ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il management

Progettazione ed implementazione di un'applicazione mobile per la gestione dei dati di vendita aziendali

Relatore:
Chiar.mo Prof.
MARCO DI FELICE

Presentata da:
NICOLÓ GAMBARELLI

Sessione III
Anno Accademico 2013/2014

Indice

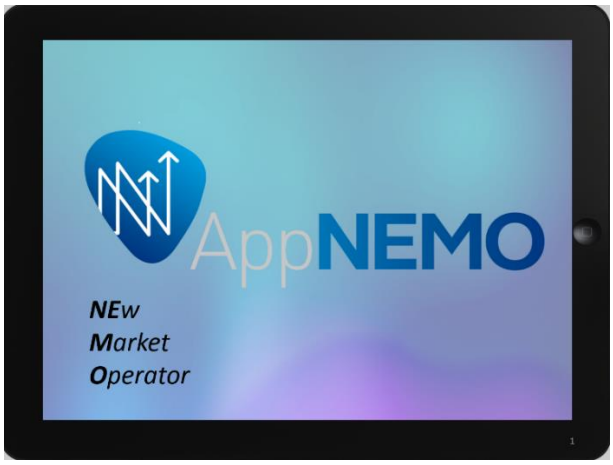
1. Introduzione

Il tirocinio, all'origine del progetto illustrato in questa tesi di laurea, è stato svolto presso la SIOR SRL di Varese, società di consulenza informatica di medio-piccole dimensioni, attiva dal 1982 nella progettazione e realizzazione di software applicativo.

Sior opera in diversi settori:

- Progetti di Application Management in Outsourcing
- Sistemi informativi aziendali integrati, basati su Dynamics AX della Microsoft
- Sviluppo di software applicativo su specifiche del cliente
- Consulenza sistemistica
- Realizzazione di portali e siti internet
- Applicativi mobile per tablet
- Migrazioni di sistemi informativi, con il re-hosting dai mainframe e dagli ambienti proprietari a piattaforme più moderne, più efficienti e molto meno costose di quelle originarie.

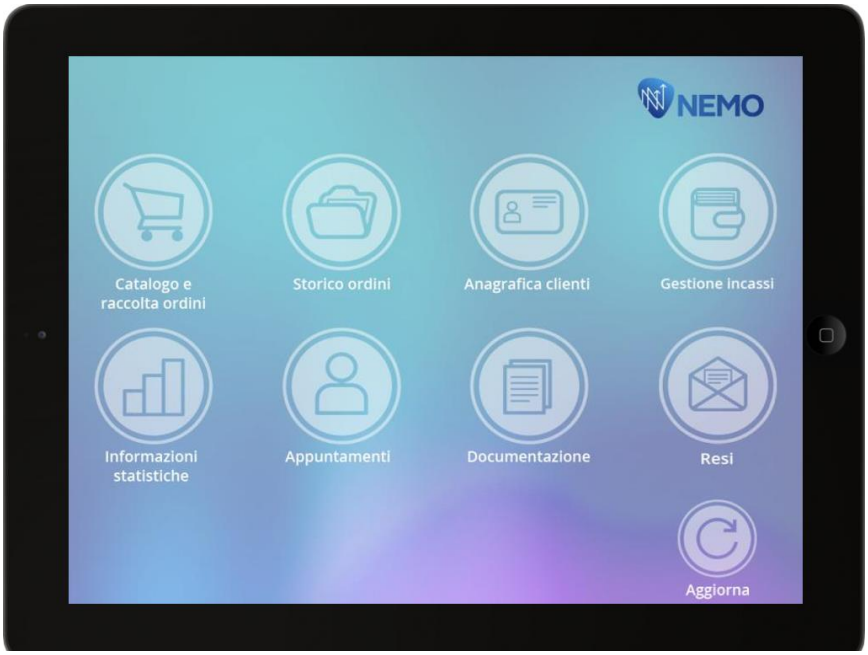
All'interno di un team mi sono occupato di curare nella produzione di una applicazione per la forza vendita dei moduli da me assegnatomi dal project manager relativi allo "storico ordini" e alla "documentazione", in questo ambito ho analizzato, progettato e implementato il software e successivamente testato.



Partendo dalla descrizione dell'applicazione nel suo complesso identifico il target di aziende a cui è rivolta specificandone le caratteristiche e nei settori in cui è stata utilizzata spiego alcune particolari personalizzazioni che ci sono state richieste; nel settore sono presenti altre realtà, ne identifico alcune e ne spiego caratteristiche e peculiarità. Per spiegare la progettazione dell'applicazione, parto dal sistema nel suo complesso, specificando il flussi di informazione tra tablet e server con i relativi

tracciati, il database su tablet e la strutturazione del software attraverso alcuni diagramma UML. Nel capitolo riservato all'implementazione introducendo alle tecnologie necessarie per la produzione di software per sistema operativo Android, identifico alcuni frammenti di codice che ritengo maggiormente interessanti. In conclusione pubblico screenshot dei settori dell'applicazione di cui mi sono occupato in prima persona e spiego le procedure utilizzate per il testing della stessa; in seguito ipotizzo delle future funzionalità che ritengo utili per l'utilizzatore.

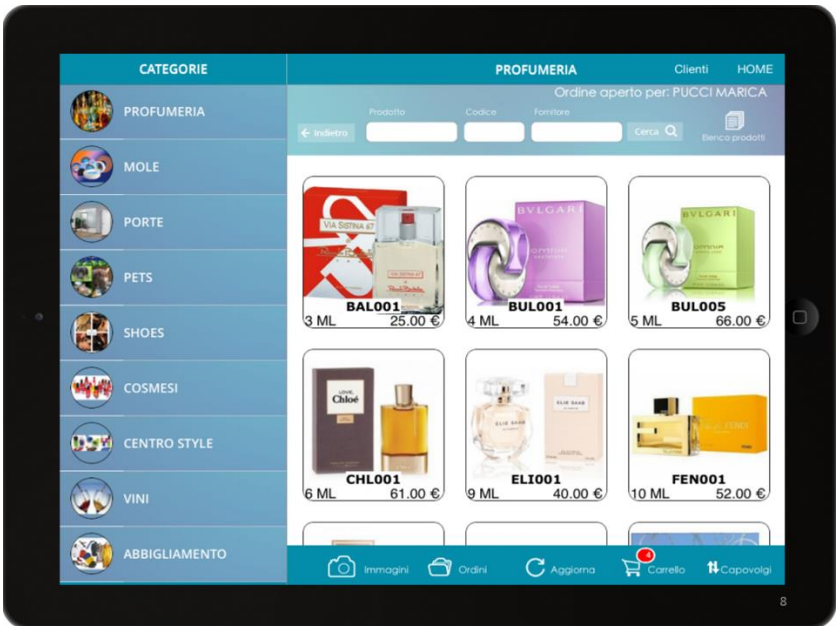
2. Descrizione della "App" NEMO - NEw Market Operator



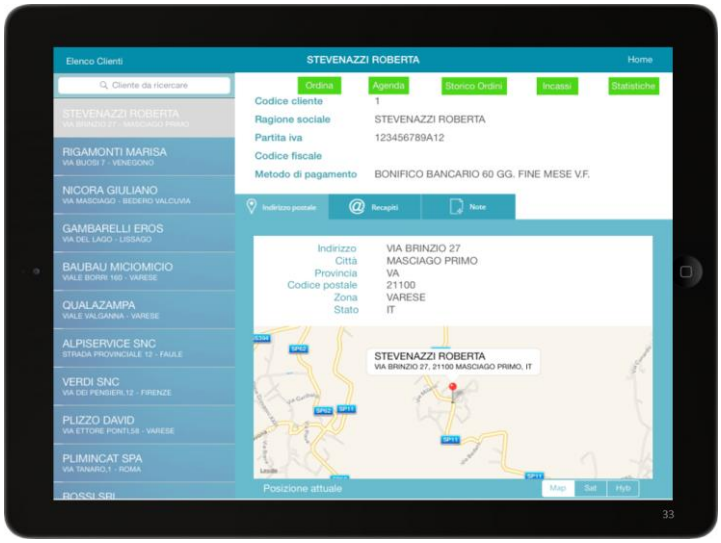
Vista principale.

L'applicazione, modulare e personalizzabile, semplice da utilizzare ed intuitiva, è stata realizzata in origine per il sistema operativo iOS con l'obiettivo di ottimizzare le procedure del processo di vendita.
Comunicando con il sistema informativo di gestione aziendale in modo bidirezionale, per mezzo di interfacce opportunamente predisposte, (sono già attive le interfacce per i package Zucchetti, Passepartout, ACG e diversi altri applicativi) NEMO mette a disposizione di ogni venditore ,di norma una volta al giorno (l'intervallo di tempo viene scelto dal cliente in funzione delle sue esigenze , e quindi potrebbe essere più ridotto) tutte le informazioni (giacenze di magazzino , nuovi prodotti, nuovi clienti, listini prezzi, situazione contabile dei clienti, ecc...) necessarie a svolgere le attività di vendita

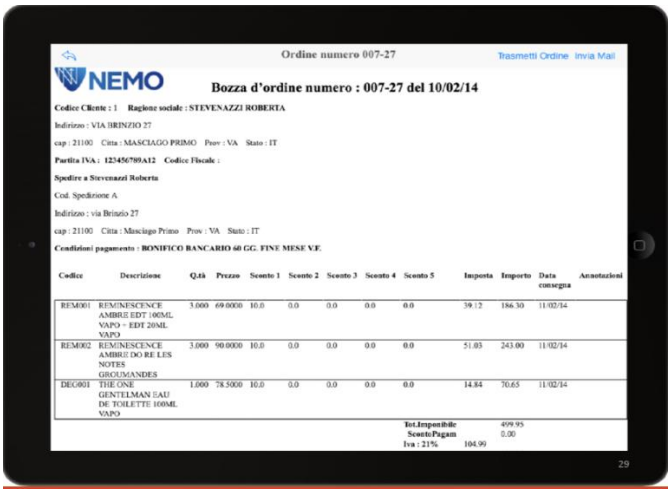
e promozionali , e contestualmente di inviare all'azienda gli ordini raccolti per innescare il processo di spedizione.
Con questa modalità operativa si evita il reinserimento manuale dell'ordine da parte degli addetti della sede.
La frequenza di interazione con il sistema informativo aziendale potrebbe sembrare insufficiente, non trattandosi di un tempo reale vero e proprio; la scelta è stata ponderata e voluta, in quanto ogni altra soluzione avrebbe implicato onerosi investimenti al cliente su un gestionale preesistente, e si considerato che il mercato potenziale delle aziende commerciali di medio-piccola dimensione, sarebbe stato più ampio.
L'applicazione, strutturata per lavorare anche in assenza di connessione internet, in quanto i dati vengono memorizzati localmente e possono essere inviati in un tempo successivo, è composta da diversi moduli, integrati con il modulo base di **Gestione Catalogo Articoli** e **Raccolta Ordini.**



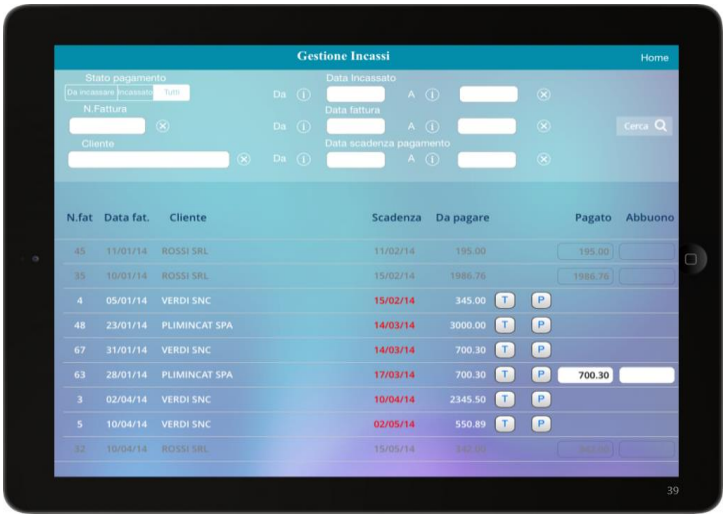
Il Catalogo può essere corredato di schede tecniche, di immagini, di filmati e, grazie all'alta definizione grafica e alla possibilità di ingrandire l'immagine e di ribaltarla, consente di visualizzare i dettagli dei prodotti; il catalogo è sfogliabile, accessibile per ricerca alfabetica e navigabile attraverso una struttura ad albero multilivello.
Dall'articolo, a prescindere dalla modalità utilizzata per raggiungerlo, è possibile innescare automaticamente la riga d'ordine, con il metodo classico del "carrello": visualizzazione delle righe inserite, possibilità di modifica, valutazione del totale d'ordine, etc.



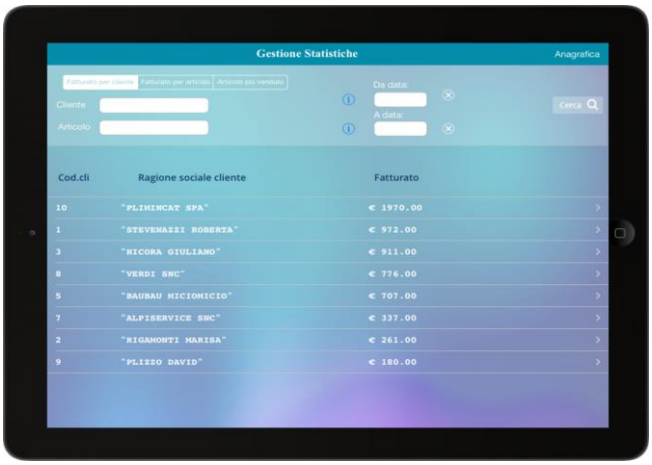
La modalità di **Raccolta Ordini** più logica è quella di accedere al modulo **Anagrafica Clienti** , collegato anche agli altri moduli, che contiene le informazioni di base relative al cliente: oltre all'indirizzo, la mappa (per l'ottimizzazione dei percorsi), le note relative ai recapiti e ai giorni di accesso e tutto ciò che è utile per effettuare una vendita ottimale; l'applicazione consente anche la raccolta dell'ordine da un nuovo cliente , non ancora catalogato, attraverso l'inserimento dei dati indispensabili, che saranno successivamente sincronizzati con la sede .



Conclusa la raccolta dell'ordine o degli ordini, è possibile inviarli, oltre che alla sede aziendale per essere direttamente elaborati dal sistema gestionale, ad indirizzi prestabiliti o digitati al momento; questa possibilità consente di inviare una copia dell'ordine al cliente stesso, a titolo di conferma.

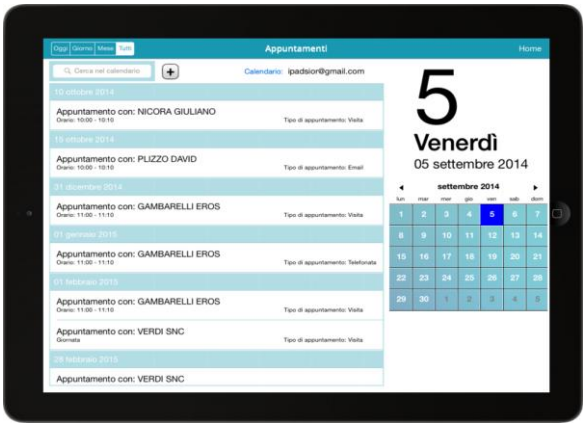


Con il modulo **Incassi** il venditore ha modo di conoscere le fatture scadute e in scadenza, e può provvedere, seguendo le disposizioni aziendali, a registrare gli incassi totali o parziali, a concedere abbuoni o sconti, sempre che sia nelle sue facoltà. Anche questi dati possono essere trasmessi, immediatamente o in un secondo tempo, al sistema gestionale per la loro elaborazione diretta.
In modo particolare sono ovviamente trattate le situazioni critiche o i crediti diventati inesigibili: in questi casi il venditore ha le segnalazioni necessarie per gestire o bloccare il prosieguo del processo di vendita.

