UNIVERSITÀ
DEGLI STUDI
FIRENZE

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

TITOLO IN ITALIANO

TITLE IN ENGLISH

YURI BACCIARINI

ROSARIO PUGLIESE

Anno Accademico 2018-2019

INDICE

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.

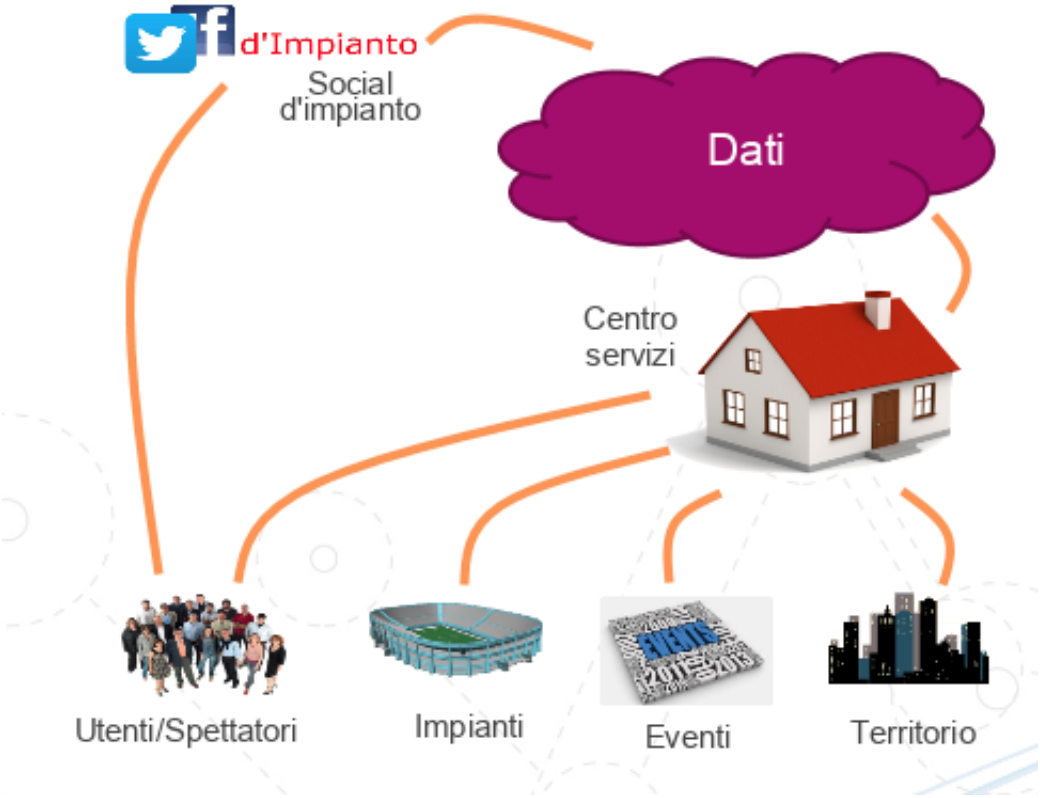
Si tratta quindi di un insieme di processi, che sono gestiti in modo assolutamente innovativo, sino a creare un processo evoluto, costituito dalla piattaforma eServant, che viene industrializzato in una struttura hardware e software altamente replicabile.

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

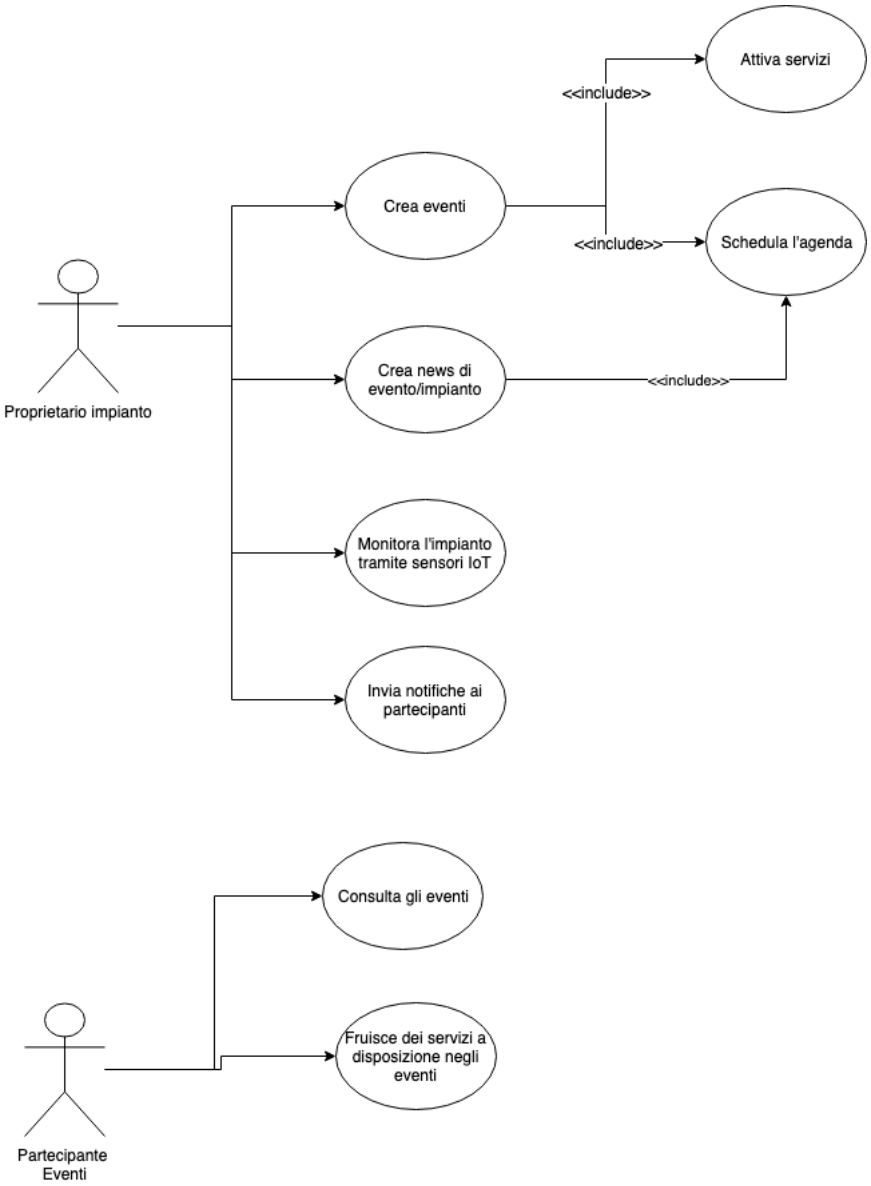
Quid Informatica spa Come capofila di progetto, Quid Informatica spa si è occupata di sviluppare mettere a disposizione l'infrastruttura tecnologica per fruire i moduli sviluppati dai vari partner. Sokom srl Sokom srl si è occupata dei cablaggi e disposizione dei vari access point necessari per fare 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 ottenere la posizione delle persone all'interno dell'impianto di sperimentazione anche in assenza di GPS.

1.3 CASI D'USO



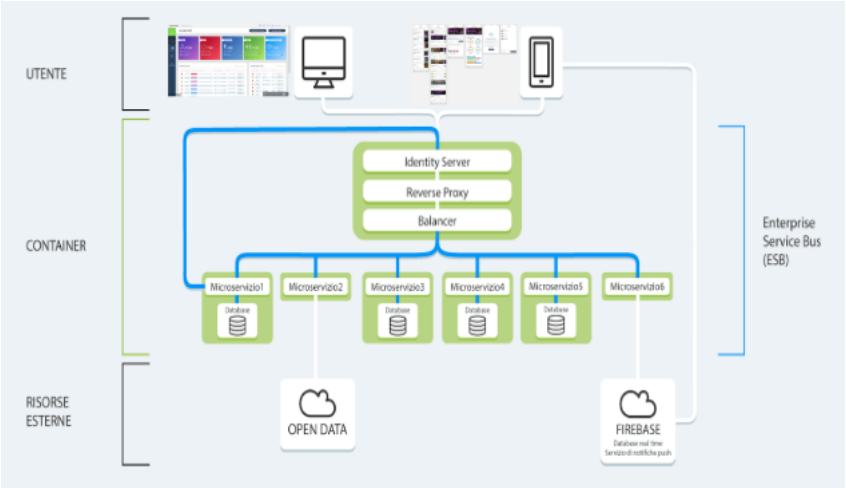
1.4 BACK-END

OSGI (Open Service Gateway Initiative) è una specifica che permette di costruire applicazioni modulari a componenti (detti "bundle") e che introduce una programmazione Service Oriented, permettendo una separazione tra interfaccia ed implementazione molto più rigorosa di quella

nativa Java. Il nucleo delle specifiche è un framework che definisce la gestione del modello del ciclo di vita del software, i moduli (chiamati bundle), un service registry e un ambiente di esecuzione. L'aumento della complessità in un prodotto software, sia esso embedded, client o server, richiede codice modulare ma anche sistemi che siano estensibili dinamicamente, molto più in applicazioni tipo quelle di eSERVANT. Il framework OSGI implementa un modello a componenti completo e dinamico cioè quello che manca all'ambiente Java. OSGi risolve molti dei problemi legati allo scarso supporto di Java nella modularità e nel dinamismo ed in tal senso OSGI è un framework che permette di fornire: un sistema modulare per la piattaforma Java; un sistema dinamico, che consente l'installazione, l'avvio, lo stop e la rimozione dei moduli a runtime, senza quindi necessitare di riavvii; un sistema orientato ai servizi, i quali possono essere dinamicamente registrati ed utilizzati nella macchina virtuale Java .

Per tutti i motivi indicati OSGI viene adottato all'interno dell'architettura di eSERVANT, come strumento per lo sviluppo delle applicazioni di backend di tutti i partner del progetto.

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: id, che serve per identificare univocamente il documento (è comparabile,semanticamente, alla chiave primaria dei database relazionali) rev, che viene utilizzata per la gestione delle revisioni, ad ogni operazione di modifica infatti la chiave rev viene aggiornata.

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;

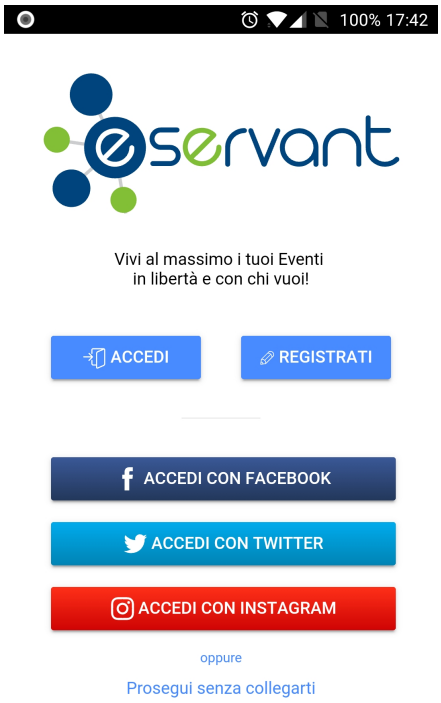
modularità: l'architettura di Angular 2 è altamente modulare e favorisce la scrittura di applicazioni modulari e la maggior parte dei componenti del framework è opzionale ed è possibile sostituirli con altri di terze parti o sviluppati in proprio;

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).

2

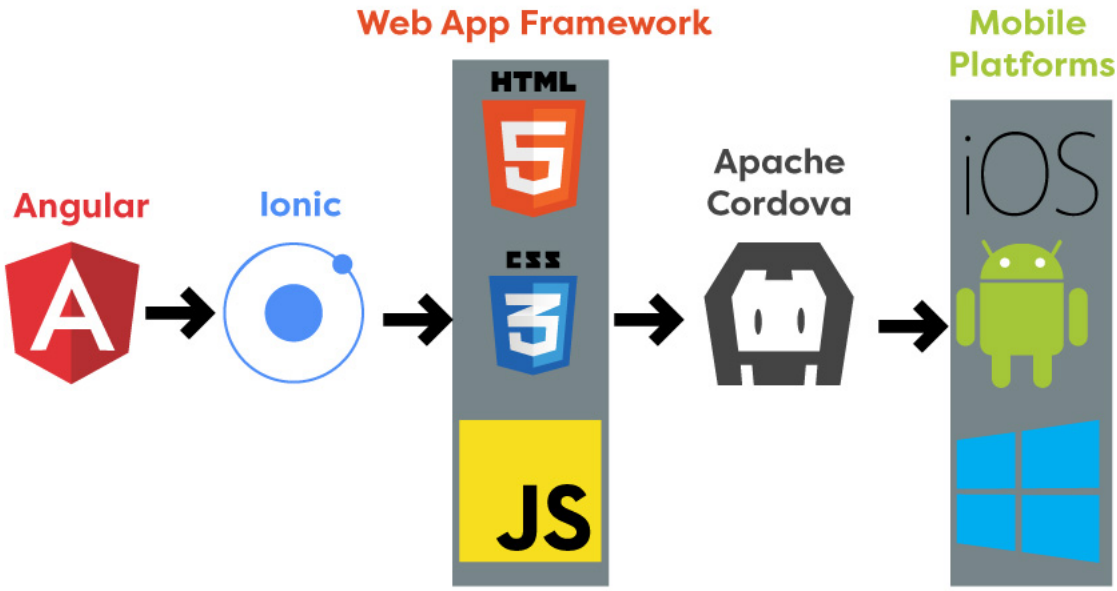
APPLICAZIONE MOBILE



L'interfaccia della piattaforma eSERVANT verso gli utenti finali è come da progetto una APP per mobile. Per questo insieme a Sintra Consulting s.r.l. siamo andati a disegnare in modo molto avanzato tutte le interfacce andando a quadrare requisiti di sistema/casi di uso con mockup montati all'interno di una vera e proprio demo dell'APP finale attesa. Tale strumento ci ha permesso di valutare non solo la user experience degli utenti, ma anche di poter fornire agli sviluppatori uno strumento specifico sul quale andare a traguardare la loro attività. Tutte le slide sono state montate e condivise tra tutti i partner del progetto all'indirizzo https://share.proto.io/U2B8EV/, anche se per motivi di praticità non sono state sviluppate tutte le articolazioni della stessa, ma solo le funzioni di maggior impatto e complessità. Il prototipo di APP è quindi stato creato andando a realizzare slide sui cinque temi di fondo:

- presentazione e riconoscimento;
- gestione evento e socialità;
- gestione delle funzioni di aiuto;
- gestione delle funzioni di mobilità;
- gestione delle funzioni di navigabilità e mappe;
- gestione del proprio profilo.

2.1 FRAMEWORK IONIC



Ionic è un HTML5 SDK lanciato nella sua prima versione nel 2013 e progettato per essere la base per lo sviluppo di app mobili ibride. Sviluppare un app mobile ibrida significa scrivere un unico code base per i due OS più famosi: IOS e Android.

La struttura di Ionic è costituita da Angular, Apache Cordova (ex-PhoneGap) e SASS. Ciò consente la creazione di applicazioni mobili ricche di funzionalità che utilizzano esclusivamente tecnologie web.

Angular

Angular è un framework web open source creato da Google ed è il successore di AngularJS (primissima versione). Angular permette ai developer di costruire applicazione fruite nel web, su piattaforme mobile o desktop. Angular capisce solo TypeScript dato che il framework è stato proprio scritto in Typescript.

Elementi positivi emersi sviluppando in Ionic:

- essendo tra i più famosi web framework, la **velocità di sviluppo** è un aspetto caratterizzante di Ionic
- **semplice** da capire
- supporta il **live reload** permettendo così al developer di vedere immediatamente le modifiche direttamente sul dispositivo mobile.

Elementi negativi emersi:

- l'accesso alle funzioni native tramite Cordova è **limitato**. Non è assolutamente paragonabile alle API native dell'OS.abilità e mappe;
- **instabilità** nell'accesso nativo. Generalmente le librerie Cordova non sono allineate con le ultime API rilasciate da Android/IOS.

Apache Cordova, ex-PhoneGap e futuro Capacitor

TODO: descrizione cordova

SASS
TODO: descrizione SASS

2.1.1 *NgRx gestore stato*

TODO: descrizione ngrx + flow

2.1.2 *Android*

TODO:

2.1.3 *IOS*

TODO:

2.2 SOCIAL LOGIN

Al primo accesso all'APP eServant è necessario dare il consenso per almeno uno dei seguenti social:

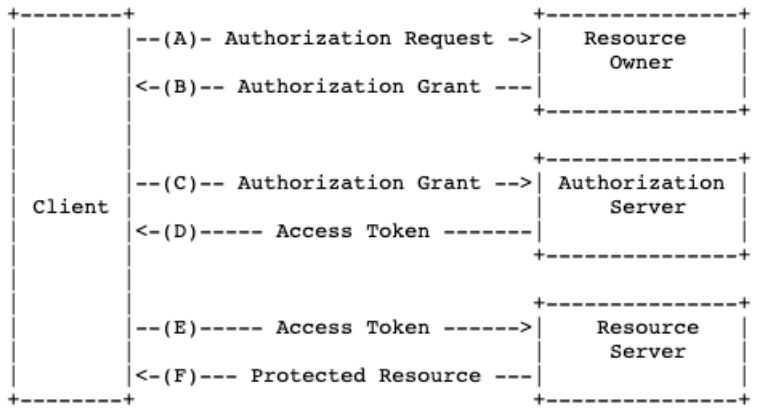
- Facebook
- Instagram

- Twitter

Altrimenti viene data la possibilità di fare un accesso privato tramite username e password. L'autenticazione verso i social network sopra citati avviene tramite protocollo OAuth2.0.

Il framework di autorizzazione OAuth 2.0 abilita un'applicazione di terze parti a ottenere un accesso limitato a un servizio HTTP, attivo per conto di un proprietario di risorse orchestrando un'inerazione di approvazione tra il proprietario della risorsa e il servizio HTTP. Questa specifica sostituisce e obsoleta il protocollo OAuth 1.0 descritto in RFC 5849.

OAuth 2.0 flow



Attori

resource owner: un'entità in grado di approvare l'accesso a una risorsa protetta.

authorization server: il server che invia i token di accesso al client dopo essere stato autorizzato dal resource owner

resource server: il server che ospita le risorse protette. Il resource Server è in grado di accettare e rispondere alle richieste di risorse protette utilizzando i token di accesso.

client: un'applicazione che fa richieste di risorse protette per conto di proprietario di risorse e con la sua autorizzazione. Il termine "client" non implica particolari caratsteristiche di implementazione (l'applicazione può essere eseguita su un server, un client web o altri dispositivi).

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'