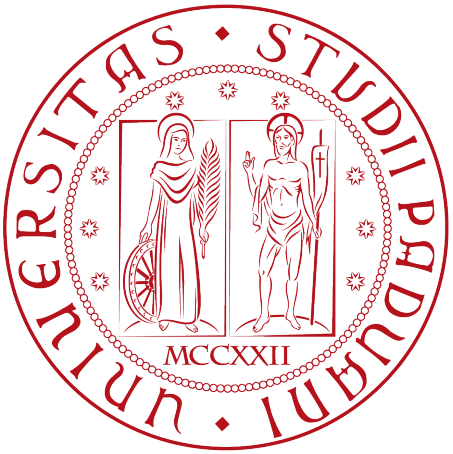
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Tecnologie web per lo sviluppo di applicazioni mobile ibride

Tesi di laurea

Relatore

Prof.Paolo Baldan

Laureando

Nicola Salvadore

Anno Accademico 2019-2020

For, after all, how do we know that two and two make four? Or that the force of gravity works? Or that the past is unchangeable? If both the past and the external world exist only in the mind, and if the mind itself is controllable – what then?

— George Orwell

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal laureando Salvadore Nicola, presso l'azienda Alternative Studio di Costa Francesco, della durata di circa trecento ore.
L'obiettivo dello stage è stato il rifacimento della applicazione per dispositivi mobili a disposizione di UCIS (Unità Cinofile Italiane da Soccorso), che si occupa di registrare la geolocalizzazione delle unità cinofile durante le esercitazioni, gli addestramenti, e anche le operazioni di emergenza.
Il progetto mi ha coinvolto nello studio e nell'approfondimento delle moderne tecnologie per lo sviluppo di applicazioni mobili, le quali verrano descritte in modo dettagliato in questo documento.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Paolo Baldan, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.
Desidero ringraziare con affetto i miei genitori per il sostegno e per avermi insegnato cosa sono l'impegno e la responsabilità.
Desidero ringraziare la mia nonna che prima di morire desidera vedermi laureato e che spero riesca a vedere anche la laurea del mio fratello più piccolo.
Ho desiderio di ringraziare poi i miei amici per aver fatto sembrare questi anni di studio e lavoro molto più leggeri di quanto non lo siano stati.

Padova, Settembre 2020 Nicola Salvadore

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

Figura 1.1: Logo di Alternative Studio.

Alternative Studio è una web agency che fornisce soluzioni professionali su misura, costruite secondo le esigenze del cliente. Si occupa principalmente di sviluppo web e marketing. È un'azienda piccola che raccoglie poche risorse umane, ma molte energie che continuano a spingere per crescere. Opera da appena sei anni nel settore del web development, ma ha abbracciato anche altre iniziative, collaborando in progetti più grandi con altre realtà. Negli ultimi anni l'azienda si è cimentata nello sviluppo di un gestionale per l'organizzazione Unità Cinofile da Soccorso (UCIS) e di una sua API volta alla ricezione e all'elaborazione di attività registrate durante addestramento, soccorso o esercitazioni.

1.1.1 Metodo di Lavoro

Dato le contenute risorse umane a disposizione Alternative Studio adotta un ciclo di sviluppo software incrementale con qualche introduzione di processi da quello agile, in

particolared dal metodo Scrum. Durante lo stage lo studente è stato anche incaricato di introdurre qualche concetto delle metodologie di sviluppo moderne all'interno del contesto aziendale, come quello di Sprint e di Daily Stand-up.

1.1.2 Processi di sviluppo

La mia figura è subentrata durante la fase di Analisi dei requisiti. Per questo i primi compiti affidatomi sono stati quelli di studio delle tecnologie adatte al progetto. Durante questa fase si sono svolte alcune riunioni con il tutor, tramite videochiamata, e redatto alcuni documenti di report riguardo le ispezioni e le ricerche. Inoltre il lavoro individuale (come compilare un "Hello World" con Ionic) si è svolto tramite tasks. Le riunioni con il tutor sono proseguite anche durante la fase di progettazione architetturale, durante la quale abbiamo chiarito la visione generale dell'applicazione.

1.1.3 Strumenti di supporto allo sviluppo

Ci sono alcuni software da citare utilizzati nella gestione del progetto, che Alternative Studio utilizza abitualmente.

Gestione progetto e Versione

Gitlab è un software open source per la gestione di repository Git e supporto alla Continous Integration.

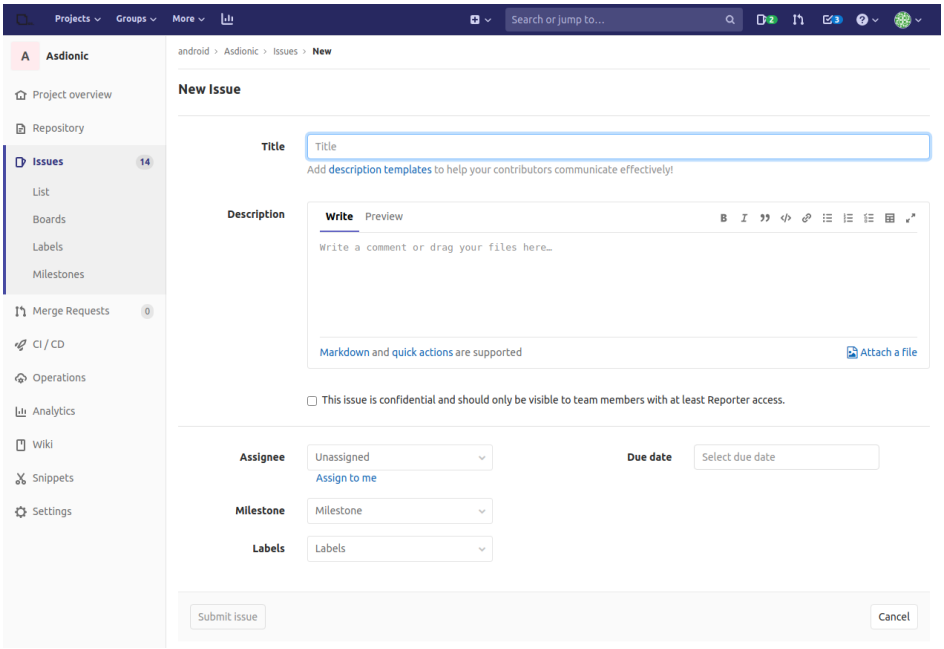


Figura 1.2: Schermata di nuova issue del software Gitlab.

Per la prima parte è stato importante il suo strumento di Issue Tracking System con il quale si sono gestite le tasks e le loro scadenze. Infatti, in questo caso le issues sono state utilizzate come metodo per tracciare i compiti da svolgere, piuttosto che per

segnalare problemi all'interno del software, come bug e affini.
Ogni task possiede un titolo significativo, una descrizione approfondita aggiornabile in caso di cambiamenti durante l'esecuzione e una scadenza. Inoltre è possibile assegnarla a più membri del team, aggiungere informazioni o dialogare con il tutor attraverso la sezione commenti e specificare la milestone alla quale è collegata.

Comunicazione

Causa il telelavoro, durante il progetto sono stati fondamentali gli strumenti di comunicazione:

* Slack: per le comunicazioni ufficiali e come strumento di notifica per i cambiamenti nella repo.
* Telegram: come servizio di messaggistica istantanea, diretta con il tutor.
* Skype: per le videochiamate, essenziale per lo svolgersi delle riunioni a distanza.

Codifica

In dotazione ad Alternative Studio ho utilizzato WebStorm, potente IDE parte della suite di JetBrains. Una delle sue caratteristiche principali sono la sua modularità, grazie allo store di plugins disponibili che aggiungono funzionalità, come il conteggio delle ore di programmazione e gestione del repository Git locale.
Inoltre si integra molto bene con le tecnologie web, come Angular, mettendo a disposizione il suo ambiente di building e di testing direttamente all'interno dell'IDE.

(a) Logo di WebStorm

(b) Logo di Android Studio

Android Studio

Android Studio fa parte sempre del pacchetto IDEA di JetBrains ed è una versione molto ridotta di Intellij. A differenza del suo fratello maggiore è open source e ottimizzato per lo sviluppo Android. I linguaggi nativi utilizzabili in questo IDE sono principalmente Go e Java. Utilizza gradle come strumento automatico per la build dei progetti ed è estensibile come le altre applicazioni IDEA tramite plugins.

Postman

Programma a supporto della codifica, il quale mi ha aiutato nell'interfacciarmi alle API. Postman permette di fare chiamate HTTP ad un'API e di visualizzarne il risultato. È un software molto utile nel quale puoi, ad esempio, testare le stringhe che utilizzerai nel tuo codice, o per analizzare il risultato per capire come utilizzarlo.

1.2 Descrizione dello stage

La scelta dello stage è stata molto difficile. Ero alla ricerca, infatti, di un contesto specifico per la mia prima esperienza lavorativa nel settore informatico. Consultando tutte le varie proposte, durante l'evento Stage-IT, ho fatto fatica a trovare una proposta che mi convincesse a fondo. Alcune delle aziende che ho contattato si sono dimostrate disponibili nei miei confronti, ma per le tempistiche legate alla laura ho dovuto rinunciare alle loro proposte.

1.2.1 I contenuti

Come già accennato nel primo capitolo Alternative Studio è un'azienda giovane che da 5 anni collabora con UCIS. Durante questi anni è stato sviluppato un gestionale e un sito per la loro organizzazione.

All'interno di questo gestionale è stata sviluppata un'API, volta alla registrazione di log contenenti la posizione tramite coordinate e alla creazione di una relativa traccia. A questo software attualmente si interfaccia un'applicazione sviluppata 5 anni fa, non da Alternative Studio, e pubblicata nel 2018 con il nome di UCIS Report Tool. L'applicazione ha come funzionalità principale la creazione e la gestione di attività, con la registrazione e l'invio di log contenenti la posizione del dispositivo sul quale esegue. Questa, però, non risulta essere un prodotto professionale e utilizza tecnologie già vecchie e difficilmente manutenibili o aggiornabili. Inoltre sono stati lamentati da UCIS alcuni bug e pochissima precisione nella registrazione della posizione. Dall'associazione nasce così l'esigenza di aggiornare o riscrivere completamente l'applicazione.
Alternative Studio ha colto, quindi, l'opportunità di sviluppare una sua versione dell'applicativo e di proporlo poi all'organizzazione con la quale già collabora.

Problematiche dell'applicazione UCIS Report Tool

L'applicazione attualmente utilizzata da UCIS è stata sviluppata interamente in linguaggio nativo, esclusivamente per il sistema operativo Android. Il principale problema rimane in ogni caso la scarsa precisione nella lettura delle tracce GPS. L'invio dei log e troppo dilazionato nel tempo rendendo la traccia risultante frastagliata e alle volte incomprensibile. Essendo la funzione primaria questo risulta essere un grosso problema, ma non è certamente l'unico. La disposizione delle viste non è logica e lascia l'utente disorientato. Inoltre i form risultano essere errati: alcune delle opzioni non devono essere presenti e la selezione di una non provoca effetti su un'altra. Ad esempio se volessi creare una nuova attività dovrei selezionare la macrocategoria e di conseguenza filtrare la selezione della categoria successiva. Ciò non avviene e provoca un'errore nell'invio del form.
Forse il problema più grande è la non manutenibilità di questo software. Il codice risulta poco modulare e la lettura delle classi ostica. È privo di documentazione formale e di una progettazione sensata. Risulta essere un prodotto morto e non professionale. Un altro punto negativo dell'applicazione è il kit grafico utilizzato, risulta essere vecchio e di sicuro non accattivante per l'utente e meno ancora per il cliente.

1.2.2 Vincoli

Questo capitolo si occupa di descrivere i vincoli ai quali sono stato sottoposto durante lo stage. Tengo a precisare che quelli imposti dall'azienda non sono mai stati un

ostacolo, bensì delle guide precise per raggiungere al meglio gli obiettivi. I vincoli sono stati raggruppati nelle sezioni successive.

Vincoli tecnologici

Il vincolo più importante imposto è stato quello di non scrivere l'applicazione in linguaggio nativo per ogni sistema operativo. Nonostante sia stata sviluppata solamente per il sistema operativo Android, l'idea iniziale era quella di produrla cross-platform e di portarla con poche modiche su qualunque sistema operativo mobile e non. È per questo che il tutor mi ha imposto la ricerca di un framework ibrido.

Vincoli metodologici

Il mio stage si è svolto, purtropp o, durante un periodo soggetto a restrizioni negli spostamenti dovuti all'epidemia di SARS-CoV-2. Dopo un periodo iniziale, però, sono riuscito a svolgere lo stage in parte in presenza. Quinid ho concordato con il tutor di poter accedere ai locali di Alternative Studio almeno due giorni a settimana, solitamente il lunedì e il venerdì.
Per quanto riguarda il metodo di lavoro ho cercato diverse volte consiglio presso il tutor. Riuscire a svolgere lo stage in presenza, anche se in parte, è stato molto importante in questo senso. Infine per aderire in parte al metodo agile abbiamo organizzato una breve riunione giornaliera, che aveva come scopo quello di allineare il lavoro e programmare le tasks successive.

Vincoli temporali

Oltre al vincolo di 320 ore, imposto dalla natura dello stage e dall'azienda stessa, sono dovuto sottostare alle scadenze delle varie tasks impostate su Gitlab. In particolare si è scelto di lavorare 40 ore alla settimana.
Non essendo stato uno stage completamente in presenza, non si è dato nessun vincolo sulla singola giornata lavorativa. Solitamente l'orario lavorativo iniziava alle 9:00 e si concludeva alle 18, con un'ora di pausa pranzo. Questo per me è stato un problema da affrontare, perché ho notato che sono molto più produttivo quando ho delle restrizioni sugli orari. Il problema è stato fatto presente al tutor che ha collaborato nel controllo e nel conteggio delle ore effettiva di lavoro.

1.3 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

1.4 Scopo del documento

In questo documento andrò a descrivere la mia esperienza di stage, raccontando la mia preparazione, lo sviluppo e le mie valutazioni riguardo ad esso. Inoltre descriverò in modo approfondito le tecnologie che sono andato ad analizzare e studiare e le mie opinioni su di esse.

1.4.1 Organizzazione del testo

Il secondo capitolo approfondisce i contenuti del progetto di stage e i motivi della scelta.

Il terzo capitolo descrive in modo dettagliato ciò che è stato fatto durante lo stage.

Il quarto capitolo contiene le considerazioni a posteriori dell'esperienza di stage.

Capitolo 2

Analisi dei requisiti

2.1 Analisi

Il mio stage è iniziato quando ormai l'analisi era al termine, quindi non ho avuto nessun contatto con il proponente. Tuttavia sono stato messo subito al corrente dei requisiti individuati e sono stati inseriti negli obiettivi del progetto di stage, quindi già esplicati nel capitolo precedente. Essendo un refactoring di un applicazione già esistente parte dei requisiti erano già stati stabiliti, quindi elencherò i principali cambiamenti richiesti. Utilizzerò come notazione R+(F|Q|V)+X+(D|O), dove:

* R: requisito;
* F: funzionale;
* Q: qualitativo;
* V: di vincolo;
* X: numero progressivo;
* D: desiderabile;
* O: obbligatorio;
* Z: opzionale.

Alcuni dei requisiti più importanti individuati sono riportati nella tabella Tabella 2.1, con codice e relativa descrizione:

Tabella 2.1: Tabella del tracciamento dei requisti funzionali

Requisito	**Descrizione**
RF1O	L'interfaccia permette di fare il login nel sistema gestionale di UCIS
RQ2O	L'applicazione riconosce se è già stato effettuato un login e utilizza i dati salvati per effettuarlo

Tabella 2.1: Tabella del tracciamento dei requisti funzionali

Requisito	**Descrizione**
RQ3O	L'applicazione deve funzionare offline se la persona è già autenticata
RV5O	I dati dell'account non vanno salvati nella memoria, ma nel sistema Android
RF8O	L'interfaccia permette di creare attività
RF9O	L'interfaccia permette di visualizzare un'attività
RF10O	L'interfaccia permette di avviare la registrazione di un'attività
RF11O	L'avvio di un attività provoca l'invio di log che registrano la posizione dello smartphone
RF12O	L'applicazione invia i log registrati al sistema gestionale UCIS
RV18O	L'applicazione deve essere scritta con uno strumento che permetta lo stesso codice per tutte le piattaforme

2.2 Studio di Fattibilità

2.2.1 Framework per lo sviluppo mobile

Come specificato dal requisito RV18O si era alla ricerca di un framework crossplatform. Di seguito alcune delle tecnologie che abbiamo analizzato e alcuni dei motivi per cui non sono stati scelti.

Cordova e PhoneGap

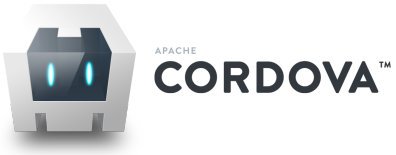


Figura 2.1: Logo di Cordova

La prima tecnologia con cui sono stato a contatto è stato il framework Cordova. Il software è stato acquisito da Adobe nel 2011 mantenendolo open source e rilasciandolo nel 2013 con il nome di Apache Cordova. Stiamo parlando di un framework che permette di creare applicazioni utilizzando CSS3, HTML5 e Javascript, evitando allo sviluppatore di affidarsi alle API specifiche del sistema operativo. Il pregio di questa

tecnologia è il fatto che la stessa applicazione non dovrà essere riscritta piu volte, ad esempio in Java per glsAndroid o in glsswift per iOS, ma avrà un codice unico che sarà adattato per i singoli sistemi operativi.
Tutto ciò funziona tramite le WebView native. Le WebView sono delle componenti istanziate dal sistema operativo utilizzate per visualizzare contenuti web, cioè renderizzano a schermo ciò che è stato scritto con linguaggio HTML5, CSS3, Javascript. Per facilitarne la comprensione si possono immaginare come delle pagine web visualizzate all'interno di un browser, senza gli elementi tipici di esso (come la barra dell'URL, delle tab o le opzioni).
Le applicazioni sviluppate in questo modo si possono considerare sia in parte , sia in parte native perché utilizzano e hanno accesso a tutti i componenti hardware della piattaforma sulla quale vengono eseguite. Ciò avviene tramite i plugin, che possono essere parte del framework, oppure, tramite API apposite, scritti dallo sviluppatore stesso. Inoltre l'applicazione può essere impacchettata, installata o caricata negli store ufficiali come una qualunque altra nativa.
La forza di Cordova e del fatto che sia open source, sta nel possedere una vastissima libreria di plugin che forniscono un controllo quasi totale sul dispositivo, come se si sviluppasse dal linguaggio nativo. Tutto questo grazie alla sua natura open source, che ha sviluppato una community, che produce e revisiona le componenti, facendo crescere il progetto molto velocemente. Uno dei plugin che ho immediatamente ricercato, è quello che si occupa del tracciamento GPS continuo in background. Sono andato quindi a creare un primo prototipo di applicazione installando solamente quel plugin per semplice test. Durante questo veloce procedimento ho però riscontrato diversi problemi nell'installazione delle varie componenti o nella compatibilità con altre librerie (come nel mmio caso con le Software Development Kit di Android). L'impressione di questo software non è stata positiva, perché mi è sembrato vecchio e poco mantenuto. Informandomi poi ho notato che le varie aziende, che lo utilizzavano come framework di sviluppo si stavano muovendo su altri software nuovi e più aggiornati.
Menzione d'obbligo è da fare ad Adobe PhoneGap, nato come versione non open source di Cordova, che fornisce a pagamento servizi molto interessanti, come ad esempio la build in cloud dell'applicazione, eliminando i problemi derivati dall'ambiente di sviluppo sul quale si sta lavorando.

Ionic e Capacitor

Figura 2.2: Logo di Ionic

Ionic è un framework open-source rilasciato da Drifty Co. nel 2013. La prima versione di Ionic non era altro che un SDK che metteva a disposizione una piattaforma AngularJS per lo sviluppo di applicazioni con Cordova. Le ultime versioni di Ionic hanno aggiunto nuove funzionalità, tra le quali la possibilità di scegliere di sviluppare in Angular, React o Vue. Inoltre è stato rilasciato un software per la build e il run-time di

applicazioni, chiamato Capacitor, successore spirituale di Cordova, del quale mantiene gran parte dei plugin, semplicemente adattandoli. Derivando da Apache Cordova, contiene tutte le sue funzionalità, e di conseguenza tutti i suoi pregi, ma anche alcuni dei suoi difetti.
Testando Ionic mi sono accorto di alcuni punti deboli delle applicazioni ibride.
In primo luogo le performance. Le applicazioni ibride hanno bisogno della WebView che non è altro che un processo di supporto per renderizzare pagine web. Inoltre è un dato di fatto che le applicazioni web sono molto esose dal punto di vista delle risorse rispetto a un applicazione scritta in linugaggio nativo per quel dato sistema operativo.
In secondo luogo non si ha controllo totale sui plugin che servono a interfacciarsi con il dispositivo. Le librerie native sono molto più ricche e specifiche, ma soprattutto non richiedono l'intervento di software di terze parti, come Ionic e Capacitor, per funzionare. Questi problemi sono emersi subito: una volta installato ionic e fatta una veloce build dell'applicazione ci siamo accorti della poca reattività su dispositivi datati e della difficoltà nel personalizzare i plugin messi a disposizione.

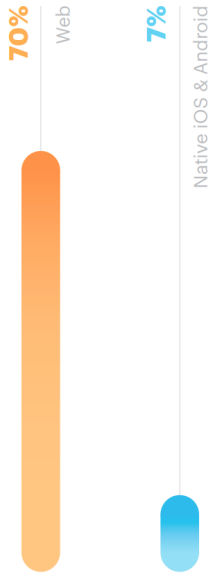


Figura 2.3: Caption1

Da alcune ricerche da noi effettuate è emerso che la maggior parte delle aziende che sviluppano per mobile si stanno muovendo verso le applicazioni ibride. I motivi non sono difficili da comprendere. Il fatto che un unico codice può essere eseguito su diversi dispositivi è sicuramente un fattore molto importante. Finora le aziende dovevano sviluppare applicazioni diverse quindi moltiplicando il costo e le risorse. Supponendo di utilizzare solo linguaggi nativi, Alternative Studio avrebbe dovuto sviluppare almeno due appplicazioni differenti, una per Android e una per iOS, riutilizzando pochissimo codice e impiegando almeno il doppio di risorse. Sarebbero stati necessari almeno altri due sviluppatori e i tempi di fornitura sarebbero stati decisamente più lunghi.
Inoltre gli sviluppatori con esperienza in API native non sono molto comuni. Non sono rari invece sviluppatori web, come dimostra la figura Figura 2.3.
Da notare anche come il mondo si stia muovendo molto velocemente verso il web. Moltissime delle applicazioni tuttora si basano su tecnologie che sono sviluppate per esso, come AWS di Amazon, Azure di Microsoft, Google Cloud e molti altri. Sviluppare in questo senