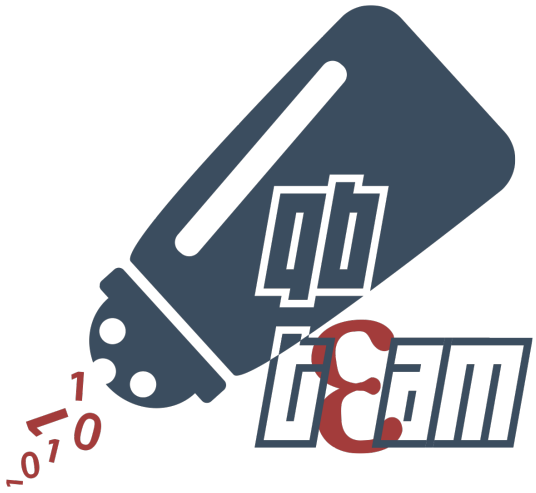

Studio Di Fattibilità

Gruppo qbteam - Progetto Stalker

qbteamswe@gmail.com

Informazioni sul documento

Versione	1.0.0
Approvatore	Azzalin Tommaso Salmaso Enrico
Redattori	Cisotto Emanuele Lazzaro Maria Davide Perin Federico Baratin Riccardo
Verificatori	Mattei Christian Drago Francesco
Uso	Interno
Distribuzione	Vardanega Tullio Cardin Riccardo Gruppo qbteam

Descrizione

Questo documento si occupa di descrivere l'analisi dei capitolati d'appalto realizzata dal gruppo al fine di valutarne la fattibilità

Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
1.0.0	14-12-2019	Azzalin Tommaso, Salmaso Enrico	Responsabili di Progetto	Approvazione del documento
0.6.4	10-12-2019	Mattei Christian, Drago Francesco	Verificatori	Creazione documento Latex, adattamento in Latex bozza documento e verifica output corretto
0.6.1	02-12-2019	qbteam	Analisti e Responsabili di Progetto	Discussione ed unione dei capitolati nella bozza documento
0.6.0	30-11-2019	Mattei Christian	Verificatore	Verifica capitolato C6
0.5.0	30-11-2019	Drago Francesco	Verificatore	Verifica capitolato C5
0.4.2	28-11-2019	Lazzaro Maria Davide	Analista	Scrittura capitolato C6
0.4.1	26-11-2019	Baratin Riccardo, Perin Federico	Analisti	Scrittura capitolato C5
0.4.0	26-11-2019	Mattei Christian	Verificatore	Verifica capitolato C4
0.3.0	25-11-2019	Drago Francesco	Verificatore	Verifica capitolato C3
0.2.1	25-11-2019	Cisotto Emanuele	Analista	Scrittura capitolato C4
0.2.0	23-11-2019	Mattei Christian	Verificatore	Verifica capitolato C2
0.1.0	23-11-2019	Drago Francesco	Verificatore	Verifica capitolato C1
0.0.4	21-11-2019	Lazzaro Maria Davide, Cisotto Emanuele	Analisti	Scrittura capitolato C3
0.0.3	21-11-2019	Baratin Riccardo, Azzalin Tommaso, Salmaso Enrico	Analista e Responsabili di progetto	Scrittura capitolato C2, controllo lavoro
0.0.2	21-11-2019	Perin Federico	Analista	Scrittura capitolato C1
0.0.1	20-11-2019	gbteam	Analisti	Creazione bozza documento, introduzione e paragrafi

Indice

1 Introduzione

1.1 Scopo del Documento

Questo documento contiene la stesura del studio di fattibilità riguardante i sei capitolati proposti, per ciascuno di essi verranno evidenziati i seguenti aspetti:

- Titolo del capitolato;
- Descrizione generale;
- Prerequisiti e tecnologie coinvolte;
- Vincoli;
- Aspetti positivi;
- Aspetti critici.

Infine per ogni capitolo verranno esposte le motivazioni e le ragioni per cui il gruppo ha scelto come progetto il capitolato C5 Stalker a discapito degli altri cinque capitolati proposti.

1.2 Glossario

Si è scelto di redigere un documento di supporto "Glossario" al fine di rendere chiaro il documento e di evitare ogni possibile ambiguità dovuta al linguaggio utilizzato. In questo documento vengono raccolti vari termini che possono essere incontrati durante la lettura dei vari documenti prodotti dal gruppo, e per ognuno dei termini inseriti nel glossari verrà esposta una definizione e una descrizione. Per facilitare la lettura i termini presenti nel glossario verranno indicati con il marcatore "G".

