**B.smart - Progetto *Etherless***

# Studio di Fattibilità

| **Informazioni essenziali** | |
|---|---|
| **Nome documento** | Studio di Fattibilità |
| **Versione** | 1.0.0 |
| **Stato** | Approvato |
| **Redazione** | Matteo Lucato |
| | Denis Benato |
| **Verifica** | Maria Morra |
| | Erik Nucibella |
| **Approvazione** | Andrea Didoné |
| **Uso** | Interno |
| **Distribuzione** | B.smart |
| **Destinato a** | B.smart |
| | Prof. Tullio Vardanega |
| | Prof. Riccardo Cardin |
| **Recapito email** | b.smart.swe@gmail.com |

## Registro delle modifiche

| Versione | Data | Nominativo | Ruolo | Descrizione |
|---|---|---|---|---|
| 1.0.0 | 2020-04-01 | Andrea Didoné | Responsabile | Approvazione documento |
| 0.2.0 | 2020-04-01 | Maria Morra | Verificatore | Verifica §5, §6, §7 |
| 0.1.1 | 2020-03-28 | Denis Benato | Analista | Stesura §5, §6, §7 |
| 0.1.0 | 2020-03-27 | Erik Nucibella | Verificatore | Verifica §2, §3, §4 |
| 0.0.3 | 2020-03-25 | Matteo Lucato | Analista | Stesura §2, §3, §4 |
| 0.0.2 | 2020-03-23 | Denis Benato | Analista | Stesura Introduzione §1 |
| 0.0.1 | 2020-03-23 | Matteo Lucato | Analista | Creata la struttura del documento in LaTeX |

# Indice

# 1 Introduzione

## 1.1 Scopo del prodotto

*Etherless*$_G$ è una *DApp*$_G$ accessibile attraverso una *CLI*$_G$, che ha lo scopo di consentire agli utenti della piattaforma la pubblicazione del proprio software in ambiente *serverless*$_G$, permettendo a tutti gli utilizzatori del servizio di eseguirlo pagando un corrispettivo attraverso la rete *Ethereum*$_G$.

## 1.2 Glossario

Al fine di evitare possibili ambiguità relative al linguaggio utilizzato nei documenti formali, viene fornito il *Glossario 1.0.0*. In questo documento vengono definiti e descritti tutti i termini tecnici e/o specifici che necessitano di un possibile approfondimento. Per facilitare la lettura, i termini saranno contrassegnati in corsivo e da una $_G$ a pedice.

## 1.3 Riferimenti

### 1.3.1 Normativi

- **Norme di progetto**: *Norme di Progetto 1.0.0*.

### 1.3.2 Informativi

- ***Capitolato*$_G$ d'appalto 1 (Autonomous Highlights Platform):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C1.pdf`;
- ***Capitolato*$_G$ d'appalto 2 (Etherless):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C2.pdf`;
- ***Capitolato*$_G$ d'appalto 3 (NaturalAPI):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C3.pdf`;
- ***Capitolato*$_G$ d'appalto 4 (Predire in Grafana):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C4.pdf`;
- ***Capitolato*$_G$ d'appalto 5 (Stalker):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C5.pdf`;
- ***Capitolato*$_G$ d'appalto 6 (ThiReMa - Things Relationship Management):**
  `https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C6.pdf`.

# 2 Capitolato C1 - Autonomous Highlights Platform

## 2.1 Descrizione del progetto

Il *capitolato*$_G$, proposto dall'azienda Zero12, ha come scopo quello creare una piattaforma web in grado di estrarre i momenti salienti da un video di un evento sportivo, per una durata massima di 5 minuti.

## 2.2 Obiettivo del progetto

La creazione del video avviene con le seguenti modalità:

- L'*utente*$_G$ farà l'upload del proprio video dell'evento sportivo usufruendo di una *CLI*$_G$;
- Verranno poi identificati ed estrapolati dei momenti salienti;
- Seguirà poi la costruzione del video di sintesi;
- L'*utente*$_G$ potrà seguire tutte le operazioni precedentemente descritte interfacciandosi ad una piattaforma web visualizzando poi il risultato finale.

## 2.3 Tecnologie interessate

- ***AWS*$_G$ (Amazon Web Services):** piattaforma che offre servizi di *cloud computing*$_G$;
- **ECS (Elastic Container Service):** servizio per la gestione di contenitori;
- **Amazon DynamoDB:** database non relazionale a prestazioni elevate. Utilizzato per immagazzinare informazioni;
- **Amazon Elastic Transcoder:** servizio che permette la conversione di diversi formati video;
- **Amazon SageMaker:** servizio adibito all'allenamento di modelli *machine learning*$_G$;
- **Amazon Rekognition video:** servizio che permette il riconoscimento di elementi all'interno di un video;
- **Bootstrap**: *framework*$_G$ per realizzare l'interfaccia grafica della piattaforma web;
- **HTML5, CSS3 e *JavaScript*$_G$:** insieme di linguaggi per sviluppare la piattaforma web;
- **NodeJS:** *runtime*$_G$ *JavaScript*$_G$ per lo sviluppo di *API*$_G$ a supporto dell'*applicativo*$_G$;
- **Python:** linguaggio per lo sviluppo sei servizi dedicati al *machine learning*$_G$.

## 2.4 Aspetti positivi

- Possibilità di apprendere e utilizzare le tecnologie *AWS*$_G$. Questo tipo di conoscenza è stata ritenuta, da molti componenti del gruppo, utile in diversi ambiti del mondo informatico;
- Progetto basato su concetti di *machine learning*$_G$ anche questi estremamente interessanti.

## 2.5 Criticità

- L'apprendimento e il corretto utilizzo della piattaforma *AWS*$_G$ può richiedere del tempo.

## 2.6 Conclusione

Il gruppo aveva fin da subito concentrato le proprie attenzioni su questo *capitolato*$_G$ ma un incontro con l'azienda non è mai stato possibile a causa delle lente risposte da parte di Zero12 dovute alla *pandemia*$_G$ attuale. Si e' scelto quindi di preferire il *capitolato*$_G$ C2 in quanto richiedeva anch'esso l'utilizzo di *AWS*$_G$.

# 3 Capitolato C2 - Etherless

## 3.1 Descrizione del progetto

*Etherless*$_G$ è una piattaforma che permette agli utenti di caricare prodotti *software*$_G$ *JavaScript*$_G$ su servizi *cloud*$_G$ mettendole a disposizione di tutti. Il cliente finale poi pagherà per eseguire le suddette funzioni attraverso la rete *Ethereum*$_G$ e la gestione di *smart contract*$_G$.

## 3.2 Obiettivo del progetto

L'obiettivo è la creazione di un ambiente in cui degli utenti sviluppatori creano e mettono a disposizione dei prodotti *software*$_G$ *JavaScript*$_G$ configurando poi il loro costo di esecuzione. Altri utenti poi potranno usufruire dei prodotti creati pagando il costo precedentemente impostato dal creatore senza così riscrivere la procedura.

La piattaforma deve essere conforme ai seguenti punti:

- Gli utenti avranno la possibilità di vedere i prodotto *software*$_G$ disponibili, caricarne di nuovi, aggiornarli, eseguirli ed eliminarli il tutto attraverso una *CLI*$_G$;
- Gli utenti saranno in grado di decidere un costo di esecuzione per ogni singolo prodotto *software*$_G$ che sviluppano. I costi inoltre non potranno superare la soglia minima calcolata dal sistema;
- Il ricavo proveniente dal pagamento dei costi di esecuzione di un prodotto *software*$_G$ deve essere spartito tra l'autore dello stesso e i proprietari della piattaforma *Ethereum*$_G$;
- I componenti di *Etherless*$_G$ dovranno utilizzare la rete *Ethereum*$_G$ in modo tale da poter utilizzare gli *smart contract*$_G$ e permettere quindi un'interazione completa col sistema;
- Utilizzo di una infrastruttura *serverless*$_G$ per contenere la logica di *back-end*$_G$.

## 3.3 Tecnologie interessate

- ***AWS*$_G$ (Amazon Web Services):** piattaforma che offre servizi di *cloud computing*$_G$;
- ***AWS - Lambda*$_G$:** servizio che permette di eseguire codice direttamente in *cloud*$_G$;
- ***Ethereum*:$_G$** piattaforma decentralizzata per la creazione e pubblicazione di *smart contract*$_G$ e gestione di pagamenti in *cryptovalute*$_G$;
- **Solidity:** linguaggio *Object-oriented*$_G$ per la creazione di *smart contract*$_G$;
- **Truffle:** *framework*$_G$ che permette lo sviluppo di *smart contract*$_G$ sulla rete *Ethereum*$_G$;
- **Web3:** *API*$_G$ *JavaScript*$_G$ che permette di interagire con nodi di *Ethereum*$_G$;
- ***MainNet*$_G$:$_G$** rete principale del sistema *Ethereum*$_G$;
- **Ropsten:** rete *Ethereum*$_G$ pubblica usata per testing prima della pubblicazione dell'*applicativo*$_G$ sulla *MainNet*$_G$;
- **Ganache:** ambiente con cui è possibile creare una rete *Ethereum*$_G$ locale per permetterne il test e eventuali analisi di sistema;
- **TypeScript 3.6:** linguaggio di programmazione standardizzato e *open source*$_G$ che permette la realizzazione di codice JavaScript;

- **Node.js:** *runtime*$_G$ *JavaScript*$_G$ per lo sviluppo di *API*$_G$ a supporto dell'*applicativo*$_G$;
- ***Serverless*$_G$ Framework:** *framework*$_G$ che permette la costruzione di ambienti *serverless*$_G$;
- ***Smart Contract*$_G$**: *protocollo*$_G$ digitale che aiuta e controlla la corretta esecuzione di un contratto.

## 3.4 Aspetti positivi

- Possibilità di apprendere e utilizzare le tecnologie *AWS*$_G$ e *Blockchain*$_G$. Questo tipo di conoscenze sono state ritenute, da molti componenti del gruppo, utili in diversi ambiti del mondo informatico;
- Possibilità di familiarizzare ed imparare a lavorare con l'ambiente *Ethereum*$_G$.

## 3.5 Criticità

- L'apprendimento e il corretto utilizzo della piattaforma *AWS*$_G$ può richiedere del tempo;
- Presenza di molte tecnologie con cui il gruppo ha poca familiarità.

## 3.6 Conclusione

Dopo il rifiuto di Zero12 il gruppo ha scelto subito questo *capitolato*$_G$ in quanto il progetto richiedeva l'uso di AWS. Tecnologia che aveva attirato l'attenzione dei membri dal *capitolato*$_G$ C1. Inoltre per conseguire questo progetto occorrono dei fondamenti di *programmazione asincrona*$_G$ che il gruppo considera necessari conoscere anche per ambiti futuri.

# 4 Capitolato C3 - NaturalAPI

## 4.1 Descrizione del progetto

Vista la possibile ambiguità che il linguaggio umano comporta tra *stakeholders*$_G$, "Natural API" propone un terreno comune con l'aiuto di un linguaggio di programmazione.

## 4.2 Obiettivo del progetto

NaturalAPI ha come scopo quello di ridurre la distanza tra le specifiche di un progetto in lingua inglese e le *API*$_G$ utilizzando la metodologia del *Behaviour Driven Development (BDD)*$_G$. Il progetto si può suddividere in tre parti logiche:

- **NaturalAPI Discover:** logica in grado di estrarre da un documento le entità coinvolte in esso assieme ai loro processi e le relazioni. Verranno quindi estratti dal documento: verbi, nomi e predicati. Il tutto con relativa frequenza con cui compaiono;
- **NaturalAPI Design:** logica il cui compito è quello di utilizzare la lista delle parole precedentemente estratte per creare una *business application language*$_G$ con la quale poter poi scrivere le *API*$_G$;
- **NaturalAPI Develop:** logica che converte il linguaggio creato nel punto precedente in modo da poterlo utilizzare per la creazione delle *API*$_G$ e testarne poi il funzionamento.

## 4.3 Tecnologie interessate

- **Gherkin:** linguaggio usato dal programma Cucumber per la gestione degli scenari e linguaggio naturale;
- **Cucumber:** software che supporta lo sviluppo basato sul *Behaviour Driven Development*$_G$;
- **HipTest:** piattaforma per il test di logiche *Behaviour Driven Development*$_G$;
- **Jbehave:** *framework*$_G$ per il *Behaviour Driven Development*$_G$.

## 4.4 Aspetti positivi

- Possibilità di apprendere e utilizzare le tecnologie di elaborazione linguaggi naturali. Cose completamente nuove per tutti i membri del gruppo.

## 4.5 Criticità

- Lavorare con linguaggi naturali ed interpretazione può essere dispendioso in termini di tempo e le nuove tecnologie in uso potrebbero contribuire al problema.

## 4.6 Conclusione

Il *capitolato*$_G$ non ha suscitato particolare interesse per il gruppo che era voglioso di sperimentare la tecnologia *AWS*$_G$ di altri progetti. Inoltre NaturalAPI è stato considerato poco interessante anche per una futura prospettiva lavorativa.

# 5 Capitolato C4 - Predire in Grafana

## 5.1 Descrizione del progetto

La Zucchetti ha scelto Grafana per migliorare il rapporto tra la fabbrica del software e gli operatori che erogano il servizio in modo tale da perfezionare la qualità dei lavori forniti. Si possono monitorare i sistemi in modo tale da utilizzare i dati ricevuti per segnalazioni e per compiere delle previsioni visualizzate poi sotto forma di grafici.

## 5.2 Obiettivo del progetto

L'obiettivo del progetto richiede la creazione di due *plug-in*$_G$ di Grafana scritti in *JavaScript*$_G$ che, una volta prelevati i dati dal sistema monitorato, creano le previsioni del caso. Il prodotto finale deve essere conforme ai seguenti punti:

- Creare un file formato JSON contenente i parametri per il calcolo delle previsioni utilizzando la Regressione Lineare o con SVM;
- Leggere i *predittori*$_G$ del file JSON per compiere la previsione;
- Applicare la previsione e fornire i dati a Grafana;
- Visualizzare i dati creando opportuni grafici per la loro immediata interpretazione.

## 5.3 Tecnologie interessate

- **Grafana:** software *open-source*$_G$ capace di creare grafici e altri elementi visivi per una facile interpretazione di diversi flussi dati;
- **SVMJS:** libreria *JavaScript*$_G$ che permette l'utilizzo modello di *machine learning*$_G$ SVM;
- **Regression-js:** libreria *JavaScript*$_G$ che permette l'utilizzo del modello di *machine learning*$_G$ del metodo statistico di regressione lineare;
- ***Javascript:***$_G$ linguaggio di programmazione utilizzato per sviluppare i *plug-in*$_G$.

## 5.4 Aspetti positivi

- Possibilità di approfondire modelli di *machine learning*$_G$ e modelli statistici.

## 5.5 Criticità

- Presenza di tecnologie poco stimolanti.

## 5.6 Conclusione

Il gruppo ha ritenuto poco stimolanti le tecnologie utilizzate nel progetto in quanto alcune già affrontate durante l'anno accademico. Si è invece creato un particolare interesse per le tecnologie *AWS*$_G$ che hanno portato a scartare questo *capitolato*$_G$.

# 6 Capitolato C5 - Stalker

## 6.1 Descrizione del progetto

Le nuove normative vigenti in merito alla sicurezza dei locali pubblici, richiedono una corretta gestione degli individui al loro interno. Il *capitolato*$_G$ quindi propone un prodotto software in grado di tracciare il numero di persone presenti in una struttura e controllarne la disposizione.

## 6.2 Obiettivo del progetto

Lo scopo del progetto è quello di sviluppare un'applicazione per sistemi operativi mobili in grado di segnalare e gestire sia l'ingresso che l'uscita dell'utilizzatore dalle aree designate. L'applicazione deve avere le seguenti specifiche:

- Un sistema di *signup*$_G$ e login per gli utilizzatori;
- Poter recuperare e memorizzare lo storico delle posizioni e spostamenti;
- Poter controllare in tempo reale la posizione attuale nell'area designata ed il tempo trascorso all'interno di essa;
- Una sezione dedicata agli amministratori che potranno gestire varie informazioni ed impostazioni riguardanti le aree d'interesse.

## 6.3 Tecnologie interessate

- **Java/Python/Node.js:** come linguaggi da utilizzare per lo sviluppo della parte *back-end*$_G$;
- **IAAS Kubernetes/PAAS/Openshift/Rancher:** piattaforme per il rilascio delle componenti Server e la gestione della scalabilità orizzontale;
- ***iOS***$_G$**/Android:** sistemi operativi mobili che utilizzeranno l'applicazione.

## 6.4 Aspetti positivi

- Possibilità di approfondire tecnologie di geolocalizzazione.

## 6.5 Criticità

- Difficoltà nel gestire il servizio di localizzazione in varie tipologie di ambiente;
- Difficoltà nell'adattare l'applicazione ai diversi dispositivi mobili presenti sul mercato.

## 6.6 Conclusione

Il *capitolato*$_G$ è stato scartato velocemente dal gruppo per il poco interesse nel costruire un'applicazione mobile e per i possibili e vari problemi di affidabilità che potrebbero avere le tecnologie richieste in esso.

# 7 Capitolato C6 - Things Relationship Management

## 7.1 Descrizione del progetto

Sanmarco Informatica propone lo sviluppo di un software per la gestione ed elaborazione di grandi quantità di dati provenienti da singole aziende. Si cerca quindi di velocizzare ed ottimizzare la notifica delle informazioni più urgenti a chi di interesse.

## 7.2 Obiettivo del progetto

Lo scopo del progetto consiste nel creare un software in grado di ricevere misurazioni da sensori di diverse tipologie ed accumulare i dati all'interno di database. Le informazioni più importanti verranno poi notificate tramite Telegram.
Il software ThiReMa finale deve contenere i seguenti punti:

- Prevedere un sistema di elaborazione dei dati provenienti dai diversi sensori;
- Un processo per la scrittura dei dati all'interno di database;
- Una serie di componenti adibiti all'elaborazione dei dati all'interno del database;
- Invio delle informazioni basato su Telegram.

## 7.3 Tecnologie interessate

- **Apache Kafka:** piattaforma *open source*$_G$ per la gestione di feed e dati in tempo reale;
- **Java:** linguaggio di programmazione utilizzato per sviluppare le varie componenti;
- **PostgreSQL:** database relazionale;
- **TimescaleDB:** database per la gestione delle tipologie di dati temporali;
- **ClickHouse:** database improntato alla gestione dei dati per colonna;
- **Bootstrap:** *framework*$_G$ per realizzare l'interfaccia grafica della piattaforma web;
- **Docker:** progetto *open source*$_G$ per la virtualizzazione di applicazioni in ambienti virtualizzati permettendone la pubblicazione;
- **GitHub:** *sistema di versionamento*$_G$;
- **Telegram:** servizio di messaggistica istantanea tramite il quale notificare le aziende.

## 7.4 Aspetti positivi

- Possibilità di approfondire tecnologie specifiche per $IoT_G$;
- Possibilità di approfondire tecnologie specifiche per grandi quantità di dati in tempo reale e loro elaborazione.

## 7.5 Criticità

- Difficoltà nel gestire diverse tipologie di sensori;
- Difficoltà nel gestire la mole di diverse tecnologie proposte.

## 7.6 Conclusione

Sebbene le tecnologie in uso nel progetto risultino interessanti ed utili si è preferito concentrarsi su $AWS_G$ in quanto a confronto risultato più interessante per l'intero gruppo.