Agrichain

Tecnologie Blockchain per il controllo della filiera di prodotti agro-alimentari

CyberSecurity
Research Lab
UniPG

Scopo del progetto

Scopo del progetto

Salvaguardare la qualità dei prodotti alimentari locali della regione Umbria verso i cittadini, proponendo una una soluzione innovativa, elaborata con competenze multidisciplinari presenti nell'Università di Perugia grazie alla collaborazione tra i dipartimenti di Matematica e Informatica, Scienze Agrarie Alimentari ed Ambientali e Ingegneria.

Scopo del progetto

Agrichain intende offrire uno strumento di prevenzione del fenomeno delle frodi commerciali e di garanzia della supply chain, attraverso l'utilizzo di tecnologie informatiche basate sulla blockchain, automatizzando e certificando in modo sicuro ed inalterabile le materie prime, il processo produttivo e distributivo.

Tecnologia adottata

Tecnologia adottata

L’infrastruttura dell’applicazione sarà basata sulla tecnologia **blockchain**:

- “**Catena**” di informazioni suddivisa in **blocchi**.
- **Sicura** grazie alla crittografia.
- Transazioni accessibili in modo efficiente, verificabile e **permanente**.

Tecnologia adottata

Alla luce di queste sue proprietà, la tecnologia blockchain funge da infrastruttura per lo sviluppo di applicazioni che richiedono decentralizzazione, trasparenza, sicurezza e immutabilità.

In particolare, per rendere Agrichain ancora più sicuro verso possibili nodi malevoli della rete, esiste un progetto, supportato da *The Linux Foundation*, chiamato **Hyperledger**, il quale permette di sviluppare una blockchain di tipo *permissioned.*

Hyperledger

Hyperledger

Una blockchain **permissioned** permette di aggiungere un livello aggiuntivo di sicurezza all'intero sistema, questo perché, operazioni come accesso alla rete ed esecuzione di determinate operazioni, possono essere svolte unicamente da soggetti adeguatamente autorizzati, pur rimanendo comunque trasparente al pubblico.

Per questi motivi, questo tipo di blockchain è popolare tra progetti di tipo industriale.

HYPERLEDGER PROJECT

Hyperledger

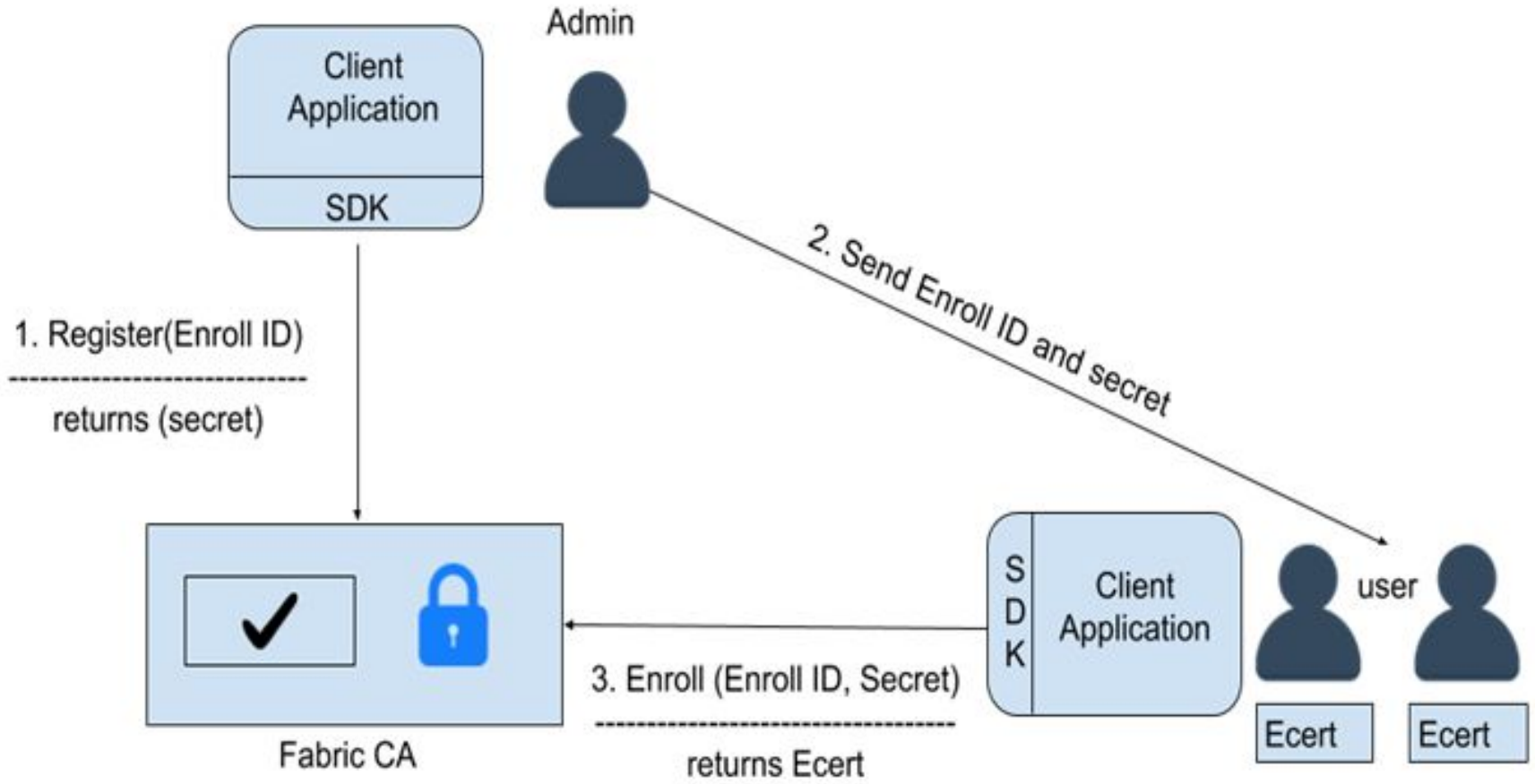
Un framework che permette uno sviluppo di una blockchain adatta al caso è **Hyperledger Fabric**. Fra le sue caratteristiche sono da menzionare:

