Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

OLIO SB
ANALISI DELLA STARTUP INNOVATIVA NEL SETTORE DELL'AFTERMARKET AUTOMOBILISTICO

Relatore:
Chiar.mo Prof.
Davide Rossi

Presentata da:
Francesco Montanari

Sessione 3
Anno Accademico 2021/2022

Indice

1 Introduzione

La diffusione dell'e-commerce e il continuo sviluppo tecnologico hanno aperto nuove opportunità per le aziende che operano nel settore dei ricambi automobilsitici. Un settore con un'alta opportunità di crescita, dovuta alla continua domanda di soluzioni comode ed efficienti per l'acquisto di pezzi di ricambio e accessori. Una startup, che da poco ha acquisito il titolo di "innovativa" in questo campo, è Olio, un'azienda che offre una vasta gamma di articoli per veicoli a prezzi competitivi.

Con il termine "innovativa" il Ministero delle Imprese definisce: "La startup innovativa è un'impresa giovane, ad alto contenuto tecnologico, con forti potenzialità di crescita e rappresenta per questo uno dei punti chiave della politica industriale italiana".

La tesi si propone di analizzare i fattori chiave del successo di Olio, indagando su come l'azienda è riuscita a creare e sviluppare un'offerta di prodotti competitiva e a raggiungere un'alta gamma di clienti in tutta Europa. Verranno inoltre esaminate le sfide che Olio deve affrontare per mantenere e aumentare la propria quota di mercato in un contesto in continua evoluzione.

2 Settore

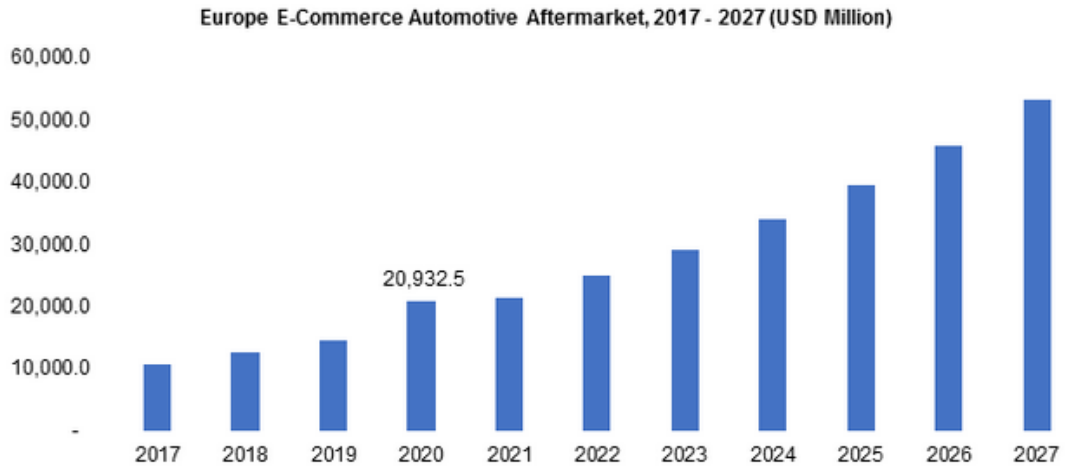

Figura 1: Andamento mercato ricambi automobilistici con previsione futura
Fonte: *https://www.graphicalresearch.com/industry-insights/1803/europe-e-commerce-automotive-aftermarket*

La pandemia di COVID-19 ha sconvolto l'industria automobilistica e ha avuto un impatto negativo sulla catena di fornitura. Ha costretto i governi di tutto il mondo a im-

porre severi blocchi e ad attuare norme di distanziamento sociale per ridurre al minimo la diffusione del virus, causando la diminuzione dell'utilizzo di veicoli per far spazio allo 'smart working' da casa. Ciò a portato l'attenzione dei clienti a spostarsi dall'impegno diretto con i venditori, all'utilizzo di piattaforme di e-commerce per le esigenze dell'aftermarket automobilistico. Questo avvenimento ha ulteriormente aumentato e creato nuove opportunità per lo sviluppo della vendita online in quanto è stato esentato dalle rigide misure di blocco.

Per quanto riguarda le fonti di ricambi auto, è presente una vasta gamma di scelta: produttori originali (OEM), produttori indipendenti (aftermarket) e venditori di ricambi usati. I ricambi OEM sono prodotti dalle stesse case automobilistiche che producono i veicoli, mentre i ricambi aftermarket sono prodotti da terze parti. I ricambi 'usati' sono pezzi di ricambio utilizzati, ma ancora funzionanti, che possono essere acquistati a prezzi più bassi rispetto ai ricambi nuovi. Il mercato dei ricambi aftermarket è in continua crescita a causa dell'aumento della domanda di soluzioni convenienti ed economiche per la manutenzione e riparazione dei veicoli. Inoltre, la maggiore durata delle automobili, a causa dei progressi tecnologici e dei miglioramenti nella qualità dei ricambi, sta aumentando la domanda per veicoli più vecchi.

3 Obbiettivo

Olio si propone sul mercato come alternativa commerciale per attori della filiera dell'aftermarket automotive che dispongono di un magazzino "a terra", siano essi produttori o distributori. Grazie alla connessione con i maggiori marketplace europei specializzati nella ricambistica auto, offre ai suoi fornitori un'ampia visibilità delle giacenze aumentandone l'opportunità di vendita, anche grazie all'esportazione in altri mercati esteri con bisogni differenti. Viceversa, per i marketplace, Olio rappresenta un aggregatore di dati già pronti ed erogabili in formato digitale ed automatizzato, capace di aiutarli a soddisfare sempre più clienti. Alla base del progetto ci sono due fattori chiave che determinano la buona riuscita:

- Conoscenza del mercato, dei suoi attori lungo la filiera e della visione commerciale
- Un'infrastruttura tecnologica, agile e flessibile, in grado di modellarsi velocemente alla visione commerciale e al contesto di riferimento, che risulta e sarà sempre in continua evoluzione e cambiamento.

Vendere online è una strada potenzialmente complessa, soprattutto per fornitori strutturati per il mercato tradizionale. Richiede risorse sia nella fase di analisi, per capire il contesto, sia in quella di strategia, per adattare i propri processi al contesto.

Una strada che può sembrare un labirinto di resi, insoluti, software, sviluppatori, ecommerce manager e tutte parole di cui si conosce vagamente il significato.

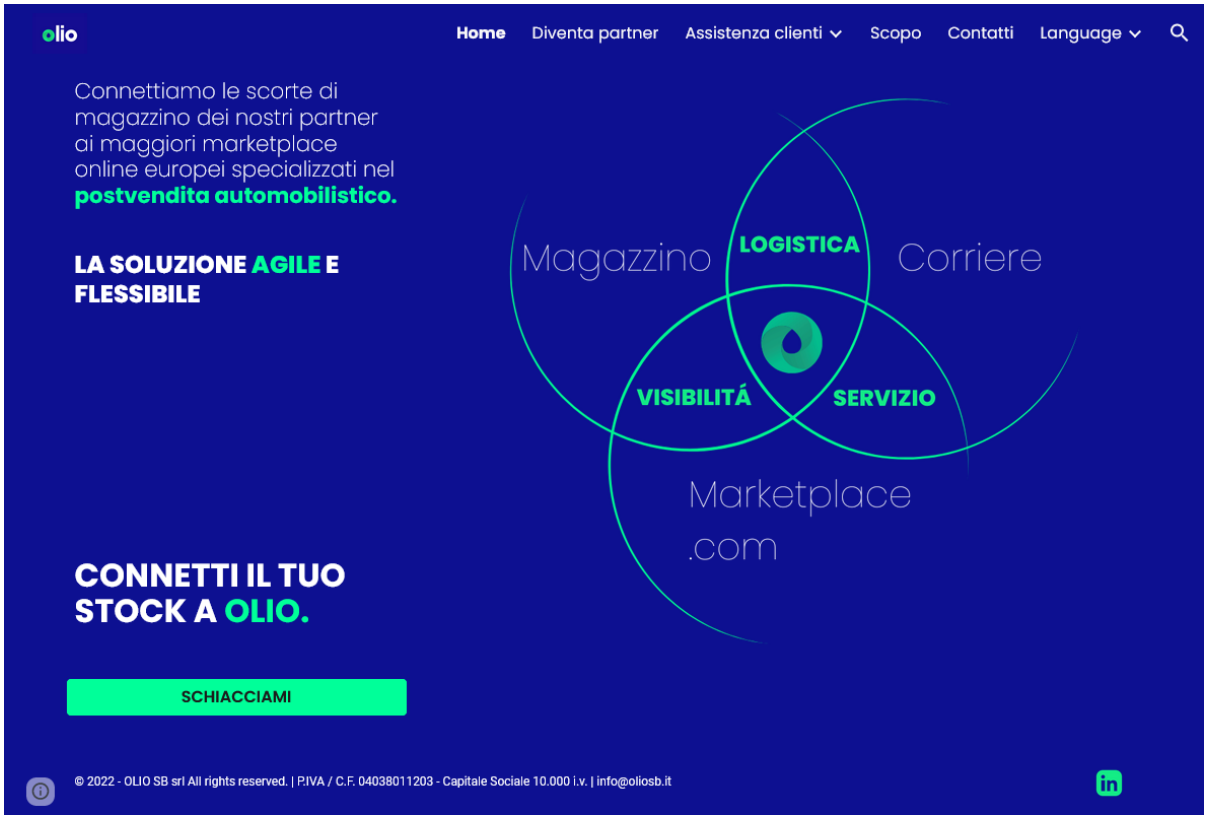

Figura 2: Sito Olio SB *https://www.oliosb.it/*

4 Fornitori

I fornitori di Olio sono rivenditori e produttori di ricambi automobilistici, i cui magazzini si estendono per tutta Italia. Per poter collaborare con Olio ci sono però delle condizioni necessarie da rispettare:

- Avere un magazzino fisico in cui gestiscono il "picking" ovvero l'attività di prelievo, smistamento e ripartizione di materiale da un'unità di carico ad altre. Ciò implica la disponibilità di operatori a spostarsi nel magazzino per gestire le merci. Infatti le spedizioni di Olio partono dal magazzino del fornitore, il quale deve essere in grado di poter impacchettare gli articoli e consegnarli al corriere. D'altro canto il fornitore, una volta consegnata la merce, viene esonerato da qualsiasi problema riguardante la spedizione, la quale passa sotto la responsabilità di Olio. Inoltre egli non ha bisogno di effettuare contratti con i corrieri, in quanto è Olio ha programmare i ritiri a suo nome.

- Gestione magazzino in modo informatico con possibilità di esportare il catalogo in formato digitale

- Necessità di uno strumento per l'effettuazione di ordini in modo veloce. Di solito si usa un sistema EDI (Electronic Data Interchange) sotto forma di e-commerce in cui effettuare gli ordini in maniera rapida oppure tramite chiamate API ad un loro servizio.

5 Canali di vendita

I canali di vendita di Olio si dividono in tre categorie: marketplace, e-commerce e agenti.

- I marketplace sono dei siti paragonabili a dei veri supermercati online, in cui ogni produttore può mettere in vendita i propri prodotti e gli utenti possono confrontare i vari articoli e scegliere l'opzione migliore per loro. Oltre ad offrire una scelta maggiore di articoli, offre solitamente maggiori garanzie al cliente durante il completamento dell'acquisto.

- Un e-commerce, a differenza del marketplace, supporta un singolo venditore, ovvero il proprietario del negozio. La responsabilità sull'acquisto e sulla spedizione dei prodotti è soltanto del proprietario, il quale vende gli articoli dei fornitori come se fossero suoi.

- Con il termine agenti andiamo a definire le officine che comunicano direttamente con Olio per l'acquisto dei prodotti. Al momento questa possibilità è ristretta a pochi singoli ma in futuro si sta già pensando a sistemi per fare in modo di acquistare direttamente da Olio senza passare per la tassazione del marketplace o e-commerce. Per attuare ciò si dovrebbe sviluppare un e-commerce di Olio in cui vengono messi in vendita i prodotti dei fornitori. Un'alternativa sarebbe l'utilizzo di whatsapp business per gestire gli ordini tramite un bot programmato ma sarebbe una cosa difficile da gestire e implementare

6 Analisi dei processi

Iniziamo analizzando il funzionamento generale legato alla vendita dei prodotti e ai passaggi che portano la merce dal fornitore al cliente.

Step 1 - Ricezione prodotti

Ogni fornitore carica quotidianamente il proprio catalogo in formato *csv* su una cartella personale in un server *ftp* di Olio. Per ogni prodotto è richiesto al fornitore di passare determinate informazioni necessarie ad Olio per poter rivendere l'articolo. Le informazioni più importanti sono: codice di produzione, brand, descrizione prodotto, quantità disponibile e prezzo. Il database di Olio raccoglie tutti i prodotti in una tabella per

ciascun fornitore e, una volta importati i dati, viene eseguito un "update" del catalogo ufficiale di Olio.

Step 2 - Aggiornamento prodotti

Con questo update, il software prende il codice del prodotto importato e va a cercare se esiste già nel catalogo interno, in caso positivo allora si limita ad aggiornare prezzo e disponibilità di quel prodotto in una tabella apposita, mentre in caso negativo va ad aggiungere il nuovo articolo al catalogo. Grazie a questo aggiornamento tramite solo il "codice del fornitore", è possibile tenere un catalogo pulito con i dettagli finali dei prodotti, senza che vadano ad essere sovrascritte ogni volta. Infatti può capitare che diversi fornitori abbiano prodotti uguali ma con descrizioni diverse e in questo caso sorge un problema: quale fornitore passa le informazioni migliori e più dettagliate riguardanti al prodotto? Al momento Olio non è in grado di affrontare questo problema e importa le informazioni del primo prodotto che gli arriva, dopo di che dovrà essere una persona ad andare a compilare eventuali campi quali peso, larghezza, altezza, profondità e categoria per avere un catalogo completo.

Ma serve davvero avere tutte queste informazioni sui prodotti? La risposta non è facile e per lo più sarebbe un bel "dipende!". Infatti ogni marketplace gestisce i prodotti in modo diverso. Alcuni si appoggiano su cataloghi già esistenti e quindi hanno solo bisogno del codice prodotto e del brand per poi associare direttamente l'articolo alle informazioni aggiornate di questo catalogo universale. Altri invece creano il loro catalogo in base a tutte le informazioni dei listini dei fornitori, un po' come se Olio dovesse aprire un e-commerce e vendere i prodotti con solo le proprie informazioni, il che diventa abbastanza complicato principalmente per il fatto che non si ha un'immagine del prodotto. Dopo aver aggiornato il catalogo si è quindi pronti all'esportazione degli articoli.

Step 3 - Esportazione prodotti

In questa fase vengono generati dei file *csv* diversi per ogni marketplace, i quali corrispondono alle esigenze di informazioni che ognuno richiede. L'esportazione avviene anche in questo caso su un server *ftp* il cui proprietario però è la sorgente di vendita. In fase di esportazione vengono inoltre calcolati i prezzi finali dei prodotti tramite una formula che comprende diversi fattori: margine di profitto di olio, tasse del marketplace, prezzi di spedizione calcolati in base alla nazione di vendita e altre variabili. Così finisce la parte legata all'esportazione dei prodotti e comincia il processo successivo: la gestione degli ordini.

Step 4 - Gestione ordini

Quando un prodotto viene venduto nelle varie piattaforme, viene inviata una mail ad Olio con i dettagli dell'ordine, i quali comprendono: il prodotto venduto, il cliente a

cui effettuare la spedizione e i dati di pagamento. Inoltre viene caricato un file *xml* sul server *ftp* del fornitore e, tramite uno script che si attiva ripetutatemente, viene scaricato e importato nel database di Olio. Questo file *xml* contiene tutti i dati dell'ordine e viene processato automaticamente inserendo i dati nelle varie tabelle del database:

- cliente: dati acquirente e destinatario che non sempre coincidono. E' infatti possibile che un utente acquisti un prodotto e lo spedisca da un'altra parte. Oppure che l'acquisto venga fatto a nome personale ma poi l'articolo sia spedito alla propria officina
- prodotto: viene automaticamente collegato, tramite un identificativo univoco, al relativo prodotto presente nel catalogo
- informazioni di pagamento e di fatturazione. I pagamenti spesso sono effettuati tramite SEPA, in cui è Olio a richiedere i soldi alla banca dell'acquirente tramite iban e un mandato che certifica la regolarità del pagamento
- spedizione: il riassunto delle informazioni viene inoltre riportato in una tabella che comprende tutti i dati necessari al corriere per poter inviare l'articolo ai clienti

Step 4.5 - Ordini tramite API

Al momento Olio ha implementato un servizio *api* riservato ad Autodoc, uno dei principali e-commerce europei del settore. Con queste chiamate l'e-commerce può:

- Controllare la disponibilità di un prodotto, con relativo prezzo e quantità
- Creare un ordine inserendo i relativi prodotti e le informazioni del destinatario
- Controllare lo stato di un ordine: ricevuto / spedito / consegnato / annullato
- Ottenere il link per seguire lo stato della spedizione
- Annullare un ordine

Step 5 - Gestione spedizioni

Per tutti gli ordini arrivati entro le 12, vengono create le spedizioni degli articoli. Una chiamata *api* al servizio del corriere permette di inviare automaticamente tutti i dati necessari ad egli per poter consegnare la merce. La chiamata restituisce le etichette da applicare sugli articoli e i dati per tracciare la spedizione. Le etichette vengono poi inviate ai fornitori e applicate da loro sui prodotti che il corriere passa a ritirare nel pomeriggio. I dati di tracciamento, che consistono in un link tramite cui si può vedere lo stato della spedizione, vengono memorizzati nel database e giornalmente inviati alla rispettiva

sorgente dell'ordine cosicché possa fornirli al cliente. Attualmente, una volta al giorno, un operatore controlla lo stato delle spedizioni per confermare che non ci siano problemi. Può capitare che alcune spedizioni rimangano ferme e quindi è necessario contattare il corriere per verificarne la situazione. Una volta che gli articoli sono stati consegnati, la spedizione viene fatta passare in stato "delivered" e scompare dalla visualizzazione nel front-end, rimanendo però sempre memorizzata nel database in caso di bisogno.

