RedRoundRobin

STUDIO DI FATTIBILITA'

v0.0.4

www.redroundrobin.site — info@redroundrobin.site

INFORMAZIONI SUL DOCUMENTO

Versione	0.0.4
Uso	interno
Stato	in redazione
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
Redattori	Giuseppe Vito Bitetti
	nome cognome
Verificatori	nome cognome
	nome cognome
Approvazione	nome cognome

Descrizione
Breve descrizione del documento

Registro delle modifiche

Versione	Descrizione	Data	Autore	Ruolo
0.0.3	Stesura valutazione C5	05-12-2019	Giuseppe Vito Bitetti	Redattore
0.0.2	Stesura valutazione C4	04-12-2019	Giuseppe Vito Bitetti	Redattore
0.0.1	Creazione iniziale del documento	02-12-2019	Alessandro Tommasin	Redattore

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 e Documenti esterni

Per evitare possibili ambiguità relative alle terminologie (che andranno indicate in MAIUSCOLETTO)utilizzate nei vari documenti, verranno utilizzate due simboli:

- Una *D* al pedice per indicare il nome di un particolare documento.
- Una *G* al pedice per indicare un termine che sarà presente nel GLOSSARIO v0.0.1$_D$.

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

2.1 Capitolato 6 - ThiReMa

2.1.1 Informazioni generali

- **Nome:** ThiReMa - Things Relationship Management
- **Proponente:** Sanmarco Informatica

2.1.2 Descrizione capitolato

Sviluppo di un software che, dopo aver ricevuto misurazioni da sensori eterogenei, li accumola in un database centralizzato. Questa applicazione viene poi completata da un servizio di dispatching per inoltrare in modo tempestivo le informazioni utili per gestire le azioni urgenti. I dati messi a disposizione dal database centralizzato dovranno essere suddivisi in due macro-categorie: dati operativi e fattori influenzanti.

2.1.3 Finalità del progetto

Creare una WEB-APPLICATION$_G$, che permetta di valutare la correlazione tra dati operativi (misure) e i fattori influenzanti. Tale applicazione si potrà focalizzare nella definizione di uno o più algoritmi per la successiva analisi dei dati al fine di essere in grado di effettuare delle previsioni sull'andamento dei dati stessi ed offrire, ad esempio, dei servizi di MANUTENZIONE PREDITTIVA$_G$. Per ogni tipologia di informazioni rilevate dovrà anche essere possibile assegnare il monitoraggio ad un particolare ente. Analizzando un determinato sensore, in base ai dati ricevuti, si può prevedere un deterioramento complessivo tale da generare una necessaria azione di manutenzione preventiva. La web-application dovrà essere suddivisa in 3 macro-sezioni:

- Censimento dei sensori e dei relativi dati;
- Modulo di analisi di correlazione;
- Modulo di monitoraggio per ente.

2.1.4 Tecnologie interessate

- **APACHE KAFKA$_G$:** Il progetto mira a creare una piattaforma a bassa latenza ed alta velocità per la gestione di feed dati in tempo reale;
- **API PRODUCER$_G$, CONSUMER$_G$, CONNECT$_G$ e STREAMS$_G$:** API$_G$ consigliate per la produzione di componenti custom per Kafka;
- **POSTGRESQL$_G$, TIMESCALEDB$_G$, ClickHouse:** Implementazioni database suggerite per contenere i dati relativi alle misurazioni, agli utenti e le loro informazioni di autorizzazione;

- **DOCKER$_G$:** Tecnologia di containerizzazione che consente la creazione e l'utilizzo di container Linux. Nel progetto risulterebbe utile per l'instanziazione di tutti i componenti;
- **Java:** Linguaggio di programmazione ad alto livello orientato agli oggetti e a tipizzazione statica;
- **Bootstrap:** Raccolta di strumenti open source per la creazione di siti e applicazioni per il Web.

2.1.5 Aspetti positivi

- Tecnologie già in parte conosciute dal gruppo con la possibilità di ampliarne le conoscenze;
- L'azienda mette a disposizione figure di diverso livello per rispondere alle varie esigenze del gruppo e per facilitare la creazione di ambienti di sviluppo e test;
- Consistente set di dati su cui testare l'applicativo;
- Interfacciarsi con l'harware tramite API;

2.1.6 Criticità

- Protocolli proprietari, la documentazione su di essi potrebbe essere limitata;
- Il capitolato richiede un'analisi dei dati più avanzata rispetto agli altri.

2.1.7 Conclusioni

Il capitolato ha suscitato l'interesse del gruppo, dando la possibilità di ampliare tecnologie già in parte conosciute ed al contempo molto attuali, quali IoT e Big Data. C'è stato inoltre molto entusiasmo per la tipologia di web-application da sviluppare. Nell'insieme questo capitolato è stato accolto con forte interesse da tutti i componenti del gruppo.

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$_G$ è creare una PIATTAFORMA WEB$_G$ 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$_G$, dovrà essere dotata di un MODELLO DI MACHINE LEARNING$_G$ 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$_G$ ed in particolare:

- **ELASTIC CONTAINER SERVICE$_G$ O ELASTIC KUBERNETES SERVICE$_G$:** è un servizio che permette la gestione di contenitori, altamente dimensionabile e ad elevate prestazioni;
- **DYNAMODB$_G$:** è un database non relazionale per applicazioni che necessitano di prestazioni elevate su qualsiasi scala.
- **AWS TRANSCODE$_G$:** è un servizio di transcodifica di contenuti multimediali nel CLOUD$_G$;

- **SAGE MAKER$_G$:** è un servizio completamente gestito che permette a sviluppatori e data scientist di creare, addestrare e distribuire modelli di apprendimento automatico;
- **AWS REKOGNITION VIDEO$_G$:** è un servizio di analisi video basato su APPRENDIMENTO APPROFONDITO$_G$.

3.1.5 Linguaggi di programmazione

- **NODEJS$_G$:** linguaggio ideale per sviluppare API RESTFUL$_G$ JSON a supporto dell'applicativo;
- **PYTHON$_G$:** 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$_G$ responsive come Twitter.

Vincoli del progetto:

- Utilizzo di SAGE MAKER$_G$;
- L'architettura dovrà essere basata su MICRO-SERVIZI$_G$, 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, è stato accolto con molto interesse dal gruppo, che si è dimostrato interessato ad approfondire l'argomento.
- Il proponente fornisce attività di formazione sulle principali tecnologie AWS e WIREFRAME$_G$ 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$_G$ per effettuare il training dell'algoritmo;
- Buona parte del tempo verrebbe utilizzato per lavori ripetitivi e che non stimolerebbero il gruppo.

3.1.8 Conclusione

Dopo aver valutato gli aspetti positivi e le criticità, si è deciso che questo capitolato, nonostante tocchi delle tecnologie piuttosto interessanti, richiede, secondo il gruppo, un'eccessiva mole di lavoro ripetitivo.

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$_G$ che permette, agli sviluppatori di fare il DEPLOY$_G$ di funzioni JavaScript, mentre agli utenti finali di pagare per l'esecuzione delle suddette funzioni (si basa su CAAS$_G$, 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$_G$ 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$_G$ 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$_G$ 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 emetterà un evento nella BLOCKCHAIN$_G$ 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$_G$ e SERVERLESS$_G$, più precisamente utilizza:

- **Ethereum:** è una piattaforma che permette ai suoi utenti di scrivere applicazioni decentralizzate (dette ĐAPPS$_G$) che usano la tecnologia BLOCKCHAIN$_G$;
- **Ethereum Virtual Machine (EVM):** è una MACCHINA VIRTUALE DECENTRALIZZATA$_G$ 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$_G$ o ETH), dati o una combinazione di entrambi;
- **Gas:** è il carburante che permette alla EVM$_G$ di eseguire un programma;
- **MAINNET$_G$, ROPSTEN$_G$:** sono reti sulle quali vengono eseguiti i protocolli di Ethereum.
- **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$_G$, Backend-as-a-Service);
- **AWS LAMBDA$_G$:** 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$_G$, AWS DYNAMODB$_G$, AWS S3$_G$:** 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 STATICO$_G$ per 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$_G$, 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

- Dato l'interesse riscosso dalla tecnologia BLOCKCHAIN$_G$, 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 dopo un'attenta analisi non ha riscosso molto interesse.

3.2.8 Conclusione

Dopo aver valutato attentamente il capitolato, il gruppo ha deciso di orientarsi verso un altro capitolato.

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$_G$.

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$_G$ 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$_G$, 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 dell'interfaccia 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 di 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);
- Gli strati 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

- Il capitolato affronta un problema molto importante, quale il confronto tra persone abituate ad esprimere gli stessi concetti in modi diversi, cercando di trovare una soluzione.

3.3.7 Criticità

- Il capitolato ha riscosso scarso interesse nella maggior parte dei membri del gruppo;
- Il progetto implicato dal capitolato non sembra particolarmente concreto.

3.3.8 Conclusione

Nonostante il capitolato C3 cerchi di risolvere un problema molto sentito, la mancanza di concretezza del progetto stesso e lo scarso interesse mostrato per quest'ultimo, ha portato il gruppo ad orientarsi verso un'altra scelta.

3.4 Capitolato 4 - Predire in Grafana

3.4.1 Informazioni generali

- **Nome:** Predire in Grafana - Monitoraggio predittivo per DevOps
- **Proponente:** Zucchetti SPA

3.4.2 Descrizione capitolato

La società Zucchetti, per eseguire il monitoraggio dei propri sistemi, ha scelto Grafana, un prodotto open source estendibile tramite plug-in. Il capitolato chiede la realizzazione di un PLUG-IN$_G$, da integrare a GRAFANA$_G$ stessa, che effettuino delle previsioni sul flusso dei dati raccolti al fine di monitorare il corretto funzionamento del sistema e consigliare eventuali interventi alla linea di produzione del software tramite allarmi o segnalazioni.

3.4.3 Finalità del progetto

É richiesto lo sviluppo di due PLUG-IN$_G$ per Grafana che siano in grado di predire un possibile stato dell'imminente futuro della macchina. La predizioni devo essere effettuate tramite REGRESSIONE LINEARE$_G$ o SUPPORT VECTOR MACHINE$_G$. La Regressione Lineare consente di analizzare dati uniformi ed é molto utile se i dati hanno un dominio continuo e una dispersione uniforme (omoschedastici) con andamento lineare. Per avere una predizione valida si deve guardare la dispersione del grafo dei residui, nel quale una dspersione uniforme é indice di una LR corretta ed attendibile. I SVM sono algoritmi di classificazione per dati con dominio discreto e sono scalabili su spazi di dati molto grandi. Essi consistono nell'effettuare trasformazioni dello spazio dati per trovare un predizione che sia in grado di separare i dati in varie classi. L'addestramento degli algoritmi di machine learning viene effettuato in un ambiente separato ed isolato da grafana tramite suit di generazione dati come JMETER$_G$ che consente di creare dati compatibili. A fine addestramento i predittori necessitano di essere salvati in file JSON per poi essere associati al flusso dati di grafana. Una volta ottenute le previsioni sui dati essi dovranno essere resi disponibili al sistema di creazione di grafici di Grafana ed essere visualizzati sulla dashboard.

Come obbiettivi opzionali viene richiesto di dare la possibilità di definire "alert" in base a soglie raggiunte dalle previsioni; di dare la possibilità di effettuare trasformazioni sulle misure i modo da ottenere regressioni non lineare (es. logaritmiche o esponenziali); di consentire l'apprendimento constante tramite i dati raccolti dall'agente JMX. Viene richiesto, inoltre, di fornire anche i dati di bontà dei predittori. Per SVM sono "Precision", che consiste nel rapporto tra i dati veri predetti veri e i dati falsi predetti veri(veri trovati/falsi positivi); e "Recall", che il rapporto tra i veri valutati veri ed i veri non valutati tali; invece per LR si fà riferimento al coefficente "R^2" che rappresenta il rapporto tra gli errori rispetto alla retta e gli errori rispetto alla media del codominio dei dati.

3.4.4 Tecnologie interessate

- **SUPPORT VECTOR MACHINES$_G$ (SVM):** modello di classificazione basata su trasformazioni dello spazio dei dati;
- **REGRESSIONE LINEARE$_G$ (RL):** modello di predizione basata sulla differenza dei quadrati che genera una predizione lineare del tipo $y = ax + b$;
- **JMX$_G$:** agente installato sulle macchine che raccoglie i dati necessari alle predizioni;
- **JMETER$_G$:** suit di generazione di dati virtuali per l'addestramento;
- **JavaScript:** linguaggio non tipizzato orientato ad oggetti sia client side (JS, JQuery) che server side (AJAX, Node.JS).

3.4.5 Aspetti positivi

- L'azienda, consapevole del fatto che gli algoritmi di Machine Learning non fanno parte del corso di studi della laurea triennale, è disponibile alla formazione ed alla fornitura di questi tipi di algoritmi;
- I requisiti obbligatori del capitolato sono basilari dando la possibilità di ampliare notevolmente il progetto con i requisiti opzionali;
- Uso di un linguaggio flessibile, all'avanguardia e richiesto sul mercato.

3.4.6 Criticità

- Integrare un sistema già esistente non ha stimolato grosso interesse nel gruppo;
- Necessita uno studio approfondito della documentazione di vare suit e programmi per poter avere un'idea più concreta del progetto.

3.4.7 Conclusioni

Il capitolato ha stimolato un discreto interesse nel gruppo dal momento che si usano tecnologie importanti e consente di acquisire conoscenze spendibili nel mondo reale. L'idea di integrare un sistema già esistente però ha convinto il gruppo ad orientarsi su un altro capitolato.

3.5 Capitolato 5 - Stalker

3.5.1 Informazioni generali

- **Nome:** Stalker
- **Proponente:** Imola Informatica

3.5.2 Descrizione capitolato

Il proponente chiede la realizzazione di una MOBILE-APPLICATION$_G$ al fine di poter tracciare, in forma anonima e non, il numero esatto di persone presenti all'interno di uno spazio fisico indentificato da un insieme di coordinate geografiche.

3.5.3 Finalità del progetto

L'obiettivo è quello di sviluppare un'applicazione in grado di segnalare, ad un server dedicato, l'ingresso e l'uscita dell'utilizzatore dalle aree d'interesse (basandosi sulla posizione attuale dell'utilizzatore del dispositivo) in due modalità, autenticata o anonima, a seconda delle esigenze. L'applicazione deve permettere le seguenti operazioni:

- Recupero lista organizzazioni (Refresh manuale e/o temporizzato);
- Login nell'organizzazione con eventuale autenticazione;
- Storico degli accessi;
- Visualizzazione in tempo reale della propria presenza o meno all'interno di un luogo monitorato e cronometro del tempo trascorso al suo interno;
- Predisposizione di un pulsante "anonima" che permetta di risultare presente in maniera anonima all'interno dell'organizzazione;

Le comunicazioni tra applicazione cellulare e server dovranno avvenire solo nel momento d'ingresso ed uscita dai luoghi designati. Il rilevamento della posizione può essere effettuato in due modi distinti:

- **DEAD RECOKING$_G$:** dato un punto di partenza, la velocità, la direzione del movimento, il tempo trascorso e la distanza percorsa si può comprendere il punto di arrivo;
- **PROXIMITY SENSING$_G$:** la posizione del punto mobile è ricavata dalle coordinate di determinate stazioni che tracciano il segnale che viene trasmesso da esse (cell ID). Ogni stazione ha un suo pattern di segnale.

In genere con il GPS$_G$ usano la trilaterazione, usando la posizione nota di due o più punti di riferimento e la distanza misurata tra il punto mobile ciascun punto di riferimento. La triangolazione permette di calcolare la posizione sulla base di angoli di arrivo (AOA) tra punto mobile e punti di riferimento e

la distanza stessa tra i punti di riferimento. Ovviamente una precisione perfetta é difficile da raggiungere, perciò l'obbiettivo é un'approssimazione abbastanza precisa e dimostrabile della posizione. Oltre all'ottimizzazione del rilevamento viene richiesto di limitare il consumo energetico che il sistema può utilizzare in modo da estendere eventuali batterie presenti nel sistema.

3.5.4 Tecnologie interessate

Per lo sviluppo del server back-end sono consigliate le seguenti tecnologie:

- **Java:** Linguaggio di programmazione ad alto livello orientato agli oggetti e a tipizzazione statica;
- **Python:** Linguaggio di programmazione ad alto livello orientato agli oggetti adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing;
- **NodeJS:** è una runtime di JavaScript Open source multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript;
- **Continous Integration$_G$:** continua disponibilità del codice prodotto tra gli sviluppatori;
- **Continous Delivery$_G$:** gestione degli artefatti creati in produzione con relativi versionamenti;
- **Continous Testing$_G$:** il codice sviluppato viene testato,tramite test unitari, prima di essere rilasciato, per fornire una base solida per lo sviluppo da parte di altri sviluppatori interni, e consente di capire cosa é funzionante e cosa no. In generale si vuole una copertura del 85-90%.

Saranno inoltre necessari:

- Protocolli asincroni per le comunicazioni app mobile-server;
- **Pattern di Publisher/Subscriberii:** In questo pattern, mittenti e destinatari di messaggi dialogano attraverso un tramite, che può essere detto dispatcher o broker. Il mittente di un messaggio (detto publisher) non deve essere consapevole dell'identità dei destinatari (detti subscriber); esso si limita a "pubblicare" il proprio messaggio al dispatcher. I destinatari si rivolgono a loro volta al dispatcher "abbonandosi" alla ricezione di messaggi. Il dispatcher quindi inoltra ogni messaggio inviato da un publisher a tutti i subscriber interessati a quel messaggio;
- Utilizzo dell'IAAS Kubernetes o di un PAAS, Openshift o Rancher, per il rilascio delle componenti del Server nonché per la gestione della scalabilità orizzontale.

3.5.5 Aspetti positivi

- Possibilità di ampliare il bagaglio di tecnologie conosciute per lo sviluppo di applicazioni mobile;
- Il proponente non impone tecnologie specifiche per lo sviluppo del server o della UI.

3.5.6 Criticità

- Progetto lungo da sviluppare;
- Documentazione GPS complicata da interpretare;
- Difficile determinare con precisione la stima della posizione dell'utilizzatore.

3.5.7 Conclusioni

Il capitolato ha stimolato l'interesse del gruppo visto la tipologia di tecnologie che verranno utilizzate e per l'applicazione che l'azienda intende farne. Il proponente chiede di fare diversi test per ottenere una stima il più precisa possibile sull'utilizzatore, dimostrando i risultati ottenuti. Dopo un'accurata riflessione abbiamo constatato che la dimostrazione dei test poteva essere molto dispendiosa a livello di tempistiche. Nonostante l'interesse il gruppo ha deciso, dopo un conteggio delle preferenze, di non scegliere questo capitolato.

4 Conclusioni e motivazioni del capitolato scelto

Dopo una attenta analisi di tutti i seminari di approfondimento effettuati dai relativi proponenti, si è deciso di optare per un argomento che racchiudesse il maggiore grado di interesse di tutto il gruppo. Basandosi sulle proprie conoscenze, inoltre, si è cercato di determinare un capitolato adeguato che potesse produrre una applicazione concreta nella materia di interesse proposta, con l'obbiettivo di raggiungere una buona qualità per il prodotto finale da consegnare.

Inizialmente, abbiamo preso in ampia considerazione il capitolato C4 (*Predire in Grafana*), poiché richiede una analisi dei dati reperiti dalla server farm facendo uso di algoritmi di predizione e, soprattutto, si appoggia su un software di monitoraggio molto utilizzato e open source. Tuttavia, la poca competenza nella parte di machine learning e l'idea di dover sviluppare essenzialmente una piccola parte di un sistema molto più grande e complesso con un unico linguaggio di programmazione, ci ha fatto convergere verso un argomento correlato che riprende l'analisi dei dati, ma in modo più avvincente.
Infatti, si è voluto optare per il capitolato C6 (*ThiReMa*) che racchiude lo sviluppo di una web application in diretto contatto con il mondo IoT, attraversando le più moderne tecnologie (quali Docker e i database non relazionali) per giungere alla gestione remota di dispositivi fisici su larga scala.

L'idea che ha menzionato l'azienda proponente per questo capitolato, inoltre, ci ha particolarmente colpito: *nascondere la complessità e ricavare l'informazione dai dati (grezzi)*. Da questa idea si è stimolato il nostro interesse di poter approfondire queste pratiche di sviluppo per la gestione e interpretazione dei dati, tali per cui l'uso di un sistema automatizzato e ampiamente connesso permetterebbe di semplificare di gran lunga il lavoro di reperimento dell'informazione.
Chiaramente, la nostra sfida si baserà sull'apprendere il più possibile queste nuove tecnologie (come Kafka, Docker e i timeseries DB) così da realizzare un prodotto valido che soddisfi le aspettative e i requisiti richiesti dal capitolato, nonchè presentare tutta la documentazione aggiuntiva richiesta dal proponente prima e dopo lo sviluppo del software.
Concludendo, con la scelta questo capitolato si vuole ampliare il proprio bagaglio di conoscenza in questa materia, facendo luce sui modi di integrare le principali funzionalità del sistema, usando tecnologie moderne e di grande interesse per tutto il gruppo, senza trascurare gli aspetti formali del progetto.