1.3 Riferimenti

1.3.1 Normativi

- Norme di Progetto 1.0.0.

1.3.2 Informativi

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

2 Capitolato C1

2.1 Titolo del capitolato

Il capitolato in questione si chiama "Autonomous Highlights Platform", il proponente è l'azienda zero12 e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.2 Descrizione del capitolo

Il capitolato presentato si pone come obiettivo finale di realizzare una piattaforma che, ricevendo in input dei video di eventi sportivi, generi in maniera automatica un breve video (di circa 5 minuti) con i soli momenti salienti (gli highlights, appunto). La piattaforma deve essere dotata di un modello di machine learning[G] per comprendere in autonomia quali sono i momenti salienti di un filmato e per raggiungere questo scopo, viene richiesto di concentrarsi su un unico sport. L'analisi e il controllo dell'elaborazione dei filmati deve essere gestibile tramite un'interfaccia web.

2.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- NodeJS[G];
- HTML5[G], CSS3[G] e Javascript[G].

Il capitolato richiede in gran parte l'utilizzo di tecnologie fornite da Amazon Web Services[G] (AWS). In particolare, di questo è richiesto l'uso di:

- Sage Maker: servizio che permette di compiere tutto il lavoro necessario per creare un modello di apprendimento automatico, dall'analisi dei dati alla sua formazione, fino al controllo delle previsioni e distribuzione;
- Rekognition Video: servizio che permette di analizzare filmati per comprendere le persone o gli oggetti presenti, i loro movimenti ed eventualmente i contenuti inappropriati;
- Transcode: servizio che permette una conversione efficiente di video in numerosi formati per la loro successiva distribuzione;
- DynamoDB: database NoSQL[G] ad elevate prestazioni da utilizzare per memorizzare i dati delle elaborazioni dei video e altre informazioni di supporto;
- Elastic Container Service (ECS) o Elastic Kubernetes Service (EKS): servizio che permette la gestione di numerosi container, ad alte prestazioni e ad alta scalabilità.

Inoltre, viene ritenuto opportuno l'utilizzo di linguaggi come Python[G] per lo sviluppo delle componenti che gestiscono il modello di apprendimento automatico e di JavaScript[G] per la creazione di REST[G] API[G] in Node.js[G]. Per la creazione dell'interfaccia web è consigliato l'utilizzo del framework Bootstrap[G], il quale richiede conoscenze pregresse di HTML5[G], CSS3[G] e JavaScript[G].

2.4 Vincoli

Sebbene l'azienda non richieda esplicitamente di utilizzare tutti i servizi appena descritti, seppur li raccomandi, ritiene obbligatorio l'utilizzo di AWS Sage Maker e l'implementazione di:

- un'architettura a micro-servizi: ovvero ogni componente deve poter essere compilata ed implementata autonomamente;
- un servizio di caricamento dei video da linea di comando;
- un'interfaccia web[G] per l'analisi e il controllo dell'elaborazione dei filmati.

Zero12 vorrebbe essere coinvolta dal team di progetto soprattutto durante la fase di analisi preliminare, ed offre attività di formazione sull'apprendimento automatico e se necessario, repository[G] git[G]. Infine, richiede di condividere:

- diagrammi UML[G] relativi agli use case di progetto;
- schema della base di dati a supporto del sistema;
- documentazione dettagliata di tutte le API[G];
- piano di test di unità.

2.5 Aspetti positivi

- È richiesto l'utilizzo di numerosi servizi forniti da AWS[G], alcuni più trasversali, altri meno, che potrebbero risultare molto utili in altri ambiti ed in futuro. I più trasversali possono essere ad esempio ECS, EKS, Dynamo DB;
- Nel capitolato si tratta l'argomento del machine learning[G], un argomento molto interessante e molto in voga negli ultimi anni;
- L'azienda risulta essere molto disponibile per seguire il gruppo nello sviluppo del progetto, e questo è senz'altro positivo per rimanere in linea con le richieste del capitolato.