- Architettura modulare.
- Divisione di ruoli fra i nodi della rete.
- Supporto di **Smart Contract** (*chaincode*) in diversi linguaggi di programmazione.

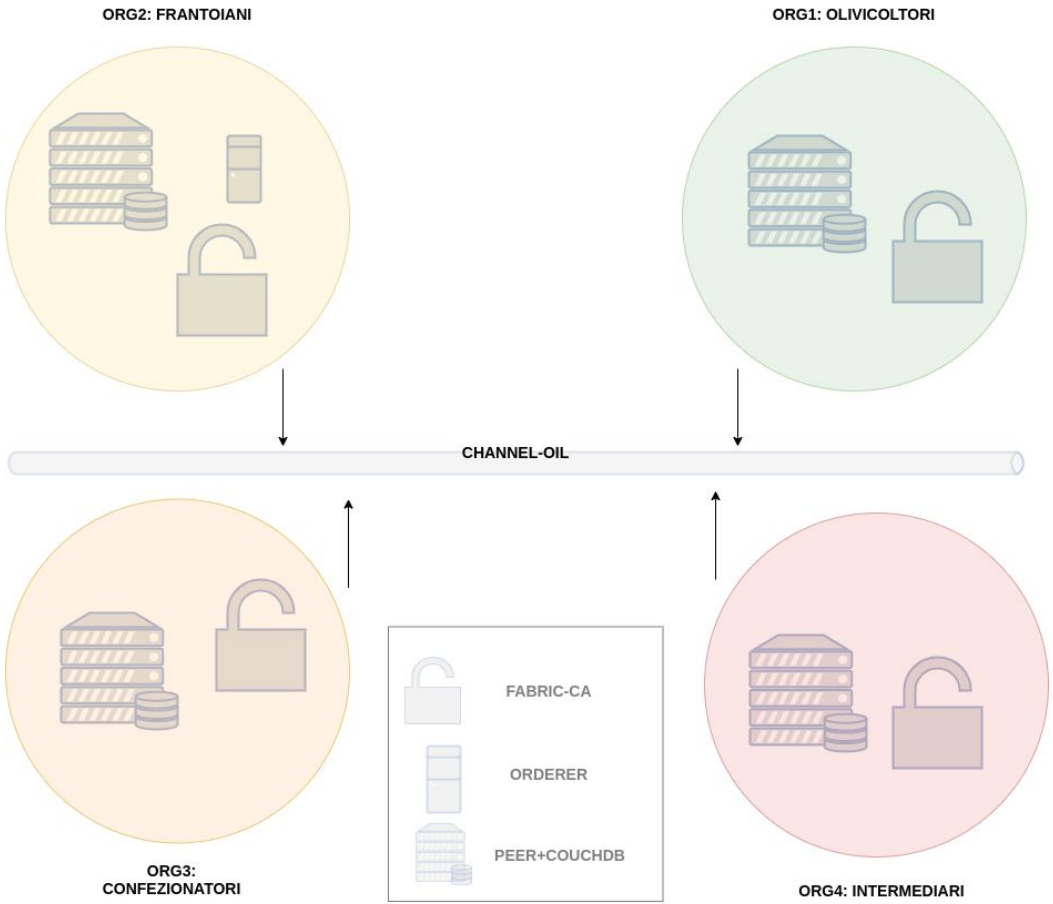
Hyperledger

In Hyperledger Fabric abbiamo più nodi / ruoli:

- **Client**: può essere un'applicazione o, in generale, un'interfaccia per inviare dati sulla rete.
- **Peer**: il "nodo" della rete.
  - *Endorser*: simula le transazioni prima di andarle a convalidare.
  - *Committer* si occupa della validazione.
- **Orderer**: un servizio che decide come le transazioni vengono salvate nel ledger.


Admin
Client Application
SDK
1. Register(Enroll ID)
returns (secret)
2. Send Enroll ID and secret
Fabric CA
3. Enroll (Enroll ID, Secret)
returns Ecert
S D K
Client Application
user
Ecert
Ecert

Esempio di rete con Hyperledger Fabric



Hyperledger

Hyperledger Convector è un tool con il quale è possibile sviluppare dei prototipi e, in generale proof of concept per blockchain di tipo permissioned. Convector possiede i seguenti componenti:

- *Hurley*: programma per un rapido set-up per progetti con Hyperledger Fabric.
- *Convector CLI*: interfaccia a riga di comando per generare file e configurazioni.
- *Convector REST Server*: generatore di API RESTful, tramite le quali è possibile esporre i propri Smart Contract al mondo esterno.

Struttura del progetto:

Chaincode con Convector

Chaincode con Convector

Hyperledger Convector permette di astrarre la supply chain del mondo reale tramite un pattern di *Software Engineering* collaudato, conosciuto come ***Model/Controller***.

- **Model**: per descrivere le <u>entità</u> (asset, organizzazioni, ecc) che interagiranno nella filiera produttiva.
- **Controller**: per definire le <u>*regole*</u> e stabilire le azioni possibili che queste dovranno seguire.

Chaincode con Convector

Modellare lo scenario della filiera tramite due differenti model (classi):

- **Participant**: indica il membro del consorzio:
  - Olivicoltore
  - Frantoiano
  - Confezionatore
  - Intermediari (altri)
- **Oil**: si tratta dell'oggetto olio vero e proprio, che nasce come oliva fino a trasformarsi in prodotto finito

Chaincode con Convector

Ogni model possidede diverse funzionalità, espresse tramite il relativo controller:

Participant

- register: per aggiungere un nuovo membro.
- get: ottenere informazioni su un certo membro.
- getAll: genera la lista di tutti i partecipanti.

Oil

- create: per creare una nuova istanza dell'oggetto olio.
- harvest: per inserire i dati relativi alla raccolta.
- extract: funzione invocata dal frantoiano.
- bottling: per inserire i dati del confezionamento.
- transport: per le informazioni sul trasporto dei prodotti.

Chaincode con Convector

Riconoscimento dell'identità del participant all'interno del chaincode:

1) L'identità viene registrata nel ledger (operazione a basso livello).
2) Si registra il participant via chaincode (alto livello).
3) Viene effettuato il binding fra il certificato nel ledger e la nuova istanza di participant.

Chaincode con Convector

La lettura e scrittura dei dati avviene tramite le librerie di Convector, in particolare i nostri model e controller vanno a estendere le classi:

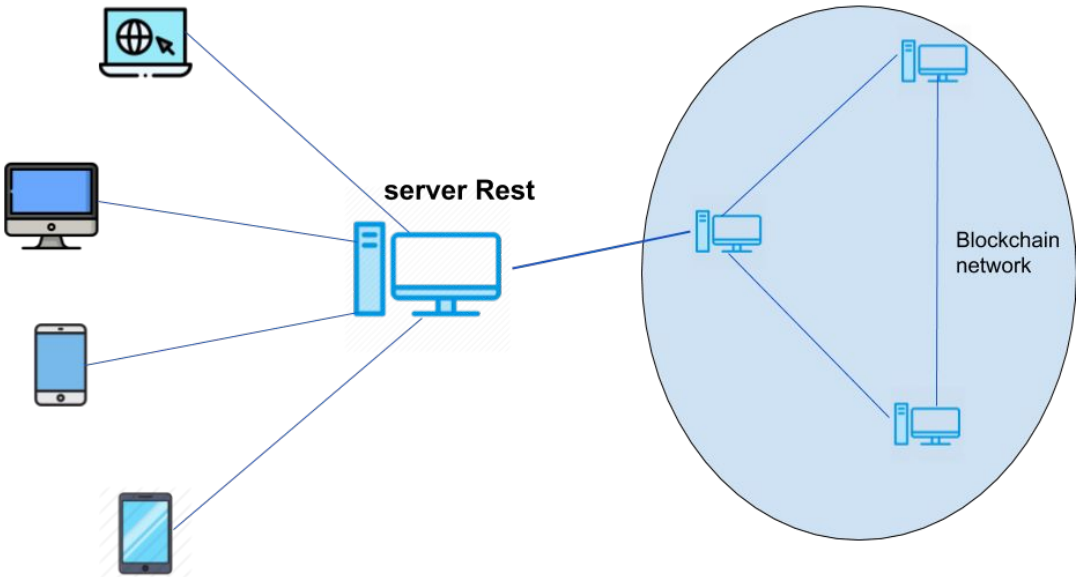
- **ConvectorModel**
  - getOne(id).then(onModelFound);
  - getAll('my.models.test').then(onModelsFound);
  - query({ amount: { $lte: 5 } }).then(onModelsFound);
  - save()
- **ConvectorController**
  - this.sender
  - initControllers()

Struttura del progetto:

Server-REST

Server-REST

Una volta realizzato il chaincode e dopo averne testato il funzionamento via test, lo step successivo è realizzare una “interfaccia” che permetta di richiamare le funzioni del chaincode tramite API REST/JSON.



Server-REST

Nest (NestJS) è un framework moderno dedicato allo sviluppo di applicazioni lato server per Node.js.

- Scelta ideale per lavorare con Convector perché supporta il linguaggio Typescript.
- Basato sul server HTTP Express (come il tool conv-rest-api)

Server-REST

Nest è molto utile per costruire delle api sicure, in quanto l'autenticazione è gestita da un sistema di passport e token (JWT).

Ad alto livello, questi sono i passi da fare per raggiungere l'autenticazione:

1) Verifica delle credenziali (username/password).
2) Rilascio di un token per la sessione dell'utente.
3) Questo token (JWT) dovrà essere incluso nelle successive richieste (come bearer nell'header) per poter accedere alle api "protette".

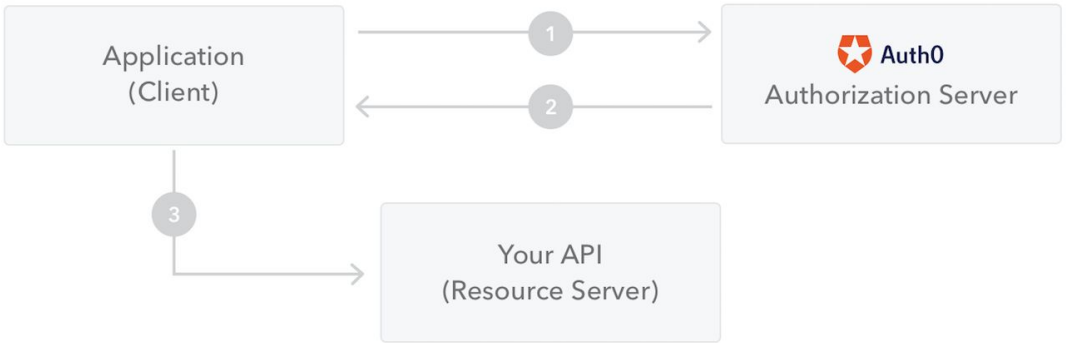
Server-REST

Passport possiede una caratteristica, chiamata strategy: un file di configurazione che va a indicare appunto il tipo di strategia dedicata all'autenticazione degli utenti.

- Un set di opzioni, ovvero quale tipo di strategia (es local, JWT, ecc).
- Un "callback di verifica", cioè controllare che un utente esista, dove andare a reperire le credenziali e tutte le operazioni legate alla convalida dell'identità.
- Il passport è arricchito dalle Guard.

Server-REST

Il JSON Web Token (JWT) è un Internet Standard (RFC 7519), definisce un modo compatto per trasmettere in modo sicuro delle informazioni tra soggetti diversi in forma di oggetto JSON. Queste informazioni possono essere verificate e trusted, in quando sono firmate digitalmente.



Server-REST

Swagger definisce un'interfaccia standard e indipendente per le API RESTful, che consente agli utenti di vedere e comprendere il funzionamento dell'applicazione senza accedere al codice sorgente.

L'interfaccia utente di Swagger è uno strumento open source che genera una pagina Web che documenta le API generate dalle specifiche fornite

Swagger
Supported by SMARTBEAR

Oil ChainCode 1.0

[Base URL: /]

Convector Oil ChainCode API

Schemes
HTTPS

Authorize

auth

POST /api/login Login user

GET /api/me Get user profile

participant

GET /api/participant Get all Participants

POST /api/participant Register Participant

GET /api/participant/{id} Get Participant

Responses Response content type application/json

Curl

```
curl -X POST "https://161.35.200.154:3444/api/login" -H "accept: application/json" -H "Content-Type: application/json" -d "{ \"username\": \"oliv1\", \"password\": \"rossi123\"}"
```

Request URL

```
https://161.35.200.154:3444/api/login
```

Server response

Code	Details
201	**Response body**

```
{
  "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJvbGl2MSIsInVzZXJuYW1lIjoicm9zc2kiLCJpYXQiOjE1OTE5Nzk3MDYsImV4cCI6MTU5MTk4MDYwNn0.gxOz9uDq6Yt3xCkt5DAZ1On4d6bgWSThJ74D
}
```

Download

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 190
content-type: application/json; charset=utf-8
date: Fri12 Jun 2020 16:35:06 GMT
etag: W/"be-TGd52SkGy3Bn9fNrBF8Vn4YVqng"
x-powered-by: Express
```

Struttura del progetto:

React Front-end

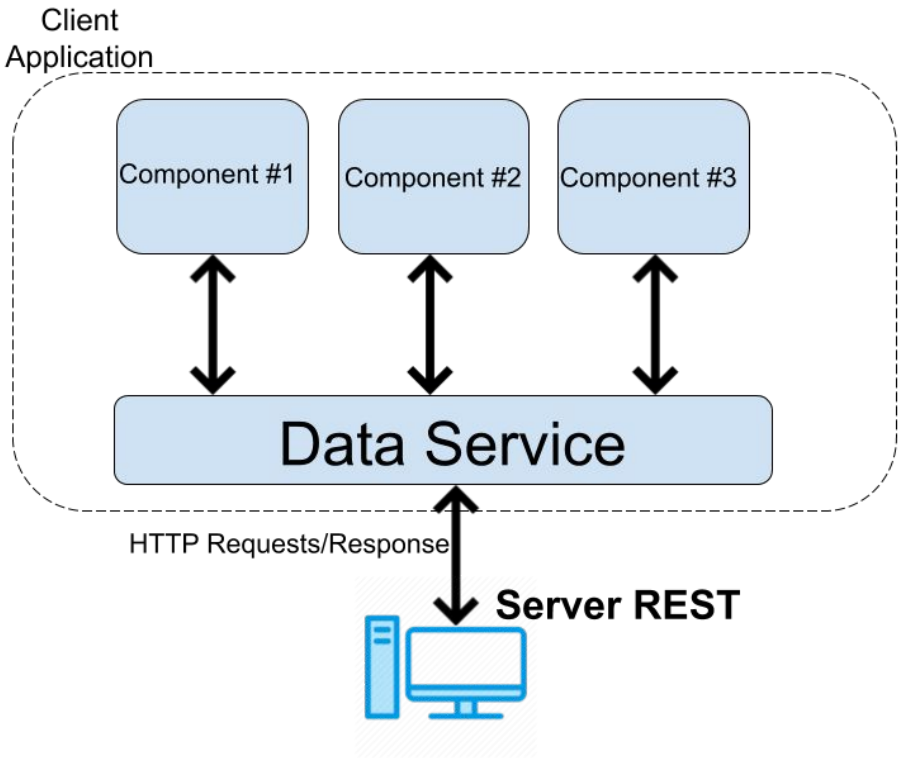
React Front-end

La scelta per la costruzione di un’interfaccia utente in forma di web-app è ricaduta su React:

- Sito web dinamico
- Caricamenti rapidi
- Supporto al linguaggio Typescript
- Grande community e vasta documentazione

React Front-end

Il progetto è strutturato in componenti, ognuna delle quali va ad interfacciarsi con il server-rest, il quale a sua volta richiama le funzioni dal chaincode.



Login User Logout

WELCOME TO THE LOGIN PAGE OF AGRICHAIN

Username

Password

Login User

Login User Logout

Welcome to the Profile page

ID	Name	Username	Email	Location	Roles
oliv1	Azienda agricola Rossi	rossi	aziendarossi@mail.com	Perugia	OLIV

Show all participants

Show oil records

Create oil record

Add oil attribute

Logout

Crea un nuovo oggetto Olio

OilID

Location

Acid

Create

Add a new attribute to the Oil Chain

OilID

ID

CertifierID

Content

Harvest

Users List

Search

Search

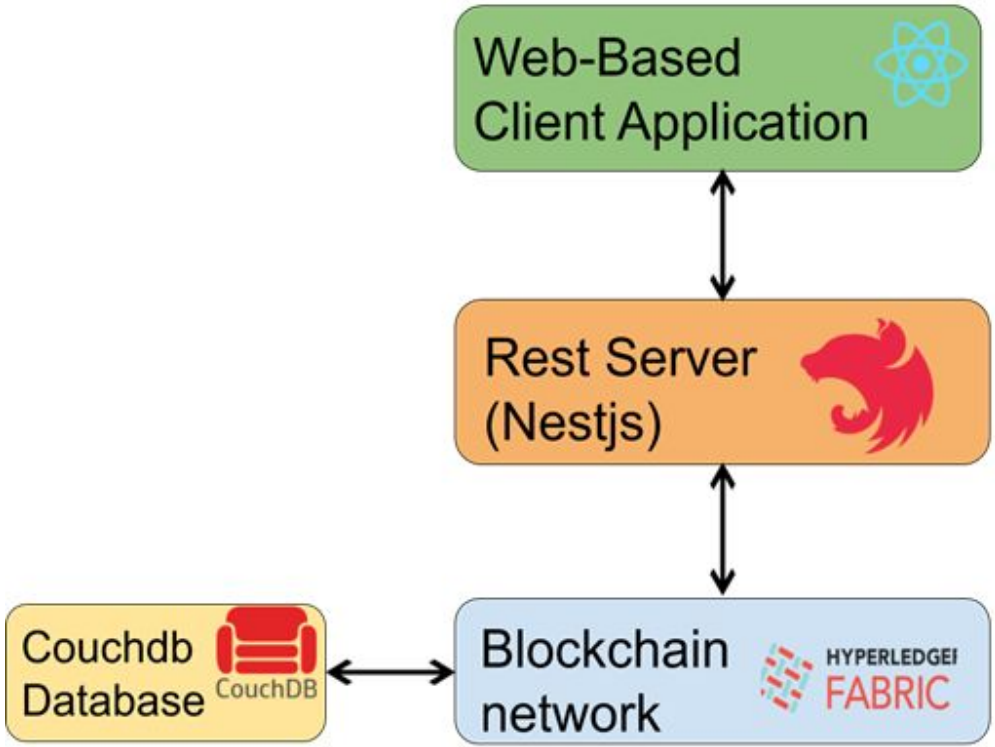
ID	Name	Username	Email	Location	Roles
conf1	Bottiglieria Verdi	verdi	aziendaverdi@mail.com	Magione	CONF
frant1	Olio Anima	cuore	olioanima@mail.com	Ponte S. Giovanni	FRANT
oliv1	Azienda agricola Rossi	rossi	aziendarossi@mail.com	Perugia	OLIV
trasp1	OPS Trasporti	opsss	opstrasporti@mail.com	Gubbio	TRASP

Riepilogo

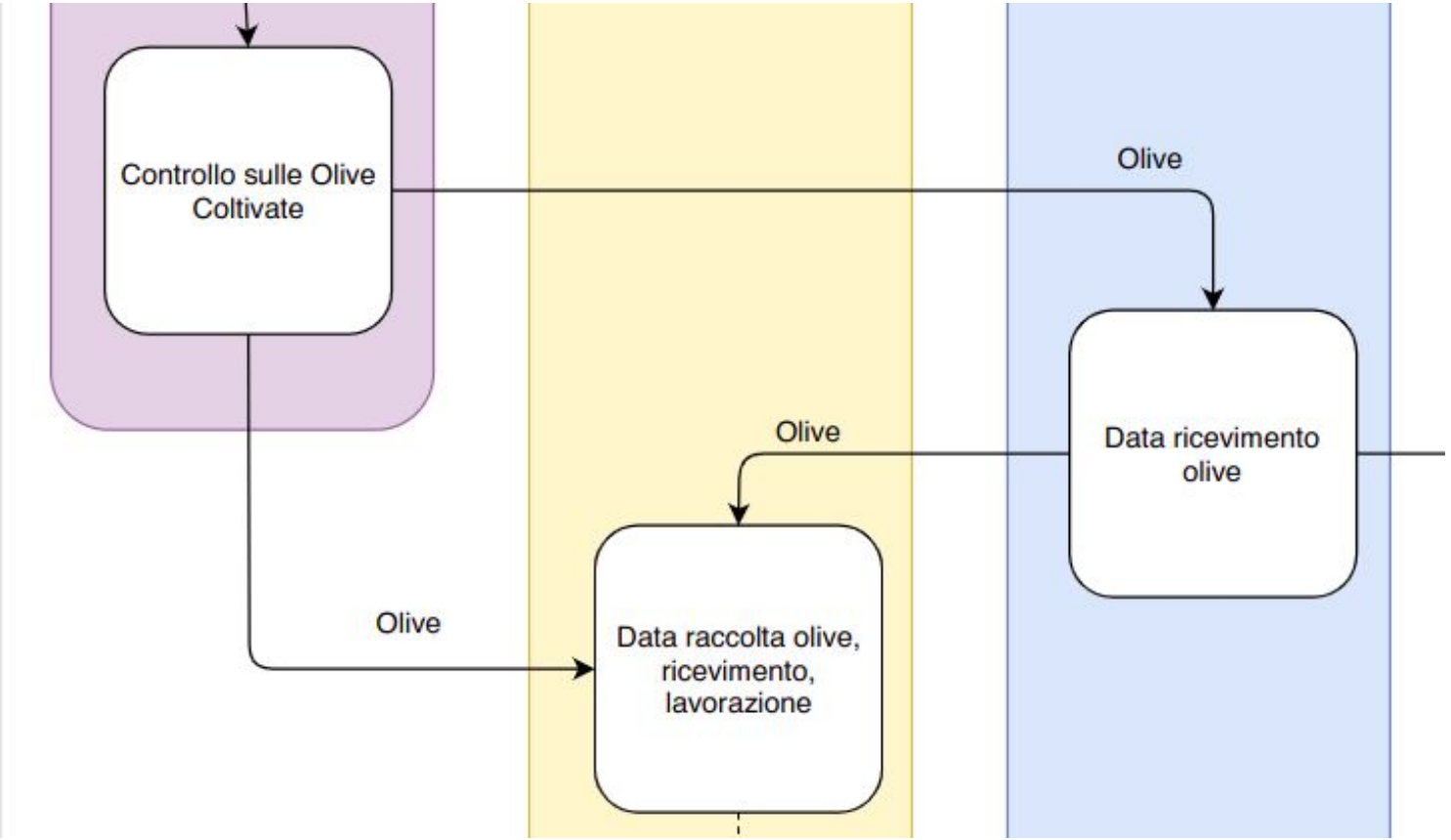
Lo stack su cui si basa il progetto è il seguente.

Typescript e Javascript sono i linguaggi utilizzati.

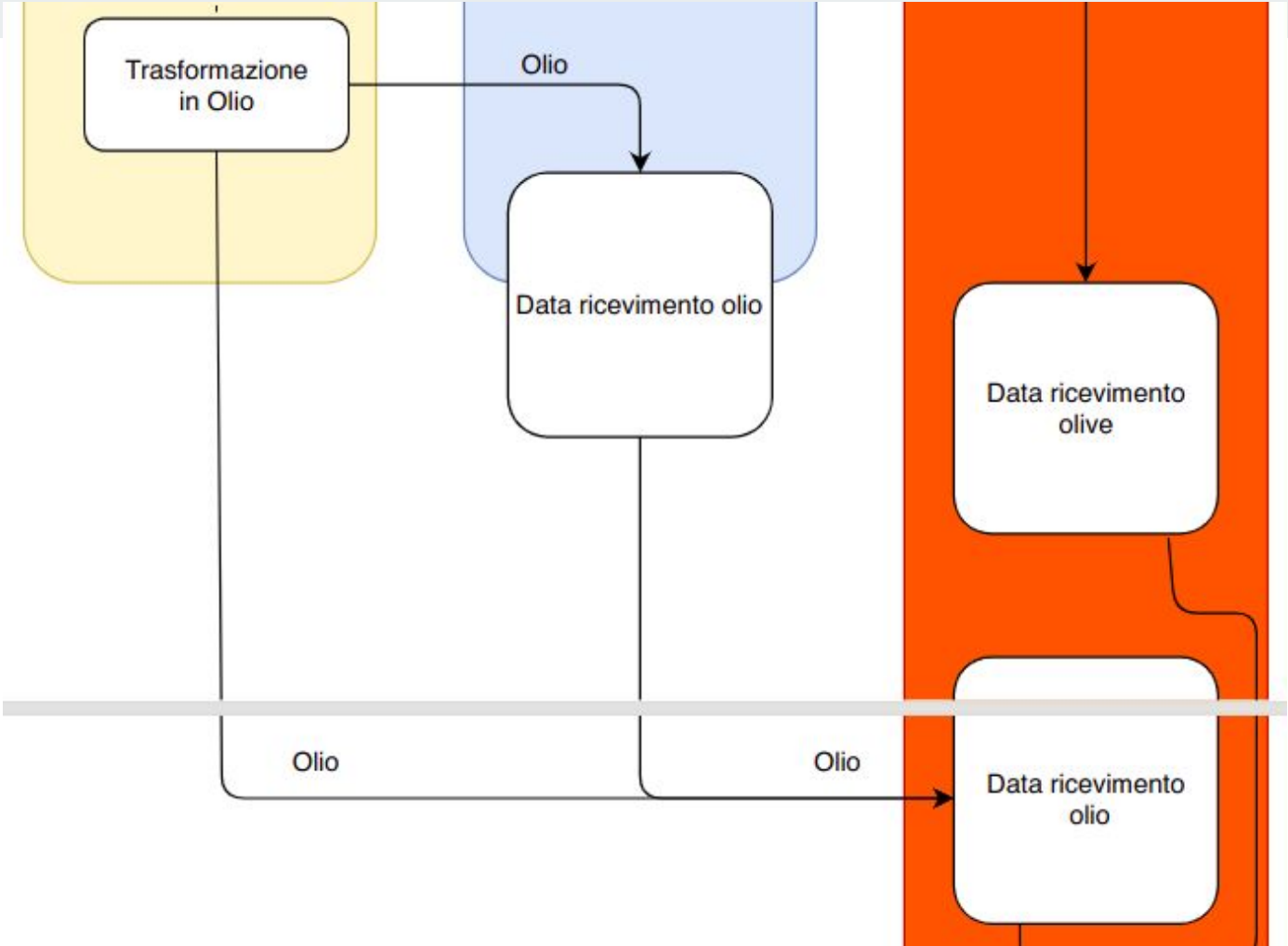
CouchDB è il DBMS su cui si basa Fabric.



Filiera Olio



Filiera Olio



Sviluppi futuri

Sviluppi futuri

Le finalità di AgriChain e futuri possibili sviluppi sono orientati a:

- Rinforzare la **rintracciabilità** in modo automatico tramite per esempio Barcode di nuova generazione o *RFID* applicati sui prodotti (singoli o bancali di prodotto) per tracciare ad esempio in modo sicuro e inalterabile tempi di lavorazione, stagionatura, consegna.
- Ridurre drasticamente le frodi documentali cartacee: sostituendole con il **digitale** blockchain, la successiva alterazione è automaticamente impedita.
- Ridurre la possibilità di frode alimentare di tipo commerciale, riguardanti per esempio falsa provenienza o qualità.

Grazie per l'attenzione

CyberSecurity Research Lab UniPG