologie interessanti, che sicuramente mirerò ad approfondire maggiormente anche in via personale.

Oltre alle conoscenze acquisite di ambito tecnico professionale, il periodo di *stage* mi ha permesso di arricchire le mie competenze trasversali. È sicuramente migliorata la capacità di analisi e risoluzione di un problema, la capacità di gestione delle risorse e la capacità di rispettare le scadenze legate a un progetto. Inoltre, l'interazione con il *tutor* aziendale e con altri colleghi ha sicuramente accresciuto la mia attitudine nel relazionarmi con le persone.

7.3 Valutazione personale

L'esperienza di *stage* presso l'azienda Ergon Informatica S.r.l. si è rilevata estremamente positiva e formativa. Questo periodo di formazione non ha rappresentato solo un completamento del mio percorso accademico, ma anche una preziosa opportunità per accrescere le mie competenze professionali e per comprendere in modo più approfondito il funzionamento reale di un'azienda di *software*.
Un aspetto significativo di questa esperienza è stato quello di trovare un ambiente di lavoro stimolante e collaborativo. Il mio *tutor* aziendale e anche tutti gli altri numerosi colleghi hanno dimostrato fin da subito grande disponibilità e supporto nei miei confronti. Si è sempre trovato lo spazio per dei confronti, per la condivisione delle conoscenze e di consigli utili.
La più grande difficoltà incontrata è stata adattarmi al *way of working* dell'azienda. Si tratta di un processo estremamente lungo e complicato da raggiungere in un periodo di tempo così limitato, tuttavia già nelle ultime settimane di *stage* è stato evidente come certe meccaniche cominciavano a diventare più automatiche e rapide. Lo sforzo è stato mitigato grazie ai consigli del *tutor* e necessario era cercare sempre di mantenere una costanza e dedizione nel raggiungimento degli obiettivi prefissati.
Considero questa esperienza come un elemento fondamentale del mio percorso formativo e professionale. Essa mi ha permesso di consolidare le mie conoscenze teoriche, di acquisire nuove competenze pratiche e di sviluppare un approccio professionale, offrendomi una visione concreta delle dinamiche interne ad un azienda.
Sono quindi estremamente grato a Ergon Informatica S.r.l. per l'opportunità che mi è stata concessa e sono certo che le nozioni apprese durante questo *stage* costituiranno un solido punto di partenza per la mia carriera professionale.

Glossario

Angular è un *framework open source* per lo sviluppo di applicazioni *web*. È sviluppato da *Google* ed permette di progettare e implementare progetti strutturati per la realizzazione di interfacce utenti, con immediati vantaggi in termini di robustezza del codice, testabilità e manutenibilità, creando applicazioni che sono anche veloci e performanti. 9

CRM (*Customer Relationship Management*) è il processo con cui un'azienda o un'altra organizzazione amministra le sue interazioni con i clienti, in genere utilizzando dei dati per studiare grandi quantità di informazioni. 1, 9

CRUD (*Create-Read-Update-Delete*), creazione, lettura, aggiornamento e rimozione. Sono le quattro operazioni basilare per la gestione persistente dei dati . 69, 70, 73, 74

Entity Framework è un *framework open source Object-Relational-Mapping* (ORM) che permette di lavorare con i dati sotto forma di oggetti con le proprie proprietà specifiche del dominio, senza doversi preoccupare delle tabelle e delle colonne del *database* in cui sono archiviati questi dati. 70, 73, 83

ERGDIS è un ERP sviluppato interamente da Ergon Informatica S.r.l. composto da vari moduli che ricoprono ogni aspetto della gestione aziendale: dall'area amministrativa al controllo direzionale, dall'area commerciale alla pianificazione e al controllo della produzione, dalla gestione acquisti alla logistica di magazzino, dall'archiviazione ottica alla gestione della qualità. 1

ERGTRA è un ERP sviluppato interamente da Ergon Informatica S.r.l. composto da vari moduli che ricoprono ogni aspetto legato al mondo dei traspor-

ti nazionali: dalla bollettazione alla fatturazione, al business intelligence, all'archiviazione digitale, alla tracciabilità delle spedizioni su *web*. 1

ERP (*Enterprise Resource Planning*) software gestionale che integra tutte le funzioni aziendali più importanti, come ad esempio gestione del magazzino, vendite, acquisti, finanza, contabilità e molto altro. 1, 9

RDBMS (*Relational Database Management System*) è un *software* che gestisce dati organizzati sotto forma di tabelle relazionali, le quali sono collegate tra di loro tramite delle chiavi. L'interrogazione e la manipolazione dei dati è possibile eseguila tramite il linguaggio SQL garantendo l'integrità e supportando le transazioni per mantenere la coerenza dei dati. 9

React è una libreria *open-source*, *front-end*, *JavaScript* per la creazione di interfacce utente. È mantenuto da *Meta* e da una comunità di singoli sviluppatori e aziende. React può essere utilizzato come base nello sviluppo di applicazioni a pagina singola ma è utilizzabile anche su *mobile* tramite React Native, una libreria sempre sviluppata da *Meta* che tramuta i componenti React in componenti nativi (*iOS* e *Android*). 8

UML (*Unified Modeling Language*) è un linguaggio di modellazione che permette, tramite l'utilizzo di modelli visuali, di analizzare, descrivere, specificare e documentare un sistema software anche complesso. 18, 20, 21, 23, 25, 26, 28, 30, 32, 35–37, 39, 40, 44, 46, 47, 51, 53, 54, 58, 59

VMWARE Azienda americana che si occupa di sviluppare *software* per la realizzazione di macchine virtuali. 1

Vue è un *framework JavaScript open source* in configurazione *Model–view–viewmodel* per la creazione di interfacce utente e *single-page applications*. 9

Bibliografia

[1] *Linguaggio C#*. URL: https://learn.microsoft.com/it-it/dotnet/csharp/.

[2] *Windows Forms*. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0.

[3] *ASP .NET Core*. URL: https://dotnet.microsoft.com/en-us/apps/aspnet.

[4] *DevExpress*. URL: https://www.devexpress.com/.

[5] *Informix*. URL: https://www.ibm.com/it-it/products/informix.

[6] *PHP*. URL: https://www.php.net/.

[7] *Visual Studio*. URL: https://visualstudio.microsoft.com/.

[8] *REST API*. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api.