2.6 Aspetti critici

- Il capitolato richiede l'apprendimento di molti servizi, sconosciuti dai membri del gruppo, che si basano su argomenti non conosciuti (come il machine learning[G]) e non affrontati nel corso di laurea triennale. Quindi gran parte del tempo andrebbe dedicato alla formazione per l'utilizzo di questi concetti e strumenti, ed infine di applicare queste conoscenze;
- Oltre al tempo impiegato per l'apprendimento di questi strumenti c'è da tenere in considerazione il tempo per cercare il materiale video per addestrare il modello visionarlo ed etichettarlo.

2.7 Conclusioni

Nonostante sia stato manifestato grande interesse da parte del gruppo e analizzando gli aspetti positivi e quelli critici, il gruppo ha ritenuto troppo elevata la mole di lavoro necessaria per svolgerlo e quindi il capitolo è stato rigettato.

3 Capitolato C2

3.1 Titolo del capitolato

Il capitolato in questione si chiama "Etherless", il proponente è l'azienda RedBabel e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

3.2 Descrizione del capitolo

Etherless è una piattaforma cloud[G] che permette agli sviluppatori di distribuire le proprie funzioni JavaScript[G] nel cloud[G] e fare pagare agli utenti finali l'utilizzo di queste (modello "Computation as a Service[G]", Caas), sfruttando gli smart contract Ethereum[G]. Etherless è gestita e mantenuta da utenti chiamati Administrators. Altri utenti chiamati Developers possono pubblicare in Etherless le loro funzioni. Gli utenti finali del prodotto sono chiamati Users e possono eseguire le funzioni presenti nella piattaforma pagando le quote stabilite dai Developers proprietari di quelle funzioni. Ogni quota pagata è in parte trattenuta da Etherless come compensazione per l'esecuzione (ricordarsi che la blockchain è una struttura distribuita, tutti i nodi eseguono le computazioni). L'obiettivo di Etherless è integrare due tecnologie: Ethereum[G] e Serverless[G]. Ethereum[G] si occuperà dei pagamenti e di scatenare (trigger) l'invocazione delle funzioni. Serverless[G] si occuperà invece dell'esecuzione delle funzioni. Una interfaccia a linea di comando (command line interface, CLI) farà da ponte fra users e developers. La comunicazione (virtuale) fra Ethereum[G] e Serverless[G] sarà ottenuta ascoltando ed emettendo eventi Ethereum[G].

3.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- avere un'idea dei contenuti del corso di Programmazione Concorrente e Distribuita[G] (programmazione asincrona, programmazione funzionale);
- conoscenza del linguaggio TypeScript (è un linguaggio molto simile a JavaScript[G], che va compilato e il prodotto della compilazione è un file JavaScript[G]);
- serve conoscere il Serverless[G] Framework che implica Node.js[G].

Per poter svolgere il capitolato è necessario avere chiaro il funzionamento delle seguenti tecnologie:

- Blockchain: è un database condiviso, chiamato anche ledger (registro). Ogni dispositivo collegato con una copia del ledger è chiamato nodo. Questi ledger tengono traccia della proprietà valutaria (criptovaluta[G]) dove ogni nodo può verificare i conti di chiunque. Questa appena descritta è la parte decentralizzata della blockchain. Le interazioni tra account in una rete blockchain sono chiamate transazioni. Possono essere transazioni monetarie oppure sono trasmissioni di dati. Ogni account sulla blockchain ha una firma unica, che consente a tutti di sapere quale account ha avviato la transazione.
- Ethereum[G]: è una piattaforma che permette all'utente di scrivere più facilmente delle applicazioni decentralizzate (Dapps[G]) che usano la tecnologia blockchain. L'Ethereum[G] blockchain può essere descritto come una blockchain con un linguaggio di programmazione integrato.
- Ethereum Virtual Machine (EVM): è la parte del protocollo che gestisce lo stato interno della rete e del calcolo. EVM può essere utilizzato come un grande computer decentralizzato contenente oggetti chiamati "accounts", ciascuno in grado di mantenere un database interno, eseguire il codice e comunicare. L'EVM consente al codice di essere verificato ed eseguito sulla blockchain. Questo codice è contenuto in "smart contracts", che vengono utilizzati dai Dapps[G] per l'elaborazione dei dati sull'EVM.
- Smart Contract[G]: è il codice che viene eseguito sull'EVM. Esso può accettare ed archiviare Ethereum, dati o la combinazione di entrambi distribuendoli ad altri account o persino ad altri smart contracts[G].
- Dapps[G]: le applicazioni che utilizzano contratti intelligenti per l'elaborazione sono denominate "applicazioni decentralizzate", e ognuna delle loro parti è individualmente in grado di fare il suo lavoro senza dipendere dalle altre.

- Gas: è il metodo di pagamento, pagato in criptovaluta[G] Ethereum[G], che viene calcolato ogni volta che l'EVM esegue uno smart contract[G]. Quindi ogni volta l'utente deve pagare per tale esecuzione, indipendentemente dal fatto che la transazione abbia esito positivo o negativo.
- Reti Ethereum[G]: la principale blockchain pubblica di Ethereum[G] si chiama MainNet[G], ma esistono altre reti, poiché chiunque può creare la propria rete Ethereum[G]. Su MainNet[G] i dati sono pubblici e chiunque può creare un nodo e iniziare a verificare le transazioni. Sono anche presenti reti di test locali, reti di test pubblici (ad esempio Roposten, che è quella ufficiale).
- Eventi Ethereum[G]: gli eventi ed i registri sono importanti in Ethereum[G] perché facilitano le comunicazioni tra gli smart contracts[G] e la loro interfaccia. Ci sono 3 principali cause dove eventi e registri possono essere usati:
  - un evento può essere il tipo di ritorno dello smart contract[G];
  - trigger asincroni[G] con dati (lo smart contract[G] genera un evento in modo da attivare la UI[G] che ascolta eventi);
  - una forma di archiviazione.
- Architetture Serverless[G]: sono applicazioni di progettazione che includono servizi "Backend as a Service[G]" (BaaS) di terze parti, codice personalizzato e container temporanei su una piattaforma di esecuzione "Function as a Service[G]" (FaaS). Le architetture Serverless[G] potrebbero trarre beneficio dalla riduzione dei costi delle operazioni, complessità e tempi di consegna. Essi sono sistemi basati su cloud[G] di eventi, dove lo sviluppo di applicazioni è basato esclusivamente su una combinazione di servizi di terze parti, logica lato client e chiamate a procedure presenti nel cloud[G].
- Serverless[G] computing: è un modello cloud dell'esecuzione di calcoli nella quale il fornitore cloud[G] avvia il server e dinamicamente gestisce le allocazioni delle risorse informatiche richieste. Inoltre, devono essere utilizzati i seguenti framework e linguaggi:
  - TypeScript 3.6: linguaggio che supporta JavaScript[G] ES6 in tutto e per tutto, aggiungendo funzionalità, in particolare la tipizzazione;
  - typescript-eslint: strumento che permette di verificare che il codice TypeScript scritto rispetti correttamente la sintassi di linguaggio;
  - Solidity[G]: linguaggio per implementare gli smart contract;
  - Serverless[G]: framework per la realizzazione di applicazioni che verranno poi eseguite in architetture serverless[G] (come AWS Lambda, Google Cloud Functions, Microsoft Azure Functions).

3.4 Vincoli

- smart contract[G] aggiornabili;
- utilizzo di TypeScript 3.6, in particolare dei costrutti Promise e async/await;
- utilizzo di typescript-eslint;
- etherless-server deve utilizzare il Serverless[G] Framework;
- il codice deve essere pubblicato e versionato su GitHub[G] o GitLab[G], deve essere pubblicata, assieme al sorgente, la documentazione per gli users e per i developers.

3.5 Aspetti positivi

- È sicuramente interessante il fatto di unire in un unico progetto due delle tecnologie più discusse nell'ultimo periodo come blockchain e serverless[G] computing;
- Nel capitolato viene tratto l'argomento dei pagamenti elettronici che nei ultimi tempi risulta essere molto discusso;

- Poter imparare linguaggi come TypeScript e JavaScript[G] e rafforzando l'uso corretto mediante typescript-eslint è formativo. Inoltre, essendo questi due linguaggi alla base di moltissimi altri framework esistenti, non sarebbero conoscenze fini a sé stesse;
- Poter sperimentare i concetti di calcolo distribuito e le potenzialità offerte dai linguaggi funzionali imparati nel corso di Programmazione Concorrente e Distribuita[G] (seppur visti in Java).

3.6 Aspetti critici

- I concetti da imparare sono molteplici e di fatto i membri del gruppo sono completamente all'oscuro delle tecnologie richieste (eccetto per la blockchain) quindi gran parte del tempo andrebbe impiegato nell'apprendimento di queste;
- Solidity[G] è un linguaggio relativamente nuovo ed è ancora molto soggetto a variazioni da una versione all'altra, che potrebbero avvenire anche durante il corso della realizzazione del progetto;
- L'argomento riguardante il pagamento attraverso criptomonete come Etherium[G] nonostante sia di molto interesse per via della tecnologia usata (blockchain), risulta essere mal visto dai principali enti bancari;
- Essendo un'azienda residente all'estero, la comunicazione potrebbe rivelarsi più problematica.

3.7 Conclusioni

Si riconosce il fatto che nel capitolato vengono trattati argomenti di grande interesse per la società moderna come Blockchain, criptomonete e pagamenti elettronici ciò nonostante per via del grande numero di nuove tecnologie da imparare e che essendo di recente sviluppo esse sono soggette a frequenti cambiamenti, analizzando gli aspetti positivi e quelli critici e dopo una discussione fra i membri del gruppo, il capitolato è stato scartato.

4 Capitolato C3

4.1 Titolo del capitolato

Il capitolato in questione si chiama "Natural API", il proponente è l'azienda teal.blue e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

4.2 Descrizione del capitolo

Considerando i vari personaggi che prendono parte allo sviluppo di un progetto (clienti, project managers, developers, ecc.), è chiaro che non tutti si esprimano nello stesso linguaggio. Difficoltà nel descrivere l'idea che si vuole trasmettere può ripercuotersi sull'efficienza[G], l'efficacia[G] e il costo per sviluppare il progetto. Natural API[G] mira a sviluppare un toolkit[G] in grado di colmare la distanza tra le specifiche di progetto e le API. In pratica è composta da tre processi distinti: il primo (Discover) deve estrarre automaticamente una lista di predicati, nomi e verbi da file testuali dati in input (per esempio guide, manuali ecc.), creando il Business Domain Language (BDL). Il secondo (Design) deve combinare le specifiche degli stakeholders in gherkin e il BDL per creare il business application language (BAL), ovvero lo step intermedio tra linguaggio naturale e codice. Il BAL deve poter integrare requisiti addizionali e poter fondere o dividere azioni ed espandere nomi in oggetti. Il terzo (Develop) invece tramuta in codice il BAL in un linguaggio di programmazione concreto. Deve essere interattivo in modo da specificare dei dettagli del linguaggio di programmazione. Il capitolato pone quindi come obiettivo l'implementazione di un sistema in grado di trasformare specifiche definite in linguaggio naturale in codice di un reale linguaggio di programmazione.

4.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- gherkin[G]: `https://cucumber.io/docs/gherkin/`
- natural language processing[G]: `https://en.wikipedia.org/wikiNatural_language_processing`
- API[G] generation: `https://github.com/OAI/OpenAPI-Specification`
- code generation: `https://swagger.io/`
- behavior-driven development[G]: `https://dannorth.net/introducing-bdd/`
- cucumber[G]: `https://cucumber.io/docs`
- clean architecture: `https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html`

Tecnologie consigliate:

- Qt[G];
- Python[G];
- React[G];
- cucumber[G].

4.4 Vincoli

Sono imposti i seguenti vincoli:

- Ogni parte dell'applicazione deve soddisfare degli standard sull'output che produce almeno due modi di interagire con l'applicazione tra i seguenti:
  - linea di comando;
  - interfaccia grafica[G];

  - interfaccia web[G].

- Rispettare delle specifiche sui logic layer[G]. Si incoraggia l'applicazione del Joel Test I logic layers dovranno essere rilasciati in uno dei seguenti metodi:
  - come una libreria (statica o dinamica);
  - come parte di un eseguibile;
  - come un processo/servizio indipendente, in locale o remoto.
- Per la delivery mode sarà sufficiente l'accessibilità da almeno un sistema operativo tra:
  - Linux;
  - Window;
  - OS X.
- input e output in codifica UTF-8[G];
- licenza Open Source;
- Il codice del progetto dovrà essere locato in un repository[G] facilmente accessibile dal pubblico (ad esempio GitHub[G]).

4.5 Aspetti positivi

È di ovvia utilità un sistema che sia in grado di soddisfare le esigenze sopracitate, poiché sarebbe applicabile in ogni azienda dove si sviluppi software.

4.6 Aspetti critici

- Risulta poco chiaro il passaggio dal BDL al BAL;
- L'obiettivo finale del capitolato è prodotto per le aziende e il linguaggio utilizzato (gherkin) è di nicchia e la conversazione tra stakeholder e UML[G] sono ancora lo standard.

4.7 Conclusioni

L'argomento trattato dal capitolato è risultato essere poco interessante secondo il gruppo, questo perché viene richiesto l'utilizzo di tecnologie poco conosciuta e di nicchia è il prodotto finale risulterebbe poco soddisfacente per il gruppo dopo il lungo lavoro richiesto per realizzarlo. Analizzando perciò gli aspetti positivi e quelli critici non è emerso un forte interesse per il capitolato.

5 Capitolato C4

5.1 Titolo del capitolato

Il capitolato in questione si chiama "Predire in Grafana", il proponente è l'azienda Zucchetti e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

5.2 Descrizione del capitolo

Si vuole realizzare un plug-in[G] per lo strumento di monitoraggio Grafana[G]. Il plug-in[G] deve calcolare una predizione con l'applicazione del Support Vector Machine[G] (SVM) e della regressione Lineare[G] (RL) al flusso di dati ricevuti da una data sorgente. La predizione sarà utile per decretare quando sia il caso di mandare dei segnali id allarme ai manutentori/sistemisti dei server (o più in generale delle apparecchiature a rischio di sovraccarico) per agire preventivamente.

5.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- Cenni di calcolo numerico (RL[G] e SVM[G]);
- JavaScript[G] per la realizzazione dei plug-in[G];
- Algoritmi di Machine Learning[G] (l'azienda è disposta a fornire la formazione necessaria a riguardo).

Tecnologie da coinvolte:

- JavaScript[G], linguaggio di programmazione richiesto per costruire i plug-in di Grafana[G] ;
- Libreria in JavaScript[G] per le SVM[G];
- Libreria in JavaScript[G] per la RL[G];
- Libreria in JavaScript[G] per le reti neurali;
- Prodotto di monitoraggio Grafana[G], software open-source che, ricevuti i dati in input, consente di raccoglierli in un cruscotto, visualizzarli, analizzarli, misurarli e controllarli;
- Orange Canvas[G], strumento per l'analisi dei dati.

5.4 Vincoli

- La realizzazione dei plug-i[G]n deve essere fatta attraverso JavaScript[G];
- il programma dovrà svolgere le seguenti compiti:
  - Produrre un file json dai dati di addestramento con i parametri per le previsioni con Support Vector
  - Machine (SVM[G]) per le classificazioni o la Regressione Lineare[G] (RL o LR da Linear Regression in inglese)
  - Leggere la definizione del predittore dal file in formato json[G].
  - Associare i predittori letti dal file json al flusso di dati presente in Grafana[G].
  - Applicare la previsione e fornire i nuovi dati ottenuti dalla previsione al sistema di Grafana.
  - Rendere disponibili i dati al sistema di creazione di grafici e dashboard[G] per la loro visualizzazione.

5.5 Aspetti positivi

- La capacità di analizzare i dati e fornire previsioni sono skill ampiamente richiesta tra le varie aziende;
- Viene lasciata abbastanza libertà al gruppo nella scelta delle tecnologie da utilizzare per la realizzazione del prodotto richiesto, ciò nonostante l'azienda Zucchetti suggerisce che tecnologie adottare;
- L'azienda si presenta come la prima software house italiana, per storia e dimensione;
- Poche tecnologie nuove da integrare tra conoscenze per sviluppare il prodotto richiesto;
- L'azienda tra tutte le aziende coinvolte nei capitolati proposti risulta essere la più vicina tra i vari membri del gruppo, infatti tra le varie sedi presenti, l'azienda risulta essere presente sia a Padova sia a Treviso.

5.6 Aspetti critici

- Non è chiaro come le librerie fornite debbano essere implementate e con che flessibilità possa analizzare diversi tipi di dati in input;
- La presentazione del capitolato è stata molto approssimativa;
- L'utilizzo di conoscenze di apprese durante il corso di calcolo numerico (RL^{G} e SVM^{G}) risulta essere poco stimolante.

5.7 Conclusioni

Nel capitolato viene trattato l'argomento del machine learning^{G}, purtroppo marginalmente e l'utilizzo del software Grafana non ha stimolato grande interesse all'interno del gruppo, perciò analizzando gli aspetti positivi e quelli critici e dopo una discussione fra i membri del gruppo, è emerso un discreto interesse per il capitolato ciò nonostante è stato preferito accantonare il capitolato .

6 Capitolato C5

6.1 Titolo del capitolato

Il capitolato in questione si chiama "Stalker", il proponente è l'azienda Imola Informatica e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

6.2 Descrizione del capitolo

Stalker richiede la realizzazione di un sistema di monitoraggio delle posizioni delle persone (in versione anonima o non) grazie ad un'applicazione "complementare" installata sugli smartphone delle persone interessate. La necessità del servizio sorge dal dover monitorare il numero e la posizione delle persone all'interno di fiere/musei/locali ai fini di sicurezza oppure per verificare la presenza e la locazione dei dipendenti di un'azienda. Nel primo caso si userà il tracciamento anonimo, nel secondo caso invece il tracciamento non anonimo. L'applicazione dovrà risalire alla posizione attuale grazie all'ausilio di infrastrutture e/o servizi resi disponibili dallo smartphone stesso, come per esempio beacons[G], GPS[G], dati del cellulare, ecc. Quando l'applicazione calcola la posizione aggiornata deve anche verificare se rientra in una zona da essere tracciata, in tal caso comunicherà col server remoto per autenticarsi (anonimamente o non) e risultare tracciabile dal sistema.

6.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- Java.

Tecnologie coinvolte:

- Utilizzo di Java (versione 8 o superiori), python[G] o nodejs[G] per lo sviluppo del server back-end[G];
- Utilizzo di protocolli asincroni per le comunicazioni app mobile-server;
- Utilizzo del pattern di Publisher/Subscriber, ovvero mittenti(publisher) e destinatari (Subscriber) di messaggi dialogano attraverso un tramite(dispatcher[G]);
- Utilizzo dell'IAAS Kubernetes[G] o di un PAAS[G], Openshift[G] o Rancher[G], per il rilascio delle componenti del Server nonché per la gestione della scalabilità orizzontale[G];
- API[G] Rest[G] attraverso le quali sia possibile utilizzare l'applicativo;
- Utilizzo del GPS[G] o altre soluzioni per monitorare un utente;
- Utilizzo di un'IDE[G] per la creazione di applicazioni mobile (Android o iOS);
- Utilizzo di LDAP[G] (Lightweight Directory Access Protocol) che è un protocollo standard per l'interrogazione e la modifica dei servizi di directory, come ad esempio un elenco aziendale di e-mail o una rubrica telefonica, o più in generale qualsiasi raggruppamento di informazioni che può essere espresso come record di dati organizzato in modo gerarchico.

6.4 Vincoli

- Il server deve essere in grado di scalare in base al numero di utilizzatori in modo dinamico sia in aggiunta che in riduzione;
- Viene richiesto di garantire una precisione sufficiente che permetta di certificare la presenza della persona all'interno degli edifici;
- Effettuare test di tipo end-to-end[G];
- Creazione di un'applicazione Android/iOS con relativa interfaccia grafica.

6.5 Aspetti positivi

- Il prodotto richiesto risulta essere accattivante ed utile poiché ci sono applicazioni concrete su vasta gamma di contesti;
- Essendo Android molto diffuso la documentazione necessaria per realizzare l'applicazione è di immediata disponibilità;
- Utilizzo di java che risulta essere una tecnologia conosciuta da tutti i membri del team;
- L'azienda ha esposto in modo chiaro i vari vincoli, i vari casi d'uso presenti nel capitolato, e inoltre fornisce una sorta di glossario su alcuni termini utilizzati all'interno del documento di presentazione del capitolato;
- L'azienda è disponibile a fornirci diversi strumenti per il testing e a tenere lezioni per spiegarne il funzionamento e l'utilizzo.

6.6 Aspetti critici

- Essendo su un dispositivo mobile il servizio dell'applicazione che si connette al server per inviare la propria posizione deve essere molto efficiente in termine di consumo della batteria;
- Non sono chiari alcuni aspetti della modalità anonima[G];
- Bisogna rispettare le normative vigenti in tema privacy.

6.7 Conclusioni

La proposta del capitolato offerto dall'azienda Imola Informatica è stata accolta con grande interesse. Il gruppo è rimasto colpito e stimolato dalla possibilità di poter creare un prodotto che possa essere impiegato in molte realtà sia aziendali sia di eventi di varia natura e dimensione. Nonostante la tecnologia Android esista da molti anni è risultata particolarmente interessante da parte del gruppo questo perché risulta essere molto supportata da un'ampia community di sviluppatori e risulta essere per il gruppo una tecnologia nuova dato che non viene tratta da nessun corso della laurea triennale. Dopo l'analisi del capitolato è emersa una unanime preferenza per il capitolato Stalker.

7 Capitolato C6

7.1 Titolo del capitolato

Il capitolato in questione si chiama "ThiReMa", il proponente è l'azienda Sanmarco Informatica e i committenti sono Prof. Tullio Vardanega e Prof. Riccardo Cardin.

7.2 Descrizione del capitolo

L'azienda è dotata di un software che raccoglie dati da numerosi sensori eterogenei, di diversi tipi di apparecchi e dislocati geograficamente, in un database centralizzato. Questi dati vengono analizzati e vengono inoltrati, se necessario, messaggi di avviso per poter gestire gli apparecchi. Questi dati si suddividono in due categorie: dati operativi (provenienti dai sensori degli apparecchi) e fattori influenzanti (per esempio la temperatura o l'umidità). L'obiettivo è realizzare una applicazione web che permetta di mettere in relazione i dati operativi con i fattori influenzanti, di analizzarli con uno o più algoritmi per poter predire l'andamento di questi ed offrire alcuni servizi, come ad esempio una manutenzione preventiva in caso di previsione di guasti. Inoltre, deve essere permesso a certi enti di poter monitorare i propri dati utili per un upselling tecnologico.

7.3 Prerequisiti e tecnologie coinvolte

Prerequisiti:

- Java;
- data base;
- web-application.

Tecnologie coinvolte:

- Apache Kafka[G]: il cluster da cui provengono i dati da analizzare;
- Java 8: è consigliato il suo utilizzo, in particolare per l'utilizzo delle API[G] Producer, Consumer, Connect e Stream[G];
- Bootstrap[G]: framework per la realizzazione di interfacce web basato su HTML5[G], CSS3[G], JavaScript[G];
- Docker[G]: strumento per la creazione di container in cui istanziare i servizi per la gestione dell'architettura;
- TimescaleDB[G], Clickhouse[G] o PostgreSQL[G]: sono DBMS(Database Management System), che hanno caratteristiche diverse e usi diversi. Possono essere usati assieme per tenere dati di diversa natura separati oppure è possibile raggrupparli in un unico database.

7.4 Vincoli

L'azienda richiede che prima di iniziare lo sviluppo del progetto siano consegnati:

- i diagrammi UML[G] relativi agli use case di progetto;
- lo schema design relativo alla base dati;
- la documentazione delle API[G] che saranno realizzate.

Materiale da consegnare a corredo del progetto:

- Lista dei bug risolti durante le fasi di sviluppo;
- Codice prodotto in formato sorgente utilizzando sistemi di versionamento del codice, quali Github[G] o Bitbucket[G].

L'aspettativa minima per la conclusione del progetto dovrà comprendere:

- Codice sorgente di quanto realizzato;
- Docker[G] file con la componente applicativa, se utilizzato Docker[G].

7.5 Aspetti positivi

- Il collante fra le varie tecnologie e servizi è Java 8, conosciuto da tutti i membri del gruppo;
- L'azienda si è mostrata molto disponibile al fornire materiale per il testing e figure professionali per il supporto agli studenti che operano per la realizzazione del prodotto commissionato;
- Vengono acquisite competenze in ambito Internet of Things[G] e Big Data[G].

7.6 Aspetti critici

- Le tecnologie richieste per la realizzazione del prodotto finale, risultano essere abbastanza conosciute e già tratta nei vari corsi della laurea triennale inoltre vengono richieste poche nuove tecnologie da apprendere;
- Limitati posti per aggiudicarsi il capitolato.

7.7 Conclusioni

Il gruppo si è mostrato abbastanza interessato da ciò che è stato proposto dal capitolato ma dopo una discussione interna tra i membri pochi del team e emersa l'insoddisfazione da parte del team per via delle poche tecnologie nuove da apprendere e inoltre a causa del limitato numero di posti disponibili e la molta richiesta si è deciso infine di scartare questa proposta.