Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Una soluzione per l'estrazione di metadati per il licensing da Microsoft Dynamics CRM

Relatore
Dott. Armir Bujari
Università di Padova

Laureando
Anna Poletti

A Riccardo

Abstract

Nel presente documento viene descritto il lavoro svolto presso l'azienda NETCOM srl durante il periodo di stage. Lo stage ha avuto una durata complessiva di 320 ore, distribuite in due mesi di lavoro a tempo pieno.
Lo stage è consistito nello sviluppo di un applicazione per la semplificazione del Software Asset Management. Alla studentessa è stato richiesto di progettare e implementare una soluzione per estrarre dati significativi per il licensing dal prodotto Microsoft Dynamics CRM. Inoltre le è stato affidato il compito di studiare il prodotto d'interesse, individuare le informazioni utili e trovare un modo per estrarle.
In concomitanza alle fasi di analisi e progettazione, la studentessa si è impegnata a fornire un'adeguata documentazione di progetto.

Ringraziamenti

Desidero ringraziare il Dott. Armir Bujari, relatore della mia tesi, per i preziosi consigli e il sostegno fornitomi durante la stesura del lavoro.

Ringrazio il tutor esterno Alessandro Strenghetto per la disponibilità che ha sempre mostrato nei miei confronti e l'aiuto fornitomi nello sviluppo del progetto.

La mia più profonda gratitudine va a Riccardo, per essere il mio luogo naturale.

Infine desidero ringraziare le mie mie fonti di ispirazione Violetta Bellocchio, Joan Didion, Claudia Durastanti, Sylvia Plath, Nick Cave e Zvonimir Boban.

Indice

Lista delle figure

Lista delle tabelle

1 Introduzione

Il seguente capitolo si pone l'obiettivo di presentare il contesto in cui si è svolto lo stage. Viene fornita una descrizione dell'azienda, della sua struttura e dei clienti a cui si rivolge. Inoltre vengono illustrate le tecnologie che vengono utilizzate dall'azienda e che sono state impiegate nel corso dello stage.

1.1 L'azienda

NETCOM nasce nel 1999 con l'obiettivo di proporre al mercato soluzioni e servizi di eccellenza per l' *IT Life Cycle Management*[G] e l' *IT Governance*[G] . Quest'ultimo, viene inteso dall'azienda come l'insieme dei processi, procedure, competenze e strumenti utili alla gestione degli item che compongono il sistema informativo, in tutte le fasi del loro ciclo di vita, per gli aspetti di:

- Gestione tecnica: per la soluzione di problemi di qualsiasi natura che il cliente può riscontrare nell'utilizzo del sistema;
- *Governance*: per la gestione di tutte le operazioni IT di livello manageriale ed allinearle alle richieste del business.

L'*IT Life Cycle Management* è un *framework*[G] che si occupa di gestire e ottimizzare il sistema informativo di un'organizzazione, riducendo costi e rischi e aumentando complessivamente la qualità. Per questo motivo, il personale NETCOM del settore tecnico, vanta certificazioni di *best practices*[G] (*ITIL*[G] Foundation e ITIL Intermediate) e standard (ISO 9001 e ISO 20000) per la gestione del sistema informativo. Essendo partner *Ivanti, SNOW* ed *EasyVista*, il personale NETCOM presenta certificazioni che attestano la competenza operativa nei rispettivi *software*. [1]

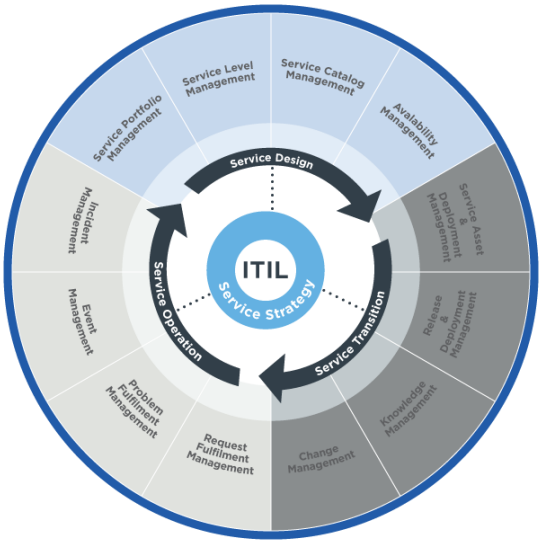

Fig. 1.1: ITIL framework

La tipologia dei principali servizi offerti da NETCOM colloca l'azienda nel mercato delle soluzioni di *System Management*, settore dedicato alla gestione di tutti gli elaboratori presenti all'interno di un'organizzazione.
Gli aspetti a cui l'azienda pone maggior attenzione riguardano attività come la distribuzione dello strato *software* destinato all'utilizzo di un dato *host*, l'installazione distribuita di applicazioni, la gestione remota di aggiornamenti e vulnerabilità, l'inventario dello stato e della configurazione delle macchine all'interno dell'organizzazione e il controllo remoto degli *host*.
L'azienda offre una vasta gamma di servizi che ne determinano la competitività:

- *Software Asset Management*: servizio di consulenza che consente di comprendere quante e quali delle licenze *software* acquistate sono effettivamente necessarie in funzione dell'utilizzo del cliente e delle relative strategie di business. Il $SAM_{[G]}$ è un approccio strutturato che si basa su una serie di processi, *best practices* e competenze specifiche, il cui obiettivo è di ridurre le complessità di gestione, i rischi ed i costi effettivi del *software*.
  NETCOM offre servizi multilivello volti a proteggere e valorizzare il patrimonio *software* delle organizzazioni, affiancandole nelle verifiche di conformità e nell' ottimizzazione relativamente al contesto operativo;

- *IT Service Management*: servizio che si occupa della modellazione dei servizi IT tracciando le linee guida per la progettazione e la realizzazione dei sistemi per l'automazione dei processi relativi ai servizi aziendali. Attraverso un ciclo costante di monitoraggio, *reporting* e revisione dei risultati dei servizi IT, il personale NETCOM è in grado di indicare al management i provvedimenti da attuare per eliminare i livelli di servizio inefficienti;

- *IT Asset Management*: servizio costituito da un insieme di *best practices*, processi e strumenti che permettono alle funzioni amministrative, contrattuali e di inventario di supportare la corretta gestione del ciclo di vita degli $asset_{[G]}$.
  NETCOM collabora con i clienti analizzando le richieste dei singoli dipartimenti e, coerentemente con strategie di business, progetta ed implementa la gestione dei processi di *Asset Management*;

- Assistenza tecnica: servizio che garantisce la continuità nel funzionamento del sistema di management e coadiuva il personale del cliente nel caso in cui si presentino problematiche durante l'utilizzo quotidiano della soluzione. NETCOM offre la possibilità di stipulare contratti annuali di assistenza tecnica che prevedono l'utilizzo di strumenti evoluti, i quali ne garantiscono l'efficacia. Tutti i problemi *software* sono poi risolti da remoto nel pieno rispetto della sicurezza aziendale. [2]

1.2 Organizzazione aziendale

NETCOM conta circa 30 dipendenti con sede principale a Padova, in via Fusinato 42. Lo stage è stato svolto all'interno del dipartimento di *Development*, che si occupa dello sviluppo di soluzioni proprie interne all'azienda.
L'organizzazione aziendale interna può essere schematizzata dal seguente organigramma:

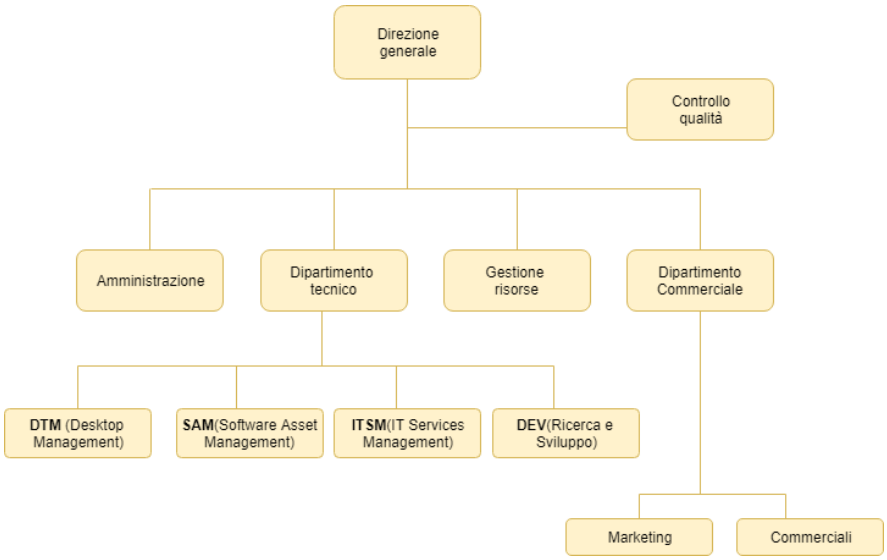

Fig. 1.2: Organizzazione aziendale NETCOM

- Direzione Generale: si occupa della pianificazione e organizzazione di tutte le attività dell'azienda;

- Controllo Qualità: definisce il sistema di qualità aziendale aderendo agli standard ISO e a *best practices* seguiti dall'azienda;
- Amministrazione: gestisce l'infrastruttura e la contabilità dell'azienda;
- Gestione Risorse: gestisce le risorse umane;
- Dipartimento Tecnico: composto da quattro sotto dipartimenti, è la parte dell'azienda che eroga concretamente i servizi.

1.3 Target

NETCOM lavora con più di 150 clienti distribuiti su tutto il territorio nazionale, di diverse tipologie e dimensioni, attive sia nel settore privato che nel settore pubblico. Per le caratteristiche dei servizi offerti, NETCOM si rivolge alla medio-grande impresa italiana che possiede tra i 250 e i 10000 dispositivi. Considerando l'ampia gamma di servizi che NETCOM offre attualmente , essa è adeguata per diverse tipologie di clientela come ad esempio le aziende che ricercano una garanzia di funzionamento o le aziende che desiderano investire in soluzioni di *System Management* e di sicurezza. Analizzando i principali servizi proposti da NETCOM, è possibile individuare il tipo di clientela a cui si rivolgono:

- Assistenza Tecnica : è un servizio di base che viene erogato al momento del bisogno, ovvero quando, durante l'utilizzo di un sistema informativo, si presentano alcune problematiche che vanno al di là della competenza del reparto IT dell'azienda del cliente. Per evitare i rischi derivanti dalla loro scarsa competenza tecnica, il personale esperto e certificato garantisce la risoluzione del problema, facendo in modo che il cliente possa concentrarsi sul *business*. Questo servizio, non richiedendo onerosi investimenti iniziali, è adatto alle aziende che hanno un budget limitato per il reparto informatico;
- *Desktop Management*: è un servizio che si occupa di permettere a un'impresa il controllo gestionale del proprio sistema informativo tramite l'installazione dei prodotti che essa offre e tramite corsi di formazione per il personale del cliente. Questo tipo di servizio costituisce una spesa più onerosa rispetto alla semplice assistenza su richiesta e può essere visto come una scelta lungimirante. Le aziende a cui può interessare questo servizio, sono quelle che desiderano provare qualcosa di innovativo accettando il rischio dell'investimento iniziale e il compromesso di gestire autonomamente il loro *asset* tramite gli strumenti e la formazione offerta da NETCOM;
- *Asset Management* Automatizzato: è il servizio più completo, poiché sono gli operatori di NETCOM a svolgere tutte le attività riguardanti la gestione del sistema informativo di un'organizzazione. Questo servizio si rivolge a entrambe e tipologie di aziende

sopra individuate: le imprese che vogliono innovarsi e che approvano la spesa iniziale puntando ad avere un ritorno economico futuro; le imprese che vogliono mantenere la loro attuale situazione ma hanno l'esigenza di delegare a qualcuno la gestione del loro sistema informativo;

- *Self Service Password Management*: la gestione automatizzata del cambio e del *reset* della password può risultare di utilità fondamentale alle aziende che intendono garantire maggiore sicurezza alla propria organizzazione, grazie all'individuazione precisa dell'identità dell'utente che richiede il ripristino della password. Inoltre si presenta adatta alle aziende con *help desk*$_{[G]}$ che desiderano affidare ad un *software* la gestione delle password, ma allo stesso tempo necessitano del rispetto totale dei requisiti di sicurezza richiesti dalle aziende di medio-grandi dimensioni;

- *Software Asset Management*: questo servizio si occupa della gestione dell'intero ciclo di vita del *software*, dall'implementazione fino alla sua ritiro. NETCOM si occupa di verificare la conformità del *software* installato rispetto alle licenze acquistate e permette l'elaborazione di una strategia di normalizzazione, che prevede l'acquisizione, rimozione o sostituzione di alcuni applicativi, al fine di garantire. Questo servizio essere considerato utile ad un'azienda che è stata sottoposta ad una verifica formale del proprio *asset software*, ma anche di un'azienda che vuole ridurre gli sprechi, assicurandosi la propria compliance rispetto alle licenze acquistate e mantenendo la garanzia di funzionamento del sistema informativo.

NETCOM offre un'ampia gamma di soluzioni per le aziende che necessitano sia di garanzie di funzionamento sia di proposte innovative, offrendo un servizio completo alla propria clientela. Inoltre la conformità del proprio *asset software* alle licenze acquistate, si rivela un servizio necessario sia in caso di *audit*$_{[G]}$, sia in caso di bisogno da parte del cliente di focalizzarsi sul proprio business lasciando a NETCOM la gestione dei degli applicativi da acquisire e installare. [2]

1.4 Tecnologie

Dal momento che NETCOM è una società che offre principalmente servizi di consulenza, è necessario distinguere le tecnologie utilizzate dall'azienda e quelle a supporto dello stage. Alla studentessa è stato richiesto di utilizzare le tecnologie usualmente impiegate dal reparto Ricerca e Sviluppo dell'azienda, che differiscono da quelle quotidianamente adoperate dall'azienda per erogare i propri servizi di *Desktop Management, Asset Managemen*t e *Software Asset Management*.

1.4.1 Utilizzate dall'azienda

Nell'ambito della consulenza, NETCOM fa uso principalmente delle seguenti tecnologie:

- *Ivanti Endpoint Management*: soluzione, illustrata in Figura 1.3, di gestione dei dispositivi fissi e mobili degli utenti, che attraverso un'unica console centralizzata, consente di monitorare lo stato dei device, distribuire *software/patch* di sicurezza ed automatizzare azioni correttive per garantirne la sicurezza e l'affidabilità;

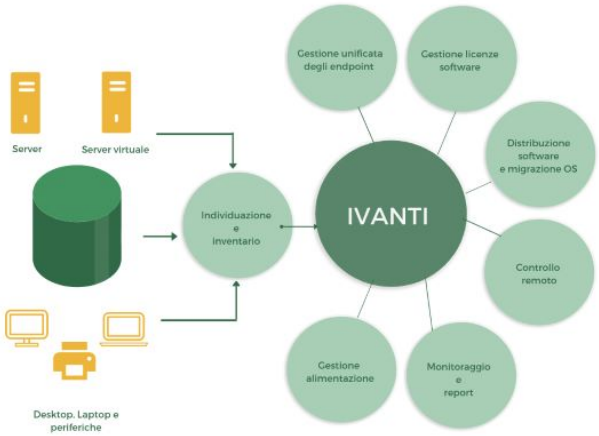

Fig. 1.3: Ivanti

- *EasyVista*: soluzione *software* di $ITSM_{[G]}$ altamente personalizzabile, pensata per realtà aziendali altamente strutturate e che necessitano di un alto grado di allineamento con il *framework ITIL*. Essendo un *software* integrato e modulare, è in grado di ottimizzare la gestione IT migliorando la qualità dei servizi e mantenendo sotto controllo i costi. Tra le funzionalità principali di *EasyVista* figurano:

  - Apertura di $ticket_{[G]}$ riguardanti incidenti e richieste di servizio;
  - Gestione da parte del reparto tecnico di tutti i *ticket* e assegnazione alle persone competenti per una risoluzione rapida e definitiva di essi;
  - Amministrazione degli *asset* aziendali.

- *Snow License Manager*: piattaforma di *Software Asset Management*, in grado di raccogliere i dati sulle applicazioni installate di computer e dispositivi mobili, determinare automaticamente lo stato di conformità delle licenze ed ottimizzarne l'assegnazione in base alla raccolta dei dati di effettivo utilizzo degli applicativi. La soluzione è progettata per ridurre i rischi, i costi e la complessità associati alla gestione del *software*, evitando gli sprechi per l'acquisto di licenze effettivamente inutilizzate e garantendo allo stesso tempo la piena conformità ai contratti d'uso. *SNOW* permette la suddivisione del progetto di *Software Asset Management* in tre attività principali:

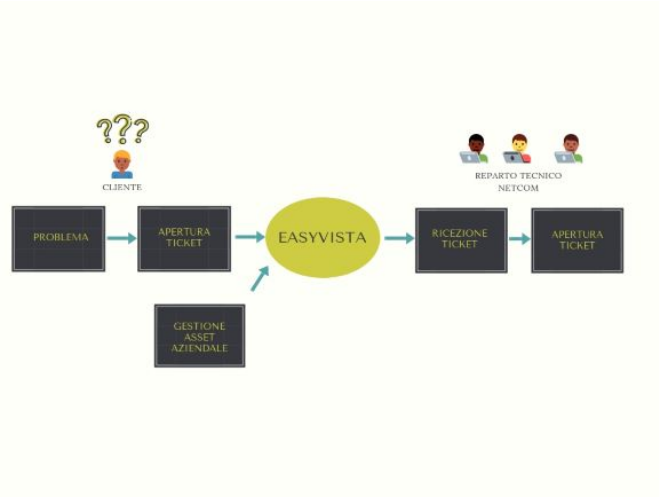

Fig. 1.4: Visione ad alto livello del funzionamento di EasyVista

- *Assessment*: come prima attività, è indispensabile formalizzare i requisiti del cliente ed analizzare lo stato degli $asset_{[G]}$ da molteplici punti di vista (installazioni, utilizzo, licenze), per poter delineare la situazione attuale e le possibili strategie di normalizzazione;
- Normalizzazione: sulla base della strategia scelta, alcuni applicativi dovranno essere acquistati, rimossi, installati o sostituiti. L'attività di normalizzazione supporta il cliente nell'applicazione delle modifiche;
- Mantenimento: questa fase prevede una serie di attività di controllo ricorrenti al fine di garantire nel tempo gli obiettivi raggiunti.

Fig. 1.5: Fasi del progetto di SAM gestite tramite SNOW

- *Hydrogen*: applicazione per permettere il *self-reset* delle password dimenticate o scadute da parte degli utenti, incrementando la sicurezza e riducendo i costi per il servizio di *help desk*. Tale servizio si integra completamente con l'interfaccia di accesso Windows (fase di richiesta login e password) aggiungendovi la funzionalità di *reset* password. Hydrogen si propone come soluzione per aziende medio-grandi, in sostituzione

all'identificazione dell'utente tramite il riconoscimento vocale da parte dell'operatore del servizio di *help desk*, poiché è plausibile pensare che l'operatore non sia in grado di riconoscere chiunque unicamente dalla voce tramite una telefonata;

- *Beryllium SPLA Manager*: soluzione che serve ad automatizzare la generazione di report, solitamente prodotti manualmente da personale tecnico specializzato, che individuano l'effettivo utilizzo del *software*. Questo progetto si pone l'obiettivo di offrire ai *service provider*$_{[G]}$ uno strumento completo che consente di monitorare e gestire i contratti $SPLA_{[G]}$ stipulati.

1.4.2 A supporto dello stage

Linguaggi

- C #: è un linguaggio di programmazione semplice, elegante e *type-safe* sviluppato da Microsoft, che consente agli sviluppatori di creare una vasta gamma di applicazioni protette e affidabili per *.NET Framework*. Essendo un linguaggio orientato a oggetti, C # supporta i concetti di incapsulamento, ereditarietà e polimorfismo;[3]

- SQL: è un linguaggio standardizzato per *database* relazionali progettato per creare e modificare schemi di *database*; inserire, modificare e gestire dati memorizzati; interrogare i dati memorizzati; creare e gestire strumenti di controllo e accesso ai dati;[4]

- PowerShell: è una *shell* caratterizzata dall'interfaccia a riga di comando e da un linguaggio di *scripting*, sviluppata da Microsoft. È basato sulla programmazione a oggetti e sul *framework* Microsoft .NET. PowerShell è la combinazione di compiti complessi e di una serie di componenti, le *cmdlets* (*command lets*, serie di comandi), che sono classi .NET progettate per sfruttare le caratteristiche dell'ambiente. [5]

Framework

- .NET Framework: è un ambiente di esecuzione gestita per Windows che fornisce un'ampia gamma di servizi alle *app* in esecuzione. È costituito da due componenti principali: *Common Language Runtime (CLR)*, vale a dire il motore di esecuzione mediante il quale vengono gestite le *app* in esecuzione, e la libreria di classi .NET Framework, che fornisce una raccolta di codice testato e riutilizzabile che gli sviluppatori possono chiamare dalle rispettive *app*.[6]

Software di sviluppo

- *Microsoft Visual Studio 2017* : è un ambiente di sviluppo integrato sviluppato da Microsoft che supporta attualmente diversi tipi di linguaggio, quali C, C++, C#, F#, Visual Basic .Net e ASP .Net, e che permette la realizzazione di applicazioni, siti e servizi in ambito *web*;[7]
- *Microsoft SQL Server Management Studio 2012 R2*: è un ambiente integrato che consente l'accesso, la configurazione, la gestione, l'amministrazione e lo sviluppo di tutti i componenti di *SQL Server*. Inoltre questa applicazione offre un'ampia gamma di strumenti grafici con numerosi editor di script avanzati per accedere a *SQL Server* dedicati agli sviluppatori e agli amministratori di *database*;[8]
- *VMWare Workstation*: è una serie di *software* che consentono di eseguire più sistemi operativi in un ambiente virtuale;[9]
- *Microsoft Dynamics CRM*$_{[G]}$ 2016: è un'applicazione *client/server*$_{[G]}$ di *customer relationship management* sviluppata da Microsoft che si appoggia al *web server IIS*$_{[G]}$ e focalizzata principalmente sui settori di vendite, *marketing* ed *help-desk*;

Software per la documentazione

- *Microsoft Word Online*: è una *web application* distribuita con licenza commerciale da Microsoft che consente di utilizzare il programma di videoscrittura Microsoft Word tramite *browser* e di lavorare sui documenti direttamente sul sito in cui sono archiviati;
- Draw.io: è un *software online* gratuito che consente di disegnare ed esportare diagrammi UML.

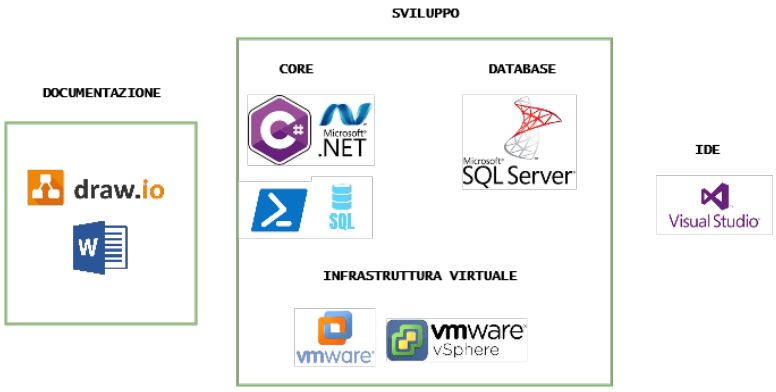

Fig. 1.6: Schema di utilizzo delle tecnologie nel progetto di stage

1.5 Struttura del Documento

Il resto della relazione di fine stage si divide nei seguenti capitoli:

- Capitolo 2: contiene la descrizione del progetto aziendale e delle motivazioni che hanno portato l'azienda a desiderare lo stage svolto. Viene analizzato il contesto in cui si colloca, le aspettative dell'azienda e i vincoli da rispettare;
- Capitolo 3: contiene la descrizione dello stage e dei risultati e attività svolte per conseguirli;
- Capitolo 4: contiene la valutazione retrospettiva dello stage.