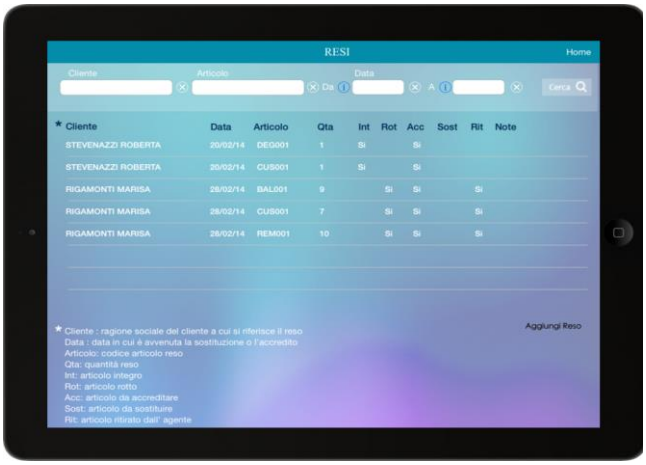


Con il modulo **Statistiche**:
- Ordinato e Fatturato per cliente, con il dettaglio per articolo in ordine decrescente, a valore e quantità.
- Ordinato e Fatturato per articolo, con il dettaglio per cliente in ordine decrescente, a valore e quantità.

Le informazioni per i venditori, per i Capi-area e per la Direzione Commerciale, disponibili quotidianamente, consentono interventi tempestivi per stimolare la vendita verso i prodotti e/o i clienti, adottando promozioni o altre iniziative più opportune.

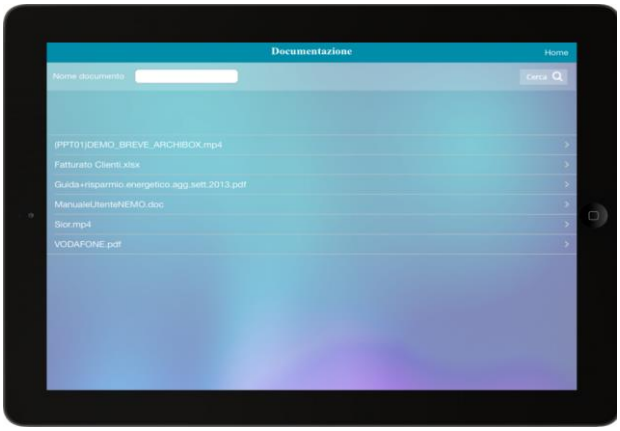


Il modulo **Appuntamenti** gestisce e visualizza, in modo sincronizzato con il tablet, le telefonate, le mail, le visite e ne tiene la storia. Per ogni evento registra le annotazioni necessarie.



Il modulo **Resi** registra il reclamo del cliente e l'eventuale reso della merce non conforme con la relativa motivazione, lo trasmette in sede e ne tiene la storia.

Il modulo **Storico Ordini** registra gli articoli acquisiti dai clienti nel tempo, e consente di visualizzarli per cliente, per articolo, per intervallo temporale, utilizzando opportuni filtri.



Il modulo **Documentazione** permette di consultare documenti e filmati, allocati dal gestionale o inserirti manualmente in un’apposita cartella; l’applicativo li riprende automaticamente, permettendone la visualizzazione. Da quanto sopra esposto si può concludere che NEMO, oltre che un efficace strumento di presentazione dei cataloghi e la raccolta degli ordini, può essere considerato anche un CRM operativo.

3. Target di mercato

Nemo è stato realizzato con l'obbiettivo di rivolgersi a:

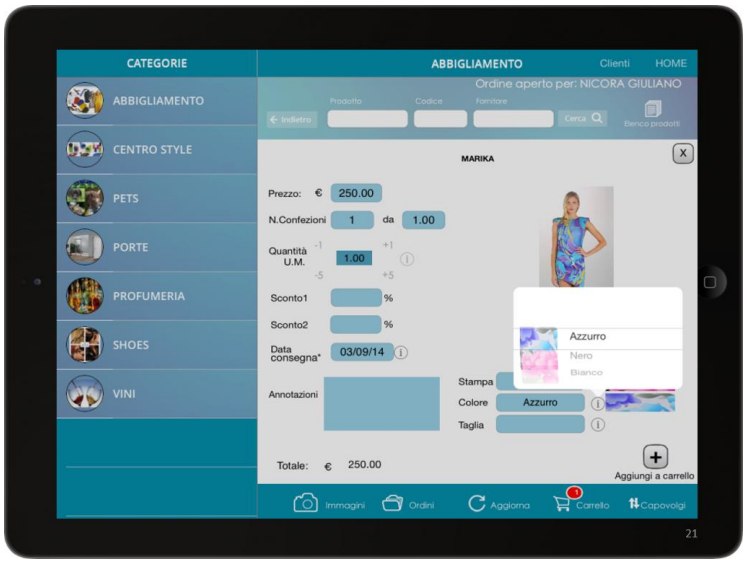
- le aziende che effettuano la vendita attraverso la visita periodica al cliente per la raccolta degli ordini
- le aziende per le quali la presentazione del catalogo riveste un'importanza determinante
- anche se non sono stati previsti limiti territoriali, poiché le attività di personalizzazione, assistenza e manutenzione sono in grande parte eseguibili da remoto, l'azienda si è orientata prevalentemente sul nord Italia, anche se eccezionalmente qualche cliente estero dimostra interesse al prodotto, grazie alla presenza su Apple Store e Google Play
- pur essendo predisposto per aziende di qualsiasi dimensione e numero di venditori, per la politica dei prezzi praticati, per l'impostazione della struttura addetta alla manutenzione e all'assistenza, Sior agevola le aziende commerciali medio-piccole, con capacità di investimento modeste, e con una struttura di vendita di una decina di venditori.
  L'impostazione del listino prezzi, strutturato a fasce in base al numero di addetti alla vendita privilegia questa tipologie di aziende; in quanto l'investimento complessivo è limitato a qualche migliaio di euro.

Passando all'esame dei settori merceologici in cui NEMO è effettivamente utilizzato occorre precisare che ogni settore merceologico ha richiesto specifiche personalizzazioni:

- Petfood e accessori Pet:
  Le proposte di vendita di NEMO sono state indirizzate sia ai Produttori che alle aziende Distributrici; su quest'ultime si sono ottenuti i maggiori successi mentre per i produttori sono in corso interessanti trattative che richiedono tempi più lunghi.
  Le personalizzazioni richieste dal settore hanno riguardato principalmente la gestione degli sconti; arrivando ad avere fino a 11 tipi di sconti dei quali alcuni agiscono in cascata secondo le regole aziendali; in funzione delle specifiche autorizzazioni, l'agente può concedere omaggi in merce.
  Sono inoltre gestiti sconti al pagamento immediato, promozioni su singoli prodotti o linee di prodotto, sconti legati alla quantità e al peso della merce ordinata.
- Distribuzione generi alimentari: il settore non ha richiesto particolari personalizzazioni.
- Distribuzione Bevande: sconti volumetrici e reso dei vuoti.

- Colorifici:

  potendo il numero degli articoli raggiungere una quantità ragguardevole (abbiamo riscontrato da un cliente fino a 180000 articoli) la problematica relativa al caricamento delle immagini dal catalogo è stata risolta da una parte riducendo la dimensione delle immagini visualizzate (e di conseguenza la qualità) e dall'altra caricando le immagini in modalità asincrona, cioè aggiornando le immagini della vista allo scorrere dei prodotti.



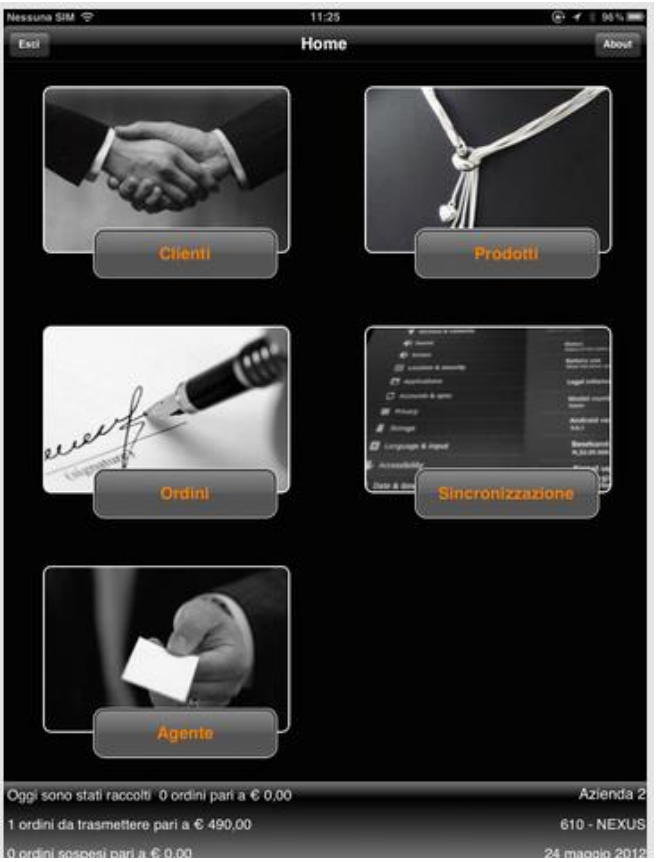
- Abbigliamento\Calzature:

  in questo ambito la particolarità è stata la gestione della taglia e del colore; per non aumentare a dismisura il numero degli articoli la taglia e il colore vengono scelti all'inserimento dell'ordine
- Cancelleria

4. Stato dell'Arte

Sono molte le aziende che hanno prodotti assimilabili al NEMO. Mi limito a citarne alcune, che sono quelle effettivamente entrate in concorrenza con Sior nelle trattative:

a) Nexus Informatica s.r.l. – Ipad Sales, Droid Sales

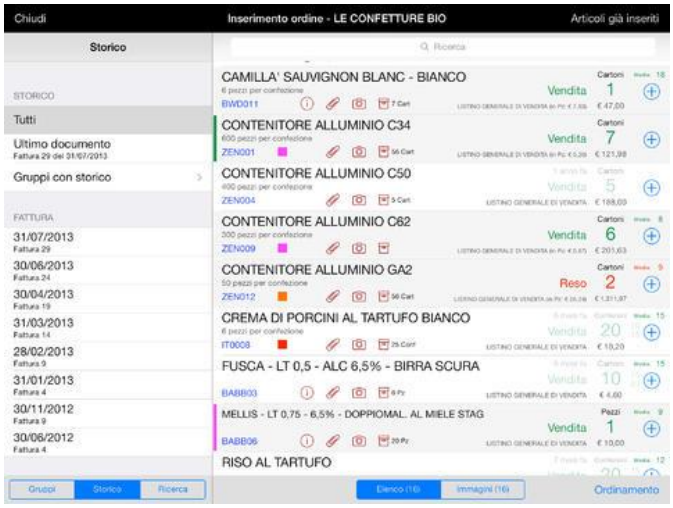


E' una software house di medie dimensioni con sede a Novara che opera con distributori in tutto il territorio nazionale, che tra le diverse specializzazioni propone un suo applicativo mobile per la raccolta di ordini, "Ipad Sales". Questo prodotto è strutturato in modo paragonabile a Nemo, cioè si compone alcuni moduli base e di altri moduli opzionali in base alle esigenze del cliente. I moduli base consistono nel modulo "Server" che si interfaccia con il gestionale, il modulo "Sync" che sincronizza i dati e il modulo "Client" che svolge le operazione su Ipad. Specularmente esiste la versione per Android , "Droid Sales".
Nexus risulta essere il più presente in concorrenza alla nostra soluzione in diverse trattative.

b) Weblink - uPP! Ordini

Software house di Varese che si occupa di sviluppo software, siti web, e-commerce, web-marketing e ha sviluppato soluzioni per il mondo mobile (iPad, iPhone, Android) come la linea di software uPP! che comprende: uPP! Doc, uPP! Catalogo, uPP! Ordini, uPP! Move, uPP! Custom.
Alcuni di questi pacchetti si sovrappongono alle funzionalità di NEMO: uPP! Catalogo consiste nella realizzazione di app personalizzate per gestione e pubblicazione del catalogo PDF su dispositivi tablet e smartphone con gestione autonoma dei contenuti e funzionalità di fruizione avanzate. La app viene utilizzata sia per finalità marketing che commerciali ed è disponibile per il download dagli store online a clienti e rivenditori.
uPP! Ordini consiste nella realizzazione, partendo dal catalogo PDF, di sistemi completi di automazione della forza vendita, agenti e commerciali. Anche questo applicativo interfaccia i sistemi gestionali, consente all'agente di sfogliare il catalogo PDF con il cliente e di creare direttamente l'ordine prodotti, avendo sempre sott'occhio lo stato degli ordini precedenti e l'anagrafica clienti. In qualche sporadico caso Sior si è trovata in concorrenza.

c) P&L information tecnology – Isell



È una società della provincia di Brescia che dal 1992 progetta e realizza sistemi software su piattaforme "mobile", in questo ambito a creato ISell, "app" multilingue le cui funzionalità sono:

- Gestione Anagrafiche Clienti e Articoli con allegati (immagini, schede tecniche, etc.) e Listini personalizzabili;
- Gestione dei Giri visite
- Gestione di Ordini con generazione copia commissione e di Preventivi
- Gestione della tentata vendita e degli incassi
- Gestione documenti
- Gestione barcode con lettori bluetooth
- Dati crittografati
- Backup dati
- Statistiche
- Storico documenti

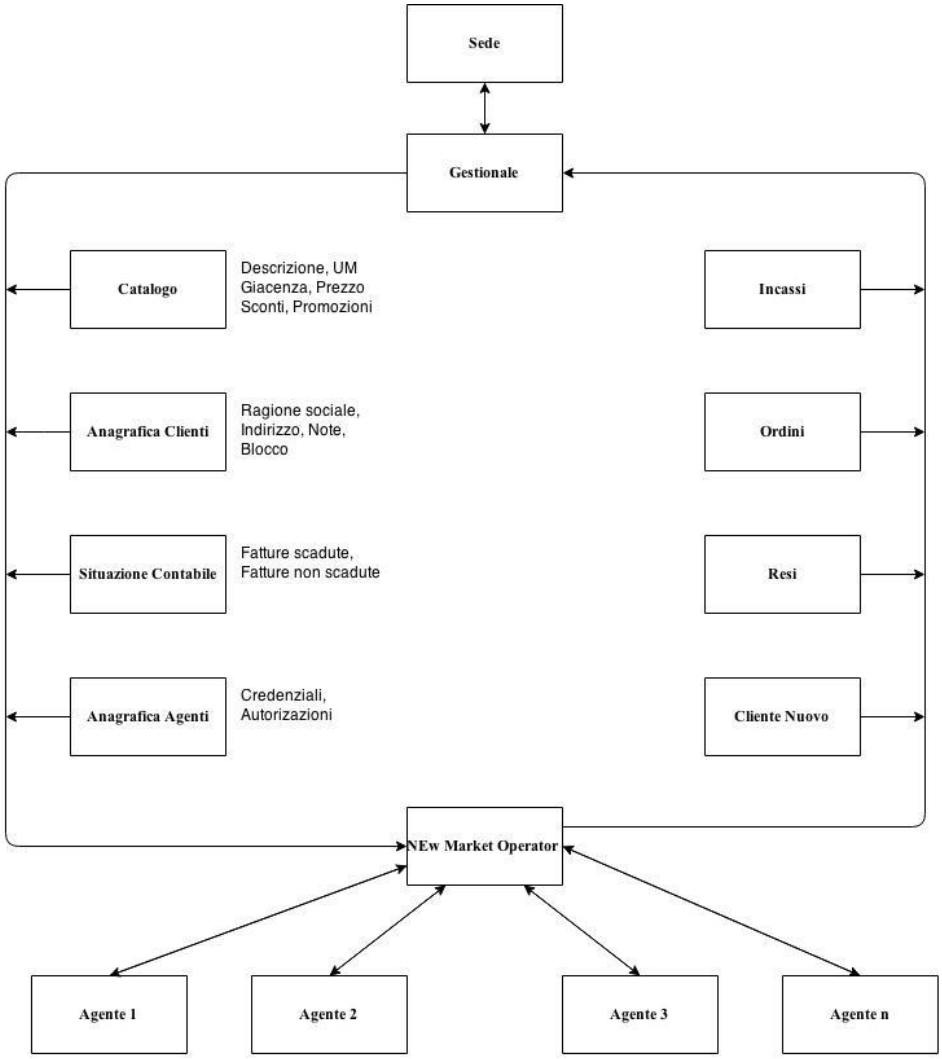
5. Progettazione di NEMO per Android

Il progetto a cui ho partecipato, che ha avuto origine dalle richieste del mercato, è stato realizzato da un team di tre persone più il sottoscritto, ha implicato un cambio di piattaforma trattandosi di un applicativo "mobile" dell'Ipad Apple da migrare verso gli ambienti Android .

La prima considerazione deriva dalla specificità dell'applicativo oggetto della migrazione: si tratta di un package realizzato per il sistema operativo iOS con il linguaggio Objective C. In secondo luogo, da una accurata analisi del codice e del database utilizzati in iOS, è emerso che mentre una serie di tecniche utilizzate nella codifica, quali ad esempio gli effetti grafici, gli effetti di transizione, e più in generale gli oggetti e le librerie specifiche non erano riproducibili con un convertitore automatico nel ambiente Android, il data base SQLite poteva essere mantenuto inalterato sia per le classi documentali che per i tracciati.
La decisione conseguentemente adottata è stata quindi quella di impostare il progetto nel modo seguente:

- confermare le stesse specifiche di analisi applicativa
- riscrivere il codice
- mantenere invariato il Database.

La struttura di NEMO, sia per le funzionalità che per il Database, è stata progettata mantenendo le linee guida utilizzate per l'Ipad, e ricodificando i programmi. Su un Server FTP avviene il passaggio bidirezionale dei dati tra il Gestionale e i tablet, mentre l'applicazione vera e propria risiede sui tablet Android. La possibilità di lavorare senza connessione Internet è stata mantenuta.



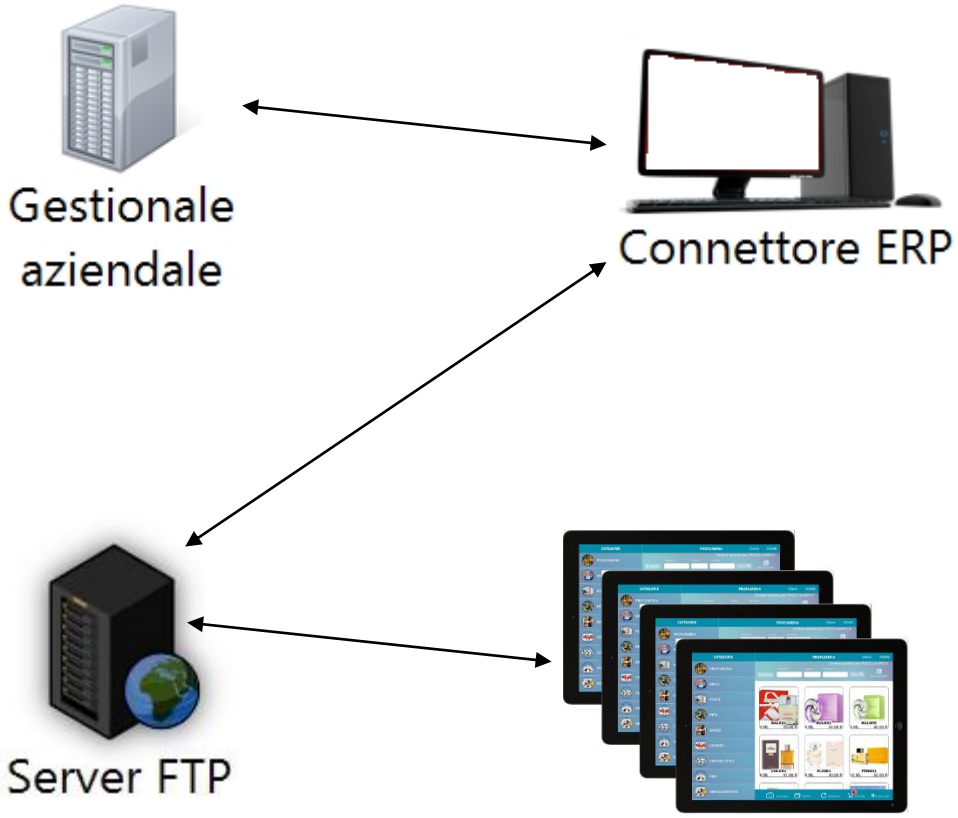
In questo diagramma vengono formalizzati il passaggio dei dati fondamentali all'applicazione, infatti il gestionale quando ritenuto necessario dal cliente aggiorna i tablet delle seguenti informazioni:

- Catalogo prodotti
- Anagrafica clienti
- Situazione contabile (incassi)
- Anagrafica agenti

I tablet viceversa aggiornano il gestionale delle seguenti informazioni:

- Ordini
- Incassi
- Resi
- Clienti nuovi

1. **Server**



Per quanto riguarda i gestionali che devono essere interfacciati, le tipologie presenti sul mercato italiano sono le più varie; i più completi sono quelli di tipo ERP (“Enterprise resource planning”, pianificazione delle risorse d’impresa).

Il gestionale (ERP, “Enterprise resource planning”) è un sistema informativo, che integra tutti i processi di business rilevanti di un'azienda (vendite, acquisti, gestione magazzino, contabilità etc.)

Componenti di un ERP:

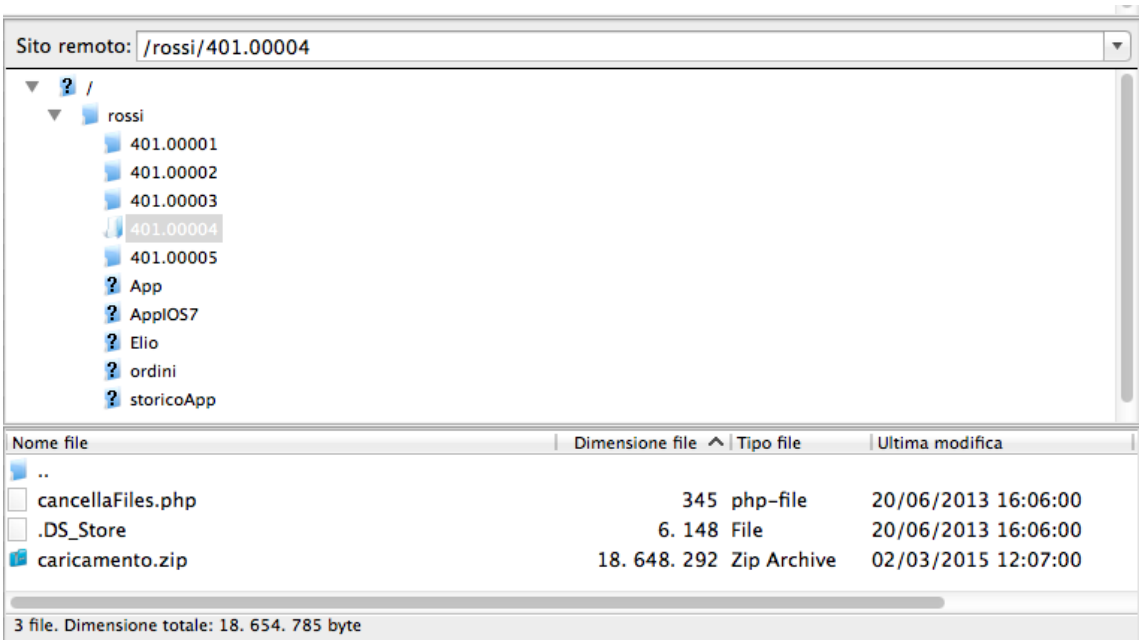
- Contabilità
- Controllo di gestione
- Gestione del personale
- Gestione degli acquisti
- Gestione dei magazzini
- Material Requirements Planning - Pianificazione del fabbisogno dei materiali.
- Gestione della produzione
- Gestione progetti
- Gestione delle vendite

- Gestione della distribuzione
- Gestione della manutenzione impianti
- Gestione degli Asset

Anche se nella maggior parte delle aziende medio-piccole è raro trovare un gestionale di tipo ERP, in tutte le aziende sono sicuramente sviluppati i moduli di contabilità e di gestione del ciclo attivo (vendite). NEMO quindi, oltre ad avere un mercato molto ampio, rappresenta un arricchimento di quest' ultima area.

Il connettore ERP è un insieme di procedure per la sincronizzazione di due o più database, basato su azioni schedulate. Ogni processo ha il compito di prendere i dati da un lato, eventualmente trasformarli, trasportarli dall'altro lato e depositarli in modo differenziale. I dati sono presi da una sorgente specifica (una tabella del db, un file, ecc…), elaborati in base ai parametri definiti e depositati nella destinazione opportuna.

Diversi clienti preferiscono occuparsi in prima persona del connettore al gestionale, in quanto di quest' ultimo padroneggiano le peculiarità. Per alcuni gestionali molto diffusi il connettore è stato predisposto in modo standard e riutilizzato in diversi casi.



Un server FTP è un programma che permette di accettare connessioni in entrata e di comunicare con un client attraverso il protocollo FTP. Il tablet remoto, detto client FTP, invia al server le richieste di trasferimento. Esistono numerosi server FTP per quasi tutti i sistemi operativi esistenti. Un server FTP solitamente attende le richieste del client sulla porta 21, mentre i dati vengono trasmessi sulla porta 20.

Sul server FTP vengono caricati dal connettore ERP del gestionale dei file di testo all'interno di un pacchetto .zip in cui ogni riga del file è un record da caricare sul Database del client e i campi sono separati dal carattere "|" (pipe) se .txt oppure ";" (punto e virgola) se .csv in base all'esigenze del cliente. Nella figura riporto la struttura e il contenuto di un server FTP; a ogni agente è associata una cartella, la cartella contiene se non ancora scaricato un file .zip (al interno del quale sono presenti tutti i file .txt o .csv da cui verrà aggiornato il tablet) e cancellaFiles.php che al termine del aggiornamento del tablet cancellerà in automatico il file .zip dalla cartella associata all'agente. La macchina su cui risiede il server FTP è una Dell dual Xeon Quad core (rack) con Windows Server 2008 e IIS7 con 10 Gb di spazio disco.

In certi casi alcuni campi possono essere superflui (indicati con F "facoltativi") e quindi non inseriti nei files, in altri casi possono essere superflue intere tabelle in questo caso i campi rappresentanti i riferimenti alle tabelle eliminate (indicati con C "condizionali") sono facoltativi ma se le tabelle fossero presenti sarebbero indispensabili, altri invece sono in ogni caso necessari (indicati con O "obbligatori") per il funzionamento del sistema.

I moduli di base richiedono al connettore ERP del Gestionale i seguenti files:

Anagrafica Articoli (articoli.txt)

Campo	**O/F**	**Descrizione**
Codice Articolo	O	Univoco
Descrizione	O	
Note 1	F	
Note 2	F	
Note 3	F	
Note 4	F	
Codice Categoria Sconto	C	non valorizzato se non esiste la tabella Sconto Categoria
Categoria principale	O	Identifica la categoria del prodotto visualizzabile nella tabella della schermata iniziale dell'applicazione
Famiglia prodotto	F	(codice)
Sottofamiglia1	F	(codice)
Sottofamiglia2	F	(codice)
Sottofamiglia3	F	(codice)
Sottofamiglia4	F	(codice)
Codice Fornitore	F	Univoco
Fornitore	F	
Peso	F	
Dimensione	F	
Unità di Misura	F	

Quantità in ordine presso il cliente	F	
Quantità in ordine presso il fornitore	F	
Quantità in giacenza	F	
Numero prodotti per confezione	C	Se viene gestito dal gestionale
Bloccato	F	
Iva	O	
Taglia	F	
Descrizione estesa famiglia	F	

Esempio.

```
ABEF1|Acqua SAN BERNARDO frizz. lt.1X12 VR|||||0|AB|0006|||||601.00055|SAN PELLEGRINO S.p.A.|||CS|0|0|139,000|1,000|N| 22  |00|0|0
ABEF2|Acqua SAN BERNARDO Frizzante cl. 50x20|||||0|AB|0004|||||601.00055|SAN PELLEGRINO S.p.A.|||CS|0|0|8,000|1,000|N| 22  |00|0|0
ABEF3|Acqua SAN BERNARDO frizz. lt.1,5X6 PET|||||0|AA|0003|||||601.00055|SAN PELLEGRINO S.p.A.|||CF|0|0|92,000|1,000|N| 22  |00|0|0
ABEF4|Acqua SAN BERNARDO frizz. lt. 1x12 pet|||||0|AA|0002|||||601.00055|SAN PELLEGRINO S.p.A.|||CF|0|0|27,000|1,000|N| 22  |00|2|0
ABEF5|Acqua SAN BERNARDO frizz. lt. 0,5x24 pet|||||0|AA|0001|||||601.00055|SAN PELLEGRINO S.p.A.|||cf|0|0|183,000|1,000|N| 22  |00|2|0
ABEL1|Acqua SAN BERNARDO liev. lt.1X12 VR|||||0|AB|0006|||||601.00055|SAN PELLEGRINO S.p.A.|||CS|0|0|43,000|1,000|N| 22  |00|0|0
ABEL3|Acqua SAN BERNARDO liev lt.1,5X6 PET|||||0|AA|0003|||||601.00055|SAN PELLEGRINO S.p.A.|||CF|0|0|86,000|1,000|N| 22  |00|0|0
ABEN1|Acqua SAN BERNARDO nat lt. 1X12 VR|||||0|AB|0006|||||601.00055|SAN PELLEGRINO S.p.A.|||CS|0|0|183,000|1,000|N| 22  |00|0|0
ABEN2|Acqua SAN BERNARDO naturale cl. 50x20 vr|||||0|AB|0004|||||601.00055|SAN PELLEGRINO S.p.A.|||cs|0|0|33,000|1,000|N| 22  |00|0|0
ABEN3|Acqua SAN BERNARDO nat. lt 1,5X6 PET|||||0|AA|0003|||||601.00055|SAN PELLEGRINO S.p.A.|||CF|0|0|149,000|1,000|N| 22  |00|0|0
ABEN4|Acqua SAN BERNARDO nat lt. 1X12 PET|||||0|AA|0002|||||601.00055|SAN PELLEGRINO S.p.A.|||CF|0|0|34,000|1,000|N| 22  |00|2|0
```

