UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

TITOLO IN ITALIANO

TITLE IN ENGLISH

YURI BACCIARINI

NOME DEL RELATORE

Anno Accademico 2016-2017

INDICE

ELENCO DELLE TABELLE

ELENCO DELLE FIGURE

1

ESERVANT

1.1 PANORAMICA

eServant è un progetto di ricerca cofinanziato con fondi POR-CReO FESR 2014 - 2020 - Bandi per aiuti agli investimenti in ricerca, sviluppo e innovazione RSI. QUID Informatica S.p.A è Capofila di un raggruppamento di Aziende ed Organismi di ricerca, finalizzato alla realizzazione del progetto.

Il focus del progetto è la gestione innovativa dei processi collegati ai grandi eventi (musicali, sportivi, del tempo libero o del lavoro) che hanno luogo all'interno di impianti e strutture, sia in termini di comunicazione e controllo, sia in termini di servizi innovativi per gli spettatori/utenti, andando a disegnare un progetto di impianto intelligente, inserito all'interno di una struttura di città intelligente, costruita per un cittadino consapevole, che valorizzi i temi della responsabilità sociale e civile e della sostenibilità.

Si tratta quindi di un insieme di processi, che sono gestiti in modo assolutamente innovativo, sino a creare un processo evoluto, costituito dalla piattaforma eServant, che viene industrializzato in una struttura hardware e software altamente replicabile.

L'obiettivo operativo è quindi quello di progettare un modello di centro-servizi per la produzione, il controllo e la distribuzione di contenuti all'interno di grandi impianti, sportivi e non, in occasione di eventi specifici.

Il concetto di impianto al quale presta i suoi servizi il progetto eServant è molto ampio. Ci sono nel settore dello sport in Italia circa 150.000 impianti sportivi, di tutte le misure e taglie. Ci sono gli impianti per

il teatro (1.162), i centri commerciali, retail park e factory outlet (956), migliaia di punti vendita della GDO, oltre 200 parchi nella sola Toscana, i comuni con decine di migliaia tra centri storici, aree monumentali, aree archeologiche, complessi monumentali e luoghi della cultura, oltre 57.000 plessi scolastici ed un numero incalcolabile di luoghi ove si esprime la socialità. Non però una socialità «generalistica» ed indistinta, come nei social tradizionali tipo Facebook (dove si parla di tutto e di nulla), ma bensì una socialità caratterizzata da tre fattori unici e comuni: tante persone raccolte in un unico spazio all'interno del quale condividono un comune interesse E' questo un tipo di contesto non esplorato, al quale il progetto eServant prova a dare delle risposte e dei servizi mirati.

La sperimentazione del progetto eServant si è svolta presso l'impianto universitario PIN a Prato nel mese di Dicembre 2018.

1.2 AZIENDE COINVOLTE

Quid Informatica spa Come capofila di progetto, Quid Informatica spa si è occupata di sviluppare mettere a disposizione l'infrastruttura tecnologica per fruire i moduli sviluppati dai vari partner. Sokom srl Sokom srl si è occupata dei cablaggi e disposizione dei vari access point necessari per fare comunicare i dispositivi IoT posti all'interno dell'impianto.

Magenta srl Magenta srl si è occupata dei sensori IoT per il conteggio delle persone tramite hardware Raspberry ed in parallelo al recupero di dati da fonti open source, come Lamma e Open Toscana.

MICC Il MICC invece si è occupato dell'ottenimento della densità di persone in alcuni punti strategici dell'impianto di sperimentazione. Sviluppando inoltre un modulo di mappe, insieme alla tecnologia precedente, per consigliare ai partecipanti eventuali percorsi alternativi in caso di affollamento. Si aggiunge al partner MICC anche il motore di affinità tra visitatori grazie al collegamento social degli stessi.

DIISM Il dipartimento Ingegneria dell'informazione e Scienze Matematiche di Siena è fornitore e configuratore dei dispositivi iBeacon che tramite la tecnologia BLE (bluetooth low energy) ci ha permesso di ottene-

re la posizione delle persone all'interno dell'impianto di sperimentazione anche in assenza di GPS.

1.3 CASI D'USO

1.4 BACK-END

1.4.1 *Microservizi*

1.4.2 *Swagger (API)*

Swagger è un insieme di specifiche e di strumenti che mirano a semplificare e standardizzare i processi di documentazione di API per servizi web RESTful, il tutto su di una piattaforma di tipo open-source. Il cuore di Swagger consiste in un file testuale (in formato sia YAML che JSON) dove sono descritte tutte le funzionalità di un'applicazione web e i dettagli di input e output in un formato studiato per essere interpretabile correttamente sia dagli umani che dalle macchine.

I vantaggi di questa standardizzazione sono molti, come una migliore e più condivisibile esplorazione delle funzionalità di un'applicazione, oltre che alla possibilità di generare codice client/server sfruttando direttamente i vincoli definiti nello schema. Infatti, i metadati presenti nel file forniscono informazioni sufficienti sia per generare il componente backend con le rotte http e la validazione degli input, sia la parte client che in modo automatico può adattarsi all'evoluzione delle API del backend. La generazione della parte server è sicuramente vantaggiosa in quanto è possibile concentrarsi direttamente sulla programmazione della logica di business, senza doversi preoccupar e di come questa va ad interagire con il sistema al quale si interfaccia. Swagger è sostanzialmente composto da 3 moduli:

- 1. Swagger Editor: è lo strumento per iniziare rapidamente nell'attività di progettazione di una nuova API o per la sua modifica in un potente editor che visualizza visivamente la definizione Swagger e fornisce feedback in tempo reale;

- 2. Swagger Codegen: è lo strumento che crea ed abilitata le varie applicazioni al consumo delle API utilizzabili su una molteplicità di linguaggi di sviluppo (Java, Javascriptm Perl, PHP, Python, Ruby, Scala;
- 3. Swagger UI: è lo strumento che permette con uno strumento visuale il consumo delle API da parte delle varie applicazioni.

Per tutti i motivi indicati Swagger viene adottato all'interno dell'architettura di eSERVANT, come strumento per lo sviluppo delle varie API di interfacciamento verso la varie applicazioni dei partner.

1.4.3 *Database (relazionale + documentale)*

I due tipi di database usati nel progetto sono stati Postgres e MongoDB, il primo relazionale e il secondo orientato a documenti.

PostgreSQL è noto come il più avanzato sistema di gestione di dati open-source disponibile sul mercato. SI tratta di un database di tipo DBMS (DataBase Management System), ossia di un vero e proprio insieme di componenti software in grado di permettere la creazione e la manipolazione di un database.

Ma PostgreSQL è anche un ORDBMS (Object Relational DataBase Management System), ossia un sistema che supporta un modello di database object oriented, che implementa oggetti, classi, ereditarietà e permette l'estensione del modello di dati con tipi di dati e metodi personalizzati. Altamente portatile (praticamente su tutte le distribuzioni di Linux, Unix, Windows, Mac OS, ecc.).

Per tutti i motivi indicati PostgreSQL viene adottato all'interno dell'architettura di eSERVANT come il database di riferimento, utilizzato da tutte le applicazioni che necessitano del supporto di un database di tipo relazione. Tutta la gestione massiva dei dati viene invece gestita attraverso il database non-SQL MongoDB.

Hibernate (talvolta abbreviato in H8) è una piattaforma middleware open source per lo sviluppo di applicazioni Java che fornisce un servizio di Object-relational mapping (ORM), ovvero che gestisce la rappresentazione e il mantenimento su database relazionale di un sistema di oggetti

Java. Lo scopo principale di Hibernate è quello di fornire un mapping delle classi Java in tabelle di un database relazionale; sulla base di questo mapping Hibernate gestisce il salvataggio degli oggetti di tali classi su database. Si occupa inoltre del reperimento degli oggetti da database, producendo ed eseguendo automaticamente le query SQL necessarie al recupero delle informazioni e la successiva reistanziazione dell'oggetto precedentemente "ibernato" (mappato su database). Hibernate in pratica esonera lo sviluppatore dal lavoro inerente la persistenza dei dati. Fortunatamente Hibernate gestisce una persistenza trasparente per "Plain Old Java Object".

MongoDB è un Database management System, orientato alla gestione dei testi che ha come obiettivo quello di strutturare delle basi di dati testuali, in particolare con dati poco strutturati. Questo DBMS eredita il meccanismo di storage dal paradigma doc-oriented che consiste nel memorizzare ogni record come documento che possiede caratteristiche redeterminate può aggiungere qualsiasi numero di campi con una qualsiasi lunghezza. Nei doc-oriented si segue una metodologia differente rispetto al modello relazionale: si accorpano quanto più possibile gli oggetti, creando delle macro entità dal massimo contenuto informativo. Questi oggetti incorporano tutte le notizie di cui necessitano per una determinata semantica. Pertanto MongoDB non possiede uno schema e ogni documento non è strutturato, ha solo due chiavi obbligatorie: ¡d, cheserveperidentificareunivocamenteildocumento(ècomparabile, semanticame

Pertanto si può interrogare il DBMS anche per versioni del documento non recenti perché mantiene in memoria tutte le versioni. MongoDB gestisce anche la ridondanza dei dati per rimediare a malfunzionamenti.

1.5 FRONT-END

Lo sviluppo lato app è stato implementato usando il framework di sviluppo applicazioni mobile ibride Ionic. Questo framework utilizza di default l'engine Angular ed aggiunge ad esso una serie di funzioni "speciali" che permettono l'accesso a routine native, es. accensione camera dispositivo. Le funzioni "speciali" sono realizzate alla libreria Cordova che implementa il ponte tra Javascript e il codice nativo Android/IOS.

Il punto di forza di Ionic è l'essere un framework ibrido, scrivendo quindi un unico progetto in Javascript è possibile eseguirlo immediatamente su 3 piattaforme diverse: web, Android e IOS.

Angular 2 è un framework Javascript pernsato per lo sviluppo di applicazioni di tipo web, sia per mobile che per desktop, basato sul progetto open-source prima di Angular e poi di Angular JS.

Angular da un lato esalta e potenzia l'approccio dichiarativo dell'HTML nella definizione dell'interfaccia grafica, dall'altro fornisce strumenti per la costruzione di un'architettura modulare e testabile della logica applicativa di un'applicazione. In questo senso Angular fornisce tutto quanto occorre per creare applicazioni moderne che sfruttano le più recenti tecnologie, come ad esempio le Single Page Application, cioè applicazioni le cui risorse vengono caricate dinamicamente su richiesta, senza necessità di ricaricare l'intera pagina, cosa particolarmente utile in una applicazione del tipo eSERVANT.

Le caratteristiche di Angular 2 sono: mobile first: uno degli obiettivi del nuovo Angular è di proporsi per lo sviluppo di applicazioni Web sia per l'ambiente desktop che per il mobile. Infatti, Angular 2 supporta di default eventi touch e gesture e promette elevate prestazioni per assicurare una interazione fluida sui dispositivi mobile;

riduzione della curva di apprendimento: Angular 2 ha ottimizzato e semplificato di molto le complessità insite nella realizzazione di interfacce sia per mobile che desktop;

modularità: l'architettura di Angular 2 è altamente modulare e favorisce la scrittura di applicazioni modulari e la maggior parte dei componenti del framework è opzionale ed è possibile sostituirli con altri di terze parti o sviluppati in proprio;

prestazioni; un'attenzione particolare è stata posta sulle prestazioni del framework ed alla riduzione dei tempi di caricamento e boostrapping delle applicazioni.

Per tutti i motivi indicati Angular 2 viene adottato all'interno dell'architettura di eSERVANT, come strumento per il disegno delle interfacce

sia su mobile (e quindi per gli utenti/spettatori) che di tipo desktop (e quindi dedicate alla centrale di controllo e gestione).

1.5.1 *Linguaggi coinvolti*

Front-end - Typescript
Backend - Java
Continuous Integration - Groovy

2

APPLICAZIONE MOBILE

Empty thesis model

2.1 FUNZIONALITA'

2.2 FRAMEWORK IONIC

2.2.1 *NgRx gestore stato*

2.2.2 *Android*

2.2.3 *IOS*

2.3 GEOLOCALIZZAZIONE

2.3.1 *iBeacon*

2.3.2 *QRcode*

2.3.3 *GPS*

2.4 NAVIGAZIONE (MAPPE INTELLIGENTI)

2.5 CHATBOT (DIALOGFLOW)

3

BACKOFFICE

Empty thesis model

3.1 FRAMEWORK ANGULAR

3.2 MATERIAL DESIGN

3.3 FIREBASE

3.3.1 *Pro/Contro servizi cloud*

3.3.2 *Real Time Database*

3.4 CI JENKINS

4

IOT

Empty thesis model

4.1 BLUETOOTH LOW ENERGY

4.1.1 *Gateway (attivo)*

4.1.2 *Smartphone (attivo/passivo)*

4.1.3 *iBeacon Box (passivo)*

4.1.4 *iBeacon Wearable (passivo)*

4.2 CONTA PERSONE

4.3 TELECAMERE DENSITA'