1.6 Convenzioni tipografiche

All'interno del documento vengono utilizzate le seguenti convenzioni tipografiche:

- *Corsivo*: indica i termini in lingua straniera o facenti parti del linguaggio tecnico;
- *Termine*$_{[G]}$: indica i termini per cui viene fornita una definizione nel glossario. Viene sottolineata solamente la prima occorrenza del termine;
- [1]: indica i riferimenti bibliografici.

2
Il progetto aziendale

Il seguente capitolo illustra gli aspetti principali dello stage, il contesto di inserimento, gli obiettivi prefissati e le prospettive dell'azienda. Questo permette di comprendere il ruolo che ricoprono i progetti di stage all'interno della strategia aziendale.

2.1 Proposta di stage

Pur essendo un'azienda che offre principalmente servizi di consulenza e assistenza, NETCOM presenta anche un reparto di Ricerca e Sviluppo, che si occupa dell'innovazione tecnologica e segue progetti per l'accrescimento dell'offerta fornita alla clientela. L'azienda, tramite l'attività di stage offerta agli studenti laureandi, si confronta con il mondo universitario e permette allo studente di acquisire competenze e applicare le proprie conoscenze all'attività lavorativa. Da anni NETCOM partecipa all'evento *Stage-IT*$_{[G]}$ e presenta alcune possibili proposte di progetti in cui il laureando può essere inserito. Personalmente, dopo aver conosciuto l'azienda a Stage-IT, ho fissato un colloquio con l'azienda e discusso sulle possibili offerte di stage. La proposta di stage offertami da NETCOM è stata quella di essere inserita all'interno del loro dipartimento Ricerca e Sviluppo per partecipare allo sviluppo di *Beryllium*, una piattaforma sviluppata dall' azienda per monitorare l'utilizzo di *software* aziendali.

2.2 Progetto Beryllium

2.2.1 Contesto di inserimento

Attualmente, un numero crescente di aziende, affida in *outsourcing* a *service provider* alcuni dei servizi utilizzati così da potersi concentrare maggiormente sugli obiettivi aziendali. I *service provider* che offrono *software* Microsoft alle aziende, sono soggetti a un particolare programma di *licensing* chiamato *Service Provider Level Agreement* (SPLA). Il programma Microsoft SPLA prevede che ogni *service provider* paghi a Microsoft una quota mensile che

dipende all'utilizzo effettivo del *software* da parte dei vari clienti e pertanto, ogni mese, un *service provider* ha la necessità di generare un *report* con l'effettivo utilizzo del *software* Microsoft da inoltrare ad un rivenditore di contratti SPLA. La generazione di questi report è solitamente un'attività manuale svolta da personale tecnico specializzato. "Beryllium SPLA Manager" è sviluppato allo scopo di automatizzare la generazione di *report* e di offriren ai *service provider* uno strumento completo che consente di monitorare e gestire i contratti SPLA stipulati. L'applicativo è composto principalmente da 3 macro-componenti:

- *Web Portal Interface*: il portale *web* attraverso cui ogni *service provider* può monitorare l'utilizzo del *software* Microsoft di ogni suo cliente, gestire i contratti SPLA e generare i *report* mensili;
- *Core Engine*: questa componente è il cuore del sistema e contiene l'intelligenza in grado aggregare i dati grezzi per renderli fruibili all'utente finale tramite il portale *web*;;
- *Data Collector*: agente installato sulle macchine virtuali e fisiche di ogni *service provider* che si occupa di raccogliere, su base giornaliera, i dati grezzi di utilizzo.

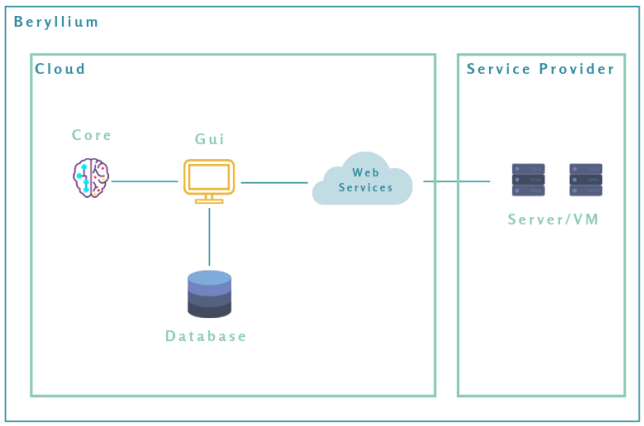

Fig. 2.1: Architettura macroscopica di Beryllium

2.2.2 Progetto di stage

Alla studentessa è stato richiesto di integrarsi con il team di sviluppo di NETCOM e collaborare con esso nella progettazione e implementazione di una soluzione per estrarre dati significativi per il *licensing* dal prodotto *Microsoft Dynamics* CRM 2016. Il Data Collector (agente) è un servizio di Windows che in maniera schedulata raccoglie tutti i dati grezzi che possono essere significativi per determinare il *licensing* dei prodotti Microsoft supportati. L'agente è stato progettato per essere modulare in modo tale che, per supportare un nuovo prodotto Microsoft, sia sufficiente aggiungere una $DLL_{[G]}$ in una cartella dell'agente. Il

progetto consiste nella realizzazione di una nuova *feature* dell'agente (una nuova DLL) contenente le funzioni atte all'estrazione dei dati significativi per il *licensing* del prodotto *Microsoft Dynamics* CRM 2016.

Il progetto di stage offerto dall'azienda comprende tutte le attività che costituiscono il processo di sviluppo di un prodotto *software*. Prima dell'inizio dello stage, il tutor Alessandro Strenghetto ha redatto un piano di lavoro per stabilire quali obiettivi dovessero essere raggiunti al suo termine. Relativamente al processo di sviluppo le attività previste dal progetto di stage sono:

- Analisi dei requisiti;
- Progettazione;
- Codifica;
- Test;
- Verifica e Validazione.

Oltre alle attività sopra elencate, il piano include una prima fase di formazione necessaria all'apprendimento del funzionamento delle tecnologie e degli ambienti di sviluppo da utilizzare in modo da acquisire familiarità con gli strumenti che saranno impiegati nell'intero processo di sviluppo; Inoltre è prevista una costante attività di documentazione, che deve accompagnare tutte le fasi del processo di sviluppo allo scopo di organizzare e tenere traccia di tutte le attività da svolgere durante lo stesso.

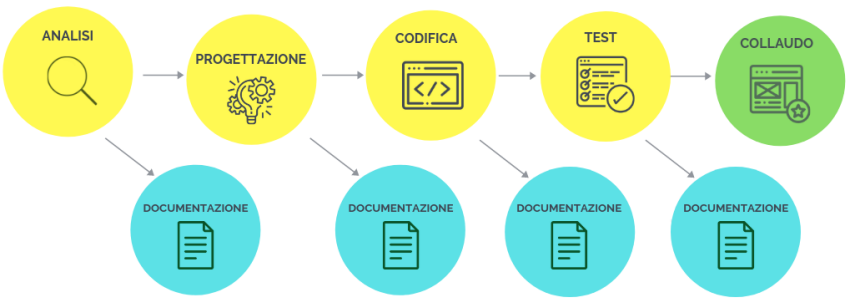

Fig. 2.2: Obiettivi e metodo di lavoro proposto da NETCOM

2.3 Vincoli

Nel Piano di Lavoro, sono stati introdotti dei vincoli, sia di tipo lavorativo, sia di tipo tecnologico. Di seguito sono riportati i vincoli suddivisi per tipologia:

- Vincoli lavorativi: questa categoria comprende i prodotti attesi al termine dello stage e le limitazioni imposte riguardanti la modalità di lavoro. Prima dell'inizio dell'attività di stage, con il tutor aziendale, sono stati stabiliti i seguenti prodotti attesi:

  - Prototipo della componente "Microsoft Dynamics CRM feature for Beryllium Data Collector" per l'ultima versione di *Microsoft Dynamics* CRM;
  - Documentazione di progetto composta dai documenti Analisi dei Requisiti e Specifica Tecnica.

  Il tutor aziendale ha condiviso un modello per la documentazione, così da permettere l'allineamento con gli standard utilizzati in azienda ed avere una traccia nello sviluppo dei documenti. Per lo sviluppo è stato reso disponibile il codice dell'applicativo da integrare, in modo da poter acquisire familiarità con il prodotto, e successivamente il personale del dipartimento DEV mi ha guidata nella realizzazione dello stesso.

- Vincoli tecnologici: questa categoria rappresenta tutte le condizioni e i limiti sulla tipologia di tecnologie da utilizzare per sviluppare i prodotti. I vincoli imposti sono stati:

  - La *feature Microsoft Dynamics* CRM deve essere compilabile con .NET framework 4.0. Eventuali librerie esterne utilizzate devono essere compatibili con .NET framework 4.0;
  - La soluzione non deve richiedere l'installazione di *tool* aggiuntivi sul *server* del cliente;
  - La soluzione deve essere retrocompatibile con la maggior parte delle versioni di *Microsoft Dynamics* CRM;
  - Il linguaggio con cui sviluppare la *feature* è il C#, con la possibilità di integrare del codice *PowerShell* all'interno del codice in C#;
  - La *feature* deve essere realizzata come libreria con estensione *.dll*;
  - La codifica della *feature* seguire la struttura decisa dall'azienda per potersi interfacciare con le altre componenti della piattaforma.

  Ogni vincolo imposto è stato controllato dal tutor aziendale durante lo svolgimento dello stage.

2.4 Piano di Lavoro

Durante la fase di pianificazione dello stage è stato redatto un piano di lavoro in collaborazione con il tutor aziendale. Allo scopo di delineare le attività da svolgere durante il corso dello stage, è stato prodotto un diagramma di Gantt, in cui è stata attribuita una certa quantità di ore che rappresenta la loro durata in termini orari di ogni attività. La durata delle attività è stata calcolata in modo da poter completare le stesse entro le 320 ore previste per lo stage formativo. Tale pianificazione ha consentito alla studentessa e al tutor aziendale di avere una visione completa delle attività da svolgere e delle tempistiche a disposizione.
Il piano di lavoro non è stato modificato nel corso dello stage al fine di poter effettuare un bilancio al termine di esso.

2.4.1 Obiettivi pianificati

Attraverso la pianificazione del lavoro sono stati stabiliti gli obiettivi da raggiungere alla conclusione dello stage. Tutte le attività sono state ripartite in tre tipologie tipologie di obiettivi, a seconda della loro priorità. In particolare si fa riferimento ai requisiti secondo le seguenti notazioni:

- O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Tale suddivisione ha permesso di quantificare il lavoro svolto durante lo stage. Alle sigle precedentemente indicate è stata aggiunta una coppia sequenziale di numeri, creando l'identificativo finale del requisito. Secondo la classificazione appena descritta gli obiettivi individuati con il tutor aziendale sono:

- Obbligatori:
  - O01: Studio dell'intero sistema *Beryllium*;
  - O02: Comprensione del mondo SAM, SPLA e del prodotti Microsoft di interesse;
  - O03: Configurazione dell'ambiente di test per il prodotto Microsoft Dynamics CRM;

- O04: Progettazione, sviluppo e test di un prototipo della componente "Microsoft Dynamics CRM feature for Beryllium Data Collector" per l'ultima versione di *Microsoft Dynamics* CRM;
- O05: Documentazione di progetto.

- Desiderabili:
  - D01: Completa integrazione con la componente *Beryllium Data Collector*.

- Facoltativi:
  - F01: Sviluppo e test della componente "Microsoft Dynamics CRM feature for Beryllium Data Collector" per almeno una versione più vecchia di *Microsoft Dynamics* CRM;
  - F02: Sviluppo e test della componente "Microsoft Dynamics CRM feature for Beryllium Data Collector" per almeno due versioni più vecchie di *Microsoft Dynamics* CRM.

2.4.2 Pianificazione attività e scadenze

Per raggiungere gli obiettivi prefissati sono state individuate una serie di attività che possono essere raggruppate in tre attività più estese:

- Introduzione: rappresenta la fase iniziale del progetto dedicata alla formazione e all'inserimento della studentessa nel contesto aziendale. In particolare:
  - Introduzione alle modalità di lavoro nel team di sviluppo: durante questa attività è prevista la descrizione delle fondamentali norme seguite dal team di Sviluppo, con particolare attenzione alla comprensione degli strumenti di versionamento utilizzati dall'azienda;
  - Formazione sul sistema aziendale di infrastruttura virtuale: l'attività prevede l'apprendimento dell'utilizzo del sistema di virtualizzazione aziendale;
  - Introduzione al progetto *Beryllium*: durante questa attività viene presentato il progetto di lavoro, rivolgendo particolare attenzione a come le componenti sviluppate dalla studentessa andranno ad interfacciarsi con il resto del prodotto.

- Analisi: rappresenta la prima attività del processo di sviluppo. In particolare:
  - Analisi dell'intero sistema *Beryllium*: attività in cui viene studiato il sistema esistente in modo da poter avere un'idea di massima sulla componente da realizzare durante lo stage;
  - Analisi del *data collector* di *Beryllium*: attività in cui viene studiato l'agente installato sulle macchine virtuali e fisiche di ogni *service provider*;
  - Analisi delle *features* già esistenti del *data collector* di *Beryllium*: attività in cui vengono analizzate le *features* precedentemente sviluppate ed integrate al fine di apprenderne la struttura e le funzionalità;
  - Identificazione requisiti: attività in cui vengono stabiliti i requisiti funzionali, di vincolo e di qualità del prodotto da sviluppare;
  - Documentazione: stesura del documento di Analisi dei Requisiti per tenere traccia del lavoro di analisi svolto.

- Progettazione: rappresenta l'attività che consiste nella determinazione dell'architettura del prodotto. In particolare:
  - Definizione architettura ad alto livello: attività che prevede la definizione della struttura della e la scelta dei *design patterns*;
  - Definizione e organizzazione componenti: attività che corrisponde alla progettazione di dettaglio, in cui vengono individuate le classi del programma, i relativi metodi e l'interazione tra esse;
  - Configurazione ambiente di test con *Microsoft Dynamics* CRM: l'attività prevede la creazione e la configurazione dell'ambiente di test e l'installazione di strumenti utili allo stage;
  - Documentazione: attività che consiste nella redazione del documento di Specifica Tecnica.

- Codifica: rappresenta l'attività che consiste nell'implementazione della soluzione individuata. In particolare:
  - Implementazione delle soluzioni individuate: attività che consiste nella realizzazione di ciò che è stato definito durante la progettazione;
  - Documentazione: attività che consiste nella redazione della documentazione della *feature Microsoft Dynamics* CRM con l'aggiunta di commenti in lingua inglese al codice prodotto.

- Test: rappresenta l'attività di verifica prevista.
  - *Bug fix* e collaudo finale: attività che consiste nell'esecuzione di test delle diverse funzionalità dell'applicazione e della sua integrazione con il resto del progetto;
  - Tracciamento dei requisiti: attività di validazione permette di riconoscere quali requisiti sono stati soddisfatti.

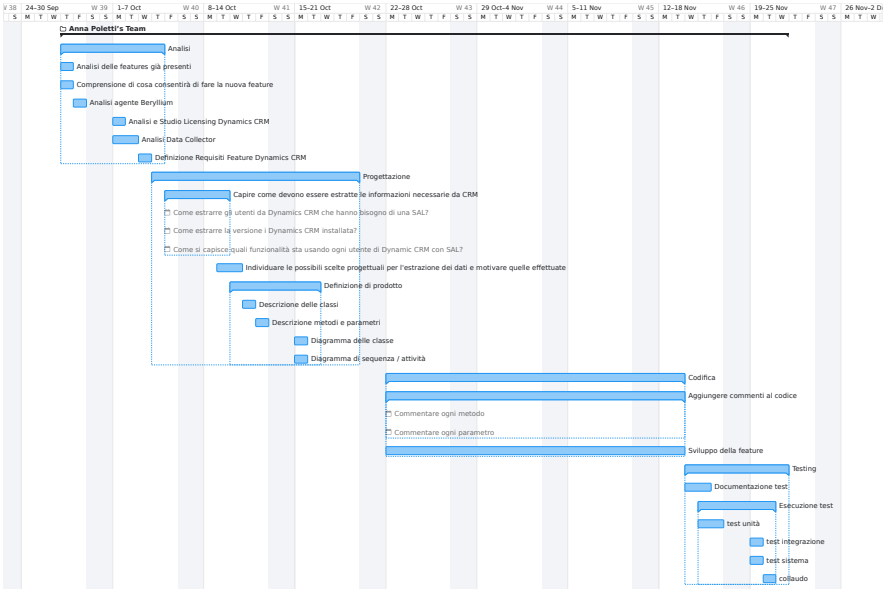

Fig. 2.3: Diagramma di Gantt: dall'Analisi al Collaudo

3
Lo stage

In questo capitolo vengono discussi i vari aspetti dello stage concernenti il processo di sviluppo. Ciascuna attività di progetto viene analizzata e descritta nel dettaglio.

3.1 Analisi

L'attività di Analisi è iniziata con lo studio dell'intero sistema *Beryllium*, in modo da delinearne le funzionalità e comprenderne i meccanismi di fondo allo scopo d comprendere gli aspetti d'interesse. Lo studio dell'intero applicativo si è rivelato molto utile, in quanto ha consentito di avere un'idea più precisa di come sarebbe dovuto essere il prodotto finale.

3.1.1 Architettura ad alto livello del sistema Beryllium

L'architettura ad alto livello del sistema *Beryllium* è così strutturata:

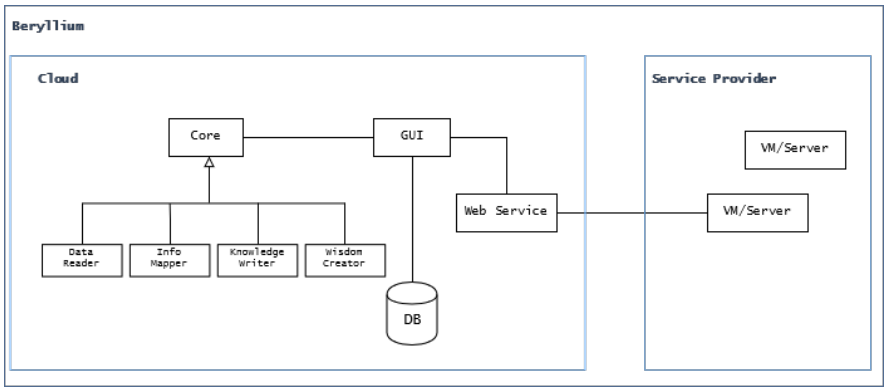

Fig. 3.1: Architettura ad alto livello del sistema Beryllium

Il sistema *Beryllium* è composto da due principali ambienti applicativi: il *cloud* e il *service provider*.

Cloud

Nell'ambiente Netcom, il *Core* e la *GUI* hanno un ruolo fondamentale all'interno dell'intero sistema. La *GUI* è l'interfaccia web che permette ad ogni *service provider* di monitorare l'utilizzo del *software* Microsoft di ogni suo cliente, gestire i contratti SPLA e generare i *report* mensili. Il *Core* riceve i dati grezzi e li aggrega e manipola allo scopo di renderli fruibili all'utente finale attraverso la *GUI*. Il *Core* si suddivide in ulteriori quattro componenti che elaborano i dati grezzi usando diverse modalità:

- *Data Reader*: legge in maniera schedulata i dati salvati nel *File System* riportati dal *service provider* tramite gli agenti installati e li inserisce nel *database*;
- *Info Mapper*: legge i dati inseriti nel *database* dal *Data Reader* e li elabora opportunamente trasformandoli in oggetti;
- *Knowledge Writer*: esegue il *merge* tra gli oggetti creati dall'*Info Mapper*, ottenendo così degli oggetti customizzati e mostrandoli come entità persistenti nel *database*;
- *Wisdom Creator*: contiene l'intelligenza in grado manipolare gli oggetti customizzati e ne calcola lo SPLA.

Service Provider

Nell'ambiente in cui opera il *service provider* è necessario che ci sia almeno un *server* in cui è installato il *software HypervisorScannerServices*, il quale interroga il *software* che gestisce l'infrastruttura virtuale e riceve una mappa che fornisce una visione ad alto livello delle macchine e del relativo stato. La mappa dell'infrastruttura permette di conoscere la disposizione delle macchine virtuali installate e rileva e riporta eventuali spostamenti che potrebbero essere significativi per il calcolo del costo di alcune particolari licenze. Sulle macchine virtuali e fisiche di ogni *service provider* è installato l'agente, il quale estrae informazioni specifiche sul *server*, per quanto concerne la componente *hardware*, e sui programmi Microsoft per quanto riguarda la componente *software*.

3.1.2 Analisi della componente Beryllium Data Collector

Il *Data Collector* è un servizio di Windows che in maniera ciclica raccoglie tutti i dati grezzi che possono essere significativi per determinare il *licensing* dei prodotti Microsoft supportati.

L'architettura ad alto livello del *Beryllium Data Collector* è così strutturata:

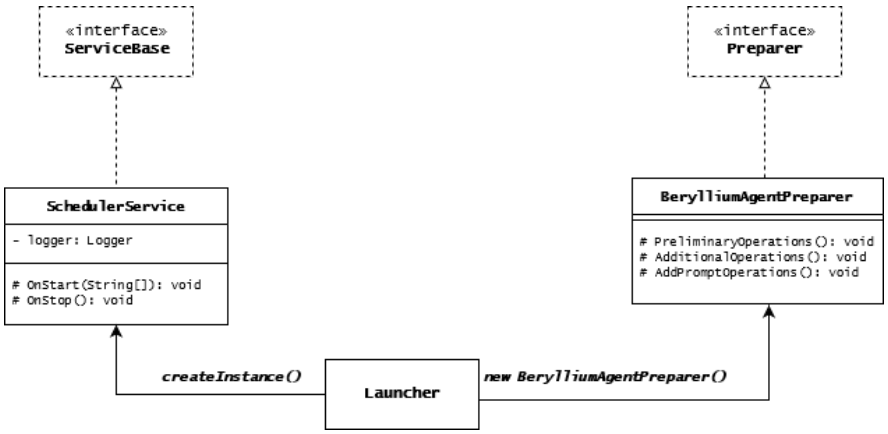

Fig. 3.2: Architettura Beryllium Data Collector

Ogni qualvolta che viene avviato un servizio Microsoft, viene eseguito il metodo *OnStart(string[] args)*, il quale crea un nuovo *thread* che invoca i metodi *ScanTask()* e *CheckForUpdateTask()*. In particolare:

- *ScanTask()*: esegue il controllo delle *features* da scansionare ciclando tutte le DLL e richiamando su ognuna di esse un metodo che verifica se la feature in questione è installata nell'host dove sta operando l'agente e in caso positivo si ne estrae le informazioni;
- *CheckForUpdateTask()*: si aggiorna se ci sono modifiche di versione L'agente interroga il *web service*$_{[G]}$ utilizzando una specifica *API*$_{[G]}$ a cui viene passato come parametro la versione attuale del *web service*. Finita la procedura che prevede la ricerca dei dati aggiornati nel *database*, le informazioni raccolte vengono aggregate in una cartella compressa. Tale cartella, che contiene i futuri *file* di configurazione del *Data Collector* e un eseguibile, costituisce il parametro di ritorno dell'API;
- *CheckForUpdateTask()* esegue il *file selfUpdate.exe*, che in caso di modifiche di versione, aggiorna i *file* vecchi.

L'agente è stato progettato per essere modulare in modo tale che, per supportare un nuovo prodotto Microsoft, sia sufficiente aggiungere una DLL in una *directory* dell'agente. Questa scelta progettuale presenta i seguenti principali vantaggi:

- Rispetto del *Single Responsibility Principle*$_{[G]}$: ogni feature ha un solo scopo e questo scopo è perseguito solamente da tale *features*;
- Rispetto dell' *Open-Close Principle*$_{[G]}$: l'aggiunta di una nuova feature non implica nessuna modifica al codice già esistente;

- Riuso di codice: le DLL contengono codice utile a più di una classe e possono essere chiamata da più classi;
- Efficienza: le DLL non necessitano di essere salvate in RAM poiché è possibile invocarle ed eseguirle a *run-time* solo quando necessario.

3.1.3 Analisi di una feature del Data Collector

Una *feature* del *Data Collector* si occupa di estrarre dati significativi dai prodotti Microsoft. L'architettura di una generica *feature* concreta è così strutturata:

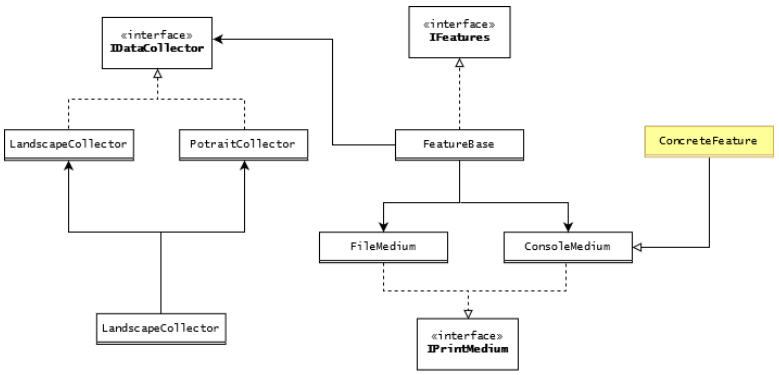

Fig. 3.3: Architettura di una generica feature del Data Collector

Ogni classe concreta derivata da *FeatureBase* implementa i metodi astratti della classe base e ha la possibilità di definire dei metodi aggiuntivi per progettazione e funzionalità della particolare *feature*. Sono presenti due modalità di raccolta dei dati che permettono di applicare un'adeguata disposizione formale ai dati raccolti. Per realizzare i due *collector* è stato utilizzato il *pattern* architetturale *Factory*$_{[G]}$ in modo da poter definire il comportamento generale e delegare alla specifica sottoclasse la costruzione del relativo oggetto. La scelta di utilizzare questo *pattern creazionale* è conseguente ad un'analisi del dominio applicativo, che nello specifico caso ha ritenuto conveniente preferire un approccio che persegue la massima astrazione, piuttosto che investire sull'estendibilità della classe base. Il *Factory* viola l'*Open-Close Principle*, ma in questa particolare situazione tale principio risulta poco significativo dato che è assai improbabile che ci sia la necessità di aggiungere una nuova classe concreta che implementa un terzo tipo di raccolta. [10]

Descrizione delle classi

Nel seguente elenco è presente una breve descrizione delle classi coinvolte nell'esecuzione di una *feature*:

- *IDataCollector*: interfaccia contenente i metodi basilari per una generica struttura di raccolta dei dati;
- *LandscapeCollector*: classe che raccoglie i dati in forma tabulare;
- *PotraitCollector*: classe che raccoglie i dati in forma *key-definition*;
- *DataCollectorFactory*: classe che fornisce l'istanza corretta del data collector a seconda la forma in cui i dati devono essere raccolti;
- *IFeatures*: interfaccia contenente i metodi basilari che verranno implementati dalle varie *features*;
- *FeatureBase*: classe astratta contenente i metodi comuni a tutte le *features*;
- *ConcreteFeature*: classe concreta che legge e raccoglie i dati richiesti da una *feature*;
- *IPrintMedium*: interfaccia contenente i metodi basilari di stampa da implementare nelle sottoclassi;
- *FileMedium*: classe che stampa i dati su *file*;
- *ConsoleMedium*: classe che stampa i dati su console.

3.1.4 Analisi del Licensing Microsoft Dynamics CRM in SPLA

I *service provider* che offrono alle aziende *software* Microsoft sono soggetti a un particolare programma di *licensing* chiamato Service Provider Level Agreement, che prevede che ogni *service provider* paghi a Microsoft un corrispettivo mensile che varia in base all'utilizzo effettivo del *software* da parte dei vari clienti. *Microsoft Dynamics* CRM è un *software* soggetto a SPLA utilizzato come strumento di gestione delle relazioni con i clienti e contribuisce all' automazione delle attività attraverso l'ottimizzazione delle vendite, del marketing e dell' organizzazione di servizi. Prendendo in considerazione il caso d'interesse, *Microsoft Dynamics* CRM 2016, essendo soggetto al programma SPLA, fa riferimento allo SPUR (Service Provider Usage Rights), il documento che definisce come un *service provider* può utilizzare i prodotti Microsoft. Lo SPUR a cui fare riferimento può variare a seconda se il cliente, al momento dell'adozione del *software Microsoft Dynamics* CRM, possiede già una licenza Microsoft. In particolare, si considera lo SPUR 2016 se il cliente ha già acquistato precedentemente una licenza compatibile, mentre si fa rifermento allo SPUR correntemente in vigore (nel nostro caso quello del 2018) se il cliente non possiede alcuna licenza al momento dell'adozione di Microsoft. [11]

Una delle opzioni di *licensing* Microsoft riguardante *software* soggetti ad un modello a sottoscrizione è la SAL (*Subscriber Access Licens*). Al variare dell'utilizzo del *software Microsoft Dynamics* CRM vengono applicate diverse opzioni di SAL, mentre per quanto riguarda l'accesso di utenti esterni tramite qualsiasi applicazione o GUI, il cliente non necessita di alcuna SAL. Nella succesiva tabella vengono indicate le tipologie di SAL a cui sono soggetti gli utenti. [12]

Essential SAL	Permette l'accesso al server per un utilizzo essenziale
Basic SAL	Permette l'accesso al server per un utilizzo di base
Professional SAL	Permette di installare e utilizzare Unified Service Desk (USD). Il permesso di utilizzare USD è limitato agli utenti a cui sono state assegnate le SALs.

Tab. 3.1: Riepilogo tipologie Subscriber Access License (SAL)

3.1.5 Requisiti

Durante l'attività di Analisi sono stati formalizzati i requisiti. In pratica, è stata redatta una lista dettagliata e completa di tutte le caratteristiche che dovrà soddisfare il prodotto *software*. Allo scopo di ottenere un prodotto di qualità è necessaria una rigorosa definizione dei requisiti dello stesso, in modo da avere una panoramica delle funzionalità e determinare i vincoli di qualità a cui il prodotto deve sottostare.
I requisiti si suddividono nelle seguenti 4 tipologie:

- Funzionali: descrivono una specifica funzionalità che il prodotto deve offrire;
- Di vincolo: indicano delle limitazioni a cui il prodotto deve sottostare;
- Qualitativi: indicano gli elementi atti ad aumentare la qualità del prodotto finale;
- Prestazionali: definiscono le performance che il prodotto deve raggiungere.

I requisiti sono stati classificati seguendo la seguente codifica:

ID Requisito: R {Categoria} {Priorità} _ {Codice}

- Categoria: indica la tipologia del requisito:
  - F: Funzionale;
  - Q: Qualitativo;
  - P: Prestazionale;
  - V: di Vincolo.

- Priorità: ogni requisito può essere:
  - Ob: obbligatorio;
  - De: desiderabile;
  - Fa: facoltativo;
  - Codice: rappresenta univocamente un requisito all'interno di una categoria.

- Codice: rappresenta un numero progressivo che permette di identificare univocamente un requisito all'interno di una categoria.

ID	Descrizione
RFOb_0	Identificare la versione di *Microsoft Dynamics* CRM installata
RFOb_1	Estrarre informazioni solo se la versione di *Microsoft Dynamics* CRM è la 2016
RFOb_2	Identificare le operazioni effettuate da ogni utente in una specifica data
RFOb_3	Distinguere le 3 tipologie di utenti (Essential, Basic, Professional)
RFOb_4	Individuare la modalità di scrittura corretta nel file CSV
RFOb_5	Analizzare i dati di *audit* solo se la modalità è attiva
RFOb_6	Stampare i dati su file CSV
RVOb_0	Essere scritta in linguaggio C#
RVOb_1	Essere compilabile con .NET framework 4
RVOb_2	Eventuali librerie esterne utilizzate devono essere compatibili con .NET framework
RVOb_3	Eventuali librerie esterne utilizzate devono essere compatibili con .NET framework
RVDe_0	La soluzione non deve richiedere l'installazione di *tool* aggiuntivi sul server del cliente
RVDe_1	La soluzione deve essere retrocompatibile con la maggior parte diversioni di *Microsoft Dynamics* CRM

Tab. 3.2: Tabella riassuntiva dei requisiti individuati

3.2 Progettazione

Nel corso dell'attività di Progettazione, viene definito il modo in cui i requisiti individuati saranno soddisfatti. Un'attenta attività di Progettazione prevede la definizione precisa di tutto ciò che deve essere implementato con la codifica in modo tale che, con quest'ultima attività, ci si possa focalizzare sulla scrittura del codice. In un primo momento è stata definita la progettazione architetturale, che definisce la struttura macroscopica del sistema con i suoi moduli e le relazioni tra di essi. Successivamente è stata costruita la progettazione di dettaglio che definisce ogni particolare delle componenti.

Dopo uno studio dettagliato di *Microsoft Dynamics* CRM e delle sue molteplici funzionalità, sono state individuate diverse possibilità progettuali tenendo conto di:

- Vincoli di compatibilità teconologica e di versione;
- Retrocompatibilità;
- Carico di autoformazione previsto.

Dovendo progettare una DLL, il cui collegamento con l'eseguibile avviene durante l'esecuzione tramite una specifica API, che accetta in input il nome della libreria e carica all'interno dello spazio di memoria dell'applicazione la DLL sviluppata, le soluzioni prese in considerazione, si basano sulla necessità di sviluppare la libreria in C#, come specificato nel RVOb_0.

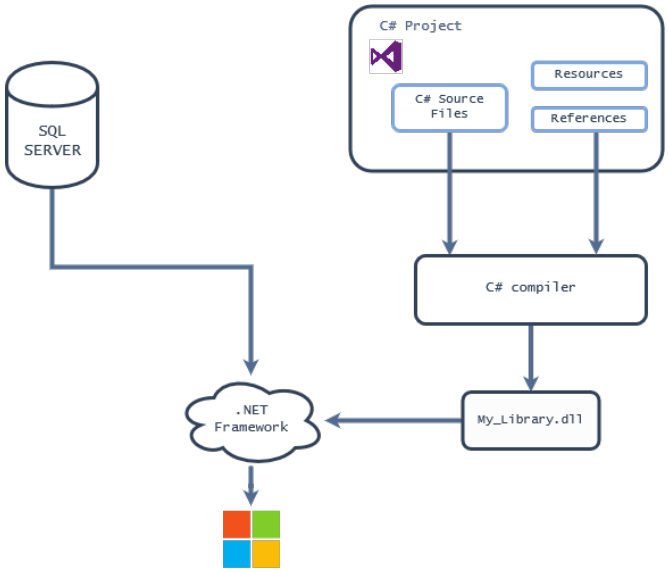

Fig. 3.4: Funzionamento dll in un progetto C#

3.2.1 Script in PowerShell

La prima soluzione individuata prevede l'implementazione di uno *script.cs1* da integrare al codice C# che viene eseguito ogni qualvolta che viene invocata la funzione che si occupa di raccogliere i dati grezzi. Andando a sviluppare una DLL che necessita di estrarre dati e identificare versioni è risulta più semplice usare i comandi di PowerShell rispetto al tentativo di collegarsi a tutte le classi .NET. Lo script in questione necessita dell'importazione del modulo `Microsoft.Crm.PowerShell` con lo scopo di poter utilizzare il comando `Get-CrmAccessLicense`. Il comando citato restituisce quantità e tipologia di licenze SAL assegnate agli utenti del sistema. Purtroppo questo comando non fornisce dati certi sull' utilizzo del *software* e delle *features*. A confermare questo dubbio sono stati eseguiti dei test da interfaccia grafica i cui risultati hanno confermato che l'assegnazione di un particolare tipo di SAL non influenza e non limita le funzionalità del prodotto. Pertanto questa soluzione non fornisce dati attendibili sull'utilizzo effettivo del *software* e il suo impiego come unica tecnologia per la raccolta dei dati va escluso. Tuttavia le informazioni restituite dal comando `Get-CrmAccessLicense` si possono rivelare utili a posteriori, dopo aver raccolto ed elaborato i dati di utilizzo effettivi, per confrontare i risultati ottenuti dalle due soluzioni e segnalare eventuali incongruenze.

3.2.2 Altre valutazioni progettuali

Dopo aver scartato l'utilizzo esclusivo di PowerShell, sono state individuate e valutate ulteriori possibilità progettuali.

Microsoft Dynamics CRM 2016 SDK

Microsoft mette a disposizione degli sviluppatori di *Microsoft Dynamics* CRM 2016 delle $SDK_{[G]}$, ovvero un insieme di strumenti per lo sviluppo e la documentazione del *software* disponibili come pacchetti $NuGet_{[G]}$ che è possibile utilizzare nei progetti di *Visual Studio*. In particolare è stata valutata la possibilità di utilizzare il *package assembly* `Microsoft.Xrm.Tooling.Connector`, in quanto offre classi e interfacce che semplificano l'estrazione di dati dal *software*. Tuttavia il *package* non rispetta il requisito di vincolo RVOb_1, poichè necessita di essere compilato con una versione di .NET uguale o superiore alla versione 4.6.2.

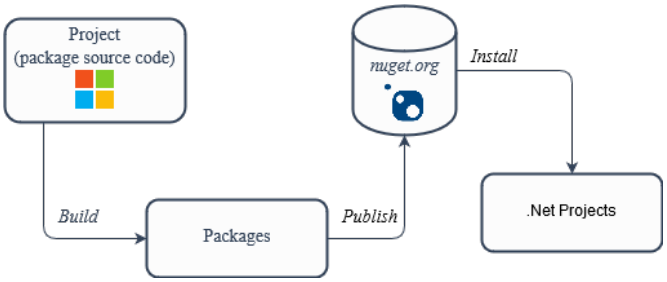

Fig. 3.5: Funzionamento di NuGet

SQL Server Reporting Services

SQL Server Reporting Services (SSRS) è un generatore di *report* per Microsoft e fa parte della suite di servizi di Microsoft SQL Server. Il vantaggio di una scelta progettuale basata su questo sull'utilizzo di tale *software* presenta il vantaggio di non aver bisogno di far installare all'utente nessuna applicazione aggiuntiva, dato che SSRS è un pre-requisito per l'installazione di *Microsoft Dynamics* CRM. Inoltre il *report* prodotto è interpretabile immediatamente da GUI. Nonostante i vantaggi sopra elencati, la soluzione è stata scartata poiché, richiedendo una versione uguale o superiore a .NET 4.5, viola il requisito di vincolo RVOb_1.

3.2.3 Scelta finale: SQL

Considerando rischi e benefici di ogni tecnologia, è emerso che estrarre i dati utilizzando procedure SQL risulta vantaggioso dal punto di vista implementativo e necessario rispetto ai vincoli tecnologici. La soluzione individuata quindi prevede la connessione al *database* in cui sono salvati i dati relativi a *Microsoft Dynamics* CRM e l'estrazione di metadati utili al *licensing* tramite *query* SQL, contenute negli appropriati metodi e classi della DLL implementata. [13]

3.2.4 Definizione di prodotto

La scelta progettuale individuata prevede la costruzione di due classi: *DynamicsCRM* e *LockPicking*. In questo modo, è possibile separare la componente che effettua la connessione al *database* ed estrae le informazioni utili, dalla componente che implementa l'interfaccia *IFeature*. Questa decisione consente di mantenere la struttura comune delle *features* concrete già esistenti e delegare ad un'altra classe l'estrazione dei dati dal database. Inoltre, alla classe concreta *DynamicsCRM*, viene integrato lo script *AssignedSALs*, il quale individua le SAL attualmente assegnate agli utenti e quindi fornisce i dati che verranno poi aggiunti a quelli raccolti dalla classe *LockPicking*. In una fase successiva, non di competenza della studentessa, tali informazioni verranno elaborate dal core dell'applicativo, che determinerà se le SAL rilevate dallo script Powershell soddisfano quelle che corrispondono all'uso effettivo del *software*.

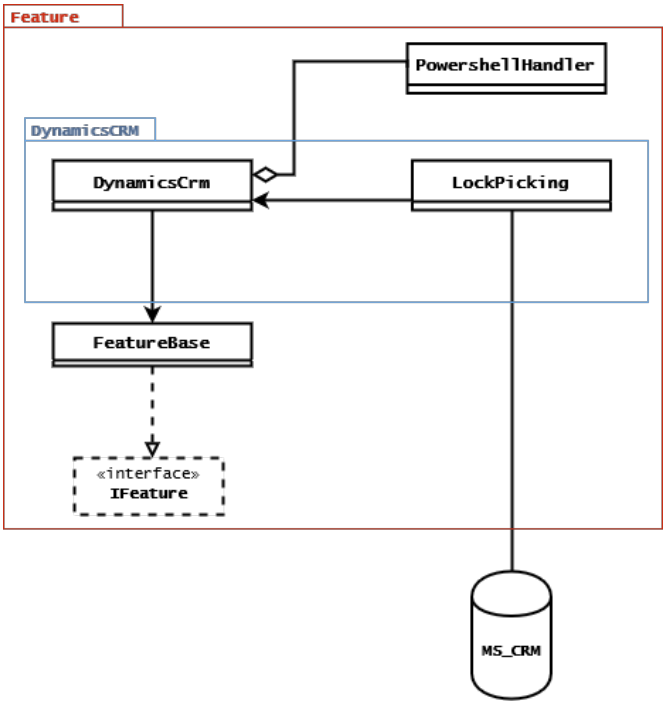

Fig. 3.6: Diagramma delle classi della soluzione DynamicsCrm

La classe *LockPicking* è stata progettata per aprire e chiudere la connessione al *database* in cui sono salvati i dati di audit del *software Microsoft Dynamics* CRM. Inoltre *LockPicking* si occupa di interrogare il *database* ed aggiungere i risultati ottenuti dalle *query* al riferimento ad un oggetto di tipo *DataCollector*, apposita classe dell'applicativo che permette di raccogliere dati in vari formati, passato dal costruttore della classe *DynamicsCrm* al momento dell'instanziazione di un oggetto *LockPicking*.

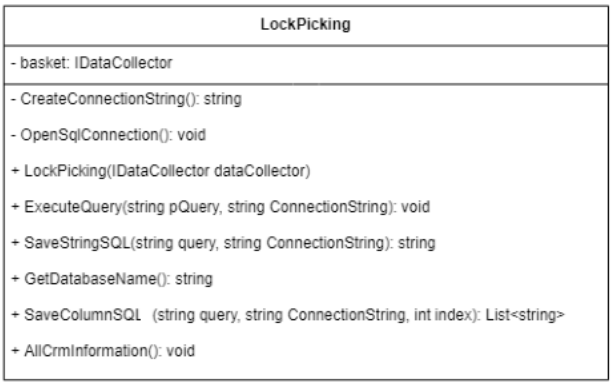

Fig. 3.7: Metodi e campi della classe LockPicking

Di seguito viene riportata una descrizione sintetica di tutti i metodi appartenenti a *LockPicking*:

- private readonly IDataCollector basket: data collector che raccoglie tutte le informazioni di Microsof Dynamics CRM;
- *private static string CreateConnectionString()*: metodo che crea la *connection string* per accedere al *database* MS_CRM;
- *private static void OpenSqlConnection()*: metodo che crea la connessione con il *database*;
- *public LockPicking(IDataCollector dataCollector)*: costruttore ad un parametro che istanzia un oggetto *LockPicking* con il data collector opportuno;
- *public static void ExecuteQuery(string pQuery, string ConnectionString)*: metodo che esegue una generica *query*;
- *public static string SaveStringSQL(string query, string ConnectionString)*: metodo che ritorna un singolo valore di una *query*;
- *public static string GetDatabaseName()*: metodo che restituisce il nome del *database* nel quale è installato *Dynamics* CRM;
- *public static List<string> SaveColumnSQL(string query, string ConnectionString, int index)*: metodo che ritorna una lista di stringhe che rappresentano una colonna della tabella risultato della *query*;
- *public void AllCrmInformation()*: metodo che raccoglie dati utili relativi alla *feature*.

La classe *DynamicsCrm* implementa l'interfaccia *IFeature* ed oltre all'override dei metodi della classe madre possiede il metodo proprio *CheckSupportedVersion()* che serve a stabilire se la versione di *Microsoft Dynamics* CRM installata è compatibile con le versioni supportate dalla soluzione. Inoltre tale classe, tramite l'override del metodo *Gather()*, manda in esecuzione lo script Powershell e aggiunge all'oggetto *DataCollector* i risultati ottenuti dallo script in questione. Di seguito viene riportata una descrizione sintetica di tutti i metodi e i campi appartenenti a *DynamicsCrm*:

- private readonly LockPicking lock_picking; reference alla classe LockPicking;
- private string possibleDynamicsCRMRegistryPath: possibile percorsi nella chiave di registro per le varie versioni di Dynamics CRM;
- private static readonly string PS_RESOURCE: contiene lo script Powershell da eseguire;

- *public DynamicsCRM()*: costruttore che definisce le informazioni di base della *feature*;

- *public DynamicsCRM(string pathToOutputFolder)*: costruttore completo che crea un'istanza di LockPicking;

- *protected bool CheckSupportedVersion(string path, RegistryView view)*: metodo che verifica che la versione installata di Microsoft Dynamics CRM sia supportata dall'agente;

- *protected void OnStepCompleted(FeatureEventArgs e)*: metodo che aggiorna il messaggio sopra la barra di avanzamento con la scritta specificata;

- *protected void OnMessageFired(FeatureEventArgs e)*: metodo che aggiorna il messaggio sopra la barra di avanzamento con la scritta specificata e si occupa dell'avanzamento di uno step della barra;

- *public bool IsEnabled()*: metodo verifica se *Dynamics CRM* è installato sul *server Beryllium*;

- *public void Gather()*: metodo che raccoglie le informazioni.

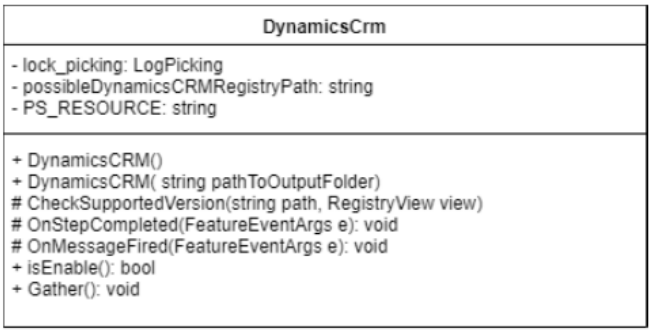

Fig. 3.8: Metodi e campi della classe DynamicsCrm

Nella seguente figura viene riportato il diagramma di sequenza che descrive le relazioni che intercorrono tra le entità del sistema nel momento in cui, dopo aver verificato che il *software Microsoft Dynamics* CRM è installato sulla macchina corrente, viene avviata la procedura di estrazione dei dati al fine di produrre il documento in formato *CSV*$_{[G]}$ con il *report* ottenuto.

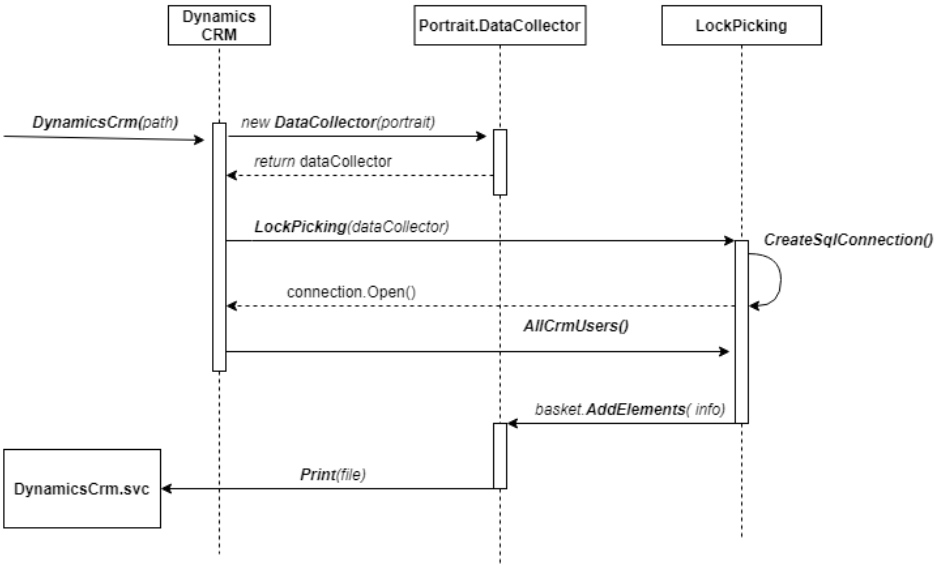

Fig. 3.9: Diagramma di sequenza della soluzione

3.3 Codifica

Prima di procedere alla descrizione dell'attività di codifica, è necessario precisare che il prototipo richiesto ha come scopo l'esclusiva verifica della fattibilità della feature. È stato volutamente tralasciato l'aspetto legislativo che regolamenta l'autorizzazione e l'utilizzo di dati degli utenti con la conseguente ripercussioni della privacy normata a livello europeo dalla $GDPR_{[G]}$. Grazie alla progettazione dettagliata e alle conoscenze pregresse di programmazione ad oggetti in Java e del linguaggio SQL, la scrittura del codice delle classi *LockPicking* e *DynamicsCrm* non è risultata particolarmente complessa. Al contrario ha richiesto più tempo del previsto implementare l'accesso e il recupero dei dati da Registro di sistema, in quanto era necessario conoscere approfonditamente il comportamento di API .NET poco documentate.
Per quanto riguarda lo sviluppo dello script in Powershell, pur non avendo esperienze pregresse con tale linguaggio, l'attività di codifica è risultata piuttosto agevole rispetto a quanto preventivato, dato che dispone di un set di comandi (*cmdlet*) progettati per gestire oggetti e quindi concettualmente più affini alle tecniche di programmazione affrontate durante il percorso universitario. Tuttavia, pur essendo avendo una sintassi intuitiva, Powershell non fornisce strumenti di segnalazione degli errori di battitura e ciò ha reso impegnativa l'attività di debugging sullo script.
L'utilizzo dell'ambiente di sviluppo *Visual Studio* 2016 ha semplificato significativamente la procedura di configurazione e integrazione della soluzione *DynamicsCrm* al progetto già esistente. Inoltre l'IDE suggerisce automaticamente i nomi dei metodi e i tipi dei parametri richiesti relativi ai *namespace* esterni utilizzati, semplificando la scrittura del codice e riducendo

gli errori sintattici.

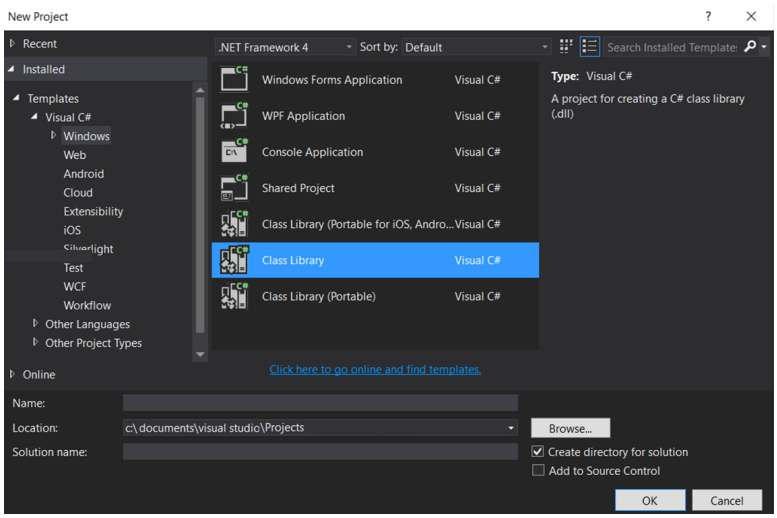

Fig. 3.10: Screenshot della finestra di creazione di un progetto DLL in Visual Studio 2016

Di seguito viene riportata la struttura della directory della *feature* sviluppata:

```
DynamicsCrm.dll
├── References
├── Scripts
│   └── AssignedSALs.ps1
├── DynamicsCrm.csproj
├── LockPicking.cs
└── DynamicsCrm.cs
```

3.4 Documentazione

L'azienda ha richiesto inoltre la produzione dei documenti di Analisi dei Requisiti e Specifica Tecnica, i quali sono stati redatti rispettivamente durante le fasi di Analisi e Progettazione. Per la stesura della documentazione sono stati utilizzati dei *template* di documentazione interni su cui si basano tutti i documenti dell'azienda.

3.5 Test

3.5.1 Ambiente di test e raccolta dati

Come previsto dal piano di lavoro, durante la fase iniziale, è stato creato e configurato l'ambiente di test. In particolare è stata implementata una macchina virtuale con sistema operativo Windows, ospitata dal *server* dell'azienda, sulla quale è stato installato il *software Microsoft Dy-*

namics CRM 2016, dopo averne soddisfatto i dipendenze *software* richiesti. In accordo con la metodologia aziendale, il tutor mi ha fornito una struttura *Active Directory*$_{[G]}$ già implementata e destinata ai test. Successivamente, accedendo all'SQL Server della macchina virtuale attraverso l'account utente avente i permessi da amministratore, utilizzando l'interfaccia grafica di *Microsoft Dynamics* CRM 2016, ho assegnato ad ogni utente una tipologia di SAL diversa, in modo da avere tutte e le tipologie di licenza e dunque di coprire tutti i casi possibili. Quotidianamente, è stato eseguito l'accesso al prodotto utilizzando gli account dei tre utenti precedentemente individuati e per ognuno di essi sono state eseguite diverse operazioni sulle entità persistenti, in modo da avere una quantità di dati sufficienti per garantire l'attendibilità dei risultati.

3.5.2 Esecuzione dei test

Durante l'ultima settimana di stage sono state realizzate tre tipologie di test:

- Test di unità;
- Test di integrazione;
- Test di sistema.

I risultati dei test sono stati poi riportati in forma tabulare, per aumentarne la leggibilità e permetterne una rapida consultazione.

Test di unità

I test di unità hanno lo scopo di verificare che tutte le unità, ovvero le componenti composte da uno o più moduli elementari, funzionino correttamente.

ID test	Descrizione	Esito	Componenti
UT_1	Si verifica che avvenga la connessione al database	Superato	CreateConnectionString(): string; OpenSqlConnection(): void

Tab. 3.3: Esempio: Struttura della tabella per i Test di Unità

Test di Integrazione

Una volta svolti i test sulle unità, si può procedere con l'esecuzione dei test di integrazione, che combinano tali unità in componenti testandone la corretta integrazione. L'approccio *bottom-up*$_{[G]}$ utilizzato pervede la seguente strategia d'integrazione: si sviluppano e si integrano prima le parti con minore dipendenza funzionale e maggiore utilità. In questo modo, vengono prima testati i package più semplici e con meno dipendenze, per poi procedere con una base solida per i test successivi.

ID test	Descrizione	Esito	Componenti
IT_1	Si verifica che la corretta integrazione tra DynamicsCrm e FeaureBase	Superato	Features
IT_2	Si verifica che la corretta integrazione tra DynamicsCrm e LockPicking, controllando che i risultati dei possibili flussi di dati corrispondano a quelli attesi	Superato	Features.DynamicsCrm

Tab. 3.4: Esempio: Struttura della tabella per i Test di Integrazione

Test di sistema

I test di sistema hanno lo scopo di verificare la copertura dei requisiti definiti durante l'attività di analisi e sono utili per prepararsi al collaudo. In particolare questi test sanciscono una versione definitiva del *software* sviluppato.

ID test	Descrizione	Esito	Requisito
ST_1	Si verifica che venga individuata la versione attesa di Microsoft Dynamics CRM installata	Superato	RFOb_0
ST_2	Si verifica che vengano identificare le operazioni effettuate da ogni utente in una specifica data	Superato	RFOb_2

Tab. 3.5: Esempio: Struttura della tabella per i Test di Sistema

Per effettuare il collaudo finale, è stato preparato un ambiente dedicato sul *server* aziendale e una volta eseguiti tutti i test, l'agente è stato eseguito dal tutor aziendale.

DynamicsCRM.csv

File Modifica Visualizza Inserisci Formato Stili Foglio Dati Strumenti Finestra Aiuto

	A	B	C
1	**Key**	**Value**	
2	Version	8.0.0000.1088	
3	SAL(0).Type	Dynamics CRM 2016 Enterprise SAL	
4	SAL(0).Count	3	
5	SAL(1).Type	Dynamics CRM 2016 Professional SAL	
6	SAL(1).Count	0	
7	SAL(2).Type	Dynamics CRM 2016 Essential SAL	
8	SAL(2).Count	0	
9	SAL(3).Type	Dynamics CRM 2016 Administrative SAL	
10	SAL(3).Count	0	
11	SAL(4).Type	Dynamics CRM 2016 Basic SAL	
12	SAL(4).Count	0	
13	User(0).DomainName	LAB\mrossi	
14	User(0).ObjectTypeCode(0).Code	UserEntityUISettings	
15	User(0).ObjectTypeCode(0).MajorOperation	Update	
16	User(1).DomainName	LAB\lbianchi	
17	User(1).ObjectTypeCode(0).Code	Contact	
18	User(1).ObjectTypeCode(0).MajorOperation	Create	
19	User(1).ObjectTypeCode(1).Code	Product	
20	User(1).ObjectTypeCode(1).MajorOperation	Update	
21	User(1).ObjectTypeCode(2).Code	UserEntityUISettings	
22	User(1).ObjectTypeCode(2).MajorOperation	Update	
23	User(0).DomainName	LAB\gverdi	
24	User(0).ObjectTypeCode(0).Code	MarketList	
25	User(0).ObjectTypeCode(0).MajorOperation	Access	

Fig. 3.11: Output prodotto dall'esecuzione dell'agente in fase di collaudo

4
Valutazione retrospettiva

Il seguente capitolo espone il bilancio complessivo dell'esperienza di stage, valutando il raggiungimento degli obiettivi posti dal Piano di Lavoro e presentando le conoscenze acquisite durante questo periodo.

4.1 Conseguimento degli obiettivi

Gli obiettivi individuati durante la stesura del Piano di Lavoro sono stati ripartiti in tre tipologie e possono essere riassunti nel seguente modo:

- Obbligatori
  - Studio dell'intero sistema Beryllium;
  - Comprensione del mondo SAM, SPLA e del prodotti Microsoft di interesse;
  - Configurazione dell'ambiente di test per il prodotto Microsoft Dynamic CRM;
  - Progettazione, sviluppo e test di un prototipo della componente "Microsoft Dynamic CRM feature for Beryllium Data Collector" per l'ultima versione di Microsoft Dynamic CRM;
  - Documentazione di progetto.

- Desiderabili
  - Completa integrazione con la componente Beryllium Data Collector.

- Facoltativi

- Sviluppo e test della componente ”Microsoft Dynamic CRM feature for Beryllium Data Collector” per almeno una versione più vecchia di Microsoft Dynamic CRM;
- F02: Sviluppo e test della componente ”Microsoft Dynamic CRM feature for Beryllium Data Collector” per almeno due versioni più vecchie di Microsoft Dynamic CRM.

L'esecuzione dei test di unità e di integrazione, hanno permesso di verificare il corretto funzionamento di tutte le componenti del *software*, mentre i test di sistema sono stati necessari al fine di capire quali dei requisiti definiti durante l'analisi erano stati implementati. Grazie all'esecuzione e al tracciamento di tali requisiti è stato possibile determinare gli obiettivi raggiunti al termine dello stage.
Come si può evincere dal grafico successivo, sono stati raggiunti gli obiettivi obbligatori e desiderabili, mentre quelli facoltativi, per mancanza di tempo, non sono stati finalizzati.

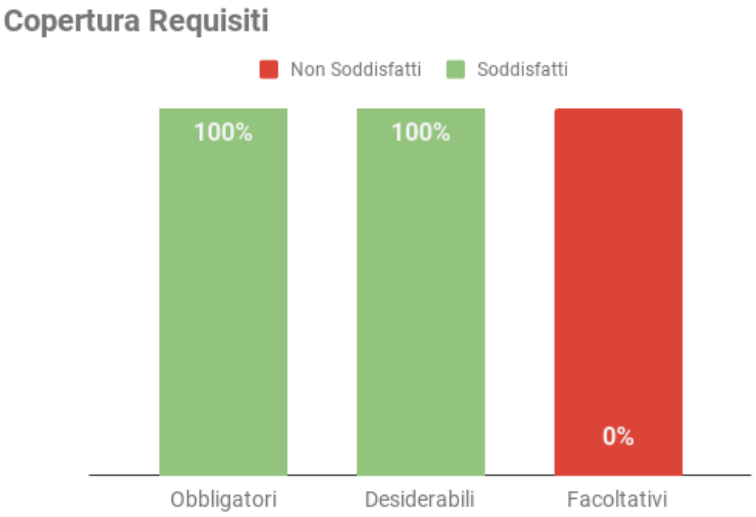

Fig. 4.1: Copertura dei requisiti raggiunta alla fine dello stage

4.2 Conoscenze preliminari

Il corso di laurea triennale in Informatica fornisce solide conoscenze di base di carattere tecnico e allo stesso tempo si occupa si far apprendere agli studenti i meccanismi fondamentali della materia e di stimolare flessibilità di pensiero e la capacità di *problem solving*. Personalmente, attraverso lo stage, ho avuto la possibilità di mettere in pratica le conoscenze e gli

strumenti ricevute dagli insegnamenti dei corsi di Ingegneria del Software, Programmazione ad Oggetti e Basi di Dati.
Dunque, per quanto concerne il processo di sviluppo del progetto a cui mi sono dedicata, le conoscenze preliminari sono risultate sufficientemente adeguate. Invece, rispetto alle esigenze del contesto aziendale in cui sono stata inserita, ho riscontrato delle difficoltà iniziali riguardo alla gestione dei sistemi informativi e della loro sicurezza, causate da una scarsa preparazione accademica di tali argomenti.

4.3 Conoscenze acquisite

Il processo di sviluppo del progetto di stage ha richiesto lo studio e la successiva applicazione di tecnologie mai affrontate in passato. Il prodotto *Microsoft Dynamics* CRM, sul quale si basava il progetto formativo, mi ha permesso di ottenere una visione consapevole riguardo al mondo dei *software* e del *licensing* Microsoft. Inoltre ho rafforzato le mie conoscenze sulla gestione di un ambiente Windows dal lato *server* e ho avuto la possibilità di apprendere il linguaggio C# e di sfruttare concretamente le potenzialità di PowerShell. In particolare ho conseguito le seguenti conoscenze:

- *Visual Studio 2016 Community*: è stato l'ambiente di sviluppo utilizzato per il progetto. Le diverse funzionalità offerte dall'IDE si sono rivelate particolarmente utili lavorando in ambiente Microsoft e la vasta documentazione fornita da $MSDN_{[G]}$ ha permesso di velocizzare in modo significativo il suo apprendimento;

- *Windows Server 2012 R2*: durante il processo di sviluppo è stato essenziale apprendere le funzionalità di base della gestione dei *server* Microsoft;

- C#: lo sviluppo della *feature* ha richiesto l'utilizzo di C# come linguaggio principale. Grazie alla ricca documentazione ufficiale la fase di codifica si è svolta agevolmente;

- *Powershell*: il linguaggio permette un accesso estremamente semplice ad una vasta quantità di funzionalità per interfacciarsi con i prodotti Microsoft e ciò ha permesso di estrarre dati utili in fase di test;

- *Active Directory*: imparare il funzionamento di questo framework si è rivelato fondamentale per l'utilizzo di *Microsoft Dynamics* CRM 2016, il quale si integrava con il dominio aziendale e perciò è stato necessario apprendere come gestirlo;

- *Microsoft Dynamics* CRM 2016: al fine di sviluppare la relativa *feature* è stato necessario installare e configurare sul *server* il prodotto. Inoltre, ho popolato le diverse tabelle con vari tipi di record, il che ha richiesto uno studio abbastanza approfondito del funzionamento di questa tecnologia;

- SAM e SPLA: durante il corso del progetto è stato necessario interagire più volte con il reparto SAM per apprendere le diverse regole di *licensing* dei prodotti Microsoft in SPLA, che mi ha permesso di ottenere una visione generale di questi concetti mai approfonditi durante il corso universitario.

Glossario

ACTIVE DIRECTORY
: A volte abbreviato in AD, è l'insieme di servizi di rete adottati dai sistemi operativi Microsoft e gestiti da un *domain controller*. Esso si fonda sui concetti di dominio e di directory , ovvero la modalità con cui vengono assegnate agli utenti tutte le risorse della rete attraverso account utente, account computer, cartelle condivise secondo l'assegnazione da parte dell'amministratore di sistema di *Group Policy*, ovvero criteri di gruppo.

API
: Abbreviazione di *Application Programming Interface*. Indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'*hardware* e il programmatore o tra *software* a basso e ad alto livello semplificando così il lavoro di codifica.

ASSET
: Nelle tecnologie dell'informazione coincidono con il complesso di *hardware*, *software* e *know-how* consolidato di proprietà dell'impresa.

AUDIT
: Attività volte a misurare la conformità di determinati sistemi, processi, prodotti a determinate caratteristiche richieste e a verificarne l'applicazione.

BEST PRACTICE
: Tradotto letteralmente dall'inglese con "miglior pratica". Si intendono,con questo termine, le esperienze di successo, o comunque quelle che hanno permesso di ottenere i risultati migliori, relativamente a svariati contesti.

BOTTOM-UP
: Tradotto letteralmente dall'inglese con "dal basso verso l'alto". Metodo di risoluzione di un problema tramite l'identificazione prima delle sue componenti base, che vengono poi suddivise in modo tale da realizzare un sistema più complesso.

CRM
: Abbreviazione di *Customer Relationship Management*, indica una strategia di business e di gestione dei processi, che attraverso il conseguimento dell'efficienza organizzativa permette di aumentare il fatturato aziendale garantendo al contempo un elevato livello di *customer satisfaction*, incentrando il business sul cliente.

CSV

Abbreviazione di *comma separated values*, è un formato di file basato su file di testo utilizzato per l'importazione ed esportazione di una tabella di dati.

DLL

Abbreviazione di *Dynamic-Link Library*, è una libreria software che viene caricata dinamicamente in fase di esecuzione, invece di essere collegata staticamente a un eseguibile in fase di compilazione.

FRAMEWORK

Nello sviluppo *software*, è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) su cui un *software* può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

GDPR

Abbreviazione di *General Data Protection Regulation*, è un regolamento dell'Unione europea in materia di trattamento dei dati personali e di privacy.

GUI Abbreviazione di *Graphical User Interface*, è un tipo di interfaccia utente che consente all'utente di interagire con la macchina controllando oggetti grafici convenzionali.

HELP DESK

Servizio destinato a fornire supporto, all'utente o al cliente, relativamente a prodotti o servizi informatici ed elettronici allo scopo di risolvere problemi o fornire indicazioni su prodotti come computer, apparecchiature elettroniche o *software*.

IIS

Abbreviazione di *Internet Information Services*, è un complesso di servizi server Internet per sistemi operativi Microsoft.

IT GOVERNANCE

Insieme di linee guida volte alla gestione dei rischi informatici e all'allineamento dei sistemi alle finalità dell'attività.

IT LIFE CICLE MANAGEMENT

Insieme di processi, procedure, competenze e strumenti utilizzati nella gestione dei *Configuration Item* che compongono il sistema informativo in tutte le fasi del loro ciclo di vita.

ITIL

Abbreviazione di *Information Technology Infrastructure Library*, è l'insieme di linee guida nella gestione dei servizi IT. Consiste in una serie di pubblicazioni che forniscono indicazioni sull'erogazione di servizi IT di qualità e sui processi e mezzi necessari a supportarli da parte di un'organizzazione.

ITSM
: Abbreviazione di *IT service management*, è la disciplina che si occupa di pianificare, progettare e gestire i sistemi IT di un'organizzazione.

MSDN
: Abbreviazione di *Microsoft Developer Network*, è la divisione di Microsoft incaricata di mantenere i rapporti con gli sviluppatori e gli amministratori di sistema. È responsabile di mantenere la maggior parte della documentazione riguardante i prodotti Microsoft.

NUGET
: Strumento di gestione dei pacchetti per le piattaforme di sviluppo Microsoft.

OPEN-CLOSE PRINCIPLE
: Secondo principio del modello di sviluppo SOLID, essoafferma che le entità (classi, moduli, funzioni, ecc.) software dovrebbero essere aperte all'estensione, ma chiuse alle modifiche; in maniera tale che un'entità possa permettere che il suo comportamento sia modificato senza alterare il suo codice sorgente.

SAM
: Abbreviazione di *software Asset Management*, si riferisce ad una pratica di business che comprende la gestione e l'ottimizzazione dell'acquisto, distribuzione, mantenimento, utilizzo e smaltimento di prodotti *software*. Il SAM è l'intera infrastruttura e l'insieme di processi necessari per l'efficacia della gestione, controllo e protezione degli asset *software* attraverso tutti gli stadi del loro ciclo di vita.

SDK
: Abbreviazione di *Software Development Kit*, indica un insieme di strumenti per lo sviluppo e la documentazione di *software*.

SISTEMA CLIENT/SERVER
: Architettura di rete nella quale genericamente un computer (client) si connette ad un server per la fruizione di un certo servizio appoggiandosi alla sottostante architettura protocollare.

SERVICE PROVIDER
: A volte abbreviato in SP, indica imprese che forniscono servizi di vario tipo e appartenenti a settori differenti.

SINGLE RESPONSABILITY PRINCIPLE
: Primo principio del modello di sviluppo SOLID, esso afferma che ogni elemento di un programma (classe, metodo, variabile) deve avere una sola responsabilità, e che tale responsabilità debba essere interamente incapsulata dall'elemento stesso. Tutti i servizi offerti dall'elemento dovrebbero essere strettamente allineati a tale responsabilità.

SPLA
: Abbreviazione di *Service Provider License Agreement*, indica una tipologia di licenze mirate alle organizzazioni che desiderano offrire servizi *software* ai loro clienti. La differenza tra una licenza SPLA e una normale è che mentre la seconda può essere utilizzata solamente da chi la ha acquistata, la prima permette al compratore di renderla disponibile a terzi. Questo permette ad alcune organizzazioni di offrire servizi basati su *software* Microsoft gestendo autonomamente le licenze, in maniera trasparente agli utilizzatori finali.

STAGE-IT
: Iniziativa che mira a mettere in contatto imprese e studenti. Offre un punto di contatto comune dove gli studenti possono scegliere tra diverse offerte di stage delle varie aziende e le aziende possono entrare in contatto con gli studenti in maniera diretta.

TICKET
: Richiesta di assistenza, tracciata e presa in carico da un servizio di assistenza tecnica.

WEB SERVICE
: Sistema *software* progettato per supportare l'interoperabilità tra diversi elaborati su di una medesima rete in un contesto distribuito. Questo permette di offrire diversi tipi di servizi tramite il *web*.

Bibliografia

[1] Wikipedia. *ITIL*. https://en.wikipedia.org/wiki/ITIL.

[2] NETCOM srl. *The Smartest Way To Manage Your Data*. https://www.netcom.it.

[3] MSDN Library. *C# Guide*. https://docs.microsoft.com/it-it/dotnet/csharp.

[4] W3Schools. *SQL*. https://www.w3schools.com/sql.

[5] MSDN Library. *Getting Started with Windows PowerShell*. https://docs.microsoft.com/it-it/powershell/scripting/getting-started/getting-started-with-windows-powershell?view=powershell-6.

[6] MSDN Library. *Get started with the .NET Framework*. https://docs.microsoft.com/it-it/dotnet/framework/get-started/index.

[7] MSDN Library. *Documentazione di Visual Studio*. https://docs.microsoft.com/it-it/visualstudio/ide/?view=vs-2017.

[8] MSDN Library. *SQL Server Management Studio*. https://docs.microsoft.com/it-it/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2016.

[9] MSDN Library. *VMWare Solution*. https://www.vmware.com/it/products.

[10] Ralph Johnson Richard Helm Erich Gamma, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[11] Microsoft Corporation. *SPLA Licensing Guide*. https://docs.microsoft.com/en-us/powershell/dynamics365/customer-engagement/overview?view=dynamics365ce-ps.

[12] Microsoft Corporation. *The Microsoft Dynamics 365 Licensing Guide*. https://mbs.microsoft.com/Files/public/365/Dynamics365EnterpriseEditionLicensingGuide.pdf.

[13] MSDN Library. *Overview of Dynamics 365 Customer Engagement PowerShell*. https://docs.microsoft.com/en-us/powershell/dynamics365/customer-engagement/overview?view=dynamics365ce-ps.