7 Back-end

Con il termine back-end indichiamo tutto ciò che opera dietro le quinte nella nostra applicazione. Rimane in ascolto sul server ricevendo le richieste dal front-end e gestisce la connessione con il database

7.1 Strumenti

Per quanta riguarda la scrittura del codice dobbiamo introdurre tre componenti principali:

- Typescript, il linguaggio utilizzato per la scrittura
- Node, un ambiente di esecuzione di codice che permette l'installazione di librerie esterne.
- Adonis, un framework, ovvero un'architettura logica di supporto per facilitare lo sviluppo del software. Offre inoltre un ecosistema stabile per la creazione di un'applicazione web e un server *api*

Prima di comprendere meglio come funziona il codice, andiamo ad analizzare la scelta di questi strumenti utilizzati.

Typescript è un linguaggio tipizzato, ovvero che permette di definire il tipo delle variabili, i parametri e il ritorno delle funzioni. Ma per comprenderlo meglio dobbiamo analizzare Javascript, un linguaggio semplice da imparare, veloce, comodo per scrivere script immediati e da cui appunto deriva Typescript. Alle variabili, in Javascript, non viene inizializzata la tipologia di dato e sarà poi l'interprete che esegue il codice a gestire in automatico il cambiamento di tipo. Questa caratteristica è molto comoda ma se si vuole sviluppare un grosso progetto potrebbe causare errori sulla tipologia di dati. Typescript è nato proprio per colmare questo problema e possiamo dire che è un super-set di Javascript in quanto comprende tutte le sue funzionalità ma aggiunge delle estensioni che integrano al meglio lo sviluppo da parte back-end. Typescript però non viene eseguito dal browser poiché non è supportato e quindi passa da un compilatore che lo traduce in Javascript. Dato che ha questa fase di semi-compilazione, è possibile accorgersi degli

errori di tipo, di sintassi, di definizione delle variabili e altro ancora, i quali avvengono in fase di compilazione e non di esecuzione come succede con Javascript.

In conclusione possiamo affermare che Typescript è un linguaggio adeguato per un progetto di grandi dimensioni, in continua crescita e incentrato sulla parte di back-end.

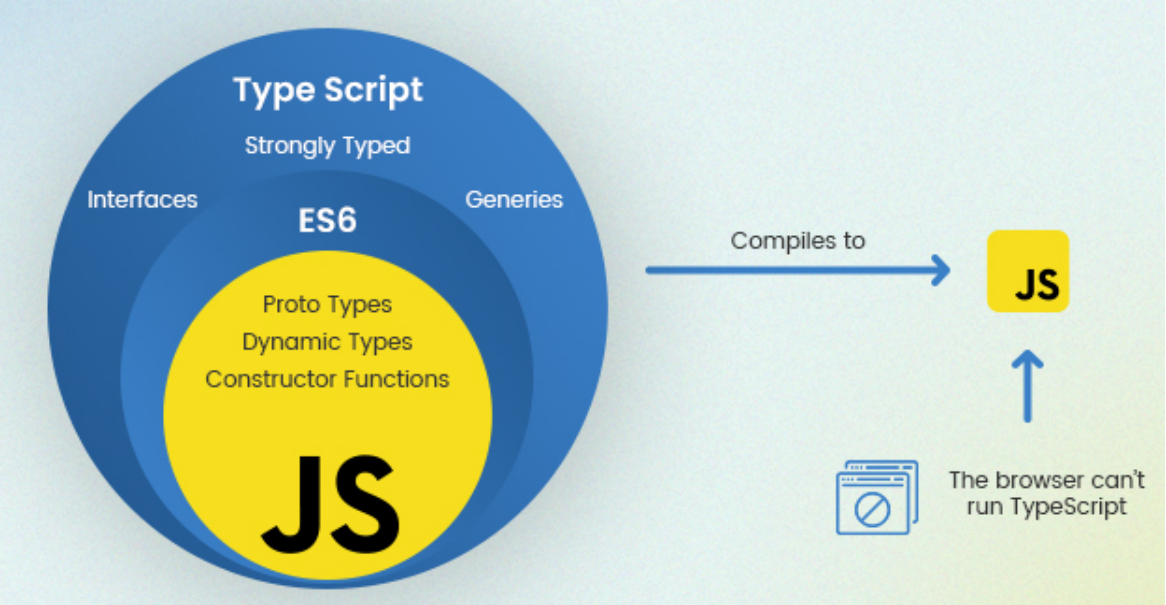

Figura 3: Typescript vs Javascript

Node è sicuramente una delle migliori scelte tra gli ambienti di sviluppo ma vediamo bene i suoi vantaggi:

- Permette un'incredibile integrazione sia dal lato database che dal lato frontend. Tramite NPM (Node Package Manager) si possono installare librerie esterne in modo veloce e intuitivo.
- Open source, con le sorgenti liberamente accessibili e modificabili dalla comunità. Ha una comunità pronta a offrire feedback e supporto in caso di bisogno.
- Multipiattaforma, ovvero disponibile su tutti i sistemi operativi.
- Sincronizzazione veloce tra server e client, perfetto per le applicazioni che si basano su eventi in tempo reale, come appunto Olio che riceve gli ordini sui propri prodotti.
- Scalabilità, essendo single-threaded, è in grado di gestire un enorme numero di connessioni contemporaneamente

Grazie a tutti questi vantaggi, possiamo affermare che Node è una scelta solida ed efficace per lo sviluppo di Olio. Esso è infatti utilizzato da molte aziende famose: Twitter, Spotify, eBay, Reddit, Linkedin e molte altre.

Infine Adonis è un framework scritto in Typescript e di tipo MVC (Model View Controller), ovvero che separa la logica dell'applicazione in tre parti: models, views e controllers. Tra i suoi punti forti ci sono: gestione di richieste *http*, query per database SQL, trasferimento di file e gestione delle mail. Fa uso dell'ORM (Object-Relational Mapping) ovvero una tecnica comune a tutti i framework utilizzata per creare un database virtuale tramite la programmazione orientata agli oggetti. Per Adonis l'ORM viene chiamato "Lucid" e ci permette di creare query in modo veloce, di effettuare il seeding del database e le migrations

Adonis mi ha davvero stupito, ha molteplici funzioni che semplificano la scrittura del codice e, nonostante la documentazione non sia molto ampia, si riesce sempre ad ottenere quello che si vuole in modo rapido e efficiente.

7.2 Struttura

Iniziamo ad analizzare la struttura del codice interno del backend, per prima cosa vediamo la suddivisione dei componenti principali:

- Applicazione

  La parte che comunica con il front-end e che comprende lo scheletro di tutte le funzionalità. Al suo interno infatti troviamo:

  1. controllers per gestire le varie richieste del front-end
  2. modelli, con l'inizializzazione di tutte le classi, che definiscono una tipologia di modello con cui opereremo. Alcuni esempi sono: la struttura del prodotto, del cliente, dell'ordine, del magazzino ma anche i vari modelli individuali per i fornitori e i marketplace
  3. la gestione delle exceptions, dei middleware e delle mail che vengono inviate quando avvengono degli errori nell'esecuzione di scripts o comandi

- Comandi

  Adonis permette l'utilizzo e la creazione di comandi tramite Ace, ovvero un framework a linea di comando incorporato dentro Adonis. In questo modo è anche possibile usare comandi già preimpostati per creare componenti, far partire seeders e migrations. Ma la parte importante è proprio la creazione di comandi che nel nostro caso i più importanti sono: l'import dei prodotti nel database, l'aggiornamento del catalogo con le nuove disponibilità, l'export del catalogo per i marketplace, il download degli ordini dai vari server dei marketplace, la registrazione delle fatture e dei vari documenti sul servizio di contabilità e altri comandi di rilevanza minore

- Configurazioni

  Comprendono le ulteriori configurazioni sulle strutture dei dati. Ad esempio per quanto riguarda i marketplace, qua troviamo le varie tasse applicate, i paesi di esportazione e le cartelle dove viene caricato il catalogo. Per quanto riguarda i fornitori troviamo la conversione delle colonne dei loro *csv* di magazzino che passano ad Olio, così da ritornare negli standard di produzione, le varie spese di imballaggio, le cartelle da cui scaricare i file *csv* e altri dettagli meno rilevanti.

- Database

  Ci sono due importanti concetti da definire e che rappresentano la struttura del database:

  1. Migrations: pagine di codice che vanno a definire la struttura del database, ovvero creano le tabelle, le chiavi e i vincoli. Per ognuna il codice si divide in 'up' in cui vengono scritti i cambiamenti e in 'down' in cui vengono rimossi questi cambiamenti. In questo modo si riesce a seguire un ordine logico e si può sempre tornare indietro lanciando la migration al contrario, infatti tramite i comandi di Adonis "run" e "rollback" si possono, rispettivamente, far partire o rollbackare una o più migrations.
  2. Seeders: i dati del database che vogliamo inserire alla creazione delle tabelle. Una volta creato il database può essere utile inserire alcuni campi dentro le tabelle, come nel caso di Olio vengono inseriti i fornitori, i marketplace, i brand dei prodotti, ecc..

- Api

  Le varie configurazioni dei percorsi per le chiamate *api* al server. Permettono ad un marketplace di poter effettuare richieste al server per creare ordini, ricevere informazioni sui prodotti, controllare lo stato delle spedizioni e avere le informazioni di tracciamento dei pacchi.

7.3 Codice

Per comprendere al meglio la struttura e la funzionalità dei processi, andiamo ad analizzare i comandi principali per la gestione dei prodotti di Olio.

Inizializzazione comandi

Prima di essere inizializzati nelle varie pagine Typescript, i comandi devono essere dichiarati nel manifest di Adonis - Ace, come in questo esempio:

```
"product:parse": {
      "settings": {
        "loadApp": true,
        "stayAlive": false
      },
      "commandPath": "./commands/ProductParse",
      "commandName": "product:parse",
      "description": "Parse and import the CSV files",
      "args": [
        {
          "type": "spread",
          "propertyName": "suppliers",
          "name": "suppliers",
          "required": true
        }
      ],
```

Listing 1: Product Parse

Product Parse

Questo comando gestisce l'import dei prodotti dai file *csv* dei fornitori al database di Olio. Come input accetta i nomi dei fornitori di cui effettuare l'import del catalogo. Quando viene eseguito va a controllare dentro le cartelle *ftp* dei fornitori se è presente il file *csv* da scaricare, in caso negativo salta quel fornitore e procede con i successivi. Quando trova un file *csv*, inizia ad effettuare controlli sul nome del file e sulla presenza di dati all'interno, in caso sia tutto corretto allora procede alla mappatura dei dati. Prima di inserirli nel database, azzera le vecchie disponibilità dei prodotti del fornitore, divide le entità da importare in gruppi da 1000 e inizia ad inserirle. Quando le inserisce controlla già se sono presenti nella tabella in base al nome di certe colonne e, in caso affermativo, aggiorna solo i dati senza ricompilare tutto da capo.

```
// Azzeramento disponibilita'
await model.query()
    .where('supplier', supplierConfig.name)
    .where('group', supplierConfig.group)
    .update('availability', 0);

// Paginazione prodotti
const batch = this.batch || 50;
for (let i = 0, len = rawEntities.length; i < len; i += batch) {
    const partialEntities = rawEntities.slice(i, i + batch)
        .map(item => {
        const undefinedKeys = Object.keys(item).filter(key => types.
   isUndefined(item[key]));
        return lodash.omit(item, undefinedKeys);
        });
```

```
    // Inserimento prodotti nel db
    const trx = await Database.transaction();
    try {
        const inserted = await model.updateOrCreateMany(['supplier',
        'group', 'warehouseId', 'code', 'brand'] as never,
        partialEntities, { client: trx });
        await trx.commit();
        importContext.count += inserted.length;
        if (importContext.count % 1000 === 0) {
        this.logger.logUpdatePersist();
        this.logger.info(`Correctly partial imported/updated
        1000 entities`);
        } else {
        this.logger.logUpdate(''.padEnd(parseInt(((importContext.count
        % 1000) / batch).toFixed(0)), '.'));
        }
    } catch (error) {
        await trx.rollback();
        this.logger.action('IMPORT').failed(`Error ${error.code}(
        ${error.errno}) while adding row`, `${error.message} -
        Detail: ${error.detail} - Stack: ${error.stack}`);
        importContext.errors?.push(error);
    }
}
```

Listing 2: Azzeramento disponibilità e inserimento dati catalogo

Una volta finito l'import, parte uno script per aggiornare i codici identificativi delle marche tramite i loro nomi. Questi codici corrispondono a degli identificati "universali" usati dalla maggior parte dei marketplace per definire i brand. Infatti nel passo successivo, ovvero l'update dei prodotti, solo gli articoli con questo codice entreranno nel catalogo interno di Olio mentre gli altri rimarranno nelle tabelle di import dei fornitori, in quanto i marketplace non li accetterebbero comunque.

```
this.logger.info(`Update all brands at once`, prefix)
    const productPossibleCodes = await UniversalCodes.query()
        .whereIn(Database.knexRawQuery('lower(unaccent(name))'), q => {
            q.from(model.table)
                .select(Database.raw('lower(unaccent(brand))'))
                .where('supplier', supplierConfig.name)
                .groupByRaw('lower(unaccent(brand))')
            })
            .debug(Env.get('NODE_ENV') !== 'production');
        const productCodes = productPossibleCodes.map
        (_ => ({ [_.name]: _.code }))
            .reduce((p, n) => ({ ...p, ...n }), {});
        this.logger.action('UPDATE CATALOGUE IDS').succeeded(`Updating
        ${productPossibleCodes.length} brands with relative ids`)
```

```
        const trx = await Database.transaction();
            try {
                for (const key in productCodes) {
                  await model.query()
                    .where('brand', key)
                    .andWhere('supplier', supplierConfig.name)
                    .update('brand_id', productCodes[key]);
                }
                await trx.commit();
              } catch (error) {
                await trx.rollback();
                this.logger.action('UPDATE UNIVERSAL IDS').failed
                (`Error${error.code}(${error.errno}) while adding row`,
                `${error.message} - Detail: ${error.detail} - Stack:
                ${error.stack}`);
                importContext.errors?.push(error);
              }
```

Listing 3: Aggiornamento codici brand

Premessa

Premetto che tutti i comandi successivi sono stati rielaborati o creati da me. Non garantisco che siano fatti usando le soluzioni migliori però ho cercato di ottimizzare le tempistiche e l'efficienza degli script tramite le conoscenze acquisite in questi anni di Università e tramite i consigli di un Full Stack Developer.

Product Update

Questo comando gestisce l'aggiornamento del catalogo di Olio con focus principale sulle disponibilità e sui prezzi dei prodotti. Quando viene fatto partire lo script è possibile inserire i nomi dei magazzini su cui lavorare oppure lasciare vuoto il campo per procedere su tutti i magazzini disponibili:

```
> node ace product:update magazzino_1 magazzino_2
>
```

Per prima cosa vengono azzerate le disponibilità dei prodotti per i magazzini richiesti. Tramite vari filtri viene creato un array con tutti gli articoli con i campi "puliti" e vengono inseriti nel catalogo del database. Subito dopo vengono inserite le disponibilità e i prezzi in una tabella di collegamento apposita. In questo modo con gli "update" successivi non verranno riscritti ogni volta i prodotti del catalogo ma andrà solo aggiornata la tabella con i prezzi e le disponibilità.

Adonis, tramite la funzione "fetchOrCreateMany" ci permette di andare a controllare le celle di una tabella tramite il valore di alcune colonne. Ad esempio, nel nostro caso vengono aggiunti nel catalogo solo i prodotti che non hanno già la coppia di chiave

"codice" e "codice brand". In caso venga trovata questa coppia già esistente allora si passa direttamente alla riga successiva.

```
try {
    await Product.fetchOrCreateMany(
    ['code','brandId'],
    partialImports,
    { client: trx });
    await trx.commit();
    totalImported += partialImports.length;
    this.logger.debug(`Imported / Updated ${totalImported} of
    ${total} imports`, prefix);
} catch (err) {
    await trx.rollback();
    this.logger.fatal(err, prefix, `Error updateOrCreateMany:
    ${err.detail + err.message}`);
}
```

Listing 4: Aggiornamento catalogo prodotti

L'aggiornamento della tabella "warehouse_products", contenente le disponibilità e i prezzi, avviene tramite una query che aggiunge o aggiorna i valori ottenuti dalle tabelle di "import" dei fornitori.

```
await Database.rawQuery(`INSERT INTO warehouse_products
 (product_id, warehouse_id, availability, price) VALUES (
 (SELECT id from products WHERE
 (code = :code and brand_id = :brandId)
 group by id limit 1),
    :warehouseId,
    :availability,
    :price
    )
ON CONFLICT (product_id, warehouse_id) DO
UPDATE SET (product_id, warehouse_id, availability, price) =
 (EXCLUDED.product_id, EXCLUDED.warehouse_id, EXCLUDED.availability,
 EXCLUDED.price)`,{
    warehouseId: newImportWarehouse.warehouseId,
    availability: newImportWarehouse.availability,
    price: priceNum,
    code: newImportWarehouse.code,
    brandId: newImportWarehouse.brandId
})
```

Listing 5: Aggiornamento tabella di collegamento

Product Export

Tramite questo comando vengono creati i file *csv* per i vari canali di vendita. Questi file contengono i prodotti del catalogo di Olio con le informazioni formattate nel metodo

richiesto dalla sorgente di vendita, la quale, prima di avviare la collaborazione con Olio, comunica il formato di output delle informazioni.

Questo comando è probabilmente quelle più complicato da analizzare dal lato del codice, quindi andiamo ad analizzare la sua logica. Come per i precedenti è possibile specificare la sorgente per cui effettuare l'export, tramite il flag "-m" e si possono indicare i fornitori di cui inserire i prodotti nel file:

```
> node ace product:export magazzino_1 magazzino_2 -m marketplace_1
>
```

Una volta partito vengono raggruppati tutti i prodotti da inserire nel file finale e, per ognuno, viene preso il prezzo minore disponibile, in quanto ci possono essere più fornitori con lo stesso prodotto ma che lo vendono a prezzi diversi. A questo punto viene calcolato il prezzo finale degli articoli passando attraverso una formula che comprende vari parametri:

- Prezzo d'acquisto che comunica il fornitore tramite il file *csv* (prezzo di partenza)
- Supplemento logistico del fornitore, ovvero le spese di imballaggio o di picking
- Carcassa. Alcuni prodotti vengono riassemblati a partire da un vecchio articolo chiamato carcassa. Quindi l'acquisto di un prodotto comprende anche il costo di questa carcassa che poi sarà rimborsato nel caso in cui venga restituito l'articolo vecchio del cliente, una volta che ha effettuato la sostituzione con il ricambio nuovo.
- Quota accantonamento imprevisti
- Quantità per confezione
- Costo della spedizione. Uno dei campi più variabili in quanto questa tariffa cambia in base al corriere e alla nazione di destinazione
- Margine di profitto di Olio
- Provvigione sorgente di vendita
- Rimborso spedizione. In certe piattaforme, a seconda delle tue regole, vengono rimborsati dei costi in caso in cui il prezzo dell'ordine sia inferiore ad un certo valore

Una volta calcolato il prezzo, viene creato il file *csv* seguendo il modello di export del relativo marketplace/e-commerce e inserendo tutte i relativi dettagli dei prodotti

```
import FinalProduct from "../FinalProduct";

export class ExportMarketplace3 {
```

```
    public InternalId: string = '';
    public ProductCode: string = '';
    public Brand: string = '';
    public BrandId: string = '';
    public EAN: string = '';
    public Description: string = '';
    public QuantityPerPack: number = 0;
    public MinQuantity: number = 0;
    public Stock: number = 0;
    public Price: number = 0;

    constructor(product: FinalProduct) {
        this.ProductCode = product.code ? product.code : '';
        this.InternalId = product.internal ? product.internal : '';
        this.Brand = product.brand ? product.brand : '';
        this.BrandId = product.brandId ? product.brandId : '';
        this.Description = product.desc ? product.desc : '';
        this.EAN = product.ean;
        this.Stock = product.availability ? product.availability : 0;
        this.QuantityPerPack = product.quanPP ? product.quanPP : 1;
        this.MinQuantity = product.minQ ? product.minQ : 1;
        this.Price = (product.price ? product.price : 0);
  }
}
```

Listing 6: Esempio modello export

A questo punto il file viene inserito dentro una cartella del server di Olio e poi, tramite uno script automatico, viene caricato nel server *ftp* della sorgente di vendita.

Order Sync

Questo comando si occupa dello scaricare e processare gli ordini arrivati. Per prima cosa introduciamo le librerie che utilizza:

- **fs** – "File System Module" permette di poter lavorare con i file presenti nelle cartelle del server
- **ftp** – permette di comunicare con un server *ftp* esterno
- **axios** – un client *http* basato sulle promesse, utilizzato per effettuare chiamate *api*
- **xml2js** – un modulo che ci permette di convertire in modo semplice i file *xml* in json

Per ogni sorgente di vendita, viene aperta la connessione al relativo server *ftp* tramite il "Client" integrato nella libreria di "ftp" e vengono scaricati i file salvandoli nelle cartelle del server tramite la funzione "createWriteStream" della libreria "fs".

```
private async getFile(path, options, marketplace, orderFolder) {
    return new Promise((resolve, reject) => {
      let c = new Client();
      c.connect(options);
      c.on('ready', () => {
        c.get(orderFolder + path, function (err, stream) {
          if (err) reject(false);
          stream.once('close', function () { c.end(); });
          stream.pipe(fs.createWriteStream(OrderSync.ordersPath +
          marketplace + "/" + path));
        })
        c.on('error', (err) => {
          console.log(err)
          reject(false)
        })
        c.on('close', (hadErr) => {
          if (hadErr) {
            console.log(hadErr)
            reject(false)
          }
          console.log(path + ' scaricato correttamente')
          resolve(true)
        })
      })
    })
  }
```

Listing 7: Funzione per scaricare un file

Prima di partire con il download dei file viene creato un "lock" che blocca l'avvio di altri processi fino al termine dell'esecuzione. Una volta scaricati i file in modo ricorsivo e per tutte le sorgenti di vendita; si parte con la creazione delle seguenti tabelle nel database.

- Cliente: colui che ha acquistato la merce
- Destinatario: colui a cui deve essere spedita la merce
- Ordine: numero ordine della sorgente, id del cliente, id della sorgente, prezzo totale di vendita e eventuali tasse di consegna
- Prodotti: articoli relativi all'ordine con prezzi e quantità
- Pagamento: dettagli sul metodo di pagamento utilizzato
- Banca: in caso in cui il metodo di pagamento sia "SEPA DEBIT" vengono memorizzati i dati della banca a cui richiedere il pagamento
- Spedizioni: le informazioni parziali della spedizioni che verranno completate quando un operatore inserirà l'ordine di acquisto relativo ai prodotti ordinati

- Ordine d'acquisto: inserisce una riga pronta ad essere popolata con i dati relativi al magazzino da cui è stato ordinato e il prezzo d'acquisto
- Tracking: inserisce una riga pronta per essere popolata con i dati di tracciamento della spedizione una volta che la spedizione verrà inviata al corriere

Una volta chiusa la comunicazione con il database, vengono rinominati i file degli ordini in:

- .done se tutto è andato a buon fine
- .error se qualcosa non ha funzionato durante l'inserimento dei dati nel database
- .fatal se il file era già .error e dopo un altro tentativo non è riuscito ad entrare nel database. A questo punto viene inviata una mail agli operatori di Olio che controlleranno la struttura del file e il motivo dell'errore

Per tutti i file che non hanno avuto problemi, viene aperta di nuovo la connessione al server *ftp* della sorgente e vengono cancellati dalle cartelle così da non essere più riscaricati le volte successive. Una volta finito viene rimosso il "lock" così da poter far ripartire il comando successivamente.

Order Cloud

Con questo comando andiamo ad inserire dentro il gestionale (Fatture in Cloud) tutti i dati necessari tramite tre principali chiamate al loro servizio *api*. Vengono analizzati tutti gli ordini ricevuti, non ancora esportati, e che abbiano già tutte le informazioni necessarie per poter passare sul gestionale. Le tre chiamate relative ad ogni ordine che consultano il database per ottenere le varie informazioni, sono:

1. Esportazione prodotti, in cui vengono richiamate le informazioni dal database dei prodotti e viene inserito nel gestionale il codice identificativo, peso, categoria e codice doganale. Il codice doganale è un valore numerico univoco che identifica una determinata categoria di prodotto negli scambi internazionali e viene usato a fini fiscali.
2. Esportazione cliente con tutte le informazioni per poter fatturare a lui
3. Esportazione ordine, in cui vengono collegati il cliente e i prodotti precedentemente esportati. Inoltre viene inserito il metodo di pagamento e le eventuali informazioni per il *ddt*

8 Front-end

Con il termine front-end indichiamo la parte dell'applicazione visibile all'utente e con la quale si può interagire. Esso è responsabile dell'acquisizione dei dati e della loro elaborazione per poi inviarli al back-end.

8.1 Strumenti

Per quanta riguarda la scrittura del codice dobbiamo introdurre cinque componenti principali:

- Typescript, il linguaggio di scrittura che abbiamo già analizzato precedentemente
- Node, l'ambiente di esecuzione anche questo visto prima
- React, un framework utilizzato per creare le interfacce utente
- Next, un framework con diverse funzionalità che vedremo tra poco
- MUI, una libreria di React utilizzata principalmente per il lato 'design' del sito web

React permette di creare componenti riutilizzabili in modo comodo, come ad esempio la barra di navigazione, un footer, un componente per il contenuto principale e molti altri. D'altro canto, React, può avere problemi con gli utenti che non hanno accesso a javascript o lo hanno disabilitato, potenziali problemi di sicurezza, tempi di caricamento della pagina notevolmente estesi e può danneggiare l'ottimizzazione complessiva del motore di ricerca del sito. Framework come Next aggirano questi problemi consentendo il rendering di parte o di tutto il sito Web sul lato server prima di essere inviato al client. Quindi la scelta di unire React e Next è stata fatta per poter ottenere il massimo dell'efficienza da entrambi i componenti. Nonostante ciò unire i due framework non era probabilmente necessario in quanto si potevano ottenere risultati ottimali utilizzando solo Next, che è attualmente il framework più utilizzato dagli sviluppatori.

La MUI, o anche detta Material UI è una libreria che comprende dei componenti con un ottimo design e personalizzabili dallo sviluppatore. Serve solo per l'aspetto grafico ma è intuitiva e con una buona documentazione

8.2 Struttura

Il front-end di Olio è ancora in parte in costruzione ma le funzionalità principali possono essere già utilizzate.

- Gestire gli ordini ricevuti: cambiare le informazioni del destinatario o dell'acquirente, modificare i dettagli dei prodotti venduti, cambiare lo stato dell'ordine, inserire l'ordine d'acquisto e inserire il metodo di pagamento. Tutti questi passaggi sono semi-automatici, infatti prima che un ordine venga spedito è necessaria l'approvazione di un operatore che controlla che tutto sia inserito in modo corretto.

- Gestire le spedizioni: modificare le informazioni che andranno inviate al corriere, inserire i dati di tracciamento per la spedizione, visualizzare le etichette dei relativi prodotti venduti

- Gestire il catalogo: visualizzare i prodotti attivi, modificarne le informazioni e poterli escludere dalla vendita

Funzioni da implementare

- Gestire i fornitori: aggiungere nuovi fornitori, modificarne le informazioni come ad esempio i costi di imballaggio i quali, una volta modificati, andranno automaticamente a cambiare il costo del prezzo finale di un prodotto

- Gestire i marketplace: aggiungere nuove piattaforme in cui esportare i prodotti e possibilità di impostare già le informazioni che si vogliono esportare per ogni articolo. Infatti ogni marketplace vuole solo certe informazioni riguardanti i prodotti

- Resi: gestire le informazioni legate al processo di reso come viene eseguito per gli ordini

- Statistiche sui prodotti e sugli acquirenti: statistiche di vendita che offrono una panoramica dell'andamento delle vendite.

9 Server

Il server di Olio è hostato su Amazon Web Services (AWS), in cui si trovano tre istanze EC2 (Elastic Compute Cloud) che svolgono da macchine virtuali per il software. Collegate ad ognuna di queste istanze, c'è un'istanza di RDS (Relational Database Service) che contiene il rispettivo database. Una domanda che sorge è: perchè avere tre istanze invece che una sola? Ogni istanza corrisponde ad uno stadio del nostro software:

- Produzione: tutto ciò con cui Olio sta attualmente operando

- Staging: ambiente identico alla produzione in cui vengono testate le modifiche prima di andare 'live' sul server di produzione

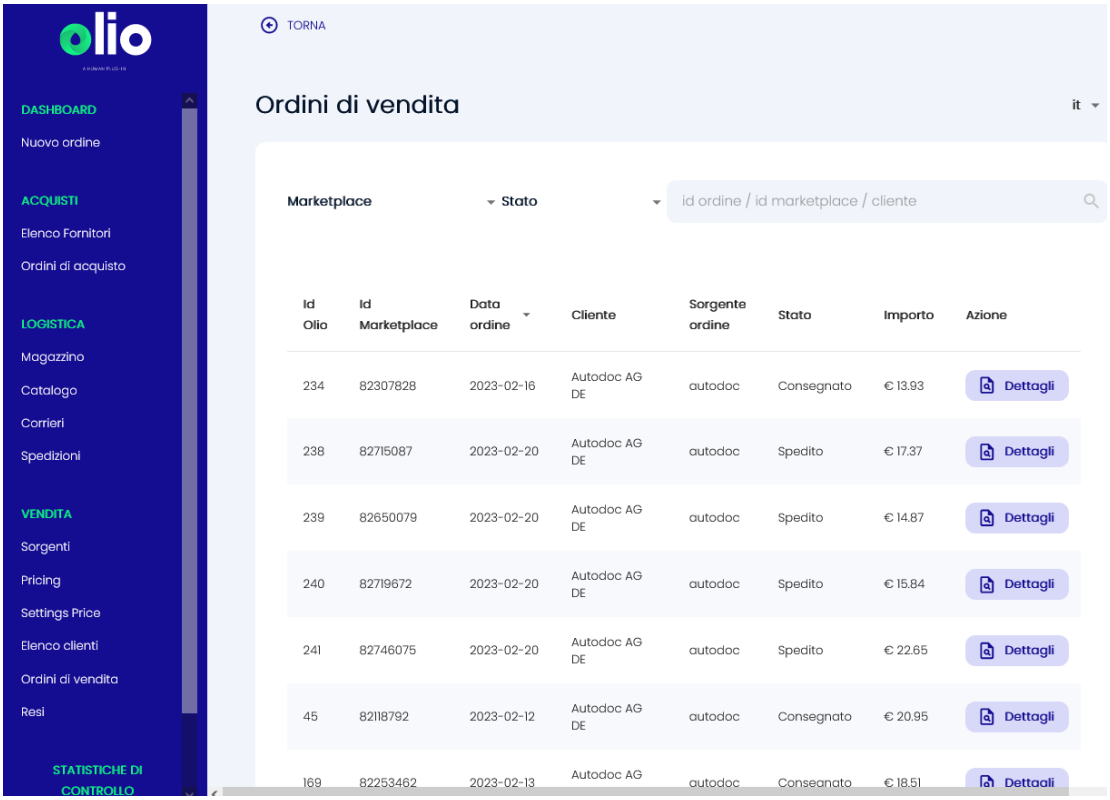

Figura 4: Esempio homepage gestione ordini

- Testing: ambiente di sviluppo in cui si può operare a piacere sul codice e sul database senza creare problemi

Il vantaggio di AWS è che, oltre all'incredibile semplicità d'utilizzo e all'alta gamma di servizi che offre, permette abbonamenti il cui costo è proporzionale all'utilizzo dei servizi. Ad esempio se una macchina ad un certo punto riceve molte richieste e ha bisogno di più memoria per poterle gestire, AWS alloca più risorse a quella macchina aumentando il costo del servizio ma solo per quel momento in cui c'è necessità.

10 Spedizioni

Olio organizza le spedizioni degli articoli ai propri clienti appoggiandosi a servizi di corrieri. Dentro il database di Olio vengono create le spedizioni con tutti i dati che il corriere necessita. Dal front-end è poi possibile confermare e inviare le spedizioni tramite una chiamata *api* al servizio web del corriere, a cui vengono inviati i dati delle spedizioni. Il corriere restituisce le etichette in un formato cifrato, decodificato in automatico e trasformato in pdf per essere mostrato sul front-end. A questo punto un operatore

controlla che sia tutto regolare e conferma l'etichettatura da front-end, avviando uno script che procede ad inviarle automaticamente ai relativi magazzini dei fornitori. I fornitori a questo punto applicano le etichette sugli articoli venduti e consegnano il tutto al corriere quando passa per il ritiro. Le spedizioni partono proprio dai magazzini dei fornitori e arrivano direttamente al cliente, senza passare per Olio. Questo è un vantaggio per Olio in quanto non necessita un magazzino in cui depositare la merce e, contemporaneamente, per il fornitore è indifferente il luogo di spedizione.

11 Fatturazione

Uno dei lati più importanti è sicuramente il tracciamento degli acquisti e delle vendite. Olio si appoggia alla piattaforma Fatture in Cloud, la quale offre molteplici servizi interessanti e semplici da utilizzare. In primo luogo analizziamo cosa contiene:

- le informazioni dei clienti a cui Olio ha venduto articoli
- le informazioni dei prodotti che sono stati venduti, come se fosse un magazzino
- gli ordini ricevuti, i quali comprendono un cliente e uno o più prodotti
- le fatture, documenti fiscali che attestano la vendita degli articoli
- i documenti di trasporto, ovvero un documento che segue la merce durante il trasporto e certifica il trasferimento del venditore al cliente. Oltre ad essere obbligatorio nel caso di controlli, garantisce la sicurezza della merce e ha valore probatorio in caso di merce smarrita o mancata consegna

Tramite un comando del server di Olio, quando un ordine viene confermato, tutte le informazioni relative a quell'ordine vengono inserite su Fatture in Cloud: cliente, prodotti e ordine.

Questo comando l'ho sviluppato in modo da interagire con l'*api* di Fatture in Cloud tramite dei 'GET' per ottenere le informazioni che abbiamo già e dei 'POST' per inserire quelle nuove. Un problema che ho riscontrato è stata l'incompleta documentazione del loro servizio di *api*, in quanto non venivano specificate le varie dipendenze tra i campi da inviare nelle richieste. Ad esempio quando si crea un ordine ci sono dei campi obbligatori da inserire che però nella loro documentazione erano definiti come 'opzionali'.

Fatture in Cloud è comunque uno strumento molto valido e ottimale per essere usato come gestionale per la propria attività

12 Statistiche

Andiamo ad analizzare un po' di numeri legati ad Olio.

12.1 Prodotti

- Nel catalogo di Olio sono presenti 236.000 articoli e 129 brand differenti
- Nelle tabelle dei fornitori sono presenti in totale 321.000 articoli di cui 30.000 senza codici "universali" dei brand

12.2 Fornitori

I fornitori attualmente attivi sono otto, due sono in fase di attivazione e uno che esporta i propri prodotti solo tramite chiamate *api* al proprio servizio ed è quindi richiesta un'implementazione di codice per poter comunicare con lui.

12.3 Mercati

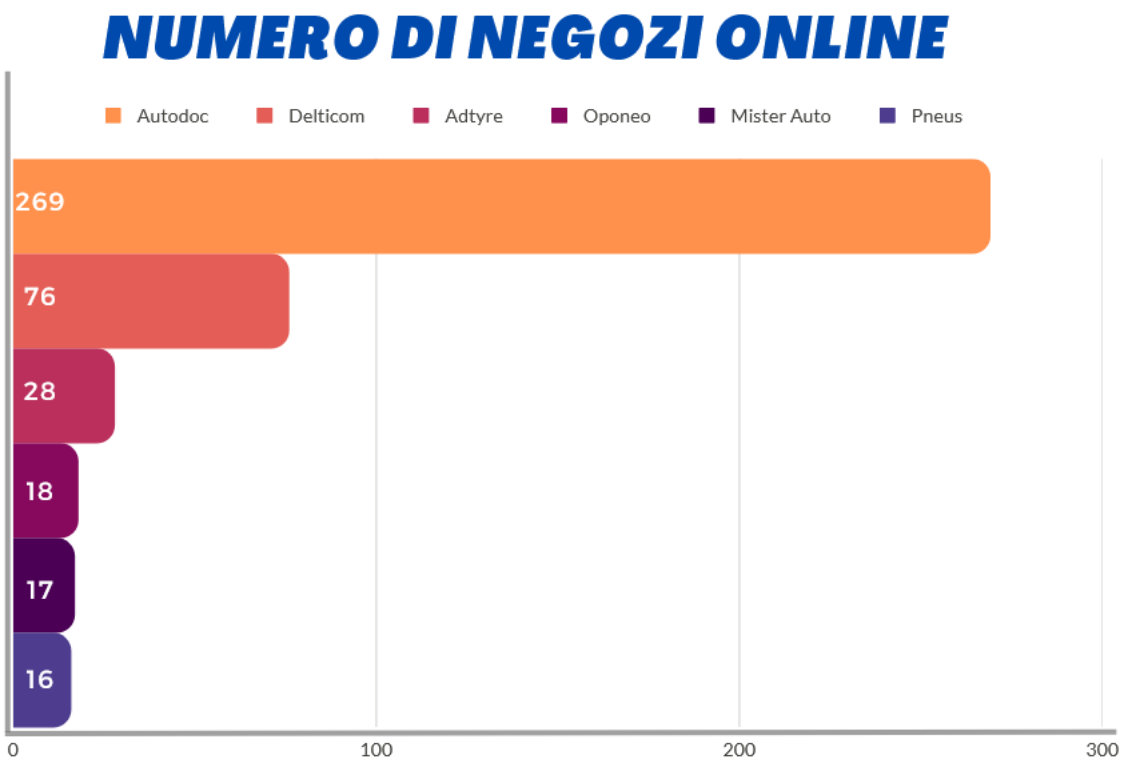

Figura 5: Analisi di 1000 siti online e divisione per compagnia

Olio opera nei seguenti mercati:

- 2 marketplace e 1 e-commerce attivi in tutta Europa, tra cui Autodoc, il principale e-commerce in Europa,

- 2 marketplace attivi in Germania
- 1 marketplace attivo in Francia
- 1 marketplace attivo in Spagna
- 2 e-commerce in attivazione
- 1 marketplace con attivazione futura: Ebay, il quale, non lavorando con il catalogo "universale" dei brand, richiede innumerevoli informazioni dei i prodotti che Olio dovrebbe mettere in vendita, come ad esempio le immagini degli articoli, il peso e le misure. Al momento Olio dispone solo delle informazioni che i fornitori comunicano e, nella maggior parte dei casi, non sono adeguate alle richieste di Ebay. Per questo si pensa ad un'implementazione futura quando si avranno i dati necessari.

13 Sguardo al futuro

Di seguito analizziamo una serie di strumenti o idee che Olio potrebbe implementare nei prossimi anni in modo da aumentare la sua popolarità e i suoi guadagni, rimanendo sempre affidabile e ponendo attenzione ai propri fornitori e clienti.

13.1 Applicazione Web

Un applicazione web per i fornitori sarebbe un salto di qualità poiché permetterebbe loro di:

- visualizzare e gestire gli ordini ricevuti
- controllare le statistiche su quali prodotti vengono maggiormente venduti
- ricevere le etichette da applicare sui prodotti pronti al ritiro e poter comunicare direttamente con l'etichettatrice per stamparle. Su questo punto introduciamo un altro progetto futuro:

  Etichettatrici

  Le etichettatrici sono uno strumento fondamentale nel lavoro di un fornitore che deve ogni giorno applicare etichette sui prodotti venduti. Olio sta pensando di equipaggiare tutti i fornitori con un'etichettatrice che comunica direttamente con l'applicazione di Olio. I fornitori ci guadagnerebbero uno strumento gratuito che semplifica ulteriormente il loro lavoro, mentre Olio otterrebbe la fidelizzazione dei propri fornitori in cambio di un costo per l'acquisto di questi strumenti.

13.2 Nuove categorie

Espandere la gamma dei prodotti è un metodo efficace per aumentare il numero di clienti. Al momento Olio si occupa solo di ricambi automobilistici, però ha già fornitori che offrono prodotti di altre categorie, quali: oli lubrificanti, grassi, pneumatici, cerchi, ruotini di scorta e batterie.

Il problema di questi articoli è che per ogni categoria ci sono informazioni diverse da memorizzare, per esempio le batterie hanno un voltaggio, una capacità, una polarità e così via. Sarebbe quindi necessario:

1. creare e collegare nuove tabelle nel database per memorizzarne le nuove informazioni

2. ottenere le fonti da cui ricavare le informazioni

Infatti i fornitori non trasmettono tutti questi dettagli ma c'è necessità di svolgere ricerche online nei cataloghi dei produttori per poterne recuperare i dati. Questo perché i brand, delle categorie in questione, non sono presenti nel catalogo usato dai marketplace e quindi non è possibile fare il "matching" tramite degli identificativi univoci. Il processo di recupero informazioni è quindi molto lento e richiede la gestione da parte di un operatore.

Probabilmente questo è il primo step a cui puntare una volta sistemati tutti i processi interni, in quanto aumenterebbe di molto i profitti andando a fare concorrenza in nuove sezioni di mercato.

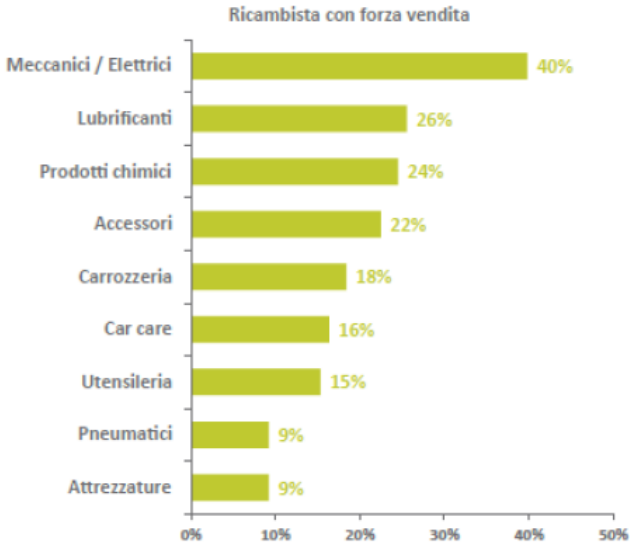

Figura 6: Statistiche sulle categorie vendute
Fonte: *Anfia aftermarket*

13.3 E-commerce

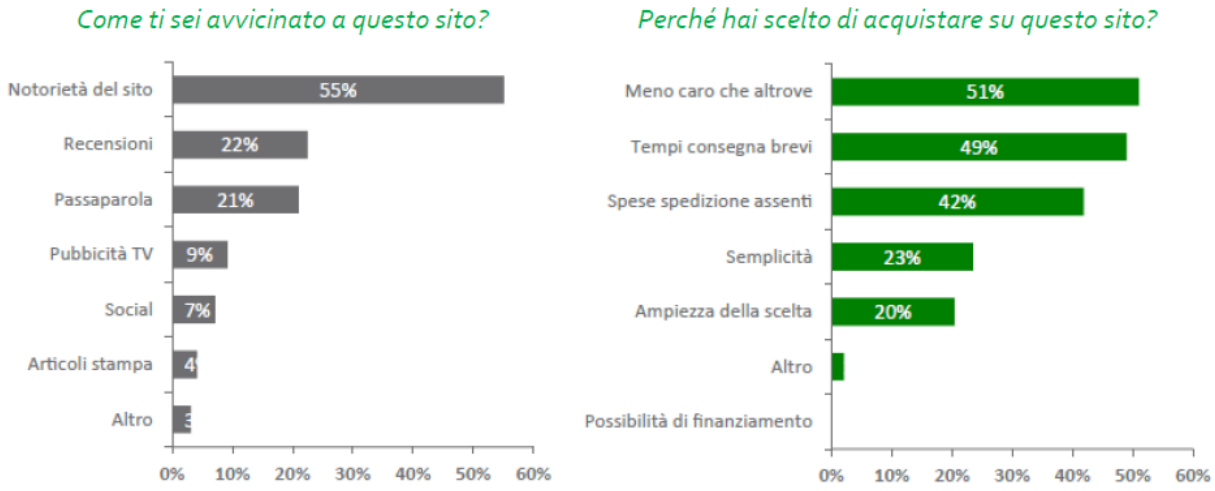

Figura 7: Statistiche su opinioni clienti e-commerce generici
Fonte: *Anfia aftermarket*

Un sito web per la vendita di prodotti comporterebbe lati positivi e negativi.

Vantaggi:

- Prezzo più competitivo in quanto viene escluso il guadagno delle sorgenti di vendita con cui Olio opera attualmente.
- Aggiornamenti in tempo reale. Quando un prodotto viene acquistato, immediatamente non risulta più disponibile e quindi non possono arrivare ordini che tanto finirebbero cancellati. Al momento invece l'aggiornamento delle disponibilità avviene quando Olio effettua l'export del catalogo aggiornato e quando il marketplace lo importa. Quindi ci sono tempistiche variabili che possono comportare anche l'aggiornamento poche volte al giorno.
- Implementazione rapida del sito dal lato codice. Il front-end permette già di visualizzare i prodotti, i link di tracciamento delle spedizioni e tutte le ulteriori informazioni presenti nel database. L'unica operazione da aggiungere sarebbe l'acquisto dei prodotti con i relativi pagamenti.

Svantaggi:

- Sicurezza. E' necessario offrire un servizio sicuro in cui i dati dei clienti non siano esposti ad eventuali attacchi di frode informatica e ciò comporterebbe un alto livello di responsabilità che Olio dovrebbe acquisire

- Disponibilità. Il sito deve essere sempre disponibile per gli utenti e gestire molteplici connessioni o transazioni. Ciò comporta dei costi di mantenimento per il server e la necessità di personale che gestisca i problemi degli utenti.
- Visibilità. Attualmente Olio rivende tramite sorgenti già conosciute con clienti fidelizzati. Con un nuovo e-commerce sarebbe necessario crearsi una nuova base di clienti fedeli tramite una buona pubblicizzazione

13.4 Catalogo aperto a tutti

L'idea della monopolizzazione attuale delle informazioni sui prodotti non va molto a genio ad Olio. Si sta infatti pensando alla possibilità di sviluppare un catalogo aperto a tutti in cui i fornitori e i rivenditori possono inserire o migliorare i dati riguardanti gli articoli del settore dell'automotive. Andiamo ad analizzare cosa comporterebbe.

Vantaggi

- Accessibilità: l'accesso alle informazioni sui ricambi diventa più facile e gratuito, anche per piccoli rivenditori e officine indipendenti, riducendo la dipendenza dalle case produttrici.
- Collaborazione: incoraggia la collaborazione e la condivisione di informazioni tra i diversi attori del settore, comprese le case produttrici, i rivenditori, le officine e i meccanici. Ciò può aiutare a migliorare la qualità e l'accuratezza delle informazioni, nonché a stimolare l'innovazione.
- Riduzione dei costi: l'accesso gratuito alle informazioni sui ricambi può ridurre i costi per le officine e i consumatori, in quanto non sarebbe più necessario pagare per accedere a tale servizio.
- Standardizzazione: può aiutare a standardizzare le informazioni sui ricambi, rendendo più facile la compatibilità tra i diversi marchi e modelli di automobili.

Svantaggi

- Qualità delle informazioni: le informazioni presenti potrebbero non essere sempre accurate e aggiornate. Inoltre, il catalogo potrebbe non coprire tutti i marchi e i modelli di automobili.
- Protezione dei dati: potrebbe contenere informazioni proprietarie che le case produttrici non vogliono condividere con altri attori del settore.
- Manutenzione: richiede una costante manutenzione e aggiornamento delle informazioni. Questo può rappresentare un onere per i gestori del catalogo, che devono assicurarsi che le informazioni siano accurate e aggiornate.

In conclusione creare un catalogo aperto è un'idea su cui bisogna riflettere e cercare di analizzare tutti gli aspetti positivi e negativi. Se l'obiettivo è quello di migliorare l'accessibilità alle informazioni sui ricambi e di promuovere la collaborazione tra i diversi attori del settore, potrebbe essere conveniente. Tuttavia, se ci sono preoccupazioni riguardo alla qualità delle informazioni, alla protezione dei dati e alla concorrenza, potrebbe essere necessario rivalutare tutti le conseguenze prima di compiere determinate scelte.

14 Conclusioni

In conclusione, la startup analizzata in questa tesi ha dimostrato di avere un modello di business solido e una forte attenzione alle esigenze del mercato. Grazie all'utilizzo delle tecnologie digitali e di una strategia di marketing efficace, la startup è riuscita ad affermarsi nel mercato dei ricambi automobilistici online, consolidando la propria posizione competitiva. Tuttavia, come ogni startup, anche questa affronta delle sfide e deve continuare a innovare per restare al passo con i rapidi cambiamenti del mercato. In futuro, sarà importante mantenere la focalizzazione sui clienti e investire in tecnologie sempre più avanzate per migliorare l'esperienza degli utenti e aumentare l'efficienza dei propri processi. In definitiva, la startup rappresenta un esempio concreto di come l'imprenditorialità digitale possa essere un'opportunità per creare valore economico e sociale.

Questa tesi ha quindi analizzato sia gli aspetti economici come la gestione di fornitori e clienti, i costi e i ricavi, la scalabilità nel tempo e il modello di business.

Dal punto di vista informatico abbiamo visto: la struttura del back-end e del front-end, la struttura dei processi interni che portano i prodotti dai fornitori ai clienti, la memorizzazione dei dati nel database e la struttura del server.

Il corso di studi di "Informatica per il Management" mi ha permesso di acquisire le conoscenze nel campo tecnologico, finanziario, informatico e di gestione di un business; così da poter sfruttare appieno le mie capacità per migliorare tutti i processi dell'azienda. Questo dimostra come sia importante che gli aspetti economici e informatici viaggino insieme per la crescita di nuove aziende e startup.

15 Ringraziamenti

Vorrei ringraziare tutte le persone che mi hanno supportato in questi tre anni e mi hanno accompagnato in questo percorso.

Ringrazio il professore Davide Rossi per avermi permesso di scrivere questa tesi su un argomento che reputo molto interessante e che si adatta agli insegnamenti informatici e economici appresi durante questo corso.

Ringrazio la mia ragazza Silvia che mi ha dato la forza e soprattutto la pazienza per scrivere questa tesi.

Ringrazio i miei compagni di corso per avermi reso la vita universitaria un'esperienza fantastica. Senza di voi non avrei mai imparato a scrivere così tante cose in poco tempo e a studiare per gli esami la settimana prima.

Ringrazio i miei più cari compagni del liceo con cui ho passato momenti indimenticabili e spero di passarne molti altri così.

Ringrazio il gruppo del Pertini in cui ho trovato amici gentili, affidabili e altruisti con cui passare bellissime giornate e serate.

Ringrazio Matteo, il fondatore di Olio, per avermi supportato nello sviluppo di questa tesi e per avermi permesso di lavorare ad un progetto appassionante e con grande potenziale

Ringrazio il mio fidato computer che da 8 anni mi accompagna negli studi, resistendo ai maltrattamenti di quando mi arrabbio e a tutte le applicazioni o schede che apro contemporaneamente

Un grande grazie anche alla mia famiglia per avermi supportato con domande del tipo "E la tesi come sta andando? Quando la finisci?" ogni volta che li incontravo. Siete stati la mia ispirazione per procrastinare ancora di più. A parte gli scherzi il più grande ringraziamento va a loro che mi hanno permesso di svolgere questo percorso universitario supportandomi in tutte le scelte fatte.

Infine, voglio ringraziare tutti coloro che, anche se non nominati, mi hanno fatto perdere tempo e concentrazione invitandomi ad uscire fuori il pomeriggio, la sera a bere o a giocare a qualche videogioco.

Grazie a tutti voi per aver reso questa esperienza universitaria un'avventura unica e indimenticabile.

Glossario

api Application Programming Interface è un software che permette la comunicazione tra applicazioni, una che effettua una richiesta (client) e l'altra che da una risposta (server)

csv Comma Separated Value è un file caratterizzato dalla presenza di dati tabulari sotto forma di testo e i cui valori delle celle sono separati da virgole o punti e virgola

ddt Documento Di Trasporto che viaggia insieme alla merce e certifica il trasferimento dal venditore a chi ha comprato i prodotti. Viene prodotto in duplice copia, una segue la merce mentre l'altra resta al venditore per eventuali controlli legali e ha inoltre valora di prova in caso di merce smarrita o non consegnata

ftp File Transfer Protocol è un protocollo utilizzato per il trasferimento di dati, basato su un sistema client-server che consente nel caricare, spostare e scaricare file all'interno di un sistema di directory

http Hypertext Transfer Protocol è un protocollo utilizzato per comunicare con i servizi web. Le richieste che vedremo saranno principalmente di 2 tipi:

- GET per richiedere una risorsa
- POST per inviare una risorsa

xml Extensible Markup Language è un linguaggio basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo