Bozza Studio di fattibilità

RedRoundRobin

26 novembre 2019

Indice

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha l'obiettivo di descrivere brevemente ciò che ogni capitolato$_G$ ha da proporre, elencando quelli che il nostro gruppo ha considerato come i loro aspetti più interessanti e le loro criticità.

1.2 Glossario

Viene fornito un *Glossario v0.0.1* per evitare possibili ambiguità relative alle terminologie utilizzate nei vari documenti. Nel documento sarà presente a pedice di quelle che riteniamo delle parole una '**G**'.

1.3 Riferimenti

1.3.1 Normativi

- **Norme di Progetto**:

1.3.2 Informativi

- **Capitolato$_G$ d'appalto C1 - Autonomous Highlights Platform:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C1.pdf`
- **Capitolato$_G$ d'appalto C2 - Etherless:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C2.pdf`
- **Capitolato$_G$ d'appalto C3 - NaturalAPI:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C3.pdf`
- **Capitolato$_G$ d'appalto C4 - Predire in Grafana:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C4.pdf`
- **Capitolato$_G$ d'appalto C5 - Stalker:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C5.pdf`
- **Capitolato$_G$ d'appalto C6 - ThiReMa - Things Relationship Management:** `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C6.pdf`

2 Valutazione capitolato scelto

3 Valutazione capitolati rimanenti

3.1 Capitolato C1 - Autonomous Highlights Platform

3.1.1 Informazioni generali

- **Proponente:** Zero12.

- **Committente:** Prof. Tullio Vardanega e Prof. Riccardo Cardin.

3.1.2 Descrizione

L'obiettivo di questo capitolato è creare una piattaforma web che è capace di ricevere in input dei video di eventi sportivi, come una partita di calcio o di tennis, e che riesca a creare autonomamente un video di massimo 5 minuti contenente soltanto i suoi momenti chiave (highlights).

3.1.3 Finalità del progetto

Il prodotto finale, ovvero la piattaforma web, dovrà essere dotata di un modello di machine learning in grado di identificare quelli che possono essere considerati come i momenti più importanti dell'evento sportivo che le è stato inviato. Un fattore importante è che andrà scelto uno sport sul quale focalizzare la propria attenzione e sul quale verrà addestrato il modello di apprendimento atto all'identificazione automatica dei momenti salienti di un evento del suddetto sport. Il flusso di generazione del suddetto highlight dovrà avere la seguente struttura:

- Caricamento del video;
- Identificazione dei momenti salienti;
- Estrazione delle corrispondenti parti di video;
- Generazione del video di sintesi;

3.1.4 Tecnologie

Le tecnologie consigliate dall'azienda riguardano la tecnologia di Amazon Web Services ed in particolare:

- **Elastic Container Service o Elastic Kubernetes Service:** è un servizio che permette la gestione di contenitori, altamente dimensionabile e ad elevate prestazioni;
- **DynamoDB:** è un database non relazionale per applicazioni che necessitano di prestazioni elevate su qualsiasi scala.
- **AWS Transcode:** è un servizio di transcodifica di contenuti multimediali nel cloud;
- **Sage Maker:** è un servizio completamente gestito che permette a sviluppatori e data scientist di creare, addestrare e distribuire modelli di apprendimento automatico;
- **AWS Rekognition video:** è un servizio di analisi video basato su apprendimento approfondito; è in grado di riconoscere i movimenti delle persone in un fotogramma e di riconoscere soggetti, volti, oggetti, celebrità e contenuti inappropriati;

3.1.5 Linguaggi di programmazione

- **NodeJS:** linguaggio ideale per sviluppare API Restful JSON a supporto dell'applicativo;
- **Python:** linguaggio ideale per lo sviluppo delle componenti di machine learning;
- **HTML5**, **CSS3**, **Javascript:** linguaggi per la realizzazione dell'interfaccia web di gestione del flusso di lavoro, utilizzando un framework responsive come Twitter;

Vincoli del progetto:

- Utilizzo di Sage Maker;
- L'architettura dovrà essere basata su micro-servizi, suddividendo il progetto in tante funzioni di base denominate servizi. Questi ultimi dovranno essere indipendenti fra loro;
- Caricamento dei video da elaborare tramite riga di comando;
- Console web di analisi e controllo degli stati di elaborazione dei video;

3.1.6 Aspetti positivi

- Per quanto riguarda le tecnologie interessate è presente molta documentazione a riguardo;
- Il tema principale del progetto, ovvero machine learning, è stato accolto con molto interesse dal gruppo, che si è dimostrato interessato ad approfondire l'argomento (che potrebbe essere un'aggiunta interessante al proprio CV);
- Il proponente fornisce attività di formazione sulle principali tecnologie AWS e wireframe dell'interfaccia della console web di analisi e controllo dello stato di elaborazione dei video;

3.1.7 Criticità

- Il proponente non fornisce nessun data-set per effettuare il training dell'algoritmo;
- Le tecnologie nonostante siano state accolte con interesse non sono conosciute dal gruppo e richiedono un apprendimento individuale avanzato sul machine learning;

3.1.8 Conclusione

DA FARE

3.2 Capitolato C2 - Etherless

3.2.1 Informazioni generali

- **Proponente: Red Babel**
- **Committente:** Prof. Tullio Vardanega e Prof. Riccardo Cardin.

3.2.2 Descrizione

Etherless è una piattaforma cloud che permette, agli sviluppatori di fare il deploy di funzioni JavaScript, mentre agli utenti finali di pagare per l'esecuzione delle suddette funzioni (si basa su CaaS, ovvero Computation-as-a-Service). Una parte di ciò che viene pagato dagli utenti verrà trattenuta dal piattaforma stessa come compenso per l'effettiva esecuzione della funzione.

3.2.3 Finalità del progetto

Il progetto finale si propone quindi di fornire un servizio di CaaS utilizzando la seguente struttura:

- **Etherless-cli:** è il modulo con il quale gli sviluppatori interagiscono con Etherless. Deve supportare diversi comandi, ovvero: la configurazione del proprio account, eseguire il deploy delle funzioni, elencare le funzioni già presenti, eseguire una funzione e visualizzare i logs riguardanti una specifica funzione.
- **Etherless-smart:** consiste in un set di contratti smart che gestiscono la comucazione e il trasferimento di denaro (detto ETH) tra Etherless-cli ed Etherless-server.
- **Etherless-server:** è il modulo che esegue le funzioni e che tramite Etherless-smart comunica con Etherless-cli. Nel momento in cui viene restituito il valore di una funzione o viene lanciata una eccezione, Etherless-server emettera un evento nella blockchain che verrà ricevuto dalla Etherless-cli che mostrarà infine il risultato all'utente.

3.2.4 Tecnologie

Il progetto si basa sull'unione di due tecnologie, ovvero Ethereum e Serverless, più precisamente utilizza:

- **Ethereum:** è una piattaforma che permette ai suoi utenti di scrivere appicazioni decentralizzate (dette ĐApps) che usano la tecnologia blockchain;
- **Ethereum Virtual Machine (EVM):** è una macchina virtuale decentralizzata che esegue script usando un network internazionale di nodi pubblici;

- **Blockchain :** è una struttura dati condivisa e immutabile; tramite questa struttura è possibile tenere traccia dei pagamenti effettuati in Ethereum;
- **Smart Contract:** sono utilizzati per le interazioni tra attori e possono contenere denaro (Ether o ETH), dati o una combinazione di entrambi;
- **Gas:** è il carburante che permette alla EVM di eseguire un programma;
- **MainNet, Ropsten:** sono reti sulle quali vengono eseguiti i protocolli di Ethereum. La prima è la rete principale mentre la seconda è la rete di test ufficiale creata dalla The Ethereum Foundation. Inoltre è possibile creare la propria rete Ethereum, ed anche simularla localmente, per permetterne lo sviluppo.
- **Eventi Ethereum:** sono una delle parti fondamentali di Ethereum in quanto possono essere: il valore di ritorno di uno smart contract, un innesco asincrono contenente dati oppure una forma di immagazzinamento più economica rispetto ad uno smart contract;
- **Architettura serverless:** è un metodo di creazione ed esecuzione di applicazioni e servizi che non richiede la gestione di un'infrastruttura (idea di Baas, Backend-as-a-Service);
- **AWS Lambda:** è un servizio che permette di eseguire codice in risposta ad eventi, senza effettuare il provisioning né gestire server.
- **Serverless Framework:** è un framework open-source che permette di sviluppare e distribuire applicazioni serverless;
- **CloudFormation:** è uno strumento che permette di gestire risorse e per fare il deploy di infrastrutture;
- **API Gateway, AWS DynamoDB, AWS S3:** componenti che possono servire da supporto alle applicazioni serverless;
- **Truffle:** è un ambiente di sviluppo per Ethereum, consigliato per creare un web server locale per gestire la front end;
- **Node Package Manager (NPM):** è un package manager congliato per installare etherless-cli;
- **ESLint:** è uno strumento di analisi del codice staticoper identificare schemi problematici nel codice JavaScript;

3.2.5 Linguaggi di programmazione

- **YAML, JSON:** sono due linguaggi che permettono di dare una struttura ai dati che poi verrà creata tramite CloudFormation;
- **Solidity:** è un liguaggio tipato staticamente che supporta l'eredità, librerie e tipi definiti dall'utente. Viene utilizzato per scrivere smart contracts.

- **JavaScript:** linguaggio utilizzato per svolgere differenti compiti all'interno del progetto;
- **TypeScript:** linguaggio basato su JavaScript che permette di definire la tipologia dei dati. Ne è consigliata la versione 3.6.

Vincoli del progetto:

- Etherless-cli deve poter essere installato con il comando bash:

```
npm install -g etherless-cli
```

- Dopo essere stato installato uno sviluppatore deve essere in grado di eseguire diversi comandi associati ai seguenti compiti: Creare o entrare in un account Ethereum, rilasciare una funzione, eseguire una funzione già presente, eliminare una funzione caricata.
- Deve essere possibile fare degli upgrade agli smart contracts;
- Etherless deve essere sviluppato usando TypeScript 3.6 usando un approccio promise / async-await;
- Deve essere usato typescript-eslint insieme a EsLint durante la fase di sviluppo;
- Etherless-server deve essere implementato usando Serverless Framework;

3.2.6 Aspetti positivi

- L'argomento Ethereum si dimostra interessante visto che negli ultimi tempi le criptovalute hanno riscosso ampio successo e questo capitolato potrebbe essere un notevole arricchimento del bagaglio di conoscenze del gruppo;

3.2.7 Criticità

- L'azienda ha sede all'estero e potrebbe fornire un supporto inferiore rispetto alle aziende che si trovano nel territorio nazionale.
- Il capitolato necessita di uno studio approfondito diverse tecnologie e potrebbe richiedere un lavoro decisamente superiore agli altri capitolati;

3.2.8 Conclusione

DA FARE

3.3 Capitolato C3 - NaturalAPI

3.3.1 Informazioni generali

- **Proponente: teal.blue**
- **Committente:** Prof. Tullio Vardanega e Prof. Riccardo Cardin.

3.3.2 Descrizione

Il progetto NaturalAPI nasce dall'idea che ogni persona in base al contesto in cui si trova nel proprio lavoro sviluppa ed utilizza un linguaggio specifico, utilizzando termini diversi per descrivere gli stessi problemi e/o soluzioni. Tuttavia le soluzioni correnti usano uno strato di collegamento tra linguaggio naturale (inglese, italiano, etc) e il codice stesso, strato che viene creato in maniera arbitraria da chi è il responsabile della progettazione delle API.

3.3.3 Finalità del progetto

Il progetto finale si propone dunque di fornire una (proof-of-concept) serie di strumenti, chiamata NaturalAPI, che possando ridurre il divario tra specifiche di progetto (espresse in inglese) e le API stesse. I tre strumenti che andranno a comporre questo toolkit saranno:

- **NaturalAPI Discover:** un estrattore del linguaggio specifico del dominio trattato, che estrarrà potenziali entità (nomi/oggetti), processi (azioni/verbi) o una combinazione di questi ultimi da documenti non strutturati (che possono appartenere a diverse tipologie come manuali, wiki o documenti veri e propri) legati al dominio in questione. Prima della fase successiva gli stakeholder del progetto selezioneranno un sottoinsieme delle entitò/processi prodotti dal NaturalAPI Discover. Questo strumento produrrà dei file con estensione .bdl (business domain language);
- **NaturalAPI Design:** responsabile della creazione di un' API specifica del dominio, utilizzando dei documenti Gherkin e dei documenti .bdl legati allo stesso dominio (prodotti dal NaturalAPI Discover); Questo strumenti produrrà dei file con estensione .bal (business application language);
- **NaturalAPI Develop:** sarà responsabile della conversione e dell'esportazione dei file .bal (prodotti dal NaturalAPI Design) in test automatici ed API in uno dei linguaggi di programmazione/framework scelti, supportando sia la creazione di una nuova repository che l'aggiornamento di una già esistente. Da notare che prima dell'esportazione finale deve essere possibile definire dei dettagli specifici legati al linguaggio scelto dallo sviluppatore delle API; questi dettagli andranno immagazzinati in file con stensione .pla (programming language adapter).

3.3.4 Tecnologie

Il progetto sfrutterà le seguenti tecnologie:

- **Gherkin:** è un formato per scrivere test in Cucumber, utilizzando delle keyword (come Given, When, Then) molto simili al linguaggio naturale. Viene proposto per scrivere dei casi d'uso nella fase di creazione dei file .bal (di cui è responsabile il NaturalAPI Design)
- **OpenAPI Specification:** definisce una descrizione dellinterfaccia standard, indipendente dal linguaggio di programmazione usato, per REST API. Ne viene consigliato l'uso nella parte di generazione finale delle API, il cui compito è affidato al NaturalAPI Develop;

3.3.5 Linguaggi di programmazione

Non c'è alcun vincolo sul linguaggio id programmazione/framework da utilizzare.

Vincoli del progetto:

- Ogni strumento facente parte di NaturalAPI deve poter essere accessibile almeno tramite due tra le seguenti interfacce:
  - Interfaccia con linea di comando;
  - Interfaccia grafica minimale;
  - Interfaccia web REST;
- NaturalAPI deve essere disponibile in almeno una tra le seguenti piattaforme desktop (Ubuntu, macOS, Windows o tramite browser);
- I layers logici devono essere rilasciati in una delle seguenti modalità:
  - Come una libreria (statica o dinamica);
  - Come parte degli stessi eseguibili delle modalità di rilascio scelte;
  - Come un processo/servizio indipendente (locale o remoto);
- le modalità di rilascio devono condividere gli stessi layers logici, che non devono avere nessuna dipendenza dalla modalità di rilascio.
- Tutte le risorse in input/output devono seguire la codifica UTF-8 e interruzioni di riga Unix;

3.3.6 Aspetti positivi

DA FARE

3.3.7 Criticità

- Il capitolato ha riscosso scarso interesse nella maggior parte dei membri del gruppo;

3.3.8 Conclusione

DA FARE