Università degli Studi di Torino
Dipartimento di Informatica

Corso di Laurea in Informatica

Tesi di Laurea in Informatica

Storie digitali per produttori agricoli: il progetto Farmchain.

Relatore:
Prof. Gian Luca Pozzato

Candidato
Giuseppe Funicello

Sessione Aprile 2021
a.a 2019/2020

Sommario

La comunicazione digitale è diventata uno strumento imprescindibile per promuovere la propria attività e nel settore agroalimentare questo rischia di lasciare indietro produttori con una minore propensione all'utilizzo di tali tecnologie. Il progetto Farmchain, sviluppato dalla startup About a bit di cui sono co-fondatore, si pone questo obiettivo: creare una piattaforma che permetta a piccoli e medi produttori agricoli di portare la propria esperienza all'interno del mondo digitale, attraverso l'utilizzo di uno strumento intuitivo. La web app oggetto della prima fase di sviluppo, realizzata all'interno del periodo di tirocinio, permette con una UI semplice di creare una "storia" del proprio prodotto, similmente a quanto accade sui maggiori social network. L'obiettivo è quello di realizzare un'etichetta digitale accessibile attraverso un QR code applicabile sui propri prodotti. Per realizzare le storie viene utilizzato il formato AMP Web Stories, realizzato da Google, che permette di rendere tali contenuti raggiungibili dal motore di ricerca, aumentando il numero di potenziali clienti. Inoltre, grazie all'integrazione delle Web Stories con il sito web dell'azienda agricola, si migliora notevolmente l'indicizzazione ed il posizionamento del sito stesso in ottica SEO. La successiva fase di sviluppo prevede la creazione di una mobile app con cui documentare in tempo reale le fasi produttive utilizzando degli Smart Contract integrati con IPFS per memorizzare timestamp, geolocalizzazione e hash dei media creati. In questo ambito, la blockchain viene impiegata per migliorare la comunicazione attraverso l'inserimento di un livello di trasparenza che aiuta ad avvicinare il consumatore al produttore. Ad integrare questo argomento, vengono analizzati i limiti di correlazione tra token digitale e fisico attualmente esistenti in questo tipo di autocertificazione.

Ringraziamenti

Ringrazio il professor Gian Luca Pozzato per la sua disponibilità e per il prezioso aiuto in fase di stesura della tesi.

Nonostante abbia conosciuto pochi altri colleghi studenti, per mia fortuna ho incontrato lungo il percorso di studi Ilaria, con cui ho condiviso progetti ed esami, a cui va il mio sentito ringraziamento.

Ringrazio Luca e le tante corse che ci hanno accompagnato per dare vita all'idea dalla quale è nato il progetto Farmchain, Francesco che ha deciso di seguirmi in questa avventura imprenditoriale e Fabrizio che si è unito con entusiasmo al gruppo. Ringrazio anche Filippo, per la fiducia ed i preziosi consigli che sempre ci regala.

Un grazie a tutte le persone incontrate lungo il mio percorso in questi anni di studio: ognuno a modo suo mi ha sempre incoraggiato ad arrivare fino in fondo.

Ringrazio mia madre e mio padre, che non hanno mai fatto mancare il loro supporto ed incoraggiamento, e le mie sorelle Angelica e Rossana, sempre pronte a fare il tifo per me. Un grazie ai miei nipoti Leonardo, Tommaso e Beatrice per avermi donato sempre gioia e buonumore ed a Daniela e Ugo, per la loro grande generosità.

Un immenso grazie a Barbara, che mi ha regalato la serenità necessaria per affrontare qualsiasi cosa, ed a Jacopo, che mi insegna ogni giorno a stupirmi della vita.

Indice

Introduzione

Nel corso degli ultimi due anni ho affiancato al mio percorso di studente universitario e lavoratore autonomo lo sviluppo di un progetto, che è culminato con la fondazione di una società insieme ad altri tre soci, che grazie al Professor Pozzato è diventato oggetto del mio tirocinio e della presente tesi.

Si tratta del progetto Farmchain, una piattaforma nata con l'obiettivo di permettere ai piccoli e medi produttori agricoli di raccontare la propria attività e utilizzarla come strumento di marketing, andando a sviluppare uno strumento centralizzato e di facile utilizzo da cui rendere accessibili ai produttori diversi canali di comunicazione digitali, nuovi e tradizionali.

All'interno del progetto Farmchain mi sono occupato principalmente dello creazione dell'infrastruttura server, dello sviluppo della parte backend e dell'integrazione della piattaforma con la blockchain. Essendo un team ridotto (al momento siamo due programmatori), ho comunque partecipato a tutti gli aspetti di sviluppo, entrando in contatto così con ogni aspetto di configurazione e progettazione di una piattaforma complessa.
Nella prima versione attualmente in fase di rilascio, abbiamo optato per una web app di veloce sviluppo basata sul framework PHP Laravel e la creazione di un editor evoluto in VueJS. L'utilizzo di queste tecnologie, da noi meglio conosciute, ci ha permesso di rilasciare velocemente la piattaforma e facilitare l'interazione con i produttori agricoli e con importanti realtà del settore in tempi brevi.

Una versione più completa, inizialmente sviluppata ma il cui rilascio abbiamo poi posticipato, prevede la creazione di una infrastruttura a microservizi sviluppata in NodeJS con sincronizzazione dei dati basata su eventi, per esporre servizi API facilmente scalabili e mantenibili, e di una mobile APP sviluppata in React Native, con cui documentare e gestire l'intero processo produttivo dell'azienda agricola. I dettagli tecnici e i motivi che ci hanno spinto a posticipare l'utilizzo di questa scelta sono dettagliati nel quarto capitolo.

Un elemento importante della piattaforma è l'integrazione della tecnologia Blockchain, già utilizzata in grandi realtà del settore agro-alimentare, con cui vogliamo aumentare la trasparenza percepita dagli utenti finali. Nella versione attuale l'utilizzo sarà limitato alla

possibilità di inserire documentazione relativa ai processi produttivi in blockchain, mentre nello sviluppo futuro il nostro obiettivo è utilizzarla per documentare in tempo reale tutti gli step della filiera in modo da offrire uno strumento in più a garanzia del processo produttivo.

Nel capitolo conclusivo, oltre a riassumere i risultati ottenuti e anticipare i prossimi sviluppi del progetto, vengono trattati i dubbi ed i limiti riscontrati nell'utilizzare la blockchain in questo contesto.

Capitolo 1

Il Progetto Farmchain

1.1 Ideazione

Il progetto nasce nel 2019, insieme ad alcuni amici e colleghi con l'obiettivo di creare un prodotto utile per valorizzare il lavoro ed i prodotti dei piccoli produttori agricoli della nostra regione.

Figura 1.1: Logo

Uno dei temi principali su cui ci siamo focalizzati inizialmente è stata la difficoltà per le aziende di piccole dimensioni di utilizzare gli strumenti digitali per far conoscere la qualità della propria produzione, limitandosi nei migliori dei casi all'utilizzo dei social.
Sempre nel corso del 2019 uno dei topic di maggior discussione, nel mondo informatico e non, era la blockchain, attenzione causata in maggior parte dal forte aumento di valore subito dal Bitcoin, la più datata e rinomata criptovaluta.

Nell'approfondire tale tecnologia, ci siamo imbattuti in interessanti utilizzi prototipali della blockchain all'interno del mondo agricolo. Le soluzioni prevedevano principalmente la notarizzazione di documenti relativi ai passaggi all'interno della filiera agricola in blockchain,

in modo da rendere più sicuro e trasparente la tracciabilità dei prodotti all'interno della filiera stessa. Tali informazioni, oltre che ad aiutare i passaggi tra i diversi attori dei processi produttivi, venivano infine utilizzate per costituire una prova di autenticità al consumatore finale.
Unendo questi due filoni, ovvero la necessità di creare nuove forme di comunicazione per le aziende agricole e le nuove possibilità offerte dalla blockchain, è nato il progetto Farmchain.

L'idea iniziale, che come si vedrà nei capitoli successivi ha subito diverse modifiche in corso d'opera, era quella di permettere al produttore di scattare foto e video tramite la nostra app per documentare e raccontare i vari passaggi produttivi.
In questa soluzione inziale, i media vengono memorizzati su protocollo IPFS insieme ad un json contenente la geolocalizzazione da cui è stato creato il media e relativo timestamp di acquisizione e viene successivamente salvato in uno smart contract l'hash del json caricato in IPFS.

Sempre dall'app, una volta finalizzato il prodotto da commercializzare, è possibile selezionare i media per costruire una cronistoria del prodotto, aggiungere eventuali contenuti extra come ricette e consigli d'uso e, una volta terminato, generare un QR code da applicare sulle etichette del prodotto per creare una vera e propria storia digitale.

Figura 1.2: Esempio di QR code da applicare sui prodotti

Per la visualizzazione dell'etichetta digitale abbiamo ricreato un formato simile a quello utilizzato nelle stories dei vari social network: una galleria di "slide" di video e foto a tutto schermo con dettagli testuali e link accessibili con uno "swipe-up". Questo per regalare familiarità agli utenti finali e sfruttare l'utilizzo mobile derivante dallo scan del QR code.

In aggiunta, abbiamo in progetto di inserire tali contenuti generati all'interno di un sito internet del produttore, automaticamente aggiornato con i contenuti generati tramite l'app, grazie all'integrazione di alcuni template di website da noi costruiti all'interno di un vero e proprio ecosistema Farmchain.

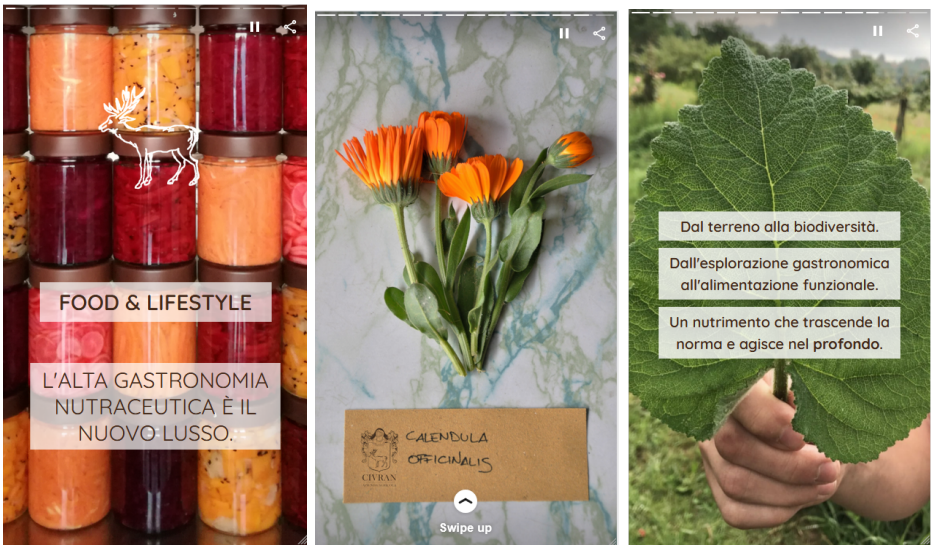

Figura 1.3: Etichetta digitale dell'azienda agricola Civran

Durante lo sviluppo di questa soluzione, come analizzato nelle sezioni successive di questo capitolo, abbiamo riscontrato limiti sia nell'adozione della blockchain, dovuti principalmente ai costi che comporta una blockchain pubblica, sia nell'introdurre una mobile App complessa, sviluppata con un team ridotto, in un settore dove gli strumenti digitali non sempre attecchiscono facilmente.

A questi problemi più tecnici, si è aggiunta la pandemia COVID-19, che ha reso ancora più difficile questa fase iniziale di testing e creazione di una rete di contatti, fattori fondamentali per permetterci di diffondere e perfezionare il nostro prodotto.
Questo però non ha fermato il nostro progetto ma semplicemente ci ha reso necessario apportare delle modifiche: inanzitutto la creazione di una vetrina digitale gratuita per produttori in difficoltà, descritta in una prossima sezione, in modo da dare un nostro contributo, seppur piccolo, al momento di difficoltà.

Abbiamo deciso di rivedere la piattaforma partendo da una web app di più veloce sviluppo e diffondibile per test e prove sul campo anche a distanza, bypassando così le difficoltà precedenti relative allo sviluppo di App mobile.
In questa accezione la blockchain assume un ruolo secondario, limitandosi a strumento di notarizzazione di documenti relativi al processo produttivo da affiancare alla narrazione costruita dal produttore, con tutti i limiti di questo tipo di soluzione che verranno approfonditi

nel capitolo conclusivo.
Abbiamo inoltre effettuato un upgrade del frontend dell'etichetta digitale, utilizzando le web stories sviluppate da Google all'interno del proprio web framework AMP (dettagliato nel prossimo capitolo), con una serie di vantaggi che vedremo in seguito.

1.2 Analisi del settore agroalimentare

Con Farmchain, fin dall'ideazione del progetto, abbiamo avuto la possibilità di confrontarci con diverse realtà del settore agro-alimentare, sia produttori sia importanti associazioni di categoria.
Quello che è emerso è che le piccole e medie aziende agricole difficilmente riescono a dare valore al proprio brand e alla storia della propria azienda perché per pubblicizzarsi si limitano ad utilizzare sporadicamente e con bassa qualità i canali digital (come i Social Network e il proprio sito vetrina) o scegliendo intermediari di distribuzione con Brand sinonimo di qualità.

Effettuare marketing digitale, sviluppare un sito-web mantenendolo costantemente aggiornato, sono attività che richiedono l'impiego di persone con competenze specifiche. Mentre quando poi si decide di affidarsi ad aziende terze, esse hanno dei costi molto alti e spesso sono poco allineate con gli obiettivi aziendali reali.
Le grandi aziende invece, avendo la possibilità di investire nel marketing digitale e tradizionale, puntano sulla creazione di una Brand-Identity che trasmetta al consumatore finale l'immagine di una filiera produttiva tradizionale e sostenibile, immagine spesso lontana dalla realtà.

Ne risulta che le PMI attive nel mondo agricolo non riescono a trasmettere in maniera efficace ed efficiente la storia del prodotto e dell'azienda, così da poter giustificare il prezzo più alto rispetto ai prodotti similari che la grande distribuzione organizzata tratta. La difficoltà nel raccontare tutti i processi produttivi e la qualità con cui vengono svolti, rendono difficile la creazione di un rapporto di fiducia attorno al proprio marchio.
Contestualmente il consumatore finale non è del tutto consapevole di ciò che sta acquistando e consumando e non percepisce l'importanza di un prodotto coltivato o allevato secondo tradizione.

Dopo aver realizzato una prima versione beta della nostra App mobile, abbiamo riscontrato numerose difficoltà nel renderla di facile utilizzo e pratica da utlizzare all'interno di una realtà agricola, nonostante la giovane età e la propensione alla tecnologia dei primi beta tester coinvolti.
É emerso che utilizzare uno strumento parallelo in aggiunta a quanto già utilizzato durante il lavoro sul campo, prevalentemente app di social network, richiede un effort che non viene accettato facilmente dai produttori, fattore che in una fase commerciale potrebbe diventare troppo negativo per noi.

Da qui, insieme al produttori coinvolti, la soluzione è diventata più "flessibile" nel permettere di utilizzare video e foto scattate in momenti diversi e con altre app, in modo da poter utilizzare contenuti già creati e facilitare la realizzazione di una etichetta narrante in un unico momento.
Tutto questo va a discapito dell'utilità della blockchain, in quanto non avendo più la possibilità di certificare i media acquisiti in tempo reale su di essa non vi è quel grado di trasparenza inizialmente previsto dal progetto.
Questo però ha portato due benefici: la possibilità di svincolare il progetto dai costi elevati derivanti dall'utilizzo di una blockchain pubblica, difficilmente sgravabili su aziende di piccole dimensioni, e il centrare Farmchain sulla narrazione del prodotto agricolo e dell'azienda, in modo da rendere più veloce la creazione di contenuti per il produttore.

1.3 Panoramica sullo sviluppo

Appena iniziato lo sviluppo della piattaforma, abbiamo deciso di optare per l'utilizzo di tecnologie nuove per due motivi: imparare nuovi linguaggi e approcci allo sviluppo applicativo e creare un'applicazione che sarebbe stato più facile far evolvere nel tempo e sopportare elevati carichi agevolmente, secondo le nostre più rosee aspettative.

La scelta è quindi ricaduta sulla creazione di una infrastruttura a microservizi, sviluppata in nodeJs, con cui esporre le API da cui poter accedere ai dati ed ai contenuti.
Parallelamente, abbiamo sviluppato un'app mobile in React Native, disponibile sia su iOS che Android, con cui l'utente potesse creare i propri prodotti, documentare le fasi produttive scattando foto e video, e infine catalogare i media ottenuti per creare l'etichetta digitale del prodotto.

Lo sviluppo è iniziato a metà 2019, e a fine anno abbiamo iniziato i primi test sul campo dell'app, grazie all'azienda agricola Civran di Trofarello.
Raccogliendo i primi feedback, abbiamo dovuto scontrarci con molte limitazioni dovute al tipo di smartphone utilizzato, che ha portato una mole di sviluppi elevata per un team così ridotto all'osso.

Abbiamo comunque cercato altri produttori per ampliare la platea di dispositivi con cui effettuare test sul campo, ma quest'attività ha subito un freno enorme a inizio marzo 2020, con l'arrivo della pandemia COVID-19 e relativo lockdown.
Dopo qualche settimana di spaesamento, abbiamo deciso di sviluppare DaTeATutti.it, un portale gratuito con cui i produttori agricoli e i piccoli venditori (negozi, banchi da mercato o produttori diretti) potessero gestire gli ordini da remoto e poi concludere la consegna della merce, cercando di aiutarli e sostituire il metodo un po' confusionario adottato degli ordini tramite whatsapp.

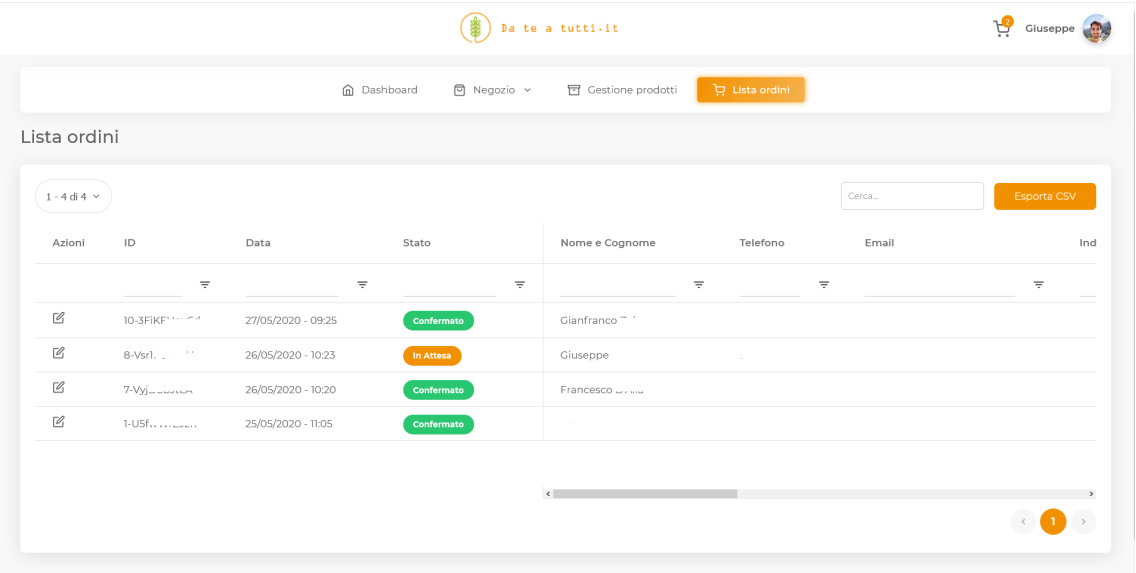

Figura 1.4: Schermata di gestione ordini di DaTeATutti.it

Dopo il rilascio del portale DaTeATutti.it, che ci ha aiutato ad entrare in contatto con diverse realtà agricole e capire meglio le loro esigenze e abitudini, abbiamo "sospeso" lo sviluppo della piattaforma per come era stata pensata e abbiamo ridisegnato l'intero progetto.
Ciò che la crisi pandemica e l'esperienza DaTeATutti.it ci ha insegnato è che l'accesso ad uno strumento digitale per il target di produttori a cui vogliamo rivolgerci deve essere il più immediato e facile possibile, ben oltre la semplificazione dell'interfaccia grafica dell'app a cui stavamo già lavorando.

Si tratta di portare al minimo i passaggi richiesti sia per accedere al servizio, quindi meglio una webapp che dover installare una app vera e propria, sia nei passaggi necessari per arrivare al prodotto finale, ovvero alla creazione dell'etichetta digitale.
Quindi la struttura di documentazione "real time" delle fasi produttive risultava troppo macchinosa in questa fase, aspetto ancor di più accentuato dal dover confrontarsi da remoto con i produttori agricoli.
Inoltre il fattore blockchain, per quanto fosse una parola attrattiva per le aziende più giovani, apportava un valore aggiunto minimo sul tipo di prodotto richiesto dal nostro target.

Abbiamo quindi deciso di ripartire con una versione "light" del progetto, incentrata esclusivamente sul far creare etichette digitali ai produttori, mettendo da parte il discorso blockchain come raccoglitore dei media creati ma relegandolo a semplice documentale facoltativo con cui arricchire la storia del prodotto.
La webapp sviluppata in questo modo, il cui primo rilascio è in corso, permette una maggior

facilità di interazione con i beta tester, essendo accessibile online, e immediatezza di utilizzo e fruizione, potendo usare i media esistenti del produttore.

1.4 Il team e la startup

Il team di Farmchain si è composto in diverse fasi e, ad oggi, conta quattro elementi, diventati poi cofondatori della startup About a Bit Srl.

Il team di sviluppo è composto da me e da Francesco D'Alia, full stack developer con esperienza in aziende di marketing digitale. Nella suddivisione dello sviluppo, pur interessandoci entrambi ad ogni aspetto essendo un team così ridotto, io mi sono occupato prevalentemente della parte infrastrutturale, backend e blockchain, mentre Francesco della parte frontend e dello sviluppo dell'app mobile.
La parte commerciale è gestita da Luca Femia, sales manager nel settore del Food, mentre la parte di comunicazione e grafica è sotto la supervisione di Fabrizio De Masi.

Dopo l'avvio del progetto, siamo entrati all'interno dell'incubatore I3P del Politecnico di Torino, con cui abbiamo sviluppato maggiormente la parte commerciale e finanziaria del progetto.
Nel settembre 2019 abbiamo fondato la società About a Bit Srl, iscritta, grazie al progetto Farmchain, nel registro delle imprese innovative.
Ad ottobre 2020, in piena fase di "refactoring" dello sviluppo, abbiamo avuto la nostra prima tirocinante dal Dipartimento di Informatica di Unito, Ilaria Frassetto, che ci ha permesso di accelerare i tempi di sviluppo.

Capitolo 2

Tecnologie utilizzate

In questo capitolo verranno analizzate le diverse tecnologie utilizzate nelle due differenti fasi di sviluppo, suddivise in base all'area tecnica di riferimento. In questa introduzione cercherò di indicare sommariamente come queste tecnologie siano state implementate nel progetto.

A livello infrastrutturale, abbiamo utilizzato i container Docker per eseguire le diverse componenti, sia per la fase di sviluppo che in produzione. Tale soluzione, oltre ad ottimizzare lo sviluppo all'interno del team, permette una gestione ottimale dei microservizi.
Utilizzando la piattaforma cloud AWS, abbiamo utilizzato il servizio ECS per eseguire i container, che come vedremo porta diversi benefici in questa fase del progetto.

La prima versione del progetto consiste in una webapp sviluppata in Php, utilizzando il framework Laravel, che comprende tutte le parti principali dell'applicazione (autenticazione, gestione dati, salvataggio media).
L'editor con cui vengono create e modificate le storie digitali è stato sviluppato separatamente in VueJS.
Le storie così costruite, memorizzate come JSON sul database della webapp, vengono poi utilizzate per creare delle pagine, le vere e proprie storie visualizzabili dall'utente finale, create attraverso il framework web Google AMP, nello specifico utilizzando il formato Web Stories.

La versione più "completa" del progetto, che verrà ripresa in una successiva fase di sviluppo, consiste in microservizi sviluppati in NodeJS che espongono API attraverso un web server NGINX. I dati vengono gestiti utilizzando il database non relazionale MongoDB. Ogni microservizio ha una propria istanza, per avere una totale indipendenza tra microservizi, e la sincronizzazione di dati comuni che devono essere replicati tra istanze differenti avviene utilizzando il gestore di eventi Apache Kafka.
Come frontend utilizzeremo una mobile App in React Native, disponibile sia su iOS che Android.
Come blockchain su cui salvare le informazioni dei media generati tramite l'app abbiamo

scelto Ethereum e sviluppato uno Smart Contract in linguaggio Solidity, integrando il protocollo IPFS per la decentralizzazione dei file.
Per integrare la blockchain con il servizio nodeJS, abbiamo utilizzato le librerie web3.js.

2.1 Infrastruttura

2.1.1 Docker

Docker è una tecnologia open-source che permette la creazione e l'esecuzione di container, ovvero di un ambiente standardizzato isolato contenente tutto il necessario per eseguire un'applicazione.
Questa tecnologia nasce con l'obiettivo di rendere semplice la distribuzione di un software, creando pacchetti facilmente portabili.

Un container è basato su un'immagine contenente il kernel Linux a cui vengono aggiunti software in base alle necessità.
Per configurare un container, come prima cosa è necessario comporre il Dockerfile: si tratta di un file di configurazione in cui specificare il tipo di immagine, in base alle necessità dell'applicazione da sviluppare, e settare diverse operazioni da effettuare sia sui file, copiandoli o rendendoli accessibili nel container, e comandi da eseguire all'interno del container in fase di build.

Di seguito un esempio di Dockerfile di una semplice applicazione NodeJS.

```
1  FROM node:8-alpine
2
3  WORKDIR /app
4
5  COPY ./src/ /app/src/
6
7  COPY package.json package-lock.json /app/
8
9  RUN mkdir /app/storage
10
11 RUN npm install && apk add --no-cache bash
12
13 CMD ["node", "src/index.js" ]
```

Una volta eseguita la build, viene creata un'immagine contenente tutta l'applicazione e il necessario per eseguirla, dalle librerie al codice sorgente.
In fase di esecuzione, è possibile settare eventuali attributi, come porte da esporre, mount di cartelle locali e altre personalizzazioni.

Per orchestrare l'esecuzione di diversi container contemporaneamente, soprattutto in fase di sviluppo, è possibile utilizzare il tool Docker-compose, che permette di creare un file

con cui configurare per ogni immagine buildata i diversi attributi di esecuzione con relative dipendenze tra container.

Di seguito alcune parti del docker-compose.yml utilizzato in Farmchain nella soluzione con microservizi.

Esempio di microservizio in ambiente di sviluppo, con utilizzo di volumi locali e npm run dev per permettere l'aggiornamento immediato dell'applicazione durante lo sviluppo senza necessità di rebuildare continuamente, e l'utilizzo di variabili d'ambiente.

```
1   version: '3'
2  services:
3    farm:
4      container_name: farm-service
5      build: ../farm-service/
6      volumes:
7       - farm-data:/app/src
8       - storage-data:${FARM_STORAGE_PATH}
9      expose:
10      - ${FARM_SERVER_PORT}
11     links:
12      - mongo
13      - kafka
14     environment:
15      - DB_HOST=${DB_HOST}
16      - DB_NAME=${FARM_DB_NAME}
17      ........
18      ........
19     command: npm run dev
```

Per poter utilizzare applicazioni terze in ambiente di sviluppo e renderle portabili, è possibile utilizzare altre immagini messe a disposizioni in repository, come ad esempio Docker Hub, configurarle e collegarle ai propri container.
Nel nostro caso, abbiamo linkato i servizi necessari alla sincronizzazione dei dati tra servizi, che in ambiente di produzione sono invece stati installati separatamente.

```
1    zookeeper:
2      image: wurstmeister/zookeeper:3.4.6
3      expose:
4      - "2181"
5
6    kafka:
7      image: wurstmeister/kafka:2.11-2.0.0
8      depends_on:
```

```
9       - zookeeper
10      ports:
11      - "9092:9092"
12      environment:
13        KAFKA_ADVERTISED_HOST_NAME: kafka
14        KAFKA_ADVERTISED_PORT: 9092
15        KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
16        KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092
17        KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
18        KAFKA_CREATE_TOPICS:
          ↪  "service.product:1:1,service.farm:1:1,service.blockchain:1:1"
19    mongo:
20      container_name: mongo
21      image: mongo
22      environment:
23       - MONGO_INITDB_ROOT_USERNAME=${DB_ROOT_USER}
24       - MONGO_INITDB_ROOT_PASSWORD=${DB_ROOT_PASSWORD}
25      ports:
26        - "27017:27017"
```

2.1.2 AWS

AWS è l'acronimo di Amazon Web Services, l'azienda di proprietà di Amazon che offre servizi di cloud computing. Offre un vasto ventaglio di servizi, e ad oggi si tratta della soluzione cloud più ampia e più diffusa.
Abbiamo avuto la possibilità, grazie alla nostra partecipazione nell'incubatore I3P del Politecnico di Torino, di avere crediti per utilizzarne i servizi per il nostro progetto.

Abbiamo quindi deciso di esplorarne le capacità: di seguito alcuni dei servizi da noi utilizzati.

- EC2 (Elastic Computing): istanze di server in cloud altamente personalizzabili, sia nelle caratteristiche tecniche che nelle modalità di utilizzo, da instanze On Demand a istanze dedicate.
- ECS (Elastic Container Service): servizio che permette l'esecuzione e l'orchestrazione di container Docker. É possibile scegliere di eseguire i container su proprie istanze server EC2 oppure utilizzando la modalità Fargate, che mette a disposizione la potenza di calcolo necessaria al carico richiesto dal singolo servizio. Questa modalità risulta ottimale in fase iniziale, in quanto ha costi minori a fronte di carichi minori.
- S3 (Simple Cloud Storage): servizio di storage scalabile e ad alte prestazioni, offre API software per diversi linguaggi di programmazione.

- ELB (Elastic Load Balancer): servizio di load balancing con cui indirizzare da un unico punto le richieste verso diverse destinazioni.
- CloudFront: offre servizio di CDN (Content Delivery Network) da collegare al proprio load balancer, in modo da ridurre al minimo la latenza nelle richieste grazie al caching geografico sulla rete.

2.1.3 MongoDB

MongoDB è un DBMS NoSQL, ovvero non relazionale: questo significa che non utilizza la struttura classica dei database relazioni, in cui gli oggetti sono ben definiti e in relazione tra di loro, ma sfrutta una struttura più dinamica memorizzando i documenti in strutture JSon-Like (chiamati Bson).

Questo tipo di database porta diversi vantaggi:

- Maggior scalabilità: non avendo una struttura di documento predefinita, i dati vengono strutturati in base a quanto descritto lato applicazione, rendendo più semplici evoluzioni nel tempo.
- Essendo strutturato a documenti, è possibile inserire tutte le informazioni necessarie in una singola entità per diminuire il numero di interrogazioni al database, non dovendo ripercorrere tutte le relazioni per reperire i dati.
- Facilità di implementazione e stretta correlazione in motori runtime come NodeJS.

Ovviamente tale archittettura non è adatta a tutte le soluzioni: infatti in contesti con una struttura dati ben definita e con relazioni importanti ai fini di integrità del dato, è sicuramente sconsigliabile utilizzare un dbms schema-free.
Inoltre, la soluzione proposta da MongoDB è solo una di quelle possibili di dbms non relazionali, ma essendo una tecnologia per noi nuova siamo partiti da quella più diffusa e maggiormente documentata.

2.1.4 Apache Kafka

Apache Kafka è una piattaforma open-sorce di stream-processing, utilizzabile per ricevere dati da diversi tipi di sorgenti (detti producer), ed elaborarli rendendoli disponibili ai riceventi (consumer).
I messaggi vengono categorizzati in topics, con cui poi interagiscono i producer e i consumer. É possibile inoltre creare dei gruppi in modo da creare un insieme di consumer iscritti allo stesso producer.

Un topic è costituito da partizioni dinamiche, ognuna di dimensioni differenti, per permettere la scalabilità del prodotto, per esempio grazie alla creazione di gruppi e associazione delle partizioni ai relativi consumer del gruppo.

All'interno della nostra piattaforma, Apache Kafka è stato utilizzato per gestire la sincronizzazione dei dati tra servizi diversi, grazie all'emissione di eventi.

Utilizzando la libreria kafka-node, ogni servizio NodeJS inizializza un client, con cui creare un consumer in ascolto sullo stream, che viene elaborato ad ogni evento, ed un producer che pubblica eventuali eventi.

Esempio di publish di un evento, in cui viene specificato il topic su cui pubblicarlo, la tipologia e il dato da condividere con i consumers.

```
....
var publishEvent =await
↪  options.kafkaService.publishEvent("service.product","create.media",media);

....
const publishEvent = async (topic,event,data) =>  {

      let payloads = [
        {
          topic: topic,
          messages: JSON.stringify({event: event, data: data})
        }
      ];
      let push_status = producer.send(payloads, (err, data) => {
        if (err) {
          throw Error(err)
        } else {
          console.log("pubblicato evento " + event + " in topic " + topic
          ↪  + "with data : " + data )
          return;
        }
      });
    }
......
```

Esempio di consumer, con l'associazione di diverse funzionalità in base all'evento ricevuto.

```
    let consumer = new Consumer(
     client,
     kafkaOptions,
     kafkaConsumerOptions
     );

```

```

    var listenerFunctions = {
      "update.farm" : (repo,farm) => {
          return updateFarm(repo, farm)
        },
      "update.dealer" : (repo,dealer) => {
          return updateDealer(repo, dealer)
        },
      "create.blockchain" : (repo,media) => {
        return updateMedia(repo, media)
      },

    }

    consumer.on('message', async function(message) {

      var message_parsed = JSON.parse(message.value);
      listenerFunctions[message_parsed.event](repo,message_parsed.data)

    })
```

2.2 Backend

2.2.1 Laravel

Laravel è un framework PHP con cui sviluppare applicazioni web, che utilizza il pattern architetturale MVC (Model View Controller).
Permette quindi facilmente di costruire una webapp con un codice solido e ben strutturato, mettendo a disposizione dello sviluppatore tutti gli strumenti necessari.

Routing

Il routing in Laravel viene gestito attraverso un file, routes/web.php, in cui definire tutti i vari path da cui è possibile accedere all'applicazione.
Per ogni route è possibile restituire direttamente una pagina, chiamata View, oppure passare l'elaborazione della chiamata ad un Controller.

Ad ogni route può venire collegato un middleware, classe che si occupa di effettuare delle verifiche prima di dare accesso alla risorsa, solitamente per verificare le permission dell'utente.

Esempio di route con gestione delegata ad un controller.

```
Route::get('/', [StoryController::class,
↪  'dashboard'])->middleware(['auth'])->name('dashboard');
```

Esempio di route dinamica con associazione direttamente nel path a variabili o oggetti univoci.

```
Route::get('/stories/content/{token}/{userID}', [StoryController::class,
↪ 'getContent'])->middleware(['web'])->name('stories.getContent');
```

Gestione dei dati

In Laravel l'interfaccia con il database è gestita tramite l'ORM Eloquent, che permette una facile trasposizione degli oggetti database in Model.
Mette inoltre a disposizione diversi metodi con cui creare e gestire le relazioni ed effettuare query.

Esempio di Model con una relazione 1 a molti tra Users e Stories.

```
class Story extends Model
{

    use SoftDeletes;

    /**
     * Get the user that owns the story
     */
    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

View

Per quanto riguarda la parte frontend, vengono utilizzate delle pagine PHP dinamiche, richiamabili dai Controller o direttamente da Routes, in cui inserire sia codice html statico che php dinamico, oltre ovviamente ad eventuali librerie javascript e fogli di stile.

Nella nostra piattaforma, abbiamo anche adattato una pagina in formato AMP per renderla dinamica in base all'etichetta digitale da caricare.

```
<body>
    <!-- Cover page -->
    <amp-story standalone title="{{ $title }}" publisher="Farmchain"
    ↪ publisher-logo-src="assets/AMP-Brand-White-Icon.svg"
    ↪ poster-portrait-src="assets/cover.jpg">

..............

```

```
7           <!-- Page 2 (Dog): 2 layers (fill + thirds) -->
8           @foreach ($stories as $id => $story)
9           @if ($id > 0)
10          <amp-story-page id="{{ $id }}">
11              @if ($story['video'])
12              <amp-video autoplay loop width="720" height="1280"
                ↪  layout="responsive" controls>
13                  <source src="{{ $story['video'] }}" type="video/mp4">
14              </amp-video>
15              @else
16              <amp-story-grid-layer template="fill" style="background: {{
                ↪  $story['style']['background'] }}">
17                  @if (preg_match('#\((.*?)\)#',
                    ↪  $story['style']['background'], $match))
18                  <amp-img src="{{ $match[1] }}" width="720" height="1280"
                    ↪  layout="responsive">
19                  </amp-img>
20                  @endif
21              </amp-story-grid-layer>
22              @endif
23
24  ............
25      </amp-story>
26  </body>
```

2.2.2 Nodejs

NodeJS nasce per dare la possibilità di utilizzare a runtime il linguaggio di programmazione javascript, utilizzato precedentemente nella programmazione web lato client.
Per fare ciò, è stato utilizzato il motore di runtime V8, sviluppato da Google all'interno del proprio browser Chrome, e reso eseguibile in modalità stand-alone.

La pecurialità di NodeJS è che permette di creare applicazioni semplici molto performanti per funzionalità specifiche.
Al contraro di Laravel, framework per gestire tutti gli aspetti di un'applicazione web, NodeJS permette di programmare API dalle funzionalità molto specifiche, quindi senza dover sovraccaricare il programma con librerie e packages non utilizzati.
Ha un gestore di pacchetti molto evoluto e funzionale, npm, con cui gestire tutte le librerie necessarie.

Per il nostro progetto Farmchain, verrà utilizzato per sviluppare le API backend dei vari servizi, utilizzando la libreria Express.js per la gestione delle chiamate.
Esempio di implementazione di una API in nodejs in Farmchain.

```
...
const app = express()
app.use(formData.parse({
            uploadDir: options.storagePath,
            autoClean: true
        }));
const farmApi = require('../api/farms')(options)
        const lotsApi = require('../api/lots')(options)
        const dealersApi = require('../api/dealers')(options)
        const productsApi = require('../api/products')(options)
        app.use('/farm',farmApi)
        app.use('/farm',lotsApi)
        app.use('/farm',dealersApi)
        app.use('/farm',productsApi)
....
    router.post('/', async (req,res) => {
        const farmData = {
            name: req.body.name,
            address: req.body.address,
            mail: req.body.mail,
            phone: req.body.phone,
            logo: req.body.logo,
            websiteURL: req.body.websiteURL,
            description: req.body.description
        }

        try{
            var farm = await repo.createFarm(farmData)
            farmData._id = farm._id
...
```

2.3 Frontend

2.3.1 VueJS

VueJS è un framework frontend javascript utilizzato per la creazione di applicazioni web e single page application.

Questo framework adotta un pattern MVVM: viene creato un binding (ViewModel) tra struttura dei dati (Model) e interfaccia grafica (View), in modo che interagendo con i dati e le variabili che compongono l'applicazione venga manipolato il DOM della pagina web in maniera automatizzata.

É un framework molto flessibile e facilmente scalabile, in quanto è possibile inserire all'interno di un'applicazione esclusivamente i componenti di cui si necessita, senza dover utilizzare tutta la sovrastruttura come in altri framework.

Nella versione di Farmchain attualmente in fase di rilascio è stato sviluppato l'editor di storie digitali completamente in VueJS, utilizzando la libreria vue-awesome-swiper per costruire uno slider touch per ricostruire l'esperienza di storie in successione presente nelle applicazioni social.

Esempio di implementazione per la creazione di una nuova storia, con un array di slides vuoto, la configurazione dello slider e un metodo per aggiungere una nuova storia.

```
1  .........
2   <swiper
3          class="swiper"
4          ref="mySwiper"
5          :options="swiperOption"
6          @slide-change="updateCurrentIndex"
7        >
8          <swiper-slide
9            v-for="(slide, index) in swiperSlides"
10           :key="index"
11           :sty
12 .......
13 export default {
14   name: "swiper-social-label",
15   data() {
16     return {
17       .............
18       swiperSlides: [
19         {
20           id: 0,
21           style: {
22             background: this.randomBgColor(),
23           },
24           title: "",
25           description: "",
26           video: null,
27         },
28       ],
29       swiperOption: {
30         effect: "coverflow",
31         grabCursor: true,
32         centeredSlides: true,
```

```
33          slidesPerView: "auto",
34          coverflowEffect: {
35            rotate: 0,
36            stretch: 0,
37            depth: 100,
38            modifier: 1,
39            slideShadows: true,
40          },
41          pagination: {
42            el: ".swiper-pagination",
43            clickable: true,
44          },
45        },
46      };
47    },
48
49    ...........
50    methods: {
51      addSlide() {
52        this.slideId = this.swiperSlides.reduce((a,b)=>a.id>b.id?a:b).id +
          ↪  1;
53        let defaultSlide = {
54          id: this.slideId,
55          style: {
56            background: this.randomBgColor(),
57          },
58          title: "",
59          video: null,
60          swipeUp: {
61            text: null,
62            url: false,
63          },
64        };
65        this.swiperSlides.splice(this.swiper.activeIndex + 1, 0,
          ↪  defaultSlide),
66          this.$nextTick(function () {
67            this.swiper.slideTo(this.swiper.activeIndex + 1);
68          });
69      },
```

2.3.2 AMP

AMP, acronimo di Accelerated Mobile Pages, è un progetto open-source di Google il cui scopo è quello di creare pagine web leggere e performanti per facilitare la navigazione mobile.

Utilizza una propria struttura di tag che riprende la classica struttura HTML di una pagina web ma ottimizzandone il contenuto, creando uno standard utilizzabile dagli sviluppatori.
Inoltre, Google ha sviluppato una Google AMP Cache in cui inserire i contenuti sviluppati tramite questo framework e renderli accessibili quasi istantaneamente tramite i propri risultati di ricerca.

Mentre questa tecnologia è ormai largamente utilizzata per velocizzare e migliorare il posizionamento delle pagine web, è stata da poco inserita una nuova componente, chiamata Web Stories, con cui creare, sempre utilizzando i tag AMP, delle pagine con una struttura simile alle storie dei social network.
Ed è proprio questa componente che viene utilizzata all'interno di Farmchain per costruire le etichette digitali dei produttori.
Oltre ai vantaggi già citati, le AMP web stories sono di facile sviluppo, in quanto si utilizzano pochi tag specifici e ben supportati offerti dal framework.

Di seguito uno stralcio di web story, con il dettaglio della struttura utilizzabile con tag per inserire immagini, sezione raggiungibile con lo swipe-up e il player dedicato ai video Youtube.

```
1        <amp-story-page auto-advance-after="3s" id="cover">
2          <amp-story-grid-layer template="fill">
3            <amp-img src="imgs/GWS 05.png"
4                width="720" height="1280"
5                layout="responsive">
6            </amp-img>
7          </amp-story-grid-layer>
8          <amp-story-grid-layer template="thirds">
9            <div class="front-layer" grid-area="upper-third">
10             <p class=" page-title bold">Il nostro cavolfiore
               ↪  fermentato.</p>
11           </div>
12         </amp-story-grid-layer>
13         <amp-story-page-attachment layout="nodisplay" data-title="I
           ↪  nostri fermentati" theme="light" data-cta-text="Scopri i
           ↪  nostri fermentati!">
```

```
14            <p class="swipe-up">In questo video vi racconto come creo i
              ↪  miei fermentati e vi presento l'Alta Gastronomia
              ↪  Nutraceutica delle WildDeerNess Box, preziosi cofanetti
              ↪  prodotti in serie limitata</p>
15            <amp-youtube
16            data-videoid="X5Ya-8AIqdQ"
17            layout="responsive"
18            width="480" height="270">
19        </amp-youtube>
20          </amp-story-page-attachment>
21        </amp-story-pa
```

2.3.3 React Native

React Native è un framework javascript con cui sviluppare applicazioni per iOS e Android.

Al contrario di altri framework cross-platform, React Native non crea view html o componenti ibridi per il funzionamento dell'applicazione, ma utilizza i componenti nativi dei due sistemi operativi, utilizzando un Bridge che permette il dialogo tra la parte Javascript e i thread in linguaggio nativo del dispositivo.
Permette inoltre di aggiungere componenti scritte in linguaggio nativo sul singolo sistema operativo mobile, aumentandone la flessibilità.

2.4 Blockchain

La blockchain è una struttura dati condivisa e immutabile, in cui i dati vengono inseriti in blocchi collegati tra di loro in ordine cronologico di inserimento. Nasce nel 2009 come tecnologia utilizzata dalla cryptovaluta Bitcoin.
Esistono diversi protocolli con cui un blocco viene autorizzato e riconosciuto da tutti i nodi della blockchain: il più utilizzato è la Proof-of-Work, un protocollo di consenso in cui diversi attori, chiamati miner, effettuano un lavoro di decodifica di hash in una sorta di competizione. Il primo che "risolve" l'hash inserisce il blocco nella catena e riceve una ricompensa, che comprende le fee pagate dai vari account che hanno inserito transazioni all'interno del blocco in oggetto.

Esistono altri protocolli di consenso, come la Proof-of-Stake, meno dispendiosi dal punto di vista energetico e basati su fattori differenti.

2.4.1 Ethereum

Ethereum è una blockchain che, a differenza di quella di Bitcoin, permette l'esecuzioni di codice al proprio interno: infatti ogni nodo della sua rete non mette a disposizione solo lo storage, detenendo una copia completa della chain, ma anche la propria potenza di calcolo, andando a creare la cosiddetta Ethereum Virtual Machine, o EVM.

Per poter utilizzare la rete ed eseguire applicazioni è necessario sviluppare delle applicazioni in Solidity ed effettuare una transazione per poter deployare il codice sulla rete Ethereum.

2.4.2 Smart Contract

Uno Smart Contract è un'applicazione sviluppata ed eseguita sulla blockchain, così chiamata perchè solitamente svolge una funzione di contratto tra parti, spesso dietro ad un'operazione di pagamento in cryptovalute. Nella realtà ormai esistono smart contract di ogni tipo, che svolgono le funzioni più disparate.
Il codice di programmazione utilizzato per sviluppare gli Smart Contract su Ethereum è Solidity, un linguaggio ad oggetti simile ad ECMAScript.

Per Farmchain abbiamo sviluppato uno Smart Contract che associa ad un singolo prodotto tutti i documenti caricati su protocollo IPFS, creando una struttura dati con tutti gli hash generati.
In questo modo è possibile recuperare dallo smart contract associato ad un prodotto tutti i relativi documenti. Nella prossima versione invece, saranno associati i json contenenti le informazioni di acquisizione dei media.

2.4.3 IPFS

IPFS, acronimo di InterPlanetary File System, è un protocollo per la condivisione di un file system decentralizzato e distribuito, con un funzionamento peer to peer simile a quello di BitTorrent.
Ogni client della rete può condividere file, i quali saranno poi raggiungibili da chiunque attraverso un namespacing globale che utilizza l'hash SHA-256 del file, reso univoco tramite la presenza di una tabella hash distribuita.

Essendo un progetto gratuito, è ottimale il suo utilizzo in coppia ad uno smart contract: si sfrutta IPFS per memorizzare i file e si limita la dimensione dello smart contract stesso relegando l'uso della blockchain esclusivamente per eseguire la logica applicativa.

All'interno di Farmchain, abbiamo utilizzato in NodeJS la libreria ipfs-http-client con cui istanziare il client IPFS e memorizzare sulla rete il file caricato dall'utente.

```
1  const client = ipfsClient(new URL('http://'+ ipfsSettings.host + ':' +
   ↪  ipfsSettings.port))
2
3      var ipfsResponse = await client.add(urlSource(mediaURL + media.src))
```

2.4.4 web3

Web3.js è una libreria con cui è possibile far interagire un'applicazione in NodeJS con la blockchain di Ethereum.

Mette a disposizione diversi metodi per creare smart contract e per richiamarne i singoli metodi.

Capitolo 3

Farmchain 1.0

In questo capitolo viene descritta l'implementazione della prima versione della piattaforma Farmchain, attualmente in fase di rilascio.

3.1 Requisti della piattaforma

A seguito del primo sviluppo, che verrà ripreso ed è dettagliato nel capitolo successivo, abbiamo potuto analizzare i feedback ricevuti dai test sul campo e dal confronto con diverse realtà del settore agricolo.
Ciò ci ha permesso di ridisegnare l'intero progetto per poterlo adattare meglio alle esigenze del nostro target.

Il primo importante feedback è stato la difficoltà di utilizzo dell'app da parte dei produttori agricoli: nonostante una UI il più possibile semplice e una UX migliorata costantemente insieme ad alcuni produttori agricoli, l'utilizzo di un'app per scopi di comunicazione non è un effort accettabile, almeno in questa fase iniziale, per la maggior parte degli utenti.
Mentre per attività legate al proprio settore (per esempio per gestione dell'attività, analisi del campo, gestionale etc..) un'app dedicata viene percepita positivamente dagli utenti, per fini di comunicazione e marketing l'effort necessario non viene facilmente digerito durante l'attività produttiva.
Sostanzialmente, gli utenti interessati all'aspetto comunicativo non desiderano dover utilizzare un'app per generare media in aggiunta a quelle social (o native) che già usano.

Lasciare quindi agli utenti la libertà di costruire la storia digitale del proprio prodotti comporta due grosse implicazioni: l'eliminazione della blockchain per tracciare la creazione dei media e la possibilità di utilizzare inizialmente una semplice web app, di più veloce sviluppo e distribuzione.
Abbiamo deciso dunque di sviluppare un'applicazione web che si focalizzasse sulle reali necessità dei produttori coinvolti nei test: creare la storia del prodotto con testi, immagini

e video e un'etichetta digitale accattivante che portasse un valore aggiunto all'esperienza dell'utente finale.

Nelle prossime sezioni del capitolo verranno analizzati i diversi aspetti dello sviluppo e le prime prove sul campo di questa versione di Farmchain.

3.2 Farmchain Webapp

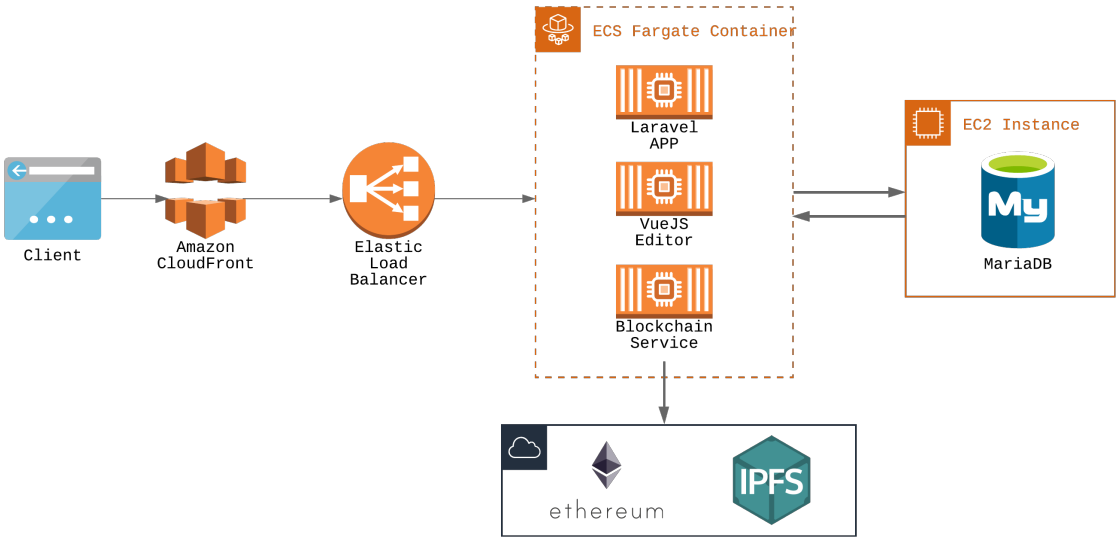

Figura 3.1: Architettura Cloud della piattaforma

Questa prima versione della webapp di Farmchain si divide sostanzialmente di due componenti: una parte "core" dell'applicazione sviluppata utilizzando il framework PHP Laravel e l'editor web delle stories sviluppato in VueJS.

Si tratta di due applicazioni separate, entrambe sviluppate all'interno di container Docker dedicati, che comunicano attraverso la condivisione di un token, cui riconoscere utente ed eventuale singolo prodotto in entrambe le applicazioni.
La scelta di dividerle è nata per permetterci in un secondo momento di modificare l'implementazione della parte backend, mantenendo invece quanto sviluppato dell'editor.

A livello di sviluppo, il mio contributo è stato principalmente relativo alla creazione dell'infrastruttura Cloud, della containerizzazione delle applicazioni e della progettazione del backend, mentre per la parte Laravel e Frontend ho contribuito principalmente in fase di ideazione e progettazione.

Per quanto riguarda l'infrastruttura ospitante i container, abbiamo utilizzato i servizi cloud offerti da Amazon AWS.
I due container Docker vengono eseguiti tramite il servizio Amazon ECS (Elastic Container Service), che si occupa di gestire i container come se fossero servizi caricando le build dei container depositate sul Repository. Una volta in esecuzione, il servizio Amazon ELB (Elastic Load Balancing) si occupa di gestire le chiamate in entrata e dirigerle verso il servizio corretto in base al path della richiesta.

Infine il servizio Amazon CloudFront si occupa di creare una url pubblica (a cui punterà il record DNS del dominio farmchain.it) collegata al bilanciatore interno, oltre ad effettuare un servizio di caching CDN (Content Delivery Network) per migliorare la velocità di caricamento del sito.

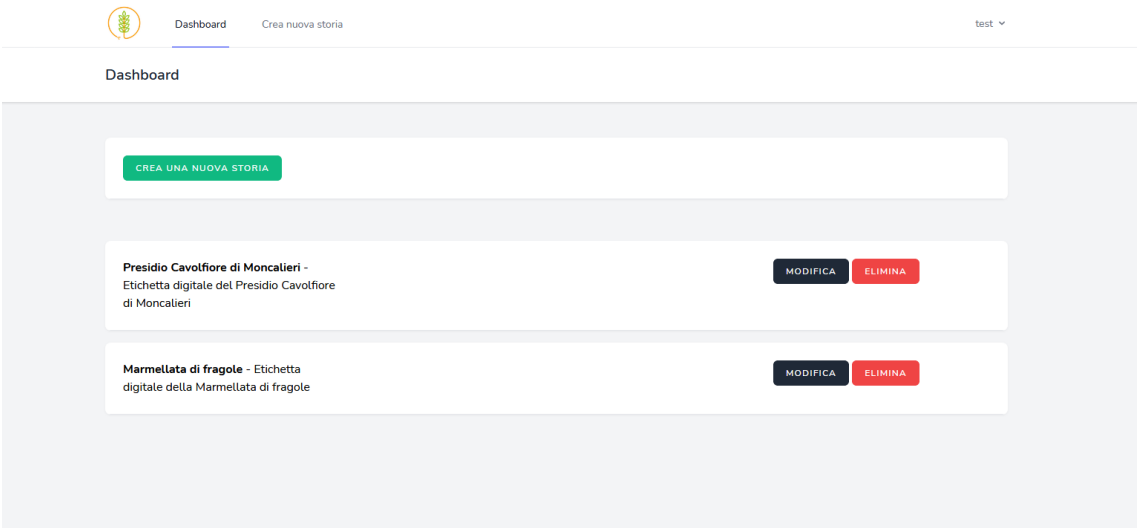

Figura 3.2: Schermata di gestione delle etichette digitali

La componente in Laravel si occupa sostanzialmente i gestire l'autenticazione, l'interfaccia dati con il database (un'istanza MariaDb) e la creazione di alcuni funzionalità basilari dell'applicazione.
Per tutte queste componenti, abbiamo sfruttato le relative funzionalità offerte dal framework, dettagliate nel secondo capitol: autenticazione e profilazione degli utenti, utilizzo dell'ORM Eloquent come interfaccia col database e utilizzo delle View Laravel per le pagine frontend. Questa scelta, per quanto non in linea con l'idea di creare una Spa (single page application), ci ha permesso di velocizzare lo sviluppo e recuperare un po' del tempo perduto.

L'editor invece vero e proprio delle stories, applicazione esclusivamente frontend, è stato sviluppato utilizzando VueJS, cercando di simulare una versione basic di quanto offerto dalle app social: si tratta di uno slider, creato attraverso il package SwiperJs, con cui

costruire le diverse storie (ogni slide è una singola storia) effettuando l'upload di foto e video, visualizzando in tempo reale una anteprima del risultato finale.

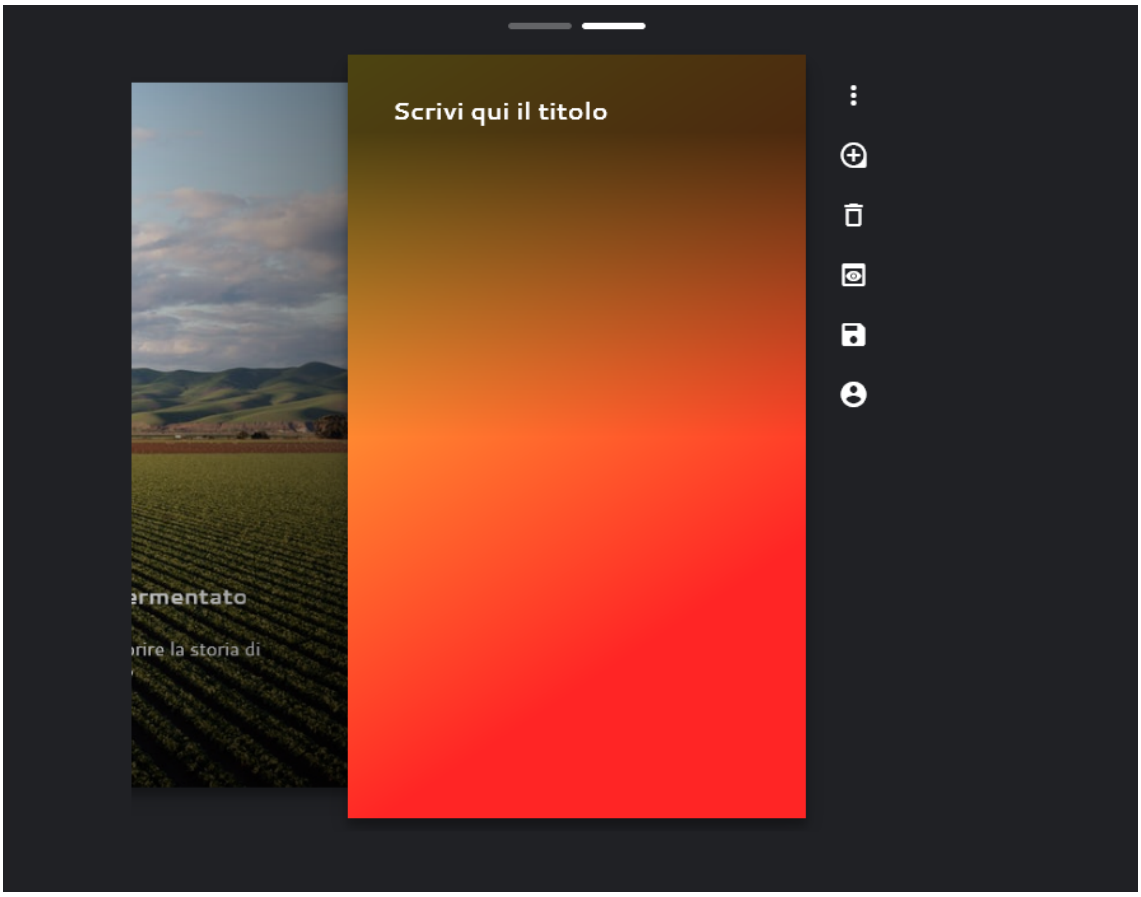

Figura 3.3: Editor delle web stories

Per non rendere troppo complessa l'app sia in termini di sviluppo sia di utilizzo, al momento non è presente una funzionalità di drag&drop di elementi grafici sulla storia, ma elementi utilizzabili prefissati, come testo, dettagli in swipe up ed eventuali link esterni. Questo permette di rendere il risultato finale più omogeneo tra produttori diversi.
I contenuti caricati sullo slider vengono memorizzati su di una struttura Json, che al momento del salvataggio viene inviata in un formdata verso una API esposta da Laravel, insieme ai relativi file caricati.
Avviene quindi la creazione di una etichetta associata all'utente, il salvataggio sul db della struttura del json e l'upload dei file sull'host del server.
In caso invece di modifica di un'etichetta esistente, verrà aperta l'app dell'editor con un token con cui da VueJS viene recuperato il json relativo tramite chiamata API verso Laravel, che effettua anche il controllo di autenticazione tramite middleware.

3.3 AMP e le web stories

L'etichetta digitale è una componente fondamentale del progetto Farmchain: si tratta della vetrina virtuale, accessibile tramite QR code e direttamente online, con cui presentare le aziende clienti ed i loro prodotti ai consumatori. É quindi l'impatto generato è fondamentale sia massimo.

All'inizio del progetto, abbiamo testato e sviluppato diverse pagine web da utilizzare come etichette digitali.
Inizialmente, prendendo spunto da altri progetti simili, abbiamo creato etichette più "informative", rendendo disponibili all'utente tutte le varie sezioni in cui navigare.

Questo approccio, oltre ad essere già utilizzato in progetti di altre aziende, non metteva abbastanza in risalto l'obiettivo del progetto, che è quello di far raccontare in prima persona al produttore la propria storia e quella del prodotto che l'utente finale si trova ad acquistare o a consumare.

Figura 3.4: Primo prototipo di etichetta digitale

Da qui, nasce l'idea di mettere in primo piano le immagini catturate durante e dopo la produzione, direttamente in azienda, per far sì che fosse ancora più diretto il contatto produttore-cliente finale.
Per far ciò abbiamo sviluppato una pagina in React Js, libreria javascript frontend, con cui simulare le storie utilizzate da diversi Social Network direttamente sul browser, accessibili dopo aver scannerizzato il QR code dal prodotto o da materiali pubblicitari dell'azienda.
Abbiamo quindi utilizzato il package react-insta-stories, un pacchetto React creato appo-

sitamente per simulare le storia Instagram su una pagina web, apportando delle modifiche per renderlo funzionante sul nostro progetto.

Si tratta ovviamente di una pagina dinamica: al caricamento della pagina, viene effettuata una chiamata verso le nostre API con l'id del prodotto richiesto, che restituisce il file json con la struttura completa della storia del prodotto con cui creare ricorsivamente le storie della pagina web.
In questo modo la stessa struttura web, accessibile nel nostro server, è utilizzabile per tutti i clienti, con le varie personalizzazioni inserite all'interno del file Json salvato nel database.

Questo formato ha ottenuto un ottimo riscontro sia nel settore, tra i diversi produttori coinvolti, sia con gli utenti finali, dando così al progetto la possibilità di provare a creare un nuovo strumento di comunicazione, come vedremo meglio successivamente nella prova sul campo.

Nel corso della fine del 2020 abbiamo però avuto modo di approfondire il framework Google AMP, che permette la creazione di pagine veloci ottimizzate per il motore di ricerca Google, e scoprire l'introduzione di un formato perfetto per i nostri scopi: le web stories.
Si tratta di un nuovo formato di Google che riprende esattamente le storie social e le trasforma in un contenuto web utilizzabile da chiunque.

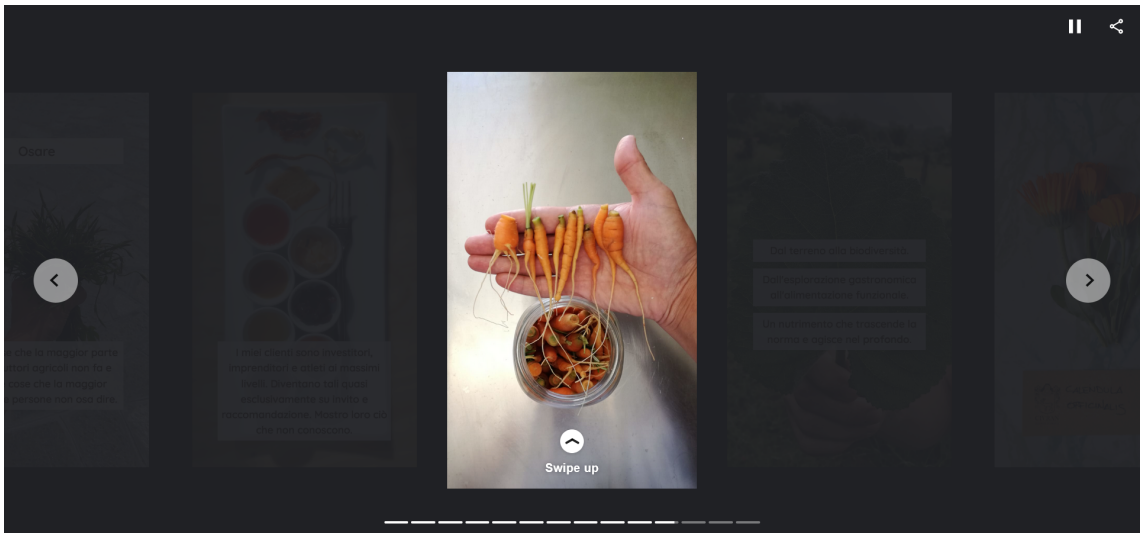

Figura 3.5: Etichetta digitale realizzata con AMP Web Stories (visualizzata da browser desktop)

Effettuare un refactor del codice della nostra etichetta digitale è stato immediato, in quanto l'implementazione di una struttura semplice e ben documentata come quella di AMP ha reso ancora più facile lo sviluppo della pagina web.

Si tratta sempre di una pagina dinamica, ovviamente con una struttura da noi prefissata, che una volta caricato il json salvato tramite l'editor permette la visualizzazione della pagina web in formato AMP. E' stato necessario effettuare un server-side rendering, per offrire al client direttamente il codice html compilato e migliorare l'efficacia dell'analisi effettuata dai motori di ricerca.

Questo formato, che riproduce ugualmente le storie social, ci ha permesso di ottenere evidenti vantaggi: solidità del codice, essendo sviluppato e mantenuto da Google, aumento delle performance da mobile, utilizzando un formato ottimizzato per i questi dispositivi, e possibilità di creare materiale per migliorare il posizionamento delle aziende sul motore di ricerca.
Infatti, i contenuti creati utilizzando questo formato sono "premiati" dai motori di ricerca e verranno meglio posizionati tra i risultati: abbiamo dunque la possibilità di rendere le etichette digitali del prodotto presenti nelle ricerche web, aumentando la visibilità delle aziende.
Inoltre, con la futura integrazione di questo tipo di contenuti all'interno di siti web delle aziende clienti, potremo aumentare il posizionamento dell'intera azienda, andando ad utilizzare il loro dominio per esporre le loro etichette.

3.4 Blockchain e IPFS

Come visto in precedenza, in questa implementazione di Farmchain il ruolo della blockchain è marginale e permette esclusivamente agli utenti di caricare alcuni documenti, relativi al singolo prodotto o alla loro azienda, da rendere poi visibili agli utenti attraverso l'etichetta digitale.

Lo scopo è aumentare la trasparenza dell'azienda nei confronti dei propri clienti, che fornisce così un'ulteriore prova della bontà della proprio attività produttiva attraverso la condivisione di certificati di qualità, iscrizione a particolari registri, etc.
La tecnica che abbiamo utilizzato per memorizzare le informazioni dei file caricati è l'integrazione tra due tecnologie: IPFS e Smart Contract.
Il problema di utilizzare solamente la blockchain per caricare un file e renderlo decentralizzato e accessibile da chiunque è il costo che ogni operazione di store comporterebbe.

Nel caso specifico di Ethereum, la blockchain più diffusa che permette l'utilizzo di smart contract, si può trattare di centinaia di euro per MB, oltre al costo della transazione. Considerando che per il nostro target anche solo questo costo può essere problematico (motivo per cui abbiamo previsto piani tariffari diversi per accedere alla funzionalità blockchain), era necessario trovare una soluzione alternativa.

Una soluzione per creare uno storage decentralizzato è l'utilizzo del protocollo IPFS per memorizzare i file e registrare l'hash restituito all'interno di uno smart contract per rendere immutabile l'assocazione.

Come visto nel secondo capitolo, IPFS è un protocollo P2P con cui salvare i file su un nodo della rete e rendere pubblici i file attraverso la generazione di un hash univoco.

Il servizio che si occupa di dialogare con IPFS e Blockchain, sviluppato inizialmente in NodeJS nell'applicazione a microservizi, è rimasto invariato anche in questa nuova versione della Webapp Laravel.
É presente un endpoint API, servito attraverso la libreria ExpressJS, che viene richiamato dal backend di Laravel nel momento in cui viene effettuato l'upload di un file da memorizzare in blockchain.

IPFS

Abbiamo installato su un server Linux in cloud (un'istanza EC2 su Amazon AWS) un nodo IPFS su cui caricare i file.
All'interno dell'applicazione nodeJS, abbiamo utilizzato il client IPFS ipfs-http-client che permette l'interazione col nodo IPFS e il caricamento dei file, restituendo l'hash (CID) generato dopo l'operazione di add.

Smart Contract

Una volta ottenuto l'hash, viene creato lo smart contract su Ethereum e caricato in blockchain.

Per sviluppare lo smart contract abbiamo utilizzato Truffle, un ambiente di sviluppo per smart contract che permette, dopo la scrittura del codice in Solidity, di generare la versione ABI (application binary interface) dello smart contract, codifica necessaria per utilizzare il codice sulla EVM.
L'hash restituito da IPFS viene memorizzato all'interno dello smart contract.

Viene quindi utilizzato la libreria web3.js, che permette l'interazione in javascript con Ethereum, per creare lo smart contract precedentemente sviluppato, passando al costruttore le informazioni ottenute precedentemente da IPFS.
In questo modo lo smart contract restituirà, attraverso il metodo getFileHash, le informazioni per reperire il file su protocollo IPFS, in cui il file sarà disponibile in maniera decentralizzata.

Come vedremo nel capitolo successivo, questa stessa tecnica verrà utilizzata non solo per memorizzare file ma anche oggetti json contenenti maggiori informazioni raccolte dall'app.

3.5 Prime prove sul campo: azienda Civran

Questa versione della webapp ha portato l'enorme beneficio di rendere più facile la fase di testing, attualmente in corso.

Come già accennato nel primo capitolo, una grande fortuna di questo progetto è stata trovare la collaborazione di un'azienda agricola giovane ed interessata alle evoluzioni del mondo digitale: l'azienda "Civran Azienda Agricola" di Trofarello, e nello specifico il suo titolare, Filippo Civran.
Si tratta di un'azienda agricola che cerca di portare innovazione nel settore, utilizzando

materie prime ricercate e tecniche di trasformazione, tradizionali e non, per sperimentare e creare un'esperienza unica. É diventato presidio Slowfood grazie alla coltivazione del Cavolfiore di Moncalieri.

Oltre al grande contributo lungo tutto il periodo dalla nascita del progetto ad oggi, con Filippo sono ora in corso test ad hoc su questa versione alla prima vera prova sul campo. Analizziamo separatamente i test relativi al funzionamento della web app e le varie etichette digitali create manualmente.

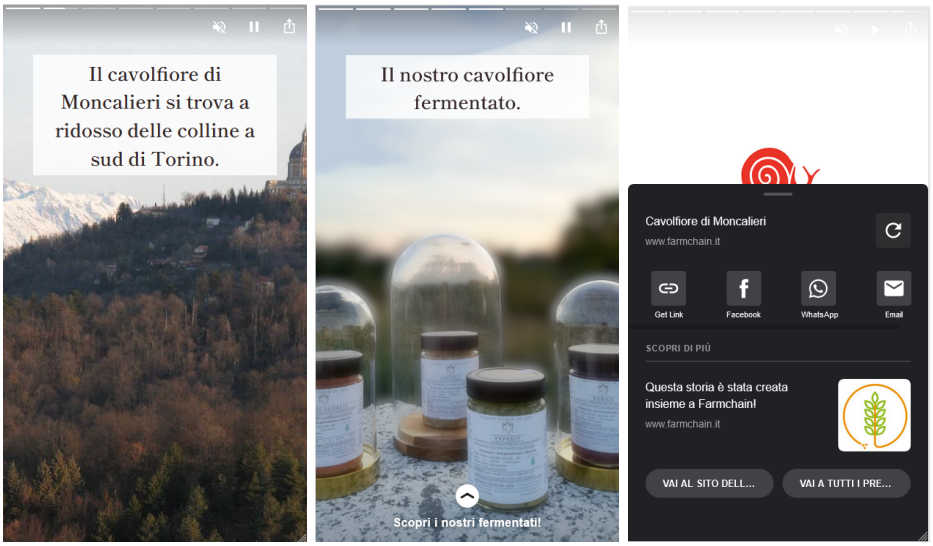

Figura 3.6: Etichetta digitale del cavolfiore fermentato dell'azienda agricola Civran

Test sulla web app

Per quanto riguarda la web app, i test sono iniziati da meno di un mese, da inizio Marzo 2021, e quindi è presto per trarre troppe conclusioni.

Un aspetto che è emerso è che offrire all'utente molte possibilità grafiche per costruire le varie stories rischia di far ottenere risultati finali disomogenei e poco funzionanti esteticamente.
Al momento le possibilità di editing sono ridotte al minimo, essendo la prima versione dell'editor, ma riteniamo utile continuare a svilupparle mantenendo una certa "guida" su come inserire i vari contenuti all'utente finale.

A tal proposito, abbiamo studiato diversi editor esistenti per le web stories di Google AMP, compreso un plugin installabile su WordPress e quindi accessibile più facilmente da chiunque utilizzi il CMS, ma essendo editor molto completi risultano più difficilmente utilizzabili fruttuosamente da un utente che non abbia abbastanza praticità col mondo digitale.

Un altro aspetto è la necessità di arrichire le funzionalità di swipe-up, che permettono di inserire maggiori contenuti non alterando l'aspetto delle storie digitali. Al momento ci siamo limitati ad una versione basica, che permette aree testuali extra o link esterni, ma aggiungeremo un editor anche per questa parte, con la possibilità di aggiungere testi, immagini e contenuti integrabili in AMP, come ad esempio il player youtube per riprodurre video relativi alla storia.

Riscontri sull'etichetta digitale

Parallelamente ai test della web app, che verranno presto estesi ad altre due aziende agricole con cui siamo in contatto, stiamo testando diversi format con cui costruire un'etichetta digitale tipo.
Per fare questi test, stiamo creando manualmente le etichette con i contenuti che ci vengono inoltrati da diverse aziende, in modo da fargli vedere immediatamente le potenzialità del prodotto finito.

Un aspetto di questo tipo di approccio è che, modificando di volta in volta l'etichetta in base ai feedback che stiamo ricevendo, arriveremo a costruire un template finale, a cui ne seguiranno altri, con il quale costruire l'etichetta attraverso l'editor, andando così ad unire i due diversi aspetti di questa fase di sviluppo.
L'etichetta digitale sta riscontrando feedback positivi, e la solidità del codice AMP si conferma grazie ai test che stiamo effettuando su dispositivi mobile molto diversi per dimensione schermo e risoluzione, in cui i contenuti si riarrangiano correttamente.

Capitolo 4

Farmchain 2.0

In questo capitolo verranno brevemente introdotte le modifiche che prossimamente saranno apportate alla piattaforma Farmchain.

4.1 Architettura a microservizi

La prima evoluzione che apporteremo alla versione attualmente in fase di rilascio è la sostituzione della parte sviluppata in Laravel con pagine frontend sviluppate con VueJS e gestione backend affidata ad una infrastruttura a microservizi sviluppata in NodeJS.

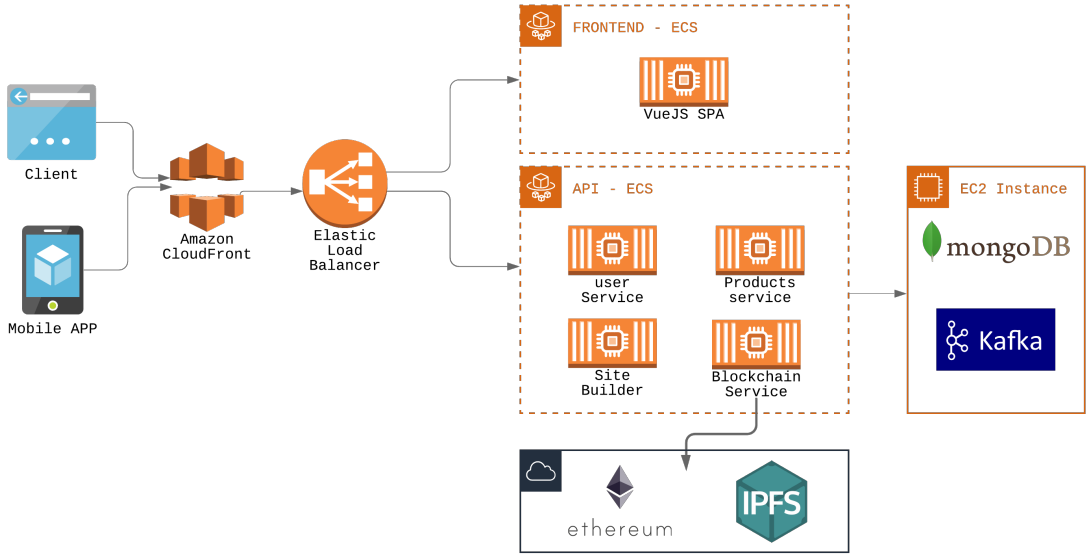

Figura 4.1: Architettura a microservizi

Microservizi

L'architettura infrastrutturale a microservizi si contrappone ad applicazioni "monolitiche" per l'utilizzo di diverse componenti più piccole e dedicate a specifiche funzioni separate tra loro. In questo modo, nel caso servano evoluzioni su un singolo servizio o sia necessario aumentarne la potenza di calcolo per un aumento specifico di carico, è possibile operare sulla singola applicazione senza intaccare la logica degli altri servizi.

Un esempio di applicazione monolitica è quella che abbiamo attualmente sviluppato in Laravel: per quanto possa andare bene in una fase inziale, con condizioni di scarso traffico e poche funzionalità, nel tempo ne renderebbe più complesso l'evoluzione, avendo le diverse componenti dell'applicazione strettamente legate tra di loro, oltre ad un'ovvia interdipendenza col framework Laravel e col linguaggio PHP.

All'interno di una applicazione in cui ogni funzionalità è indipendente dalle altre, è possibile introdurre facilmente nuove funzionalità, nuove strutture dati e, volendo, riscriverla completamente in un linguaggio che offra librerie e prestazioni più adeguate.

Traducendo le attuali funzionalità in applicazioni separate, emergono quattro servizi distinti:

- Users: si occupa dell'autenticazione e della gestione delle informazioni utente.
- Products: servizio utilizzato per la gestione dei prodotti e delle etichette digitali create dagli utenti.
- Blockchain: già presente in questa versione, si occupa di interfacciarsi con la blockchain Ethereum e con IPFS alla creazione di un prodotto e per permettere la lettura dei dati da API.
- Site Builder: servizio dedicato alla creazione e gestione di siti internet creati attraverso la nostra piattaforma.

Gestione dei dati

Rispetto a quanto sviluppato precedentemente, in questa implementazione di microservizi dovremo valutare se utilizzare istanze database dedicate, per rendere completamente indipendenti i servizi, oppure se utilizzare un database unico, per limitare la ridondanza di dati e rendere la piattaforma più facilmente mantenibile.

Nella concezione di servizi completamente separati tra di loro, abbiamo inizialmente progettato un'architettura in cui ogni servizio ha un proprio database dedicato e l'aggiornamento di informazioni condivise tra servizi diversi avviene tramite eventi e replica dei dati.

L'approccio "Event Driven" consiste nell'utilizzare degli eventi per far comunicare ai servizi eventuali modifiche ai dati o altre operazioni. Questi eventi, gestiti tramite il gestore

di stream di dati Apache Kafka, utilizzano un pattern publish-subscribe in cui, oltre al tipo di evento generato, viene passato il vero e proprio dato coinvolto.

Di seguito un esempio di comunicazione di eventi:

1. Sulla pagina frontend viene effettuato l'upload di un file da caricare in blockchain: viene effettuata una chiamata verso l'API, successivamente dirottata sul servizio "Products".

2. Il servizio "products" si occupa di salvare i dati sul proprio database, per associarlo al relativo prodotto, e pubblica un evento "create.bc.document"

3. Il servizio "blockchain", iscritto alla coda di eventi del servizio "products", viene triggerato, recupera il file oggetto dell'evento dal server e procede al salvataggio del file in IPFS e alla creazione dello Smart Contract su blockchain.

4. Una volta concluse queste operazioni, emette un evento "create.smartcontract", ricevuto dal servizio "products" (a sua volta iscritto agli eventi di "blockchain") e salva l'informazione dello smart contract all'interno del database nel relativo prodotto.

In questo modo i dati sono scollegati tra i diversi servizi ed è possibile utilizzare database differenti e strutture dati diverse.
Questa soluzione viene implementata utilizzando il database NoSQL Mongodb, che con la sua gestione a documenti risulta più flessibile, caratteristica molto utile soprattutto per quanto riguarda la gestione dei prodotti che potrebbe evolvere nel tempo.

4.2 Mobile APP

Un'evoluzione obbligata per la piattaforma sarà l'introduzione dell'applicazione mobile con cui l'utente potrà gestire tutte le sue funzionalità all'interno dell'ecosistema Farmchain.

Come accennato in diversi capitoli, una primissima versione di app mobile di Farmchain è già stata sviluppata, in quanto necessaria per poter gestire l'acquisizione diretta dei media, in modo da avere informazioni più dettagliate.
Per svilupparla abbiamo utilizzato il framework javascript React Native, che permette attraverso un codice unico di sviluppare un'applicazione funzionante sia su sistemi Apple iOS che Android, appoggiandosi su elementi nativi dei due sistemi.

L'applicazione andrà completamente riscritta, sia per le modifiche effettuate al progetto sia per le nuove versioni di React Native nate nel frattempo (essendo una libreria relativamente giovane ci siamo trovati con aggiornamenti molto frequenti durante il primo sviluppo).

Le peculiarità principale dell'app mobile è la possibilità di acquisire foto e video relativi ai diversi passaggi della filiera produttiva ed inserirli direttamente in blockhain, con le modalità descritte nel capitolo precedente e riprese nella prossima sezione, in modo da creare una sorta

di "cronistoria trasparente" del prodotto, per esempio dalla semina fino al confezionamento. In generale l'app mobile permetterà una migliore gestione dei contenuti, dalla creazione delle storie alla gestione del proprio sito integrato, funzionalità descritta a fine capitolo.

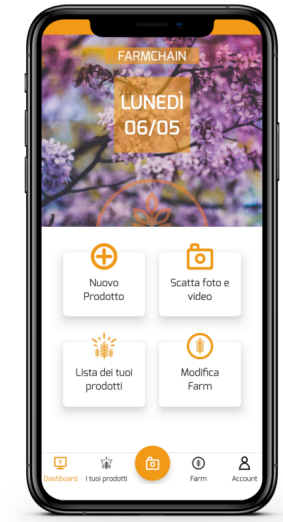

Figura 4.2: Primo prototipo della mobile app

Tuttavia, abbiamo deciso di dare priorità alla webapp ora introdotta in modo da rendere fruibili tutte le funzioni direttamente tramite web, in modo da facilitare l'accesso al servizio alle aziende interessate.
Strutturando il backend sotto forma di API in NodeJS e il frontend separato sviluppato in VueJS ,le due strade (webapp e app mobile) potranno procedere parallelamente, slegando tra di loro i diversi componenti.

4.3 Blockchain e tracciamento della filiera produttiva

Per quanto riguarda l'utilizzo della blockchain, la principale evoluzione prevista è legata alla possibilità di documentare la filiera produttiva attraverso i media acquisiti tramite app.

Attualmente la blockchain, in questa prima versione in fase di rilascio, ha un utilizzo molto limitato, che consiste nel permettere all'utente di effettuare l'upload di documentazione legata al prodotto (come per esempio certificazioni agroalimentari) che successivamente viene inserita in un nodo IPFS e il cui hash viene memorizzato in uno Smart contract su blockchain pubblica Ethereum.
Questo rende il file distribuito e immutabile, e aumenta il livello di trasparenza del produttore agricolo percepita dall'utente finale, ma offre un utilizzo molto limitato delle possibilità della blockchain.

Al momento esistono molte applicazioni differenti di blockchain legate alla filiera agroalimentare. Principalmente si tratta di gestione dei lotti tramite smart contract, con cui far interagire i diversi attori della filiera attraverso metodi e livello di autorizzazione differenti. Questo porta grandi benefici all'interno di filiere lunghe e con raggio operativo elevato, pur con i limiti che verranno descritti nel capitolo conclusivo.

Ma come già scritto precedentemente, in un contesto di piccoli produttori, nella maggior parte dei casi venditori diretti, la blockchain ha un ruolo molto più marginale rispetto ad altri fattori, come la creazione di strumenti di marketing digitale, in quanto vi sono pochi attori coinvolti e zone geografiche interessate ristrette.

Abbiamo quindi pensato ad una soluzione differente, che vada a integrare l'aspetto documentale della blockchain insieme alla centralità dei media nel racconto della filiera all'interno del progetto Farmchain.
L'evoluzione prevista è quindi quella di inserire i media dell'utente, acquisiti utilizzando una nostra mobile app, su protocollo IPFS e acquisire contemporaneamente il timestamp e la geolocalizzazione dell'evento di acquisizione.

Queste tre informazioni, timestamp, geotag e hash IPFS del media, verranno inserire in un unico file json, anch'esso inserito in IPS e il cui HASH verrà a sua volta inserito nello smart contract in blockchain.
In questo modo, oltre a limitare l'utilizzo oneroso di memoria in blockchain, avremo la possibilità di evolvere facilmente la struttura dati nel tempo, lasciando inalterate le modalità di salvataggio in blockchain.

4.4 Website builder e altre possibili integrazioni future

Un aspetto che sarà oggetto di futuri sviluppi sarà la creazione, attorno alle storie create dai produttori, di un vero e proprio ecosistema digitale da integrare all'interno della piattaforma Farmchain.
Il nostro obiettivo è quello di sfruttare i contenuti inseriti dagli utenti per renderli disponibili su diversi canali digitali automaticamente, senza operazioni aggiuntive.

Per esempio, stiamo progettando la configurazione, tramite webapp, di un sito internet per l'utente in cui inserire, oltre alle informazioni relative all'azienda, i prodotti che man mano vengono caricati su Farmchain. L'idea è creare uno o più template strutturati di pagine web che l'utente potrà scegliere come proprio sito vetrina e in questa fase iniziale di configurazione caricare il template sul dominio dell'azienda, dando supporto completo o parziale per effettuare queste operazioni di setup.

Una volta caricato online, i diversi contenuti caricati dall'azienda (dati, media, prodotti, etc..) verranno caricati dinamicamente dalla pagina attraverso delle chiamate verso le nostre API.
In questo modo, il stio web sarà aggiornabile in qualunque momento tramite la nostra webapp, e l'azienda avrà a disposizione un sito costantemente aggiornato con le varie storie dei

prodotti inserite in Farmchain.
Oltre a creare una presenza digitale d'impatto senza conoscenze tecniche o grosse configurazioni, l'azienda potrà sfruttare le pagine in AMP per ottenere un buon posizionamento nei risultati di ricerca Google.

Ovviamente sarà possibile prevedere soluzione più complesse nel tempo, anche con interventi customizzati in base alle esigenze, ma la priorità sarà sviluppare una piattaforma dinamica che riesca facilmente ad offrire siti veloci e aggiornati costantemente agli utenti.

Allo stesso modo, vorremmo integrare le storie all'interno di un portale di E-commerce in cui poter far vendere i prodotti ai nostri utenti, che gestirebbero eventuali ordini sempre attraverso la piattaforma.
Oltre ad offrire un nuovo canale di vendita, potremmo sviluppare il primo e-commerce in cui gli utenti possano vedere la storia di ogni singolo prodotto e relativo produttore prima di acquistare, creando un'esperienza immersiva di acquisto online.

Capitolo 5

Conclusioni

5.1 Risultati ottenuti

La creazione di questo progetto da zero, insieme ai soci della Startup, mi ha permesso di crescere molto sotto diversi aspetti, sia imprenditoriali che tecnici.

Nel progettare questa piattaforma, ci siamo posti inizialmente degli obiettivi molto complessi da raggiungere, che col tempo abbiamo dovuto rivedere.
E forse il più grande insegnamento è stato proprio questo: mettere da parte virtuosismi tecnici, dettati soprattutto da voglia di sperimentare e ambizione, a favore di soluzioni pratiche per raggiungere il risultato voluto, che sempre deve mettere al centro l'utilizzo da parte dell'utente.

Ho avuto modo di esplorare la tecnologia blockchain e le sue applicazioni in questo tipo di problematiche, ovvero al tracciamento di asset fisici nel settore agro-alimentare.
Sicuramente ho approfondito la parte più "applicativa" di questo mondo, concentrandomi sullo sviluppo di smart contract rispetto a questioni più "infrastrutturali", ma ad ogni modo l'interesse per questo tipo di soluzione porterà sicuramente nuove evoluzioni per questo progetto.

Nonostante sia stata accantonata temporaneamente, lo sviluppo di un'architettura a microservizi mi ha permesso di esplorare tantissimi aspetti dello sviluppo di un'applicazione web in modo molto più approfondito rispetto a quanto fatto in precendenza attraverso l'utilizzo di framework più strutturati.
Innanzitutto l'adozione di NodeJs lato backend, tecnologia nuova per me, mi ha permesso di sviluppare in libertà, potendo sperimentare diversi pattern in modo da minimizzare la quantità di codice necessario e rendere i servizi il più leggero possibile. Sicuramente nel riprendere questa soluzione apporteremo delle modifiche, ma è un linguaggio che ho apprezzato per la semplicità e la facilità di implementazione.

La progettazione di tale soluzione richiede molto tempo di sviluppo preventivo in cui focalizzare i diversi servizi in base alle loro funzionalità e progettare in maniera solida la

struttura dati per ognuno di loro, andando poi successivamente a costruire la rete di eventi con cui gestire la sincronizzazione dei database.

Per quanto riguarda la parte architetturale, ho avuto modo di approfondire molto AWS e i vari servizi offerti, che coprono qualsiasi necessità infrastrutturale di un'applicazione.
Inizialmente, nello studiare la piattaforma cloud, abbiamo valutato l'utilizzo di alcuni servizi offerti da AWS da poter utilizzare nella nostra applicazione senza doverli sviluppare e gestire, come l'autenticazione e le notifiche.

Ma nei test effettuati è emerso quanto dispendiosa sia la piattaforma, e legarci fin dalla fase di sviluppo a doppio nodo con alcuni loro servizi ci avrebbe comportato grosse difficoltà in caso di migrazione verso un altro fornitore di hosting.
Abbiamo quindi optato per sviluppare tutto in maniera indipendete dalla piattaforma Cloud, andando però a sfruttare, in fase di implementazione, alcune loro soluzioni in grado di ottimizzare le performance e i costi, come ECS per far girare i container Docker e la CDN di CloudFront per il caching web.

Per quanto riguarda lo sviluppo frontend, sia lato webapp che mobile app, il mio contributo è stato minore ma il confronto costante con gli utenti finali, veri e propri beta tester con cui realizzare un'interfaccia da zero, mi ha permesso di sviluppare una maggior sensibilità ad aspetti di usabilità e accessibilità, quanto mai indispensabili in un settore con un tasso di digitalizzazione ancora relativamente basso.

5.2 I limiti attuali della blockchain

L'utilizzo della blockchain è stato uno dei nodi cardine nella primissima fase di progettazione di Farmchain: lo scopo era quello di portare questa tecnologia ai piccoli produttori agricoli, in modo da valorizzarne l'attività produttiva.

Nel corso del tempo ci siamo però scontrati con diversi limiti, sia legati alla blockchain sia legati alla reale sua utilità all'interno del nostro progetto.
Per quanto riguarda l'applicazione della blockchain nel settore agroalimentare, uno dei limiti maggiori che abbiamo rilevato è l'associazione di token digitali con asset fisici specifici.

Nel nostro caso, per quanto la soluzione da noi scelta di notarizzare i media acquisiti con relativi timestamp e informazioni di geolocalizzazione fosse abbastanza solida nel certificare l'autenticità dell'informazione, l'associazione finale del qr code con uno specifico prodotto non presenta alcuna reale garanzia: i QR code generati nel raccontare un certo lotto li posso associare a lotti di tutt'altra provenienza.

Abbiamo riscontrato questo tipo di carenza in tutte le soluzioni al momento presenti sul mercato, e solo una compartecipazione in questo processo di documentazione con enti garanti potrà renderlo realmente funzionante al 100%.
Tuttavia, crediamo che in questo caso la blockchain perda uno dei suoi elementi caratteristici,

ovvero la decentralizzazione, in quanto si renderebbe necessario un ente terzo per garantirne l'efficacia.

Ovviamente in grandi aziende agricole la blockchain può assumere un ruolo più importante, soprattutto nell'interazione tra attori diversi della filiera. Nel caso però di filiera corta assume maggiormente il ruolo di semplice strumento di marketing, per i limiti attuali evidenziati precedentemente.
Certo è che con un'utilità così limitata, inserirla a forza nel progetto come "chiave" dell'applicazione non ci è sembrato né utile né produttivo. Abbiamo quindi posto l'attenzione sui contenuti e sulla creazione di un prodotto, l'etichetta digitale, realmente utile e calzante per il nostro target di clientela.

Un altro grosso limite riscontrato è il costo che una transazione può avere su una blockchain, e volendo interagire con uno smart contract ad ogni acquisizione di media, è un aspetto molto rilevante per la nostra offerta commerciale.
In questo senso, Ethereum, la più grande blockchain che permette l'esecuzione di Smart Contract, ha costi sempre più elevati in base al suo valore di mercato nell'ambito cryptovalute, dovuto al costo in gas necessario all'esecuzione dello smart contract. Questo tipo di oscillazioni di prezzo rende necessario, per il nostro tipo di implementazione, valutare altre blockchain con costi di molto inferiori e meno legati al mercato.

In sintesi, abbiamo per il momento scelto di mettere in secondo piano questa tecnologia nella nostra soluzione, mettendo in primo piano le reali esigenze di digitalizzazione per i piccoli produttori agricoli.

5.3 Implementazioni future

Come descritto all'interno del quarto capitolo, le implementazioni future riguarderanno diverse componenti del progetto.

In questa prima fase di rilascio e testing, le nostre attenzioni saranno dedicate allo sviluppo dell'editor di Web Stories, in parallello con il frontend AMP dell'etichetta digitale. Stiamo raccogliendo feedback, sia interni sia dai produttori agricoli coinvolti, per cercare di aggiungere funzionalità nella fase di creazione della web stories, come l'aggiunta di contenuti non previsti all'interno delle stories e la possibilità di caricare template.
Sarà inoltre interessante approfondire l'integrazione con l'ecosistema Google per cercare di valorizzare il più possibile i contenuti inseriti dai produttori. Google My Business, che permette la gestione per le aziende della propria vetrina sul motore di ricerca, potrebbe fornirci ulteriori strumenti per posizionare meglio i nostri clienti.

In questo senso, inizieremo anche lo sviluppo del Website Builder, la funzione con cui creare siti web per i produttori integrati con le loro etichette digitali. Partiremo con un semplice template HTML prototipale, in cui inserire le web stories e i dati dell'azineda come

componente dinamica con cui costruire il sito. Anche in questo caso, cercheremo di utilizzare le tecnologie AMP per rendere il sito più veloce e performante dal punto di vista SEO.

Per quanto riguarda il backend della webapp, come già anticipato, provvederemo, prima del lancio commerciale di Farmchain, a trasformare l'architettura server, ora sviluppata in PHP Laravel, creando una serie di servizi API in NodeJS, andando così a rendere più performante e mantenibile l'infrastruttura.
Si tratta principalmente di creare i diversi servizi (gestione utenti, autenticazione ed etichette) in NodeJS per sostituire la parte di Controller e Model di Laravel, mentre le varie viste utilizzate saranno riadattate separatamente, utilizzando VueJS. In questa direzione, avendo già creato l'editor come un component VueJS, sarà più semplice andare a realizzare una Spa.

Per una fase successiva, abbiamo intenzione di riadattare la mobile app precedentemente realizzata, andando ad inserirvi le funzioni ora presenti nell'editor di web stories.
In questo modo potremo ripensare anche l'utilizzo della blockchain, andando ad implementare la notarizzazione dei media acquisiti nelle diverse fasi produttive tramite app.

Per rendere sostenibile questo utilizzo, dovremo probabilmente andare a sostituire la blockchain Ethereum, economicamente troppo esosa per una implementazione del genere, con un'altra blockchain più adatta ad uno scopo applicativo di questo tipo e con costi meno soggetti alle movimentazioni di mercato della relativa cryptovaluta.

5.4 Considerazioni finali

Come spero sia emerso all'interno di questa tesi, l'entusiamo che ha portato alla nascita di questo progetto non è mai mutato.

L'esperienza universitaria mi ha permesso di far evolvere Farmchain insieme alle mie competenze, adottando di volta in volta soluzioni differenti.
Credo sia fondamentale al termine del percorso di studi poter sperimentare quanto appreso il più possibile, senza limitarsi all'applicazione dei concetti acquisiti su lavori preconfezionati o compiti assegnati.

La libertà concessa da un progetto personale permette di crescere sotto diversi punti di vista, e la possibilità di sbagliare, avendo la capacità di trarne insegnamento, crea le più grandi occasioni di crescita.
Avendo la fortuna di lavorare autonomamente e contemporaneamente portare avanti questo progetto, spero di poter continuare a espandere le mie competenze e sperimentare, senza perdere di vista i reali obiettivi che un'applicazione deve raggiungere.

Elenco delle figure

Dichiarazione di originalità

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.