UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Tesi di Laurea

Progettazione e sviluppo plugin SAP Customer Checkout 2.0

Relatore
Prof. Loris Nanni

Laureando
Toffanin Nico
mat. 1164245

Anno Accademico
2021 - 2022

Data di laurea
14 Marzo 2022

Abstract

SAP è il leader dei sistemi ERP in tutto il mondo, negli ultimi trent'anni si è evoluta fino a diventare il colosso che è oggi. Un sistema ERP è un software gestionale che aiuta le aziende che lo utilizzano ad organizzare e gestire la propria impresa, semplificando la gestione della contabilità e tutto ciò che riguarda la produzione. Datalab srl è partner SAP dalla sua nascita e tutt'oggi sviluppa addon e plugin per integrare processi produttivi e aggiungere funzioni non presenti nativamente nel software a seconda delle richieste dei clienti.
In particolare vedremo come è stato sviluppato il plugin Java per SAP Customer Checkout 2.0, che è un software per la gestione dei punti cassa nei punti vendita di molte imprese in Italia e nel mondo.

Indice

Capitolo 1

Introduzione

L'obbiettivo di questa tesi è presentare il progetto sviluppato durante il tirocinio svolto presso Datalab srl, azienda che integra soluzioni nazionali e internazionali verticalizzate per lo sviluppo di soluzioni di intelligent ERP, per lo più basate su piattaforma SAP, ma non solo. Per permettere al lettore di comprendere meglio gli argomenti trattati è necessario fare una piccola digressione, spiegando cos'è SAP e quali sono i suoi prodotti.

Datalab srl è partner SAP e la maggior parte degli sviluppi che vengono prodotti sono plugin o add-on da integrare ai prodotti SAP, per aggiungere o modificare alcuni processi in base alle esigenze dei clienti. Lo scorso anno è stato rilasciata la versione 2.0 di Sap Customer Checkout, un software dedicato a gestire vendite e processi di cassa di un qualsiasi punto vendita. Lo scopo del tirocinio era realizzare un plugin in grado di integrare diverse funzioni richieste dai clienti, in quanto il plugin che era stato realizzato per la precedente versione non era più compatibile con la nuova struttura del software e necessitava comunque di nuove implementazioni.

Capitolo 2

SAP e i suoi prodotti

In questo capitolo conosceremo meglio SAP, produttore di software per la gestione dei processi aziendali tra i più importanti al mondo, sviluppa soluzioni che facilitano l'efficace elaborazione dei dati e il flusso di informazioni tra le organizzazioni.

2.1 Storia di SAP

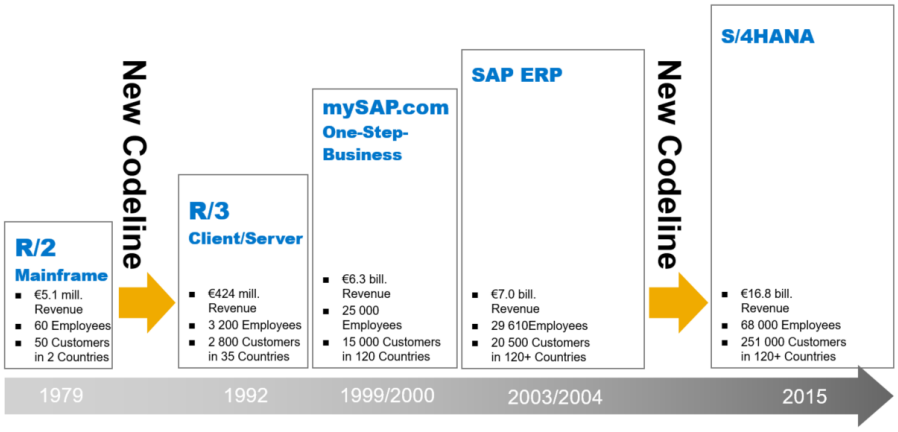

Figura 2.1: SAP evolution. [1]

Nel 1972 la società è stata fondata con il nome di System Analysis Program Development (Systemanalysis Programmentwicklung), successivamente abbandonato per la sigla SAP. Da allora la piccola impresa portata avanti da cinque persone è cresciuta fino a diventare una multinazionale con 105'000 dipendenti in tutto il mondo, con sede principale a Walldorf. L'idea nasce alla fine degli anni '60 quando 5 ingegneri tedeschi si conobbero collaborando come dipendenti di IBM. In quegli anni si utilizzavano ancora i mainframe a schede perforate, molto ingombranti e con una potenza di calcolo inferiore ad un moderno cellulare. Uno dei problemi principali era che, oltre ad essere estremamente costosi, necessitavano di personale specializzato e constringevano ogni impresa a programmare il proprio software gestionale. L'idea dei fondatori fu quella di creare un programma che potesse essere adattabile a diverse

imprese di vari settori con il minor numero di modifiche possibile, inoltre volevano che le oprazioni venissero eseguite in real time (a differenza dei sistemi esistenti all'epoca, quando era necessario eseguire più processi in maniera sequenziale per ottenere un risultato).
Nacque nel 1972 il loro primo prodotto, venne chiamato R/1 ("R" per real time), inizialmente però non potevano permettersi di acquistare un mainframe, quindi strinsero un accordo con un'altra società con sede vicino alla loro per poter utilizzare i loro computer durante la notte. Nel 1979 venne rilasciata la seconda versione del programma, R/2, e un paio di anni dopo SAP raggiunse il cliente numero 200. Una decina di anni dopo ci fù un punto di inflessione, quando venne introdotto SAP R/3 (figura 2.2), questa versione del programma integrò un'architettura a due livelli: client/server, dove il sistema centrale gestiva la base di dati e gli applicativi. Questa innovazione tecnica rese molto più economico il gestionale, che poteva essere utilizzato così da centinaia di utenti senza troppi problemi e così facendo SAP ha fissato quello che sarebbe stato lo standard nel mondo per il software ERP (Enterprise Resource Planning). Nel decennio 1990-2000 il fatturato di SAP aumentò rapidamente, così come il numero di dipendenti.

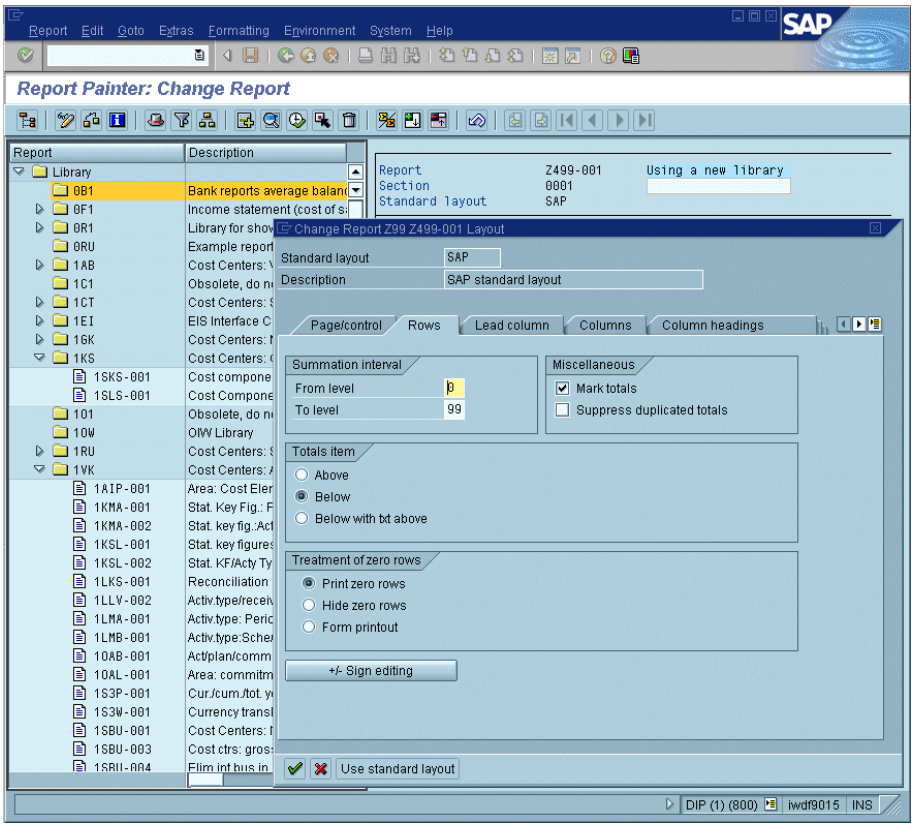

Figura 2.2: SAP R/3. [2]

In questa fase di crescita iniziale la strategia è stata semplice e efficace: sviluppare i propri software ascoltando i suggerimenti e le necessità dei clienti senza pensare troppo alla parte tecnica (database, sistema operativo, customizzazione, middleware, etc.). L'evoluzione successiva del gestionale è stata l'introduzione di SAP ERP nel 2003, basato su piattaforma middleware chiamata "NetWeaver", introducendo così la possibiltà di collegarsi con vari sistemi, anche esterni a SAP, permettendo lo

scambio di dati e l'integrazione dei processi. Sia SAP R/3 che ERP hanno una struttura modulare, dove ogni modulo gestisce una funzione aziendale differente ed è integrato con gli altri moduli aggiornando le informazioni in tempo reale.
Negli ultimi anni SAP ha ampliato la propria offerta, grazie anche all'acquisto di BusinessOne (software di una società israeliana venduto con il nome di "Menahel"), tutt'oggi uno dei prodotti di punta dell'azienda. Altre acquisizione importanti hanno contribuito all'espansione di SAP in nuovi mercati, ma l'evoluzione finale è stata passare ad offrire servizi ERP cloud, utilizzando il nuovo sistema di gestione di basi di dati HANA (sistema proprietario sviluppato da SAP stesso).

2.2 Prodotti SAP

Datalab srl come partner fornisce assistenza e sviluppi evolutivi di qualsiasi tipo sui principali prodotti SAP: BusinessOne, SAP ByDesign, SAP S/4HANA e SAP Customer Checkout in ambito retail. Li analizziamo uno ad uno nelle prossime sezioni per capire cosa sono, come funzionano e come si sviluppano gli add-on per soddisfare le necessità particolari dei clienti non implementate nativamente da questi software.

2.2.1 SAP BusinessOne

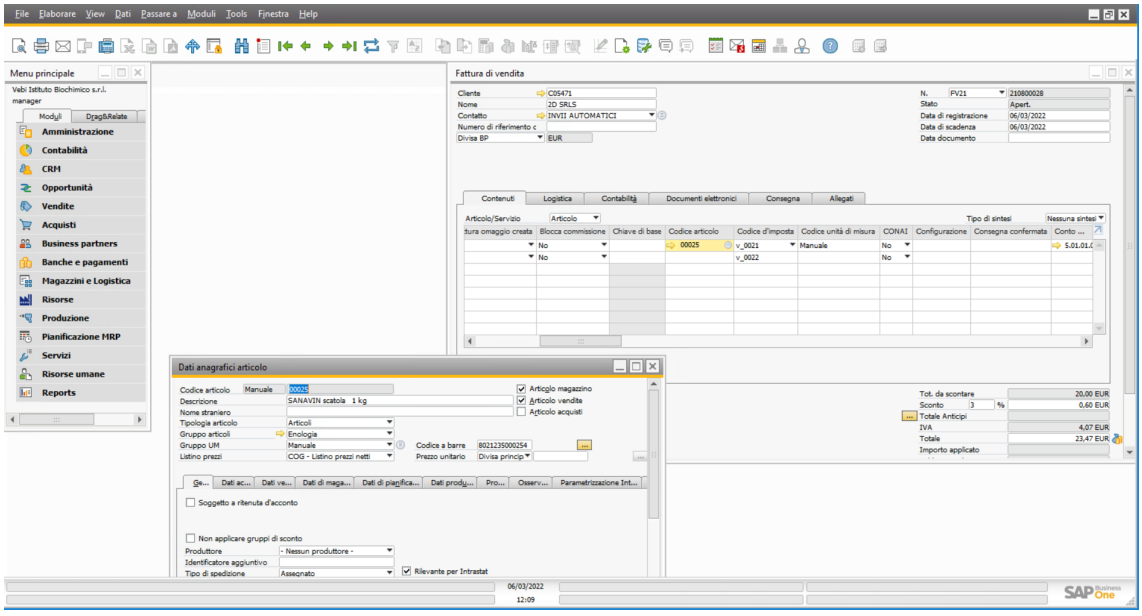

Figura 2.3: SAP R/3. [3]

SAP BusinessOne è un ERP, un software in grado di gestire tutti i processi business principali di un'azienda, è progettato per le piccole e medie imprese e consente di avere un'idea precisa dell'andamento delle attività in modo da poter progettare le azione future sulla base dei dati. La carattaristica

principale di questo ERP è la struttura modulare, che consente di gestire i processi aziendali come realtà a se stanti, ma allo stesso tempo legate tra loro perchè i collegamenti tra una fase e la successiva sono automatizzati, aumentando così la loro velocità. Di seguito una breve descrizione dei moduli:

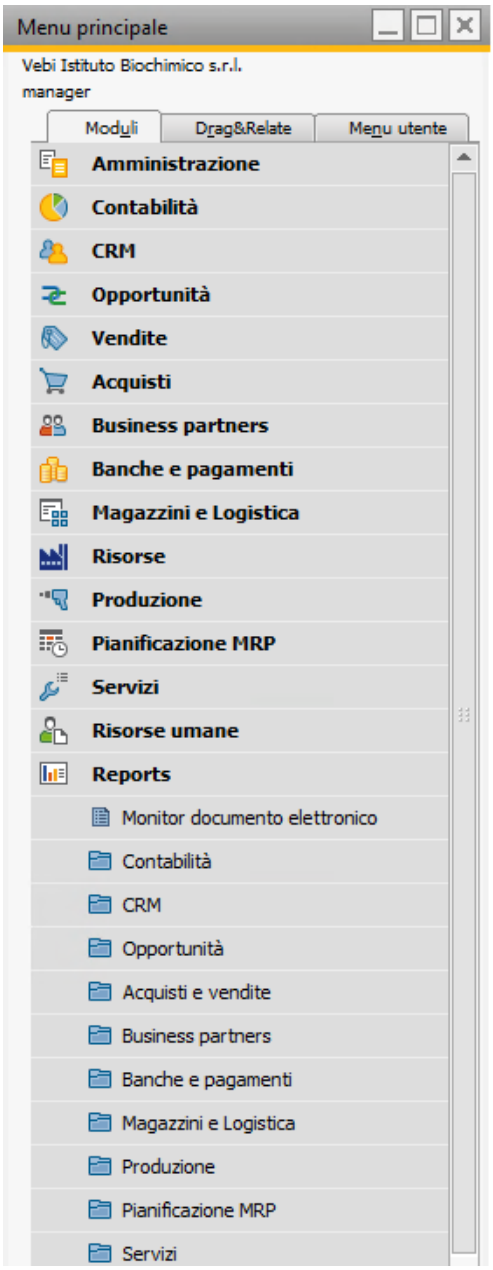

Figura 2.4: Moduli Business One.

- Gestione finanziaria: Modulo dedicato alle principali attività contabili come prime note ricorsive e calcolo delle imposte, la gestione è automatizzata.
- Vendite e clienti: Vengono memorizzati tutti i documenti di vendita in un unico luogo, come fatture per i clienti, note di credito clienti, consegne e molto altro. I dati più importanti sono sempre in evidenza in modo da comprendere al meglio le esigenze di ogni cliente.
- Acquisiti e controllo inventario: Fornisce il controllo completo del processo di approvigionamento, comprende documenti come fatture di acquisto da fornitore, note di credito fornitore, consegne e molto altro. Consente anche di comparare i vari fornitori permettendo di siglare accordi migliori.
- Piano di produzione: Permette di monitorare in tempo reale le scorte, in modo da scegliere il miglior piano di produzione e gestire più facilmente prezzi e offerte speciali.
- Business intelligence: Integrato con Office 365, consente di esportare e gestire report di molti formati e avere il controllo completo sull'accesso alle informazioni.
- Analisi e reporting: Mette a disposizione degli utenti tutte le informazioni critiche per prendere le decisioni più importanti.

Datalab srl si occupa di fornire assistenza su questo prodotto, ma soprattuto fornisce sviluppi evolutivi per integrare o modificare funzioni o processi a seconda delle richieste del cliente. Tramite lo sviluppo di add-on, generalemte utilizzando i linguaggi c# e Visual Basic fino a qualche anno fa. SAP mette a disposizione le "DI API" (Data Interface API) per comunicare con il database HANA o SQL in modo da poter manipolare i dati, sempre però facendo attenzione a mantenere tutti gli standard SAP: non è possibile eliminare o modificare record come meglio si crede, perchè essendo tutto collegato e automatizzato si rischia di provocare degli errori che potrebbero corrompere il database compromettendo l'operatività di tutto il sistema. Inoltre vengono messe a disposizione le "UI API" (User Interface API) per accedere ai controlli presenti nelle "form", come bottoni, celle di testo e qualsiasi cosa che riguardi l'interfaccia grafica di BusinessOne. Tramite l'utilizzo di queste API, il codice sviluppato è in grado di agganciarsi a degli eventi, prima e dopo l'esecuzione del codice sorgente, per modificare i dati o aggiungere flussi necessari all'operatività del cliente.

2.2.2 SAP ByDesign

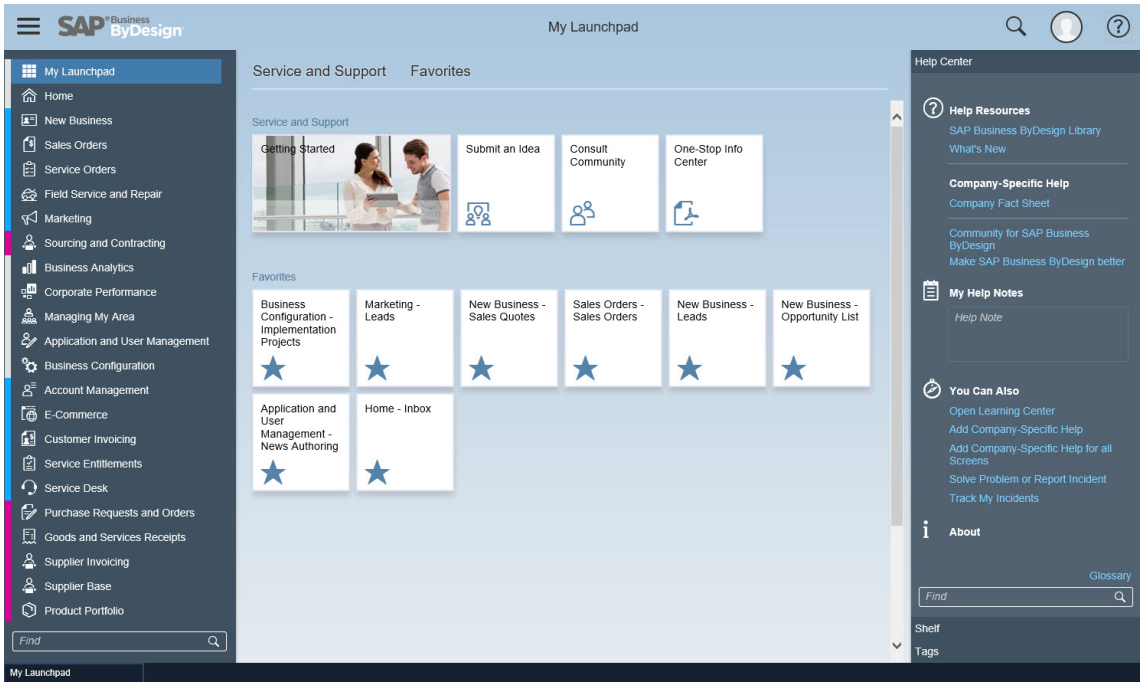

Figura 2.5: SAP R/3. [4]

SAP ByDesign è sempre un sistema ERP molto simile a BusinessOne, ma pensato per imprese medio-grandi in rapida espansione. La differenza sostanziale è che ByDesign è una soluzione on demand completamente adattabile alle necessità del singolo cliente, in quanto essendo una piattaforma 100% cloud, l'impresa non necessita di spese accessorie per tutta la struttura hardware che richiede BusinessOne ad esempio. Il cliente ha una quota mensile da pagare ed è in base ai singoli moduli di cui necessita, modificabili in qualsiasi momento in caso di crescita o espansione in

diversi ambiti. Il vantaggio di essere completamente su cloud esonera il cliente da spese infrastrutturali, non ha la necessità di acquistare server o hardware di qualsiasi tipo.
ByDesign come prima parte della sua configurazione richiede l'organigramma dell'azienda, per avere subito chiara la sua struttura e l'ambito in cui opera, in modo da poter ottimizzare al meglio tutti i processi aziendali, anche interni. I moduli di ByDesign sono molto simili a quelli di BusinessOne, con qualche variazione:

- Amministrazione, finanza e controllo: Fornisce una visione in tempo reale delle finanze, snellisce i processi di contabilità e dà la possibilità di gestire in maniera più efficare la liquidità
- CRM (Customer Relationship Managment): Sfrutta le informazioni personalizzate dei cliente per rendere più efficienti le campagne di marketing e capire su quali prodotti puntare.
- Risorse umane: Snellisce i processi HR relativi a gestione dell'organizzazione, amministrazione del personale con relativi orari, presenze, retribuzione e strumenti self-service per i dipendenti.
- Gestione progetti
- Approvigionamento
- Gestione della supply chain
- Funzionalità specifiche di settore: Questo modulo fornisce ulteriori strumenti specifici per determinati settori come ad esempio i servizi professionali, l'industria manifatturiera, la distribuzione all'ingorsso e molto altro.

In conclusione Bydesign è un software ERP molto simile a BusinessOne, ma dedicato ad imprese in rapida espansione essendo molto più adattabile alle nuove necessità, è un prodotto più completo che aiuta anche nella gestione interna dell'azienda (elemento non necessario per imprese un basso numero di dipendenti). ByDesign è un prodotto nuovo e nato da poco, da qualche mese Datalab srl ha ricevuto l'abilitazione al trattamento di questo prodotto.

2.2.3 SAP S4/HANA

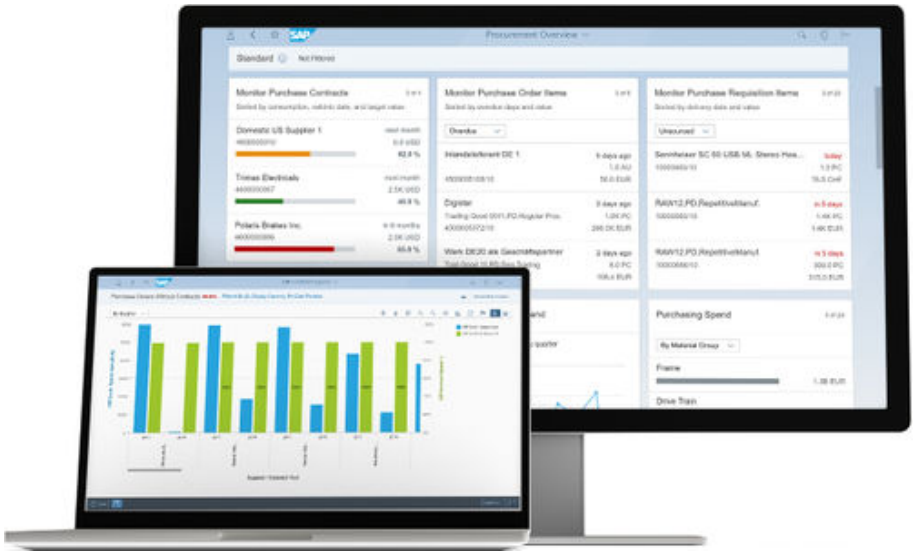

Figura 2.6: SAP S4/HANA. [5]

Differenze SAP ECC vs SAP S/4 HANA	
ECC	SAP S/4 HANA
Database Agnostic (MS SQL, Oracle, HANA ecc.)	Solo con database Hana
Anagrafiche separate Clienti e Fornitori	Uniche anagrafiche come Business Partner
Tabelle multiple e aggregate	Universal Journal (FI-CO unite) e riduzione delle tabelle di logistica
User Interface primaria: GUI	User Interface primaria: Fiori; secondaria GUI
Classico General Ledger o New GL (libro mastro)	New General Ledger
Material Ledger opzionale	Material Ledger esclusivamente
SD - elaborazione dello sconto	Gestione della Liquidazione e gestione condizione di sconto
Elemento di transazione commerciale estera	Transazione commerciale globale
WM (Warehouse Management)	EWM (Extended Warehouse Management)
Sviluppo sia in ABAP che in Java	S/4 HANA 1809 e precedenti release non supportano il dual stack
Leagucy Sistem Migration Work Bench (LSMW) come strumento per la migrazione	Leagucy Transfer Migration Cocktip (LTMC) come strumento per la migrazione
Lunghezza numero materiale 18 caratteri	Lunghezza numero materiale 18 o 40 caratteri

Figura 2.7: Differenze SAP ECC e SAP S/4 HANA.

SAP S/4HANA è la naturale evoluzione dei sistemi ERP presentati fino ad ora, le funzioni sono principalmente le stesse, le differenze essenziali sono due: la prima è che può essere interfacciata solo con database HANA (proprietario SAP), ottimizzato per avere maggiori velocità e più leggerezza, infatti il numero delle tabelle è stato diminuito e riorganizzato per essere più efficiente e meno caotico; La seconda è che l'applicazione è progettata per essere completamente in cloud, accessibile tramite interfaccia web da personal computer, smartphone o tablet a seconda delle esigenze. Utilizzando database HANA, SAP ha potuto semplificare notevolmente la struttura del proprio database, infatti nelle precendeti versioni sono presenti decine e decine di tabelle riguardanti la contabilità o le vendite, con anche rindondanza dei dati, ma erano tutte necessarie al funzionamento del programma. Abbandonando la compatibilità con gli altri database, SAP ha svolto un operazione che ha chiamato "SAP simplification" e consiste nell'accorpamento di tutte le informazioni relative ad un determinato oggetto in tabelle aggregate, adesso un informazione è memorizzata in un solo campo di una sola tabella, se poi questa informazione può essere calcolata non viene nemmeno memorizzata.

Per quanto riguarda il passaggio al cloud, SAP ha introdotto SAP Fiori, ovvero una nuova interfaccia completamente orientata al web e al mobile, che rivoluziona l'esperienza utente mantenendo l'efficacia della gestione delle attività aziendali a cui SAP ci ha abituato, in più ha solo il vantaggio di essere completamente cross-platform. SAP Fiori organizza la gestione dei processi in quattro categorie principali, viste come quattro tipologie di utente: manager, impiegato, rappresentante e venditore (figura 2.8). SAP inoltre definisce questo nuovo software "L'ERP intelligente", fra le motivazione che ci spingono a definirlo tale sono gli strumenti di Machine Learning integrati nel digital core e pronti ad essere utilizzati. Molti di questi strumenti sono in area finance e consentono di addestrare il sistema a fare operazioni ricorrenti, imparando dalle nostre operazioni e dalle correzioni applicate alle proposte che il sistema genera. Un'altra importante area dove sono presenti questi algoritmi è in CRM, riuscendo ad analizzare la domanda dei clienti e permettendo di fornire risposte efficaci. Un altro motivo per cui viene definito intelligente è il differente approccio all'operatività: un utente SAP ECC (ERP Central Component) delle precedenti versioni che si occupa dell'order desk, per capire quali sono i backorder su cui doveva lavorare, era costretto a lanciare un'apposita transazione, scegliere la variante giusta, visualizzare la lista degli ordini e capire quali necessitavano di un intervento; ora, grazie alla nuova User Experience basata su Fiori, l'operatore ha di fronte a se tutte le "tile", nelle quali sono immediatamente visibili KPI e indicatori di avvertimento, debitamente customizzato dà la possibiltà all'utente di avere una visione generale e capisce subito quale applicazione richiede la sua attenzione.

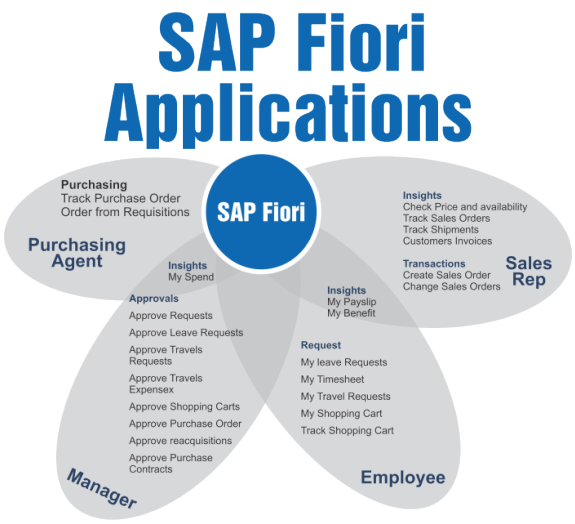

Figura 2.8: Differenze SAP ECC e SAP S/4 HANA.

2.2.4 SAP Customer Checkout

Facciamo una breve introduzione a SAP Customer Checkout, dato che lo vedremo più nello specifico nel capitolo successivo con la descrizione del plugin sviluppato per Datalb srl durante il periodo di tirocinio. SAP Customer Checkout è un'applicazione di POS (Point Of Sale) management, vendite in tempo reale e stock updates, in pratica fornisce tutti gli strumenti necessari per la vendita al dettaglio. Per le realtà aziendali nel mondo del retail è un sistema gestionale di cassa capace di auotomatizzare i punti vendita in tutti i suoi processi, inoltre si integra perfettamento con i sistemi ERP di SAP, sincornizzando in tempo reale tutte le vendite effettuate, gli articoli utilizzati in modo da aggiornare lo stato del magazzino, le fatture generate per i clienti con partita iva e molto altro.

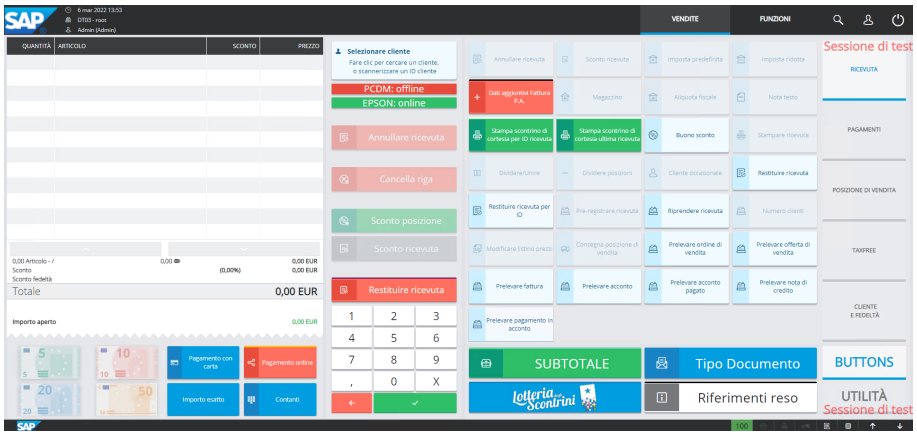

Figura 2.9: SAP S4/HANA. [6]

L'applicazione ha un interfaccia web con lo scopo di essere multipiattaforma e prefettamente adattabile alle necessità del cliente, infatti dispone anche della modalità "Table Service" per gestire i tavoli dei ristoranti e tutto ciò che riguarda questo settore.

Capitolo 3

Progettazione e sviluppo plugin per SAP Customer Checkout

SAP Customer Checkout, come tutti i prodotti SAP, prevede l'integrazione di plugins per modificare il comportamento del software in determinati casi, o implementare funzioni non disponibili nativamente. Il plugin viene sviluppato in Java e in questo capitolo vedremo cosa è cambiato da SAP Customer Checkout versione 1.0 alla versione 2.0 e quali funzioni ho avuto la possibilità di sviluppare durante il periodo del tirocinio.

3.1 SAP Customer Checkout 1.0

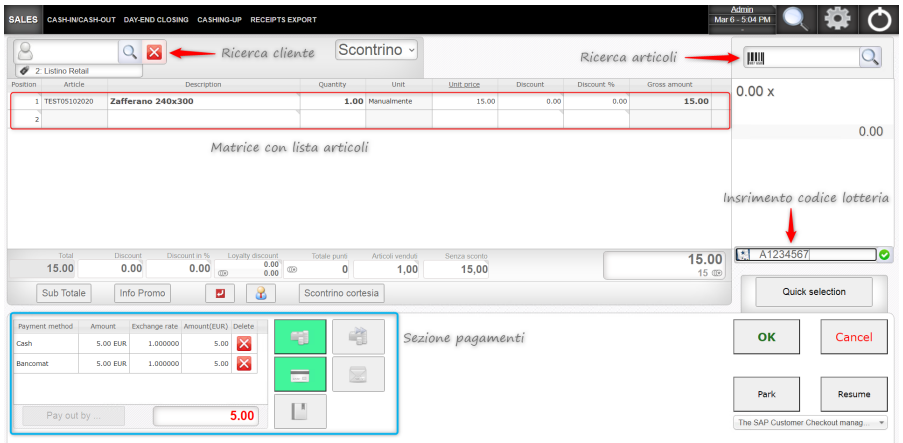

Figura 3.1: SAP Customer Checkout 1.0. [1]

L'interfaccia di Customer Checkout 1.0 si presenta come in figura 3.1, facciamo una breve introduzione al funzionamento del programma: in alto a destra è presente la barra di ricerca dove scegliere gli articoli da inserire nello scontrino/fattura; in basso a sinistra c'è la sezione pagamenti, dove è possibile scegliere il metodo di pagamento e vedere quali pagamenti sono stati inseriti; al centro è presente la matrice contenente gli articoli inseriti all'interno del documento e i relativi dati di prezzo, sconti

e altro; in alto a sinistra è presente la barra di ricerca dove scegliere il cliente al quale si sta vendendo gli articoli, campo non obbligatorio, ma necessario nel caso in cui il cliente abbia una tessera fedeltà da inserire nel documento per aderire a campagne promozionali in corso di validita. Ovviamente sono molte altre le funzioni che questo software ha da offrire, ma non è necessario conoscerle tutte per spiegare lo sviluppo del plugin, alcune le vedremo comunque durante la spiegazione.
Su questa versione ho apportato poche modifiche, in quanto gran parte delle funzioni erano già state sviluppate prima che io iniziassi il tirocinio e alcuni sviluppi importanti sono stati rimandati alla versione 2.0, la maggior parte delle funzioni sviluppate per questa versione non sarebbero state compatibili con la versione successiva (scopriremo più avanti il perchè) e quindi non avrebbe avuto senso spendere risorse per sviluppi che non avrebbero avuto seguito (anche per un discorso economico da parte del cliente). Nella versione 1.0 il plugin si divideva in una parte back-end dove era presente il vero core ed una parte front-end, necessaria soltato all'aggiunta di componenti grafici o per agganciarsi a funzioni che non erano disponibili in back-end. La parte back-end era gestita ad eventi, alcuni venivano messi a disposizione da SAP per intervenire prima o dopo con funzioni custom a differenza delle necessità, altri venivano generati dalla nostra parte front-end.

```
@PluginAt(pluginClass = IReceiptManager.class, method = "finishReceipt", where = POSITION.BEFORE)
public void pluginAtFinishReceiptBefore(Object proxy, Object[] args, StackTraceElement callStack) throws BreakExecutionException {
    if (!license.isIsvalid()) return;
    popupToConsole("FinishReceiptBefore");
    BreakExecutionManager.Reset();
    try {
        if (!jObject.isNull(fiscalPrinter) && !fiscalPrinter.isPrinterOnline())
            throw new Exception(jMessage.getString(jMessage.CultureMessages.Strings.PrinterOffline.toString()));

        ReceiptCheck receiptCheck = new ReceiptCheck((ReceiptEntity) args[0], useReceiptBean: true);

        if (receiptCheck.checkTansaction()) {
            isNextVersion();
            if (!jObject.isNull(fiscalPrinter))
                fiscalPrinter.PrintReceipt_Before((ReceiptEntity) args[0], receiptCheck);
        }

        receiptCheck.FinishTransaction();

    } catch (Exception e) {
        //BreakExecutionManager.PopupToUI(e);
        BreakExecutionManager.Break(e);
    }
    BreakExecutionManager.Reset();
}
```

Figura 3.2: Esempio evento finishReceipt. [2]

Per ogni evento c'è la possibilità di eseguire delle istruzioni, prima o dopo il flusso standard. Possiamo vedere un esempio in figura 3.2, in questo caso si tratta dell'evento "finishReceipt" ed eseguiamo il codice all'interno della funzione prima (before) del proseguimento del flusso standard. In questo caso l'evento viene lanciato alla chiusura dello scontrino e ci permette di eseguirne la stampa, nel caso qualcosa andasse storto viene lanciata un eccezione e questo provoca il roll-back, mantenendo aperto lo scontrino, perchè se la stampante è offline o genera un errore è possibile rieseguirne la stampa direttamente, senza riprendere lo scontrino dallo storico. Inoltre attraverso le annotation "@JSInject" e "@DOMInject" era possibile innestare del codice Java Script e HTML rispettivamente, in modo da aggiungere il nostro codice frontend per i motivi citati sopra (esempio figure 3.3 e 3.4).

```
@DOMInject(targetScreen = "cashingup")
public InputStream domInjectCashingup() {
    if (!license.isIsvalid()) return null;

    popupToConsole("CashingupDOMInject");

    HashMap<String, String> replaceHashMap = new HashMap<String, String>();

    return CombineResource.getCombinedResourceWithString( sourceResource: "/resources/cashingupStandard.html", replaceHashMap);
}
```

Figura 3.3: Esempio DOMInject. [3]

```
@JSInject(targetScreen = "cashingup")
public InputStream jsInjectCashingup() {
    if (!license.isIsvalid()) return null;

    popupToConsole("CashingupJSInject");

    String injStream = getCommonJSInject( includeOverrides: true);


    injStream = CombineResource.AppendInject(injStream, configValue: true, CustomJS.cashingupOverride.toString(), plugin: this);

    //injStream = CombineResource.MergeInputStream(injStream, injectDiscounts());

    injStream = customerCCO.jsInjectCashingup(injStream);
```

Figura 3.4: Esempio JSInject. [4]

In alcuni casi era necessario eseguire delle funzioni custom prima o dopo alcuni eventi che SAP non metteva a disposizione, per ovviare a questo problema era necessario individuare una funzione front-end che venisse eseguita in quel momento, veniva sovrascritta da una nostra funzione in modo da poter eseguire degli spezzoni di codice e poi riprendere il flusso standard. Ovviamente bisognava stare molto attenti a cosa si andava a modificare perchè avrebbe potuto compromettere tutte le operazione del software nativo.

```
3476 var getNewReceiptIdMutationHandlerOrig = getNewReceiptIdMutationHandler;
3477 getNewReceiptIdMutationHandler = function(mutationRecords){
3478     var _outResponse = null;
3479     if(modifyItemBalancePrice)
3480     {
3481         var _amount = parseFloat(removeSeparatorsNF($("#dialogPayment #amountInput").val()));
3482         var _paymentInput = parseFloat(removeSeparatorsNF($("#dialogPayment #CashPaymentInput").val()));
3483         var _itemBalanceRow = getRowIdFromItemId(FiscalFlowTicketItemBalance);
3484         if(_itemBalanceRow != null)
3485         {
3486             var _priceBalance = ( (parseFloat(_paymentInput))  -  (parseFloat(_amount)) ).toFixed(2);
3487             _priceBalance = String (_priceBalance).replace(".", ",");
3488             setPriceOnReceiptMatrix(_itemBalanceRow, _priceBalance );
3489             modifyItemBalancePrice = false;
3490         }
3491 
3492     }
3493     updateTotalItemSaleLabel();
3494     updateTotalLoyaltyPointLabel();
3495     updatePriceTotalWithoutDiscountLabel();
3496 
3497     _outResponse = getNewReceiptIdMutationHandlerOrig(mutationRecords);
3498     return _outResponse;
3499 };
```

Figura 3.5: Esempio wrap funzione front-end. [5]

Nell'esempio di figura 3.5 è stata creata una variabile che contiene la funzione originale ed è stata creata una funzione con lo stesso nome dell'originale, in questo modo viene chiamata la nostra funzione custom dove vengono eseguiti dei controlli e vengono eseguite le operazioni necessarie, un volta finito si ritorna la funzione originale per riprendere il flusso standard (ovviamente è possibile fare delle chiamate back-end se vanno eseguite operazioni complesse).
Per questa versione del plugin ho sviluppato l'inserimento del codice lotteria e la stampa dello scontrino di cortesia: nel primo caso è stato sufficiente inserire una testbox contenente la stringa del codice lotteria, una volta inserito e cliccato il bottone a finco veniva fatta una chiamata back-end dove veniva salvato il codice insieme ai dati dello scontrino, alla chiusura di quest'ultimo veniva inviato il codice alla stampante insieme ai dati dello scontrino e veniva aggiunto all'xml inviato all'ERP a cui Customer Checkout si appoggiava in modo da salvare l'informazione all'interno del gestionale; nel secondo caso ho inserito un bottone (sempre lato front-end) che alla sua pressione, tramite una chiamata back-end, memorizzava la richiesta e alla chiusura dello scontrino veniva stampato anche quello di cortesia.

3.2 SAP Customer Checkout 2.0

Nella seconda versione di SAP Customer Checkout è cambiato completamente il metodo di sviluppo, lato front-end non era più possibile aggiungere codice custom, infatti se c'è la necessità di aggiungere bottoni, o qualsiasi cosa riguardante l'interfaccia grafica, deve essere gestita tramite SAP Customer Checkout Manager, un software sempre basato su interfaccia web che permette di aggiungere gli elementi necessari e generare eventi custom a seconda delle necessità (figura 3.6 e figura 3.7).

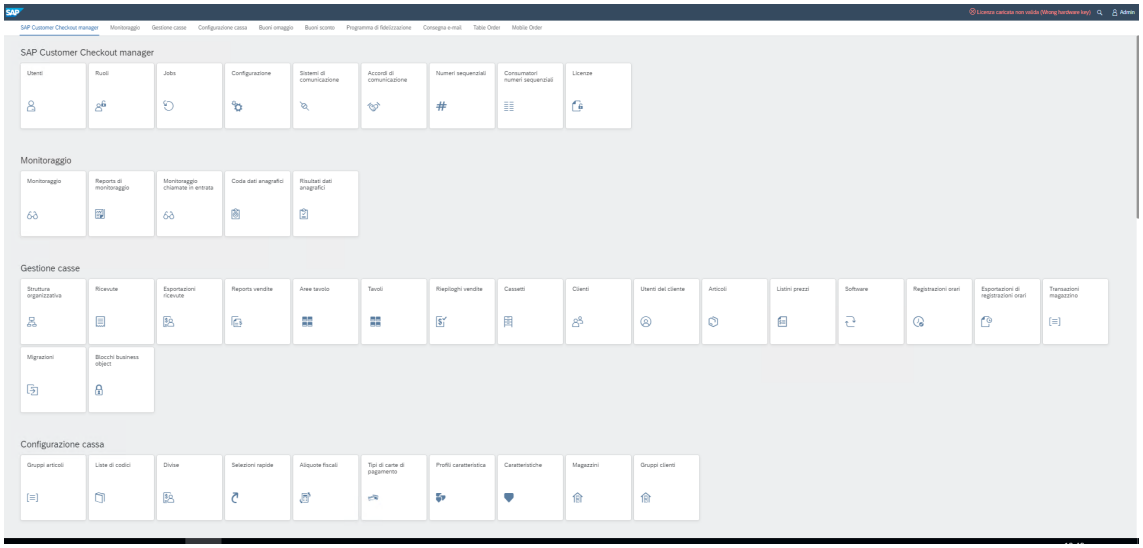

Figura 3.6: Customer Checkout manager. [6]

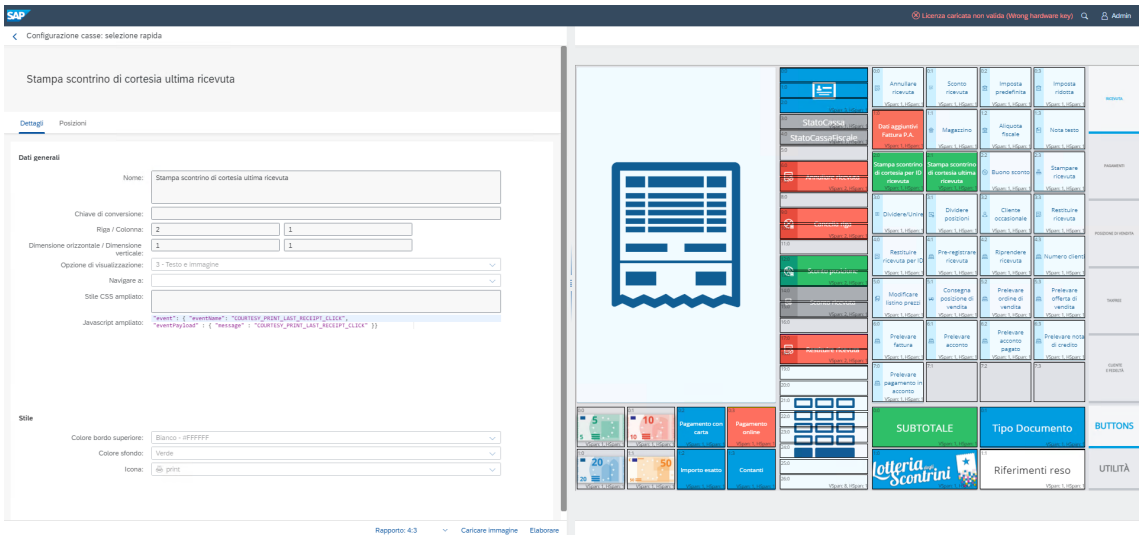

Figura 3.7: Esempio creazione bottone. [7]

L'unica cosa che si può gestire lato front-end è collegarsi all'event bus manager di Customer Checkout per agganciarsi agli eventi front-end lanciati dai nostri bottoni

custom o da quelli standard (in figura 3.8 si può vedere appunto la gestione dell'evento lanciato per il codice lotteria), infatti tramite il Manager c'è la possibilità di aggiungere qualche riga di codice Java Script per generare questi eventi anche sui bottoni presenti nativamente nel software.

```
handleEvents() {
//si aggancia all'eventbus di cco
    this.eventBus.subscribe({
       'handleEvent': (event) => {
            let payload = event.getPayload();
            if(event.getType() != "CONNECTION_MESSAGE"
            && event.getType() != "BACKEND_STATUS_POLL"
            && event.getType() != "HIDE_MESSAGE_BOX"
            && event.getType() != "CLEAR_MESSAGE"
            && event.getType() != "KEYBOARD_SHOWN"
            && event.getType() != "SHOW_KEYBOARD"
            && event.getType() != "KEYBOARD_HIDDEN"){
                var s = "dummy"; //per monitoraggio eventi frontend
            }
            if(!this.customerEventBusManager.handleEvents(event)){
                switch (event.getType()) {
                    /*
                     * Funzioni comuni e utilità
                     */
                     /*
                                            * Lotteria degli scontrini
                                            */
                    case 'LOTTERY_CLICK':
                        this.eventHelper.callBackendEvent('LOTTERY_CHECK', { });
                        break;
                    case 'CALL_BACKEND_EVENT':
                        this.eventHelper.callBackendEvent(payload.eventName, payload.eventPayload);
                        break;
                    case 'PCDM_STATUS_CLICK':
                        this.eventHelper.callBackendEvent('PCDM_STATUS_CHECK', {});
```

Figura 3.8: Esempio creazione bottone. [8]

Quando ho preso in mano il progetto è stato molto dispendioso in termini di tempo riportare tutto quello che era stato sviluppato per la precendente versione viste le nuove logiche, inoltre ho implementato la compatibilità con le stampanti fiscali Axon, Olvietti e RCH.
Ovviamento ogni stampante ha il proprio protocollo con i propri comandi per la stampa degli scontrini e tutti e tre i modelli lavoravano con un server di stampa che riceve in ingresso un file di testo che rappresenta lo scontrino/fattura e lo comunica alla stampante fiscale (in figura 3.9 e 3.10 è possibile vedere alcuni spezzoni di codice che costruscono lo scontrino per le stampanti Axon).

```
private void makeReceipt(DiscountSpecifier DiscountSpecifier) {
    boolean isReturn = false;
    clearReceipt();
    String extraDescription = jString.EMPTY;
    if (dtlTable.getReceiptHeader().haveReturnReference(receiptHeader) && receipt.getReturnReceipt()) {
        extraDescription = jString.isNullThenEmpty(dtlTable.getReceiptHeader().GetFieldDataEx(receiptHeader, fieldName: "RR_COMMENTS")); // dtlReturnBean.getT
        isReturn = true;
        returnCommand(jString.isNullThenEmpty(dtlTable.getReceiptHeader().GetFieldDataEx(receiptHeader, fieldName: "RR_ZREPORTNUMBER")),
                jString.isNullThenEmpty(dtlTable.getReceiptHeader().GetFieldDataEx(receiptHeader, fieldName: "RR_RECEIPTNUMBER")),
                jString.isNullThenEmpty(dtlTable.getReceiptHeader().GetFieldDataEx(receiptHeader, fieldName: "RR_SERIALNUMBER")),
                jString.isNullThenEmpty(
                        jDateTime.dateToString(BasePrinter.DATE_RT_NOTA, jDateTime.fromISODateToLocalDateTime(dtlTable.getReceiptHeader().GetFieldDataEx(re
                )
        );
    } else addReceiptDescription();
    int currentItem = jNumber.ZERO;
    int totItem = receipt.getItemsList().size();
    ArrayList<String> _itemsDescription = new ArrayList<>();
    for (ReceiptItemBean item : receipt.getItemsList()) {
        logger.info("item: " + item.getSaleType().name() + ": " + item.getDescription());
        if (pluginConfiguration.settings.getReceiptItemsExcluded().contains(item.getItemCode())) {
            String _lineQuantityForText = jString.EMPTY;
            _lineQuantityForText = jNumber.toString(jNumber.parseDouble(item.getQuantity()), decimal: 0);
            if (_lineQuantityForText.endsWith(".00"))
                _lineQuantityForText = _lineQuantityForText.replace( target: ".00", replacement: "");
            //addTextToReceipt(item.getDescription() + " " + _lineQuantityForText, true, true, TextDashLineLength.TEXT);
            _itemsDescription.add(item.getDescription() + " " + _lineQuantityForText);
            continue;
        }
```

Figura 3.9: Creazione scontrino per Axon (1). [9]

```
    for (ReceiptAdditionalText additionalText : additionalTextList) {
        addTextToReceipt(additionalText.getText(), additionalText.getDashLineBefore(), additionalText.getDashLineAfter(), additionalText.getDashLineLengthO
    }
    if (!jObject.isEqual(receipt.getDiscountType(), DiscountType.STANDARD))
        addDiscountReceipt(receipt, DiscountSpecifier);
    //if (ConfigurationValues.INSTANCE.getConfigurationOptionBoolean(ConfigurationValues.ConfigurableOptions.FiscalFlow_Printer_New_Firmware)) {//se è il n
    if (pluginConfiguration.printer.isLotteryEnabled()) {//se è il nuovo firmware
        if (!jString.isNullOrEmpty(receipt.getLotteryId())) {
            addLotteryCode(receipt);//aggiungo codice lotteria
        }
    }
    if (!isReturn) {
        disableCutter();
    }
    for (ReceiptPaymentBean payment : getReceiptPaymentList())
        if (!jObject.isEqual(payment.getPaymentType(), PaymentType.CONTANTI))
            addPayments(payment);
    for (ReceiptPaymentBean payment : getReceiptPaymentList())
        if (jObject.isEqual(payment.getPaymentType(), PaymentType.CONTANTI))
            addPayments(payment);
double totalPoint;
    if (receipt.getLoyaltyAccountTotalpoints() >= 0 && receipt.getLoyaltyPoints() >= 0) {
        addTextToReceipt("---RACCOLTA PUNTI-----");
        addTextToReceipt("Saldo iniziale: " + receipt.getLoyaltyAccountTotalpoints());
        addTextToReceipt("Punti sulla spesa: " + receipt.getLoyaltyPoints());
        totalPoint = receipt.getLoyaltyAccountTotalpoints() + receipt.getLoyaltyPoints();
        addTextToReceipt("Nuovo saldo punti: " + totalPoint);
    }
    addBarCode(receipt.getReceiptId());
    getLastReceiptNumber();
```

Figura 3.10: Creazione scontrino per Axon (2). [10]

Durante la migrazione di versione, Pittarello S.p.A. (nostro cliente dalla nascita di Datalab srl) ha deciso di acquistare SAP Customer Checkout come POS managment, visto anche che utilizzavano già SAP Business One come ERP, di conseguenza ho dovuto implementare alcune funzioni necessarie al cliente. La priorità era la lettura dei codici a barre per l'inserimento di articoli, sconti e fidelity, ovviamente il software nativamente implementa gia la lettura dei barcode, però Pittarello possiede una propria gestione che non corrisponde allo standard e hanno un proprio server per la gestione delle fidelity e degli sconti. Inoltre avevano la necessità di inserire ulteriori campi per gli articoli, perchè dovevano gestire anche taglie, colore, materiale, famiglia e altro di ogni prodotto venduto, sempre scannerizzando solamente il barcode,

così l'utente avrebbe dovuto solamente scannerizzare i codici, senza inserire a mano i dettagli. Per ovvi motivi non posso spiegare qual'è la gestione dei codici a barre di Pittarello, ma in figura 3.11 e 3.12 si possono vedere alcuni spezzoni di codice che analizzano le informazioni scannerizzate e salvano i dati aggiuntivi necessari, che poi verranno comunicati all'ERP di appoggio.

```
case "SALESITEM_SIZE_SAVE":
{
    SalesItemEntity salesItem = null;
    String sizeCode = null;
    if(parameters.getString( key: "selectedOption").contains("|")) {
        String[] optionSel = parameters.getString( key: "selectedOption").split( regex: "\\|");
        String itemSelId = optionSel[0];
        sizeCode = optionSel[1];
        salesItem = ReceiptHelper.getSalesItemByRowIndex(receipt, Integer.parseInt(itemSelId));
    }
    else {
        sizeCode = parameters.getString( key: "selectedOption");
        salesItem = ReceiptHelper.getLastSalesItem(receipt);
    }
    String barcode = (String) ReceiptHelper.getReceiptRowValue(receipt, salesItem,  key: "U_BARCODE");
    if (jString.isNullOrEmpty(barcode))
        barcode = salesItem.getId();
    if (barcode.length() > 10)
        barcode = barcode.substring(0, 10);

    barcode += sizeCode;
    barcode = BarcodeQrHelper.getEan13From12CharsBarcode(barcode);
    ReceiptHelper.setReceiptRowInfo(receipt, salesItem, barcode);
    MessageHandler.showMessage(MessageHandler.MessageType.INFO,  message: "Taglia impostata per l'articolo "+salesItem.getDescription()
    blockNextEvents();
    break;
```

Figura 3.11: Analisi barcode contenente la taglia di una scarpa. [11]

```
case "GIFTCARD_PAYMENT_SAVE"://gestione input utilizzo gifcard
{
    ReceiptHelper.checkReceiptSale(receipt);
    checkMandatoryFields(parameters,  ...fields: "suggestedValue", "code","itemCode", "remainingValue");
    BigDecimal suggestedValue = MathHelper.getPriceFromJsonField(parameters,  field: "suggestedValue");
    BigDecimal remainingValue = MathHelper.getPriceFromJsonField(parameters,  field: "remainingValue");
    if(remainingValue.compareTo(BigDecimal.ZERO) < 1)
        sendError("Importo non valido!");
    if(suggestedValue.compareTo(remainingValue) > 0)
        sendError("Importo superiore al residuo della GiftCard!");

    try {//aggiungo all'header le info della gift che sto usando come pagamento
        //ottengo la lista di giftcard che ho utilizzato come pagamento
        JSONObject newGift = new JSONObject();//creo il json che contiene il barcode
        newGift.put("description", Const.GIFTCARD_PAYMENT_RECEIPT_EXTRAINFO_KEY);
        //newGift.put("externalId", "NEW");
        newGift.put("code", parameters.getString( key: "code"));
        newGift.put("itemCode", parameters.getString( key: "itemCode"));
        newGift.put("remainingValueBefore", remainingValue);

        PaymentHelper.setReceiptPaymentInfo(dtlTable, configuration, receipt,  paymentId: "NEW", newGift ); //gli passo il json con le info pagamento
    } catch (Exception e) {
        throw new ManagedError(e.getMessage());
    }
    PittReceiptFidelityHelper.addGiftcardPaymentToReceipt(receipt, suggestedValue, parameters.getString( key: "code"));
    blockNextEvents();
    break;
```

Figura 3.12: Analisi barcode contenente una giftcard per il pagamento. [12]

SAP Customer Checkout mette a disposizione una sezione dove è possibile avere dei campi di configurazione che il plugin è in grado di leggere, però sono modificabili solo manualmente all'interno del programma ed è possibile averne una quantità limitata. Per comodità del nostro reparto di assistenza c'era la necessità di poter configurare il plugin su più punti cassa contemporaneamente, perchè sarebbe stato infattibile modificare la configurazione a mano su duecento punti cassa. Inoltre il numero di parametri che SAP mette a disposizione non erano sufficienti alle necessità del nostro plugin, di conseguenza abbiamo optato per una configurazione attraverso file JSON, in maniera che fosse facilmente leggibile e così è sufficiente copiare il file all'interno di ogni cassa. Il JSON è strutturato in due parti: una contenente i parametri di configurazione generale, comuni a tutte le casse, ed una parte contenente la configurazione specifica della singola cassa, come ad esempio indirizzo ip della stampante e altro.
Pittarello inoltre ha una sua gestione delle fidelity, nel loro precedente software di cassa quando c'era la necessità di inserire la fedelizzazione del cliente nello scontrino per accumulare i punti o per partecipare a campagne promozionali, si apriva automaticamente il browser e andando a comunicare con il loro web service, sceglievano il cliente ed il POS system in automatico si salvava il codice della fidelity, quindi sotto loro richiesta abbiamo riprodotto questo comportamento (in figura 3.13 e 3.14 si possono vedere le due funzioni esterne che gestiscono questa richiesta).

Figura 3.13: Funzione di apertura del browser per la ricerca della fidelity. [13]

```
115             case "USE_FIDELITY": {
116                 if(PittReceiptFidelityHelper.isReceiptValidForFidelity(receipt)) {
117                     EventCaller.showLoadingIndicator("Recupero dati fidelity...");
118                     long codiceFidelity = Long.parseLong(parameters.getString( key: "CodiceFidelity"));
119                     FidelityCaller fidelityCaller = new FidelityCaller(configuration, pittSettings, (PittDtlTable) dtlTable);
120                     int puntiCliente = fidelityCaller.getFidelityPoints(codiceFidelity);
121                     JSONObject couponInfo = null;
122                     if(parameters.containsKey("Campagna") && !jString.isNullOrEmpty(parameters.getString( key: "Campagna"))) {
123                         //Gestione coupon
124                         EventCaller.showLoadingIndicator("Recupero dati coupon...");
125                         Integer campaign = Integer.parseInt(parameters.getString( key: "Campagna"));
126                         String couponKey = parameters.getString( key: "ChiaveCoupon");
127                         couponInfo = fidelityCaller.getCouponInfo(codiceFidelity,campaign,couponKey);
128                     }
129                     ReceiptHelper.removeAppliedPromotion(receipt);
130                     PittReceiptFidelityHelper.setFidelityDataInReceiptHeader(dtlTable, receipt, codiceFidelity, parameters.getString( key: "Nome"), parameter
131                     EventCaller.hideLoadingIndicator();
132                     EventCaller.callBackendEvent( event: "SHOW_FIDELITY_RECAP",  payload: null);
133                 }
134                 else {
135                     MessageHandler.showMessage(MessageHandler.MessageType.WARNING,  message: "Operazione non supportata");
136                 }
137                 blockNextEvents();
138                 break;
```

Figura 3.14: Funzione che aggiunge l'utilizzo della fidelity allo scontrino. [14]

Capitolo 4

Conclusioni

Lo sviluppo di questo plugin è stato particolarmente complesso perchè la documentazione a disposizione non è molta e a volte è stato necessario modificare processi nativi in punti molto delicati, dove SAP non metteva a disposizione eventi che potevano essere riconosciuti dal software per intervenire. Un'altra difficoltà è stata riportare tutto ciò che che era stato sviluppato per la prima versione, perchè per alcune funzioni abbiamo dovuto modificare l'operatività dell'utente per rispettare la struttura della nuova versione del sofware e questo ovviamente ha portato i clienti a dover imparare nuovamente ad utilizzare le funzioni a cui erano abituati da anni. Tralasciando queste problematiche la nuova versione ha introdotto un interfaccia grafica più piacevole e completa, inotre utilizzando Customer Checkout Manager per apportare modifiche all'interfaccia grafica è sicuramente un vantaggio per noi programmatori, che non dobbiamo più sviluppare codice lato front-end, rendendo le modifiche più veloci e intuitive.

Ringraziamenti

Ringrazio il Prof. Nanni Loris, relatore di questa tesi, per l'opportunità e per essersi reso disponibile a farmi da tutor per il tirocinio.
Ringrazio la mia ragazza per aver sempre creduto in me e per il supporto psicologico che mi ha dato in questi anni.
Ringrazio i miei genitori, per il sostegno costante e per avermi permesso di poter studiare nelle migliori condizioni.
Ringrazio i miei amici di sempre, per i bellissimi momenti passati insieme e per aver condiviso lo stesso percorso di studi anche in università diverse.
Ringrazio gli amici che ho potuto conoscere grazie all'università: con voi ho potuto avere sempre un confronto per la preparazione pre e post esami.

Bibliografia

[1] Sito Web SAP: https://www.sap.com/index.html

[2] Sito Datalab srl: https://www.datalab-srl.com/

[3] Sito Corso SAP: https://www.corsosap.com/?nab=1

[4] Vincent A Mabert, Ashok Soni, and Munirpallam A Venkataramanan. The impact of organization size on enterprise resource planning (ERP) implementations in the US manufacturing sector. Omega, 31(3):235–246, 2003

[5] Carl Marnewick and Lessing Labuschagne. A conceptual model for enterprise resource planning (ERP). Information management & computer security, 2005.