Anagrafica Clienti (clienti.txt)

Campo	**O/F**	**Descrizione**
Codice Cliente	O	
Ragione sociale	O	Dati di fatturazione
Indirizzo	O	Dati di fatturazione
CAP	O	Dati di fatturazione
Città	O	Dati di fatturazione
Provincia	O	Dati di fatturazione
Stato	O	Dati di fatturazione
Cod Fiscale	F	Dati di fatturazione
Partita IVA	F	Dati di fatturazione
Fatturare a	F	Stringa vuota tranne per gestionale passepartout

Campo	O/F	Descrizione
Categoria sconto	C	Passare 0 o stringa vuota se non esiste la tabella Categoria Sconto
Pagamento	F	foreign key della tabella Pagamenti
Codice Agente 1	O	
Mail del cliente	F	
Mail del customer service	F	
Tipo di cliente	F	Non gestito
Cliente bloccato in	F	S=BLOCCATO;N=NON BLOCCATO
Cliente esente iva	O	1=cliente non esente; 0=cliente esente
Numero Listino	O	Passare fisso 1 per demo
Sconto1	F	sconti fissi legati al cliente prioritari rispetto agli altri tipi di sconto
Sconto2	F	
Sconto3	F	
Sconto4	F	
Sconto5	F	
Telefono	F	
Indirizzo web	F	
Zona	F	riferimento a zona_cod della tabella Zona
Nota cliente	F	

Esempio.

```
501.00470|PERAGA S.R.L.|VIA NAZIONALE, 9|10010|MERCENASCO|TO|IT|04736210016|IT04736210016|501.00470|0||610.00003|||0|N|1|1|0|0|0|0|0
501.00505|MECI MIRELA|STR. PER ZIMONE, 2|10010|PIVERONE|TO|IT|MCEMRL76M53Z129F|IT10577450017|501.00505|0|Rimessa diretta|610.00003|mirella362@virgilio.it||0|N|1|1|0|0|0|0|0
501.00502|CAFFETTERIA PESCI VIVI di Gioffrè Francesco|Corso G. Ferraris, 37|10034|CHIVASSO|TO|IT|GFFFNC82S02L063T|IT11209990016|501.00502|0|Rimessa diretta|610.00003|||0|N|1|1|0|0|0|0|0
501.00501|CAFFE' ENZO di SCAFIDI Vincenzo|Viale Cavour, 8|10034|CHIVASSO|TO|IT|SCFVCN60P12F217O|IT11146050015|501.00501|0|Rimessa diretta|610.00003|||0|N|1|1|0|0|0|0|0
501.00500|BACCO TABACCO & CAFFE' di MALDERA ANTONELLO|Via Castellazzo n° 1|10010|PIVERONE|TO|IT|MLDNNL62A28L219B|IT07567450015|501.00500|0|Rimessa diretta|610.00003|||0|N|1|1|0|0|0|0|0
501.00498|ASS.NE CHICO E NERINA|Campus ex Tav- Via Baraggia|10034|CHIVASSO|TO|IT|09102505001|IT00105243600|501.00498|0|Rimessa diretta|610.00003|||0|N|1|1|0|0|0|0|0
501.00497|GIMMY CUCINA E CAFFE' di PROBO Federica|Via Martini d'Italia, 16|10014|CALUSO|TO|IT|PRBFRC81A41E379V|IT03546830047|501.00497|0|Rimessa diretta 7 gg|610.00003|||0|N|1|1|0|0|0|0|0
501.00494|BAR TRATTORIA NAZIONALE DI FESTA BIANCHET FRANCO|Via Maestra, 49|10030|Villareggia|TO|IT|FSTFNC64E05A859C|IT02150190029|501.00494|0|Rimessa diretta|610.00003|||0|N|1|1|0|0|0|0|0
```

Listini (listini.txt)

Campo	O/F	Descrizione
Codice Articolo	O	Univoco
Prezzo1	O	
Prezzo2	F	
Prezzo3	F	
Prezzo4	F	
Prezzo5	F	
Prezzo6	F	
Prezzo7	F	
Prezzo8	F	
Prezzo9	F	

Al massimo 9 listini per articolo

Esempio.

```
ABEF1|6,60|4,78|5,06|6,60|5,97|6,60|0,00|0,00|0,00
ABEF2|8,55|0,00|0,00|8,55|0,00|8,55|0,00|0,00|0,00
ABEF3|3,38|3,02|3,19|3,38|3,04|3,38|0,00|0,00|0,00
ABEF4|6,05|5,92|5,92|6,05|4,68|6,05|0,00|0,00|0,00
ABEF5|6,30|5,57|7,09|6,30|0,00|6,30|0,00|0,00|0,00
ABEL1|6,60|4,78|5,06|6,60|5,97|6,60|0,00|0,00|0,00
ABEL3|3,38|3,02|3,19|3,38|3,04|3,38|0,00|0,00|0,00
ABEN1|6,60|4,78|5,06|6,60|5,97|6,60|0,00|0,00|0,00
```

Tabella delle Condizioni di Pagamento (pagamenti.txt)

Campo	O/F	Descrizione
Codice condizione di	O	Univoco
Descrizione	O	
% sconto pagamento	O	% sconto relativo alla condizione pagamento Passare 0 se non esiste

Esempio.

```
1|Rimessa diretta|0,00
2|Rimessa diretta 30 gg d.f. f.m.|0,00
3|Rimessa diretta 30/60 gg d.f.|0,00
4|Rimessa diretta 7 gg|0,00
5|Rimessa diretta 60 gg d.f.|0,00
6|Rimessa diretta 30 gg d.f.|0,00
7|Rimessa diretta 90 gg d.f.|0,00
8|Rimessa diretta 60 gg d.f. f.m.|0,00
```

Anagrafica Agenti (utenti.txt)

Campo	O/F	Descrizione
User ID	O	Utilizzato per l'accesso all'applicazione
Codice agente	O	Codice agente intestatario della user id.
Nome agente	O	
Tipologia utente	O	A= Può MODIFICARE DATI U= NON PUO MODIFICARE I DATI
Indirizzo Mail	F	

Esempio.

```
FONSIM|610.00001|FONDACARO SIMONE|A||0
FONANT|610.00002|FONDACARO ANTONIO|A||0
RUZLUC|610.00003|RUZZA LUCA|A||0
BAUMAU|610.00004|BAUDINO MAURO|A||0
```

Gruppi Merceologici (categoria.txt)

Campo	O/F	Descrizione
Codice categoria	O	
Codice famiglia	F	
Cod subfam1	F	
Cod subfam2	F	
Cod subfam3	F	
Cod subfam4	F	
Descrizione	F	Relativamente all'ultimo codice famiglia inserito

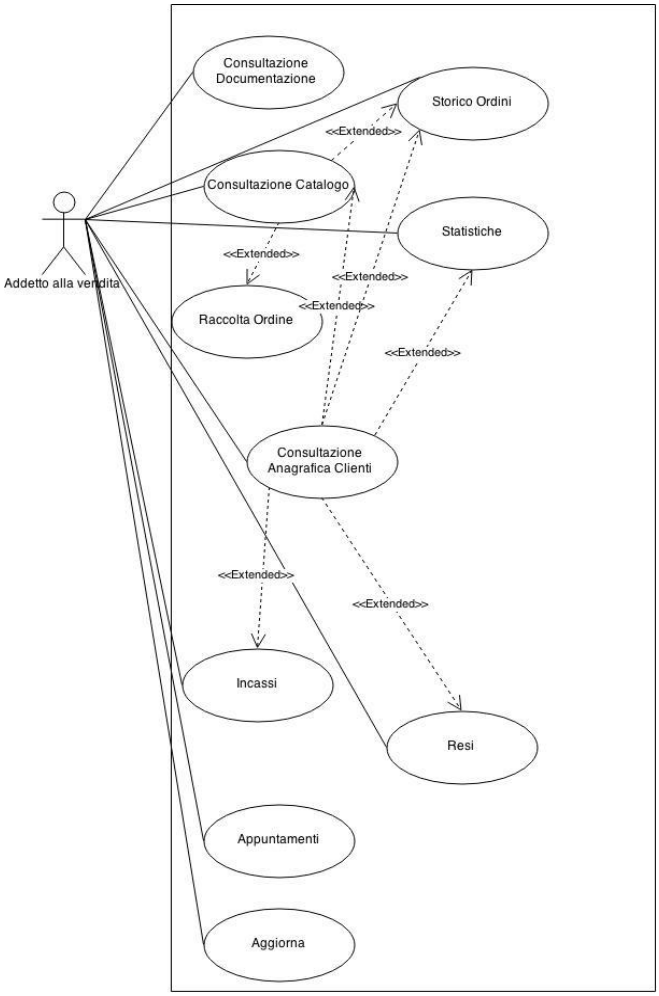
In questa tabella viene simulato un albero di categorie e sottocategorie (denominate famiglie) di prodotto la cui descrizione si riferisce alla foglia di questo "albero".

Esempio.

```
AA|0001||||PET CL 50
AA|0002||||PET LT 1
AA|0003||||PET LT 1,5
AB|0004||||VETRO CL 50 VR
AB|0005||||VETRO CL 75 VR
AB|0006||||VETRO LT1 VR
AC|0007||||BIRRE NAZIONALI
AC|0008||||BIRRE D'IMPORTAZIONE
```

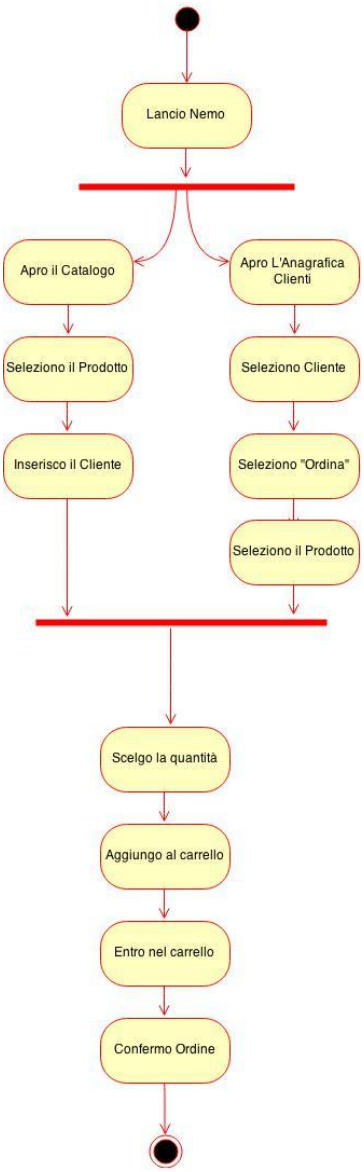
2. Client

1. Specifiche software

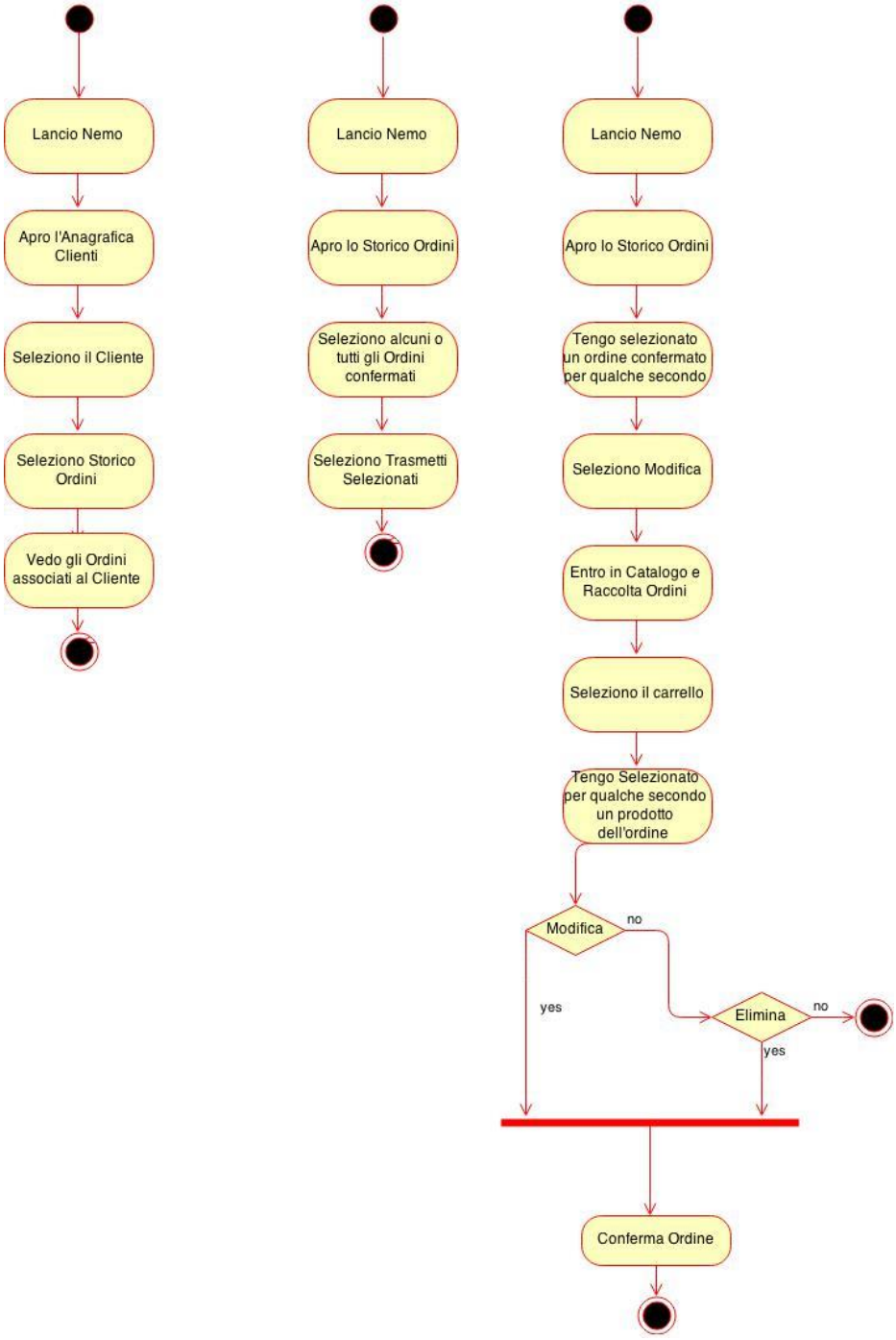


Il diagramma dei casi d'uso sopra rappresentato formalizza e spiega i requisiti funzionali del sistema in esame; infatti ogni utilizzatore può:

- Consultare il catalogo
- Consultare documenti
- Raccogliere un ordine
- Consultare l'anagrafica dei clienti
- Consultare gli appuntamenti
- Consultare le statistiche (per cliente, per prodotto)
- Consultare lo storico ordini
- Aggiornare i dati su tablet
- Visualizzare e inserire i resi e gli incassi dai clienti

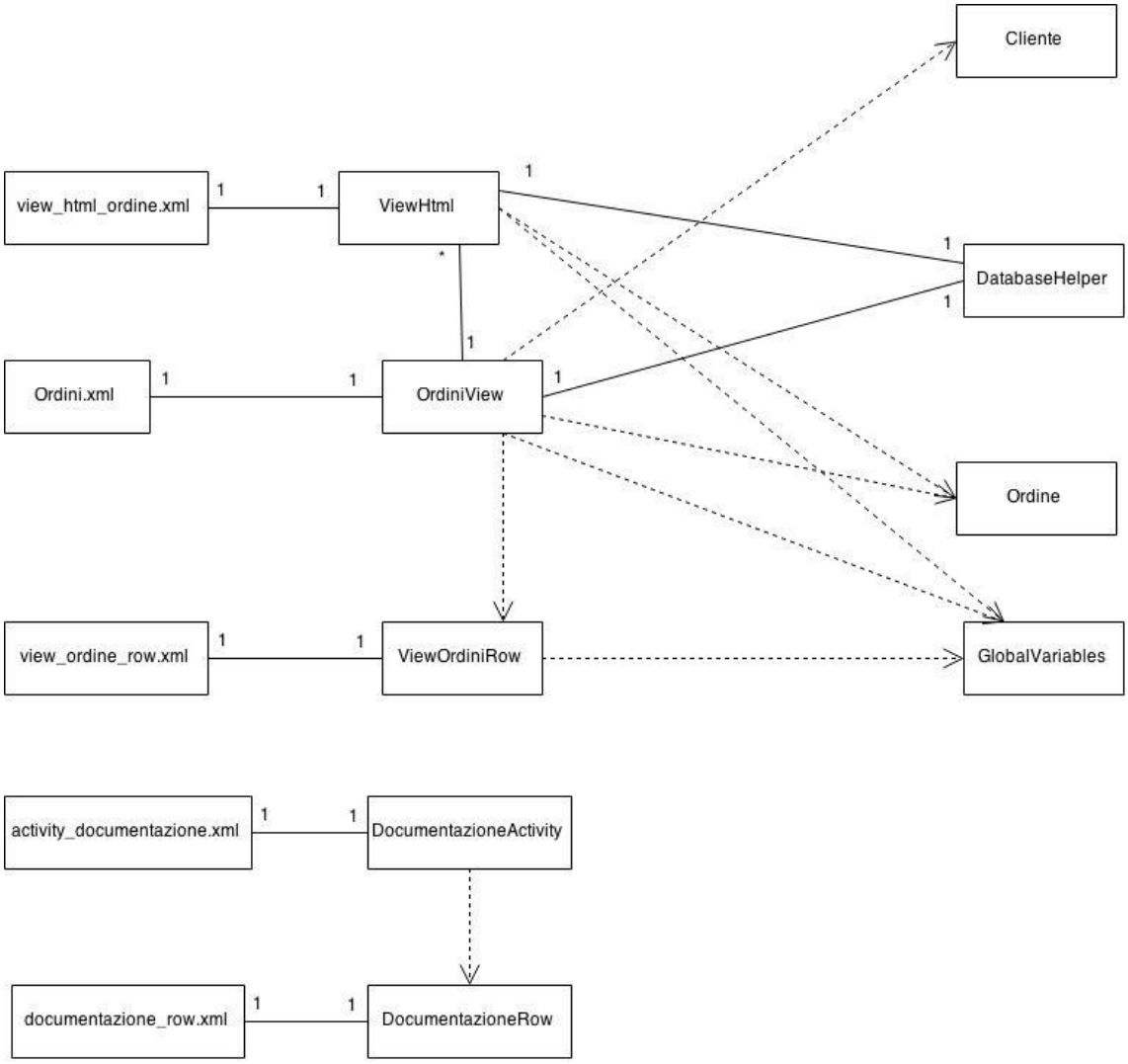


Questo diagramma delle attività esplicita le operazioni necessarie per la raccolta dell'ordine, infatti per la raccolta dell'ordine si può partire dall'anagrafica clienti e poi inserire i prodotti oppure dal catalogo partendo dal prodotto e inserendo successivamente il cliente.



I diagrammi delle attività sopra riportati formalizzano le operazioni da effettuare per:

- Visualizzare lo storico ordini di uno specifico cliente
- Trasmettere l'ordine al gestionale
- Modificare un ordine confermato (ma non ancora trasferito su gestionale)



Il diagramma delle classi sopra riportato formalizza le dipendenze nelle classi da me realizzate.

- OrdiniView: l'activity associata alla vista principale del modulo storico ordini.
- ViewOrdiniRow: rappresenta la singola riga all'interno della vista principale del modulo storico ordini.
- Ordine: rappresenta un ordine ed esegue le query necessarie all'oggetto sul database.
- GlobalVariables: rappresenta un insieme di variabili globali statiche.
- ViewHtml: l'activity associata alla visualizzazione del ordine.
- DocumentazioneActivity: l'activity associata alla vista principale del modulo documentazione.
- DocumentazioneRow: rappresenta la singola riga all'interno della vista principale del modulo documentazione.

- DatabaseHelper: rappresenta la classe che si occupa di effettuare alcune fondamentali operazioni su database.
- Cliente: rappresenta un cliente ed esegue le query necessarie all'oggetto su database.
- View_html_ordine.xml: rappresenta il layout associato all' activity ViewHtml.
- Ordini.xml: rappresenta il layout associato all'activity OrdiniView.
- View_ordine_row.xml: rappresenta il layout associato alla classe ViewOrdiniRow.
- Activity_documentazione.xml: rappresenta il layout associato all'activity DocumentazioneActivity.
- Documentazione_row.xml: rappresenta il layout associato alla classe DocumentazioneRow.

2. Database

Il Database sul client è stato creato utilizzando il tool "MesaSQLite" ed è stato strutturato come segue :

Tabella Taglie

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	Primary Key
art	Varchar	Foreign Key - Articolo
taglia	Varchar	

Tabella Statistiche

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	Primary Key
codCli	Varchar	Foreign Key - Cliente
numFat	Varchar	
dateFat	Date	
codArt	Varchar	Foreign Key - Articolo
importo	Float	default 0
ragsoc	Varchar	
desArt	Varchar	
qta	Integer	default 0
categArt	Varchar	Foreign Key - Categoria
codAge1	Varchar	
codAge2	Varchar	

Tabella Sconti Categoria

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key
categoriaCliente	Integer	Foreign Key - default 0
categoriaArticolo	Integer	Foreign Key
sconto1	Float	default 0
sconto2	Float	default 0
sconto3	Float	default 0
sconto4	Float	default 0
sconto5	Float	default 0

Tabella Sconti

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key
sconti_codCli	Varchar	Foreign Key - Cliente
sconti_codFor	Varchar	Foreign Key - Fornitore
sconti_sc1	Float	default 0
sconti_sc2	Float	default 0
sconti_sc3	Float	default 0
sconti_sc4	Float	default 0
sconti_sc5	Float	default 0

Tabella Resi

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key - Autoincrement
id	Varchar	identificativo progressivo gestito dal gestionale
cliente	Varchar	Foreign Key - Cliente
dataSostituzione	Datetime	Data sostituzione
artDaSostituire	Varchar	Foreign Key - Articolo
qtaDaSostituire	Integer	Quantità da sostituire
artSostituito	Varchar	Foreign Key - Articolo
qtaSostituita	Integer	Quantità Sostituita
resoIntegro	Varchar(1)	Flag
resoRotto	Varchar(1)	Flag
accreditato	Varchar(1)	Flag
sostituito	Varchar(1)	Flag
ritiratoDaAge	Varchar(1)	Flag
note	Varchar(500)	

codAge1	Varchar	Foreign Key - Agente
codAge2	Varchar	
ragsoccli	Varchar	Ragione sociale cliente
desartdasost	Varchar	Descrizione articolo da sostituzione

Il reso può essere sostituito oppure accreditato

Tabella Provincia

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	
prov_codprov	Varchar(2)	Primary Key
prov_desprov	Varchar	Descrizione Provincia

Tabella Particolarità

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	Primary Key
tipoParticolarita	Varchar(1)	
codiceConto	Varchar(9)	
categoriaConto	Integer	default 0
codiceArticolo	Varchar(9)	
categoriaArticolo	Integer	default 0
finoA	Varchar	
quantita1	Varchar	
valore1	Varchar	
valore2	Varchar	
quantita3	Varchar	
valore3	Varchar	
valore4	Varchar	

Tabella Paese

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	
paese_codpaese	Varchar(3)	Primary Key
paese_despaese	Varchar	Descrizione paese

Tabella Nota Clienti

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	Primary Key
codCli	Varchar	Primary Key - Codice cliente

nota	Varchar	

Tabella Listini

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	Primary Key - Autoincrement
list_codArt	Varchar(32)	Foreign Key - Articolo
list_prezArt	Float	Prezzo
list_numList	Integer	Numero listino

Tabella Disegno

Nome Campo	**Tipo**	**Descrizione - Note**
immagine	Varchar	
Serial	Integer	Primary Key
art	Varchar	Foreign Key - Articolo
disegno	Varchar	

Tabella Colori

Nome Campo	**Tipo**	**Descrizione - Note**
immagine	Varchar	
Serial	Integer	Primary Key
art	Varchar	Foreign Key - Articolo
colore	Varchar	

Tabella Anagrafica Famiglie

Nome Campo	**Tipo**	**Descrizione - Note**
Serial	Integer	
tblanagfam_codcat	Varchar	Primary Key
tblanagfam_codfam	Varchar	Primary Key
tblanagfam_desfam	Varchar(50)	
tblanagfam_ico_Ink	Varchar(50)	icona
tblanagfam_notefam	Varchar(1000)	

Tabella Anagrafica Categorie

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key - default 0
tblanagcat_codcat	Varchar	Primary Key
tblanagcat_descat	Varchar(50)	

Tabella Anagrafica Clienti

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	
cli_codcli	Varchar(10)	Primary Key - Cliente
cli_ragsoc	Varchar	
cli_indiri	Varchar(70)	
cli_cap	Varchar(20)	
cli_citta	Varchar(50)	
cli_prov	Varchar(2)	
cli_paese	Varchar(3)	
cli_cf	Varchar(14)	
cli_piva	Varchar(14)	
cli_fatturareA	Varchar(9)	
cli_categoriaSconto	Integer	
cli_pag	Varchar(10)	
cli_codage1	Varchar(8)	
cli_codage2	Varchar(8)	
cli_mailcli	Varchar(50)	
cli_mailcs	Varchar(50)	
cli_tipcli	Varchar(10)	
cli_flgblocco	Varchar(1)	
cli_iva	Integer	default 0
cli_listino	Integer	
cli_sconto1	Float	default 0
cli_sconto2	Float	default 0
cli_sconto3	Float	default 0
cli_sconto4	Float	default 0
cli_sconto5	Float	default 0
cli_telefono	Varchar	
cli_web	Varchar	
cli_zona	Varchar	Foreign Key - Tabella Zona
cli_nota	Varchar	

Tabella Anagrafica Articoli

Nome Campo	Tipo	Note
Serial	Integer	
tblanag_art	Varchar(20)	Primary Key - Articolo
tblanag_art_desc	Varchar(80)	
tblanag_note1	Varchar(400)	
tblanag_note2	Varchar(400)	
tblanag_note3	Varchar(400)	
tblanag_note4	Varchar(400)	
tblanag_cod_categ	Varchar	Categoria
tblanag_cod_famprod	Varchar	
tblanag_cod_subfam1	Varchar	subfam2,subfam3,subfam4
tblanag_prod	Varchar	
tblanag_peso	Real	default 0
tblanag_dim	Varchar(20)	
tblanag_unimis	Varchar(3)	Unità di Misura
tblanag_qtaord_cli	Integer	
tblanag_qtagiac	Integer	Giacenza
tblanag_lst_upd	Timestamp	
tblanag_bloc	Varchar(1)	Fuori Produzione - Catalogo
tblanag_iva	Integer	default 0
tblanag_taglia	Varchar(2)	
tblanag_qtaord_for	Integer	Quantità ordinata dal fornitore
tblanag_cod_categ_sconto	Integer	Codice Categoria Sconto
tblanag_desEstFam	Varchar	Descrizione estesa Famiglia
tblanag_codprod	Varchar(50)	codice produttore
tblanag_numProdPerConf	Float	default 0

Tabella Indirizzo Spedizione

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key
codcli	Varchar(10)	Cliente
ind	Varchar(10)	Identificativo indirizzo di consegna per elenco
ragsoccons	Varchar	rangione sociale cliente
indiricons	Varchar	indirizzo
capcons	Varchar(7)	
cittacons	Varchar(50)	
prov	Varchar(2)	

Tabella Tipo Pagamento Incassi

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key - Autoincrement
codtippag	Varchar	Codice Pagamento
destippag	Varchar	Descrizione

Tabella Incassi

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key - Autoincrement
azienda	Varchar(3)	Codice Identificativo del gestionale solo visualizzato
cliente	Varchar(9)	Codice Cliente
annoreg	Integer	default 0
seriedoc	Integer	default 0
numdoc	Integer	default 0
datadoc	Varchar	
prgriga	Integer	progressivo di riga - default 0
tipopag	Varchar	
statopag	Varchar	
datascad	Date	
importo	Float	default 0
tiporeg	Varchar(2)	Tipo registrazione
importopagato	Float	default 0
importoabbuono	Float	default 0
ragsoccli	Varchar	
codAge1	Varchar	
codAge2	Varchar	
Trasferito	Varchar	
dataIncasso	Varchar	
tipoPagamento	Integer	default 0

Tabella Zona

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key
zona_cod	Varchar(2)	Codice Zona
zona_des	Varchar	Descrizione Zona

Tabella Anagrafica Utenti

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	
user_id	Varchar	userid accesso applicazione - Primary Key
codage	Varchar	codice agente - Valorizzato se l'utente è un agente (può inserire ordini)
user_name	Varchar	Nome utente
Tipo_user	Varchar	A= Agente (inserire i suoi ordini) S="Super user" , area manager o altro, può vedere e modificare tutti gli ordini presenti sul tablet U= Utente generico, con sola visualizzazione catalogo
User_mail	Varchar	Indirizzo di posta dell'utente
progorrd	Integer	progressivo numero ordine - incremento ogni ordine confermato
Usr_passwd	char	Password attuale - Per ora non utilizzata
Dt_cambio_passwd	Date	Data di cambio password - Per ora non utilizzata
Password old	Char	Vecchia Password - Per ora non utilizzata

Tabella Ordini

Nome Campo	Tipo	Descrizione - Note
Serial	Integer	Primary Key - autoincrement
numOrd	Varchar(30)	Numero ordine - cod.Agente + nr. ordine
codcli	Varchar(15)	Codice cliente
ragsoc	Varchar	Ragione sociale
codage	Varchar(8)	Codice agente
codart	Varchar(20)	Codice articolo
desart	Varchar(80)	Descrizione Articolo
qtaord	Integer	Quantità ordinata
przuni	Float	Prezzo unitario
sconto1	Float	sconto 1
sconto2	Float	sconto 2
sconto3	Float	sconto 3
importo	Float	importo totale riga ordine
ragsocsped	Varchar	Rag. Soc. spedizione
indsped	Varchar	indirizzo di spedizione
capsped	Varchar	cap spedizione
cittasped	Varchar	città spedizione
provsped	Varchar	Prov. Spedizione

paesesped	Varchar	paese spedizione - chiave esterna tabella paese
dataord	Timestamp	data dell'inserimento dell'ordine (cioè la data più vecchia) - In formato YYYYMMDD HH:mm:SS
datacons	Date	data consegna - in formato YYYYMMDD
incoterm	Varchar	data di trasferimento
ordpag	Varchar	codice pagamento
note	Varchar	note
referente	Varchar	referente cliente
stsord	Varchar	stato ordine I:iniziale C:confermato T:trasferito
mailcli	Varchar	Mail cliente
mailcs	Varchar	Mail customer service
indcli	Varchar	Indirizzo cliente
capcli	Varchar	CAP cliente
cittacli	Varchar	Città cliente
provcli	Varchar	Provincia cliente
paesecli	Varchar	Paese cliente
pivacli	Varchar	P. Iva cliente
disegno	Varchar	
colore	Varchar	
ordiva	Float	
ordcf	Varchar	
ordnotecolore	Varchar	
ordconfezioni	Integer	
ordtaglia	Varchar	
codclisped	Varchar	
ord_quantitaPerConf	Float	

6. Implementazione

Nell'implementazione di un'applicazione Android è necessario conoscere Java per la programmazione dell'App e l'XML per il disegno del layout grafico (anche se è possibile disegnare il layout dinamicamente da Java).

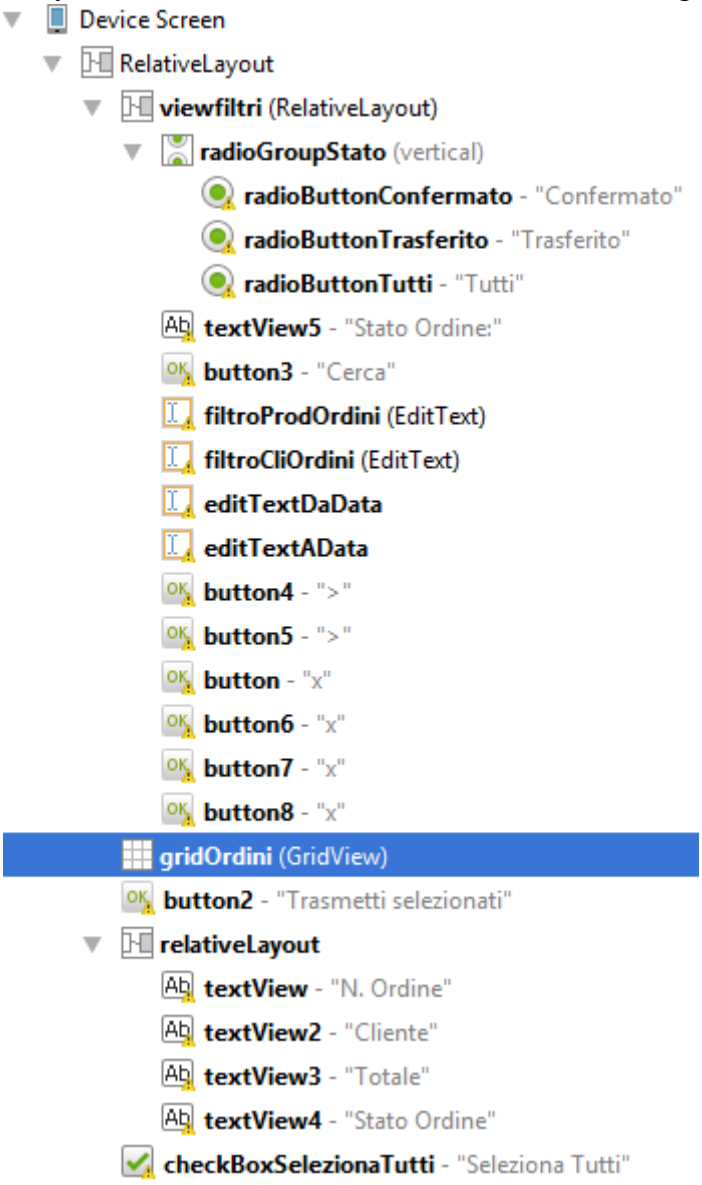
XML (sigla di eXtensible Markup Language) è un linguaggio di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo. Costituisce il tentativo di produrre una versione semplificata di Standard Generalized Markup Language (SGML) che consente di definire in modo semplice nuovi linguaggi di markup da usare in ambito web. Il nome indica quindi che si tratta di un linguaggio marcatore (markup language) estensibile (eXtensible) in quanto permette di creare tag personalizzati.

1. **Layout**
   Un layout definisce la struttura visiva di una interfaccia utente, ad esempio l'interfaccia utente per un widget, un'attività o un'applicazione. È possibile definire un layout in due modi:
   - Dichiarando gli elementi dell'interfaccia utente in XML. Android fornisce un vocabolario XML semplice che corrisponde alle classi e sottoclassi della View, come quelli per i widget e layout.
   - Istanziando elementi di layout in fase di esecuzione. L'applicazione può creare Vista e ViewGroup (e modificare le loro proprietà) durante la programmazione.

   Il framework Android offre la flessibilità di utilizzare uno o entrambi questi metodi per la dichiarazione e la gestione dell'interfaccia utente dell'applicazione. Ad esempio, si potrebbe dichiarare layout predefiniti della applicazione in XML, compresi gli elementi della schermata che appariranno e le loro proprietà. Si potrebbe quindi aggiungere al codice dell'applicazione lo stato degli oggetti di pagina, incluse quelle dichiarate in XML, in fase di esecuzione.
   Il vantaggio per dichiarare l'interfaccia utente in XML è che permette di separare meglio la presentazione della applicazione dal codice che controlla il suo comportamento. Le descrizioni della UI sono esterne al codice dell'applicazione, il che significa che è possibile modificarle o adattarle senza dover modificare il codice sorgente e ricompilare. Ad esempio, è possibile creare layout XML per i diversi orientamenti dello schermo, diverse dimensioni dello schermo del dispositivo, e lingue diverse. Inoltre, dichiarando il layout in XML si rende più facile da visualizzare la struttura per l'utente.

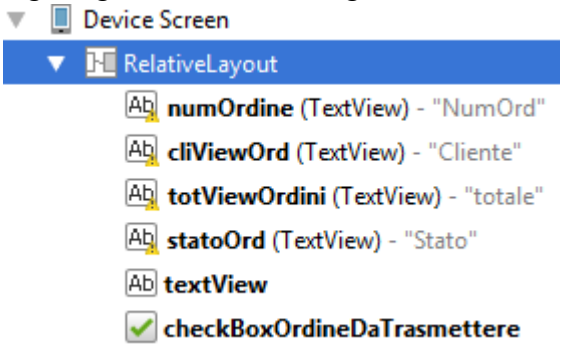
In generale, il vocabolario XML per dichiarare elementi dell'interfaccia utente segue da vicino la struttura e la denominazione delle classi e dei metodi, in cui nomi di elementi corrispondono ai nomi di classe e attribuiscono nomi corrispondenti ai metodi. In realtà, la corrispondenza è spesso così diretta che si può intuire a cosa corrisponde l'attributo XML, o indovinare quale classe corrisponde ad un determinato elemento XML. Tuttavia, non tutto il vocabolario è identico. In alcuni casi, ci sono piccole differenze di denominazione. Ad esempio, l'elemento EditText ha un attributo text corrispondente al EditText.setText ().
Nell'applicazione realizzata si è scelto di disegnare 2 tipi di layout in modo da poter essere visualizzata su entrambi i device di test a disposizione: un google Nexus 7 (7 pollici) e un Asus Trasformer Prime TF2011 da 10 pollici.

Il layout dello “Storico Ordini” è strutturato come segue:

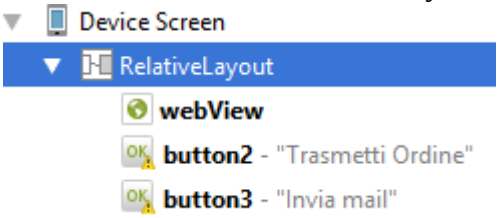


È stato scelto il RelativeLayout in quanto è un potente strumento per la progettazione di una interfaccia utente in quanto in grado di eliminare i gruppi di visualizzazione nidificati e mantenere la gerarchia di layout piatta, il che migliora le prestazioni.
Il radioGroup contiene i radioButton per filtrare gli ordini in base al loro stato Trasferito, Confermato o entrambi.
Il button “Cerca” avvia i filtri di ricerca.
L’editText filtroProdOrdini e filtroCliOrdini filtrano per descrizione prodotto e cliente gli ordini da visualizzare. Il realativeLayout figlio del principale

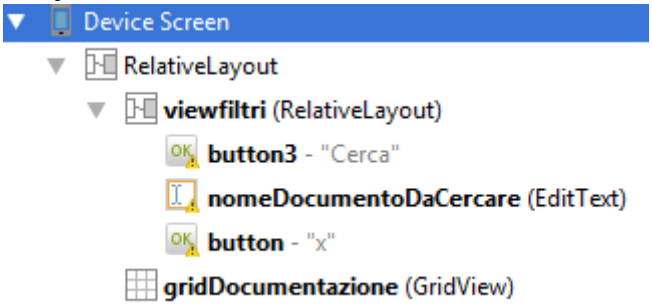
contenete 4 textView rappresenta lo schema della visione tabellare degli ordini filtrati.
La GridView gridOrdini conterrà le righe degli ordini filtrati.
Ogni riga all'interno della gridOrdini è strutturata come segue:



Se è un ordine non ancora stato trasferito tenendo cliccata la riga per alcuni secondi c'è la possibilità di modificare nuovamente l'ordine passando attraverso il carrello.
Cliccando sulla riga si aprirà una vista in cui verrà visualizzato l'ordine sotto forma di html all'interno di un layout strutturato come segue:

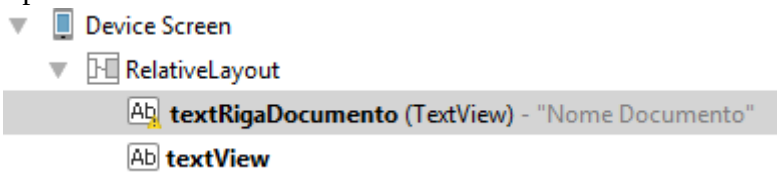


Dove il button "Trasmetti Ordine" manderà sul server FTP l'ordine da caricare a gestionale (se non ancora trasferito) e con "Invia mail" manda a un indirizzo mail come copia commissione il file html.
Il layout del modulo "Documentazione" è strutturato come segue:



Dove il button "Cerca" filtra i documenti in base all'editText nomeDocumentoDaCercare e li visualizza nella GridView gridDocumentazione.

Ogni riga della gridDocumentazione è strutturata con la TextView textRigaDocumento contenente il nome del file e una textView che fa da separatore tra i documenti.



2. **Java**

   In informatica Java è un linguaggio di programmazione orientato agli oggetti, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione. Uno dei principi fondamentali del linguaggio è espresso dal motto write once, run anywhere (WORA): il codice compilato che viene eseguito su una piattaforma non deve essere ricompilato per essere eseguito su una piattaforma diversa. Il prodotto della compilazione è infatti in un formato chiamato bytecode che può essere eseguito da una qualunque implementazione di un processore virtuale detto Java Virtual Machine.
   Java venne creato per soddisfare quattro scopi:
   - essere orientato agli oggetti;
   - essere indipendente dalla piattaforma;
   - contenere strumenti e librerie per il networking;
   - essere progettato per eseguire codice da sorgenti remote in modo sicuro.

Qui di seguito spiego le classi di cui sono autore e ne commento alcuni frammenti di codice.

OrdiniView.java è l'Activity che gestisce il layout dello "Storico Ordini" gli attributi che gestisce sono

```
//classe per gestione database
public DatabaseHelper myDbHelper=new DatabaseHelper(this);
//Ordini non trasmessi al gestionale
public ArrayList<String> ordiniNonTrasmessi;
//numeriOrdini degli Ordini filtrati
public ArrayList<String> numeriOrdini;
//statoOrdini degli Ordini filtrati
public ArrayList<String> statoOrdini;
//gestione filtri date
protected int mYear;
protected int mMonth;
protected int mDay;
//textview dove inserisco la data
protected TextView mDateDisplay;
```

All'OnCreate controlla se è stata lanciata passando dall'anagrafica clienti controllando la variabile globale relativa o se è stata lanciata dopo aver effettuato una trasmissione di ordini nel qual caso effettua una transizione.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.ordini);

    EditText txtFiltroCli=(EditText)findViewById(R.id.filtroCliOrdini);

    //controllo se l'ho richiamata da anagrafica clienti(in questo caso applico il filtro cliente)
    if (GlobalVariables.codCliGlobaleDaAnagrafica!="")
    {
        Cliente cli=new Cliente();
        cli.codCli=GlobalVariables.codCliGlobaleDaAnagrafica;

        Map<String,String> infoCli=cli.infoCliente(myDbHelper.prendiDb());

        txtFiltroCli.setText(infoCli.get("ragSoc"));
        GlobalVariables.codCliGlobaleDaAnagrafica=""; //resetto variabile globale
    }
    //Se l'activity viene lanciata dopo aver trasmesso gli ordini effettua una transizione
    if (GlobalVariables.onResumeViewOrdDopoTrasmissione == true)
    {
        GlobalVariables.onResumeViewOrdDopoTrasmissione = false;
        overridePendingTransition(0,0);
    }
    //lancio il filtro
    eseguiFiltro();

}
```

In eseguiFiltro() creo un occorrenza di Ordine e la popolo con le informazioni filtro, dichiaro e istanzio una Map rappresentante tutte le righe della tabella Ordini.

```
    //riempio i dati degli ordini
    Map<String, ArrayList<String>> righeOrdine;
    Ordine ordine= new Ordine();
    ordine.desArticolo=txtFiltroProdotto.getText().toString();
    ordine.ragSoc=txtFiltroCli.getText().toString();
    ordine.daData=txtDaData.getText().toString();
    ordine.aData=txtAData.getText().toString();
```

Per popolare l'oggetto rappresentante le righe della tabella lancio dall'oggetto di classe Ordine il metodo selectOrdineStato passando come parametri il Database e lo statoOrd filtrato attraverso i radioButton.

```
righeOrdine= ordine.selectOrdineStato(myDbHelper.prendiDb(),statoOrd);
```

In selectOrdineStato viene lanciata questa select se stato ordine è una stringa vuota:

```
String select="SELECT  ord_numOrd,ord_ragsoc,ord_stsord," +
        "SUM(ord_importo),SUM(ord_iva),ord_desart " +
        "FROM tblordini WHERE ord_numOrd!='Aperto' AND ord_codage='"+GlobalVariables.codAgeGlobal+"'" +
        " AND ord_stsord!='I' AND ord_ragsoc LIKE '%"+ragSoc+"%'"+
        " AND ord_dataord BETWEEN '"+daData+"' AND '"+aData
        +"' GROUP BY ord_numOrd,ord_ragsoc ORDER BY ord_dataord DESC";
```

Viceversa viene lanciata questa select:

```
String select="SELECT  ord_numOrd,ord_ragsoc,ord_stsord," +
        "SUM(ord_importo),SUM(ord_iva),ord_desart " +
        "FROM tblordini WHERE ord_numOrd!='Aperto' AND ord_codage='"+GlobalVariables.codAgeGlobal+"'" +
        "AND ord_stsord='"+ _statoOrd+ "' AND ord_ragsoc LIKE '%"+ragSoc+"%'"+
        " AND ord_dataord BETWEEN '"+daData+"' AND '"+aData+
        "' GROUP BY ord_numOrd,ord_ragsoc" +
        " ORDER BY ord_dataord DESC";
```

Per cui per ogni riga viene passato il numero ordine, la ragione sociale del cliente, lo stato dell'ordine, l'importo totale dell'ordine, l'importo totale dell'IVA e la descrizione di ogni articolo del ordine. Finché l'ordine non è confermato il numero Ordine è uguale ad "Aperto" per cui controllo che ne sia diverso, che il codice agente sia uguale alla variabile globale associata, che lo stato ordine sia diverso da I (iniziato, cioè ancora nel carrello), che la ragione sociale del cliente sia simile alla stringa passata come parametro di filtro e che la data dell'ordine sia tra la data di partenza e la data di arrivo del filtro, raggruppando per numero ordine e per ragione sociale e ordinando in modo decrescente per data ordine.

Per popolare le righe della tabella istanzio un oggetto viewOrdiniRow passando come parametri il contesto, il layout e le colonne associate a ciò che ci interessa visualizzare nelle righe (numero ordine, ragione sociale cliente, stato ordine, importo).

```
viewOrdiniRow customGridAdapter = new viewOrdiniRow(this.getBaseContext(), R.layout.view_ordini_row,
        righeOrdine.get("numOrd"),
        righeOrdine.get("cliente"),
        righeOrdine.get("stato"),
        righeOrdine.get("importo"));
```

Per visualizzare le righe della tabella associo la GridView all'oggetto rappresentante le righe.

```
gridview.setAdapter(customGridAdapter);
```

Aggiungo l'ascoltatore del click su un elemento della tabella, che apre file numeroOrdine.html associato all'ordine selezionato creato durante conferma dell'ordine nel carrello.

```
gridview.setOnItemClickListener((arg0, v, position, arg3) -> {
        Map<String, ArrayList<String>> righeOrdine;
        Ordine ordine= new Ordine();
        ordine.desArticolo=txtFiltroProdotto.getText().toString();
        ordine.ragSoc=txtFiltroCli.getText().toString();
        ordine.daData=txtDaData.getText().toString();
        ordine.aData=txtAData.getText().toString();

        righeOrdine= ordine.selectOrdineStato(myDbHelper.prendiDb(), statoOrd);
        ArrayList<String> numeroOrd;
        numeroOrd=righeOrdine.get("numOrd");

        Intent intent = new Intent(
                getApplicationContext(),
                ViewHtml.class
        );


        intent.putExtra("numOrdine",numeroOrd.get(position).toString());
        startActivity(intent);
});
```

Aggiungo l'ascoltatore del click prolungato su un elemento della tabella, che apre un menu contestuale che dà la possibilità, se non ancora trasferito, di modificare l'ordine.

```
gridview.setOnItemLongClickListener((arg0, v, position, arg3) -> {

        Map<String, ArrayList<String>> righeOrdine;
        Ordine ordine= new Ordine();
        ordine.desArticolo=txtFiltroProdotto.getText().toString();
        ordine.ragSoc=txtFiltroCli.getText().toString();
        ordine.daData=txtDaData.getText().toString();
        ordine.aData=txtAData.getText().toString();

        righeOrdine= ordine.selectOrdineStato(myDbHelper.prendiDb(),statoOrd);
        ArrayList<String> numeroOrd;
        numeroOrd=righeOrdine.get("numOrd");
        String statoOrdineCorrente=righeOrdine.get("stato").get(position).toString();
        if(statoOrdineCorrente.equals("C")) {
            dialogModifica(numeroOrd.get(position).toString());
        }
        return true;
});
```

La trasmissione degli ordini avviene passando attraverso questo metodo che, creando un nuovo thread, evita lo stallo dell'applicazione.

```
public void trasmettiSelezionati(View view)
{

    ordiniNonTrasmessi=new ArrayList<String>();
    final ProgressDialog dialog = ProgressDialog.show(this, "", "Uploading file...", true);

    new Thread((Runnable) () -> {

            int selezionati = 0;

            for (int i=0;i<GlobalVariables.arrayCheckBoxOrdiniDaTrasmettere.size();i++)
            {
                if (GlobalVariables.arrayCheckBoxOrdiniDaTrasmettere.get(i).toString().equals("1"))//1==checked
                {
                    if (statoOrdini.get(i).toString().equals("C"))//lo trasmetto solo se non è già trasferito
                    {
                         selezionati++;
                        avviaTrasmissioneOrdine(numeriOrdini.get(i).toString());
                    }
                }
            }
            dialog.dismiss();
```

Se per un qualsiasi motivo alcuni ordini non sono stati trasmessi viene popolato una stringa di avviso con il numero degli ordini non spediti e viene visualizzato tra Looper.prepare() e Looper.loop() per parlare con il Thread principale che è l'unico che può modificare l'interfaccia grafica.

```
if(ordiniNonTrasmessi.size()>0)
{
    String avviso=new String("Errore nel trasferimento dei seguenti ordini:");

    for (int ord=0;ord<ordiniNonTrasmessi.size();ord++)
    {
        avviso+=ordiniNonTrasmessi.get(ord).toString();
        if (ord<ordiniNonTrasmessi.size()-1)
        {
            avviso+=", ";//metto la virgola per separare i numeri ordine tranne per l'ultimo
            //perchè dopo l'ultimo non ce n'è più
        }
    }
    Looper.prepare();
    AlertDialog.Builder builder = new AlertDialog.Builder(OrdiniView.this);
    builder.setMessage(avviso);
    builder.setCancelable(true);
    builder.setPositiveButton("OK", (dialog, id) -> {
            //do things
            dialog.dismiss();
            GlobalVariables.onResumeViewOrdDopoTrasmissione=true;
            OrdiniView.this.onResume();
    });

    builder.show();

    Looper.loop();

}
```

Avvia trasmissione ordine richiede come parametro il numero del ordine da trasmettere prima dichiara e valorizza tutte le variabili necessarie (HttpURLConnection, DataOutputStream,URL) alla connessione e al trasferimento dei dati al server.

```
public int avviaTrasmissioneOrdine(String numOrd)
{
    HttpURLConnection connection ;
    DataOutputStream outputStream ;

    String lineEnd = "\n";
    String twoHyphens = "--";
    String boundary =  "*****";
    int serverResponseCode=0;
    SQLiteDatabase db=myDbHelper.prendiDb();
    String upLoadServerUri = "http://192.168.128.146:8888"+"/ANDROID/ordini/upload.php";
    try
    {

        URL url = new URL(upLoadServerUri);
        connection = (HttpURLConnection) url.openConnection();

        // Allow Inputs & Outputs
        connection.setDoInput(true);
        connection.setDoOutput(true);
        connection.setUseCaches(false);

        // Enable POST method
        connection.setRequestMethod("POST");

        connection.setRequestProperty("Connection", "Keep-Alive");
        connection.setRequestProperty("Content-Type", "multipart/form-data;boundary="+boundary);

        outputStream = new DataOutputStream( connection.getOutputStream() );
        outputStream.writeBytes(twoHyphens + boundary + lineEnd);
        outputStream.writeBytes("Content-Disposition: form-data; name=\"uploadedfile\";filename=\"" + numOrd+".txt" +"\"" + lineEnd);
        outputStream.writeBytes(lineEnd);
```

Poi popola il file prima effettuando due query (una per ottenere i dati di intestazione e l'altra per il contenuto del ordine) e poi concatenandone i contenuti ritornati dalle query.

```
//contenuto del file
String contenutoFile="";
Ordine ordine= new Ordine();

ordine.numeroOrdine=numOrd;

Map<String,String> testataOrd=ordine.selectTestataOrdine(db);


Map<String, ArrayList<String>> righeOrdine;

righeOrdine= ordine.selectDettagliOrdine(myDbHelper.prendiDb());

//cambio il formato della data per renderla più leggibile
SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss");
SimpleDateFormat formatter2 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");//ore 24
String dataOrdine="";

for (int i=0;i<righeOrdine.get("codProdotto").size();i++)
{
    try {

        Date dataDaFormattare=formatter2.parse(testataOrd.get("dataOrd"));
        dataOrdine=formatter.format(dataDaFormattare);
    } catch (ParseException e) {
        e.printStackTrace();
    }

    contenutoFile+="\"R001\""+"|"+"\""+testataOrd.get("codCli")+"\""+"|"+
            "\""+numOrd+"\""+"|"+
            "\""+dataOrdine+"\""+"|"+"\""+righeOrdine.get("codProdotto").get(i)+"\""+"|"+
            righeOrdine.get("prezzo").get(i)+"|" +
            righeOrdine.get("qta").get(i)+"|" +
            righeOrdine.get("importo").get(i)+"|" +
            "\""+testataOrd.get("pIva")+"\""+"|" +
            "\""+testataOrd.get("ragSoc")+"\""+"|" +
            "\""+testataOrd.get("ind")+"\""+"|" +
            "\""+testataOrd.get("cap")+"\""+"|" +
            "\""+testataOrd.get("citta")+"\""+"|" +
            "\""+testataOrd.get("provincia")+"\""+"|" +
            "\""+testataOrd.get("paese")+"\""+"|" +
            "\""+"telefono"+"\""+"|" +
            ""+"\""+testataOrd.get("mail")+"\""+"|" +"\""+testataOrd.get("codAge")+"\""+"|" +
            "\""+testataOrd.get("pagamento")+"\""+"|" +
            "\""+"cf"+"\""+"|" +
            "\""+testataOrd.get("note")+"\""+
            lineEnd;
}
```

Scrivo il file con il contenuto creato, trasmetto il file e controllo il reale invio del file attraverso il response code (se la trasmissione è andata a buon fine è uguale a 200) modifico lo stato del ordine in trasferito altrimenti aggiungo il numero ordine all'ArrayList ordini non trasmessi.

```
        outputStream.writeChars(contenutoFile);

        outputStream.writeBytes(lineEnd);
        outputStream.writeBytes(twoHyphens + boundary + twoHyphens + lineEnd);

        // Responses from the server (code and message)
        serverResponseCode = connection.getResponseCode();

        outputStream.flush();
        outputStream.close();

        //controllo che il file sia stato trasmesso davvero
        try
        {
            final URL urlFile = new URL("http://192.168.128.146:8888"+"/ANDROID/ordini/"+numOrd+".txt");
            HttpURLConnection huc = (HttpURLConnection) urlFile.openConnection();
            int responseCode = huc.getResponseCode();
            // responseCode=200:File esiste , 404:File non esiste
            if (responseCode==200)
            {
                //cambio lo stato dell'ordine da confermato a trasferito
                ordine.mettiStatoTrasferito(db);
            }
            else
            {
               //inserisco il numero dell' ordine nell'array degli ordini non trasmessi
               ordiniNonTrasmessi.add(numOrd);

            }
        } catch (Exception e)
        {
            //inserisco il numero dell' ordine nell'array degli ordini non trasmessi
            ordiniNonTrasmessi.add(numOrd);
        }
    }
    catch (Exception ex)
    {
        //inserisco il numero dell' ordine nell'array degli ordini non trasmessi
        ordiniNonTrasmessi.add(numOrd);
    }

    myDbHelper.close();

    return serverResponseCode;
}
```

Gli attributi della classe Ordine sono :

```
public String numeroOrdine; //numero Ordine
public String codCli; // codice Cliente
public String ragSoc; // ragione sociale Cliente
public String indirizzo; // Indirizzo
public String cap; // codice di avviamento postale
public String citta;
public String provincia;
public String paese;
public String codArticolo; // codice Articolo
public String desArticolo; // descrizione Articolo
public int quantitaOrdinata;
public String prezzoUni;//prezzo unitario
public String sconto1; //sconto
public String sconto2;
public String sconto3;
public String dataOrdine;
public String dataConsegna;
public String codPag; // codice pagamento
public String pIva; // partita iva
public String codFisc; // codice fiscale
public String note;
public String statoOrdine;
public String importo;
public String mailcli; // mail cliente
public String mailcs; //mail customer service
public String iva; //imposta sul valore aggiunto
public String seriale; // valore univoco
public String daData;
public String aData;
public String ragSocSped; //ragione sociale cliente per spedizione
public String indirizzoSped; //indirizzo spedizione
public String capSped; // codice di avviamento postale per spedizione
public String cittaSped; // città spedizione
public String provinciaSped; // provincia spedizione
public String paeseSped; // paese spedizione
```

Dalla classe Ordine per aggiungere un record sul Database lancio addRigaOrdine passando il Database come parametro.

```
public void addRigaOrdine(SQLiteDatabase db)
{
    ContentValues values = new ContentValues();
    values.put("ord_numOrd", numeroOrdine);
    values.put("ord_codcli", codCli);
    values.put("ord_ragsoc", ragSoc);
    values.put("ord_codart", codArticolo);
    values.put("ord_desart", desArticolo);
    values.put("ord_qtaOrd", quantitaOrdinata);
    values.put("ord_przuni", prezzoUni);
    values.put("ord_sconto1", sconto1);
    values.put("ord_sconto2", sconto2);
    values.put("ord_sconto3", sconto3);
    values.put("ord_importo", importo);
    values.put("ord_pag",codPag);
    values.put("ord_indcli", indirizzo);
    values.put("ord_capcli", cap);
    values.put("ord_cittacli", citta);
    values.put("ord_provcli", provincia);
    values.put("ord_paesecli", paese);
    values.put("ord_pivacli",pIva);
    values.put("ord_ragsocsped", ragSoc);
    values.put("ord_indsped", indirizzo);
    values.put("ord_capsped",cap);
    values.put("ord_cittasped", citta);
    values.put("ord_provsped", provincia);
    values.put("ord_paesesped", paese);
    values.put("ord_cittacli", citta);
    values.put("ord_provcli", provincia);
    values.put("ord_mailcli",mailcli);
    values.put("ord_mailcs",mailcs);
    values.put("ord_cf",codFisc);
    values.put("ord_noteColore",note);
```

Dall'importo dell'ordine calcolo l'IVA ed effettuo l'insert.

```
    values.put("ord_dataord",dataOrdine);
    values.put("ord_datacons",dataConsegna);
    values.put("ord_stsord",statoOrdine);
    values.put("ord_codage",GlobalVariables.codAgeGlobal);
    values.put("ord_note","");

    //calcolo importo iva
    float importoFloat=Float.parseFloat(importo.replace(",","."));
    Integer percIva=Integer.parseInt(iva);
    float importoIva=importoFloat*percIva/100;
    values.put("ord_iva",importoIva);

    // 2. insert
    db.insert("tblordini", // table
            null, //nullColumnHack
            values); // key/value -> keys = column names/ values = column values
}
```

Per ottenere il contenuto del carrello lancio questa query:

```
String select="SELECT * FROM tblordini WHERE ord_stsord='I' and ord_codage='"+GlobalVariables.codAgeGlobal+"'" ;
```

Per ottenere il totale del ordine ancora nel carrello lancio:

```
Cursor cursor = db.rawQuery("SELECT SUM(ord_importo) FROM tblordini where ord_stsord='I' and ord_codage='"+GlobalVariables.codAgeGlobal+"'" , null);
```

Per ottenere il totale dell'iva dell'ordine nel carrello:

```
Cursor cursor = db.rawQuery("SELECT SUM(ord_iva) FROM tblordini where ord_stsord='I' and ord_codage='"+GlobalVariables.codAgeGlobal+"'" , null);
```

Per effettuare le operazioni di modifica nel Database della tabella relativa agli Ordini sono state predisposte alcune query:

Se l'ordine non è ancora stato trasferito si può modificarlo nuovamente facendolo tornare nel carrelo.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_stsord='I' WHERE ord_stsOrd='C'" +
        "and ord_numOrd='" + numeroOrdine + "'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

Se l'ordine è stato trasferito si modifica lo stato del ordine.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_stsord='T' WHERE ord_stsOrd='C'" +
        "and ord_numOrd='" + numeroOrdine + "'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

Se si vuole cancellare un'ordine dal carrello.

```
db.execSQL("DELETE from tblordini " +
        "WHERE ord_stsOrd='I'" +
        "and serial='"+seriale+"'"+
        "and ord_codage='"+GlobalVariables.codAgeGlobal+"'");
```

Per aggiornare interamente un record di Ordine.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_qtaord='"+quantitaOrdinata+"'" +
        ",ord_przuni='"+prezzoUni+"'" +
        ",ord_sconto1='"+sconto1+"'" +
        ",ord_sconto2='"+sconto2+"'" +
        ",ord_sconto3='"+sconto3+"'" +
        ",ord_importo='"+importo+"'" +
        ",ord_iva='"+ iva +"'" +
        ",ord_noteColore='"+note+"'" +
        ",ord_dataCons='"+dataConsegna+"'" +
        " WHERE ord_stsOrd='I'" +
        "and serial='" + seriale + "'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

Per modificare la modalità di pagamento.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_pag='"+codPag+"' WHERE ord_stsOrd='I'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

Per aggiornare l'indirizzo di spedizione del cliente.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_ragsocsped='"+ragSocSped+"',ord_indsped='"+indirizzoSped.replace("'"," ")+"' " +
        ",ord_capsped='"+capSped+"',ord_cittasped='"+cittaSped+"'" +
        ",ord_provsped='"+provinciaSped+"',ord_paesesped='"+paeseSped+"' WHERE ord_stsOrd='I'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

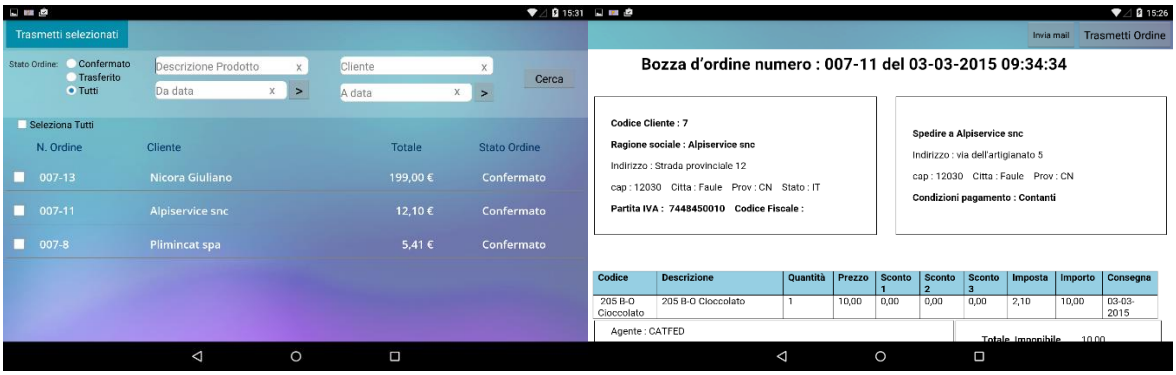
Per modificare i campi del intestazione dell'ordine.

```
db.execSQL("UPDATE tblordini " +
        "SET ord_mailcli='"+mailcli+"'" +
        ",ord_mailcs='"+mailcs+"'" +
        ",ord_note='"+note+"'"+
        "WHERE ord_stsOrd='I'" +
        "and ord_codage='" + GlobalVariables.codAgeGlobal + "'");
```

Per quanto riguarda il modulo “Documentazione” gli elementi della tabella (i files contenuti nella cartella assets/documenti filtrati per nome) sono gli elementi dell’array file che contengono la stringa filtro; cliccando su un elemento della tabella verrà lanciato l’intent di visualizzazione del file.

```
public void elencoDocumenti(View view) throws IOException {
    EditText campoNomeDoc=(EditText)findViewById(R.id.nomeDocumentoDaCercare);


    String[] file = getAssets().list("Documenti");

    final ArrayList<String> arrayFile=new ArrayList<~>();

    if (file!=null)
    {
        for (int i = 0; i < file.length; i++)
        {
            if (file[i].toUpperCase().contains(campoNomeDoc.getText().toString().toUpperCase())) {
                arrayFile.add(file[i]);
            }
        }
    }

    GridView gridDoc = (GridView) findViewById(R.id.gridDocumentazione);

    DocumentazioneRow customGridAdapter = new DocumentazioneRow(this.getBaseContext(), R.layout.documentazione_row,arrayFile);

    gridDoc.setAdapter(customGridAdapter);

    gridDoc.setOnItemClickListener((arg0, v, position, arg3) -> {
           // File root = getFilesDir();

            String path=getAssets()+"/Documenti/"+arrayFile.get(position);

            //apro il documento selezionato
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            File file = new File(path);

            InputStream in = null;
            OutputStream out = null;
            File fileNuovo = null;
```

7. Valutazione

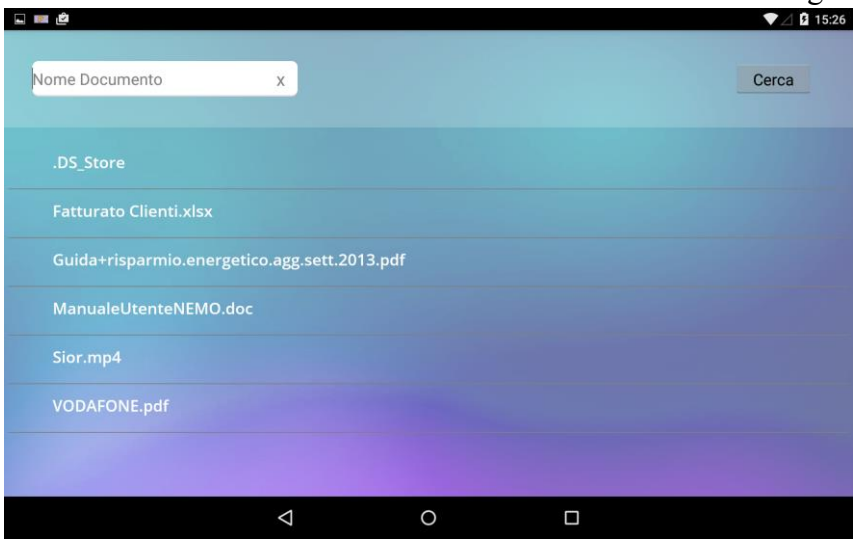


Negli ambienti di sviluppo Sior, qualunque sia il codice adottato, uno degli aspetti considerati di importanza fondamentale è il testing, che si compone di due fasi:

a) Unit test.
   Per "unit test" si intende l'attività di prova della singola unità di software (programma), che ogni sviluppatore deve effettuare per assicurarsi che l'unità di sviluppo assolva alle sue funzioni seguendo i requisiti prefissati. Questo viene considerato uno dei più importanti passaggi per poter avere un prodotto finale di buona qualità, quando si farà la verifica di integrazione tra i diversi componenti; basta pensare quanto è complesso il processo di debug alla fine del progetto se qualcosa non va nelle componenti di base. Lo "unit test" è sviluppato in maniera indipendente dallo sviluppatore, non sono cioè adottati in Sior standard o strumenti automatici.
   Pertanto, adeguandomi a queste metodologie, ho dovuto provvedere a caricare nello "Storico degli ordini", con uno specifico programma, una serie di dati casuali che mi mettessero in evidenza anomalie e incongruenze, fino a raggiungere in modo iterativo dati pienamente soddisfacenti. Su questo archivio, così creato, ho provveduto a verificare le estrazione con i filtri e tutte le altre funzioni.
   Questa attività mi ha consentito di presentare un ottimo risultato alla fase successiva del system test.
b) System test.
   Per "system test" si intende il processo di verifica dell'applicazione nel suo complesso, per constatare se si sono raggiunti gli obbiettivi prefissati, sia per quanto riguarda la funzionalità, che le prestazione e la facilità di utilizzo.

A questa fase, condotta in prima persona dal capo progetto con la presenza di tutto il team, non ho potuto che dare un contributo marginale, limitandomi a constatare che i moduli da me codificati hanno funzionato egregiamente.

Per il modulo “documenti” è stata utilizzata la stessa metodologia.



L’applicazione realizzata si può considerare scalabile in quanto, per le prestazioni può crescere o diminuire in funzione delle capacità dell’hardware; in buona sostanza abbiamo potuto verificare con il team che l’applicazione è indipendente dall’architettura utilizzata. Inoltre, essendo l’applicazione distribuita, abbiamo potuto rilevare sul campo, che le prestazioni non degradano anche in presenza di diverse decine di utilizzatori. Sulla modularità, essendo appunto Nemo realizzato con moduli integrati tra loro risulta evidente che sarà possibile in futuro corredarla di altri moduli che si ritenessero necessari o che fossero richiesti dal mercato.

8. Conclusioni

Il contributo di questo tirocinio, da un punto di vista mio personale, è stato quello di farmi comprendere la differenza tra la fase di studio vissuta finora e la realtà del mondo del lavoro, nel quale dovrò inserirmi a breve. Innanzitutto, pure essendo stato accolto con simpatia nell’ambito del gruppo di lavoro, ho dovuto rapidamente adeguarmi ai problemi derivanti dal lavorare in un gruppo, soprattutto per uno privo di esperienza rispetto a colleghi con diversi anni di lavoro alle spalle.

Da un punto di vista tecnico non ho incontrato particolari difficoltà in quanto già in possesso delle nozioni fondamentali derivanti dall’Università, anche se ho dovuto affinarle man mano in funzione dei casi che mi si presentavano. In merito agli sviluppi

futuri dell’applicazione mi sono formato alcune idee che ritengo interessanti e che intendo proporre all’azienda nel prossimo futuro. Mi riferisco principalmente alla firma digitale e alla possibilità di ottimizzare i percorsi dei venditori attraverso uno specifico modulo che potrebbe portare a risparmi di tempo e di denaro.

Appendice

A completamento dei tracciati del gestionale per moduli opzionali:

Anagrafica degli Sconti Categoria (scocat.txt)

Campo	O/F	Descrizione
Categoria Cliente	O	codice categoria cliente dal gestionale
Categoria Articolo	O	codice categoria articolo da gestionale
Sconto1	O	percentuale di sconto
Sconto2	F	
Sconto3	F	
Sconto4	F	
Sconto5	F	

Anagrafica degli Sconti Cliente (sconti.txt)

Campo	O/F	Descrizione
Codice cliente	O	codice cliente
Codice fornitore	O	codice fornitore
Sconto1	O	Percentuale di sconto
Sconto2	F	
Sconto3	F	
Sconto4	F	
Sconto5	F	

Particolarità (particolarita.txt)

Campo	O/F	Descrizione
Tipo particolarità	O	P(prezzo), S(Sconto), V(provvigione)
Codice conto	O	chiave esterna del cliente
Categoria conto	O	categoria cliente
Cod articolo	O	chiave esterna articolo
Fino a	O	Q(qta), F(fisso), P(prezzo)
Qta1	F	
Valore1	F	
Qta2	F	
Valore2	F	
Qta3	F	
Valore3	F	
Valore4	O	

Se "Tipo particolarità" è sconto (S) e "fino a" è quantità (Q) significa che fino alla quantità Qta1 ci sarà lo sconto Valore1, tra la quantità Qta1 e Qta2 ci sarà lo sconto Valore2, tra la Qta2 e Qta3 ci sarà lo sconto Valore3 e oltre la quantità Qta3 ci sarà lo sconto Valore4. Se "Tipo particolarità" è sconto (S) e "fino a" è fisso (F) significa che lo sconto è fisso a Valore4. Se "Tipo particolarità" è sconto (S) e "fino a" è prezzo (P) significa che fino alla prezzo Qta1 ci sarà lo sconto Valore1, tra il prezzo Qta1 e Qta2 ci sarà lo sconto Valore2, tra la Qta2 e Qta3 ci sarà lo sconto Valore3 e oltre il prezzo Qta3 ci sarà lo sconto Valore4. Con "Tipo particolarità" è prezzo (P) "fino a" non può essere prezzo (P). Se "Tipo particolarità" è prezzo (P) e "fino a" è quantità (Q) significa che fino alla quantità Qta1 ci sarà il prezzo Valore1, tra la quantità Qta1 e Qta2 ci sarà il prezzo Valore2, tra la quantità Qta2 e Qta3 ci sarà il prezzo Valore3 e oltre la quantità Qta3 ci sarà il prezzo Valore4. Se "Tipo particolarità" è prezzo (P) e "fino a" è fisso (F) significa che il prezzo è fisso a Valore4.

Indirizzi di Spedizione (Inspe.txt)

Campo	O/F	Descrizione
Cod cli	O	
Cod indirizzo	O	
Ragione sociale cons	O	
Indirizzo cons	O	
Cap cons	O	
Città cons	O	
Prov cons	O	

Statistiche (statisticheFatturato.txt)

Campo	O/F	Descrizione
codcli	O	Codice cliente
numfat	O	Numero fattura
datfat	O	Data fattura
codart	O	Codice articolo
importo	O	Importo per articolo
qta	O	Quantità venduta per articolo

Incassi (Scadenziario.txt)

Campo	O/F	Descrizione
azienda	F	
clifor	O	Codice cliente
annoreg	F	Anno registrazione
tiporeg	F	
seriedoc	F	serie documento
numdoc	O	Numero fattura
datadoc	O	Data emissione documento

prgriga	F	Progressivo riga
tipopag	F	Codice pagamento
statopag	F	pagato/non pagato
datascad	O	Data di scadenza del documento
importo	O	Importo scadenza da pagare

Resi (resi.txt)

Campo	**O/F**	**Descrizione**
idReso	O	Identificativo del reso
cliente	O	Codice cliente
dataSostituzione	O	
Articolo da sostituire	O	Codice articolo da sostituire
qtaDaSostituire	O	
articoloSostituito	O	Codice articolo sostituito
Qta sostituita	O	Data emissione documento
Reso integro	O	Flag che può avere valore 1 o 0… 1=Si 0=No
Reso rotto	O	Flag che può avere valore 1 o 0… 1=Si 0=No
Reso accreditato	O	Flag che può avere valore 1 o 0… 1=Si 0=No
Reso sostituito	O	Flag che può avere valore 1 o 0… 1=Si 0=No
Reso ritirato da agente	O	Flag che può avere valore 1 o 0… 1=Si 0=No
note	F	Nota relativa al reso

Reso e ordine rimangono scollegati nell'applicativo su tablet.

Metodo Pagamento Incassi (tipopag.txt)

Campo	O/F	Descrizione
Codice Met pag	O	Codice del metodo di pagamento dell'incasso
Descrizione Met Pag	O	Descrizione del metodo di pagamento dell'incasso

Per la categoria abbigliamento:

Stampe (stampe.txt)

Campo	O/F	Descrizione
articolo	O	Codice articolo a cui è legata la stampa
stampa	O	Descrizione della stampa
immagine	O	Nome dell'immagine relativa alla stampa

Colori (colori.txt)

Campo	O/F	Descrizione
articolo	O	Codice articolo a cui è legato il colore se è legato ad un articolo, se è legato a una stampa questo campo viene riempito con la descrizione della stampa a cui si riferisce
colore	O	Descrizione del colore
immagine	O	Nome dell'immagine relativa al colore

Taglie (taglie.txt)

Campo	O/F	Descrizione
articolo	O	Codice articolo a cui è legata la taglia
taglia	O	Descrizione della taglia

Ritengo doveroso un sentito ringraziamento ai componenti del mio gruppo di lavoro, nonché futuri colleghi, per l'accoglienza riservatami e per i preziosi suggerimenti che ho ricevuto in tutto il periodo del tirocinio.

Sitografia

http://developer.android.com/guide/topics/ui/declaring-layout.html

https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)

https://it.wikipedia.org/wiki/Enterprise_resource_planning

http://www.sior.it/works-nemo.html

http://www.sior.it/prodotti-nemo.html

http://www.isell.it/funzioni.php

http://forza6.com/

http://www.weblink.it/cosafacciamo-cont/mobile

http://www.uppordini.it/

http://www.nexusinformatica.biz/ipad.shtml

http://www.nexusinformatica.biz/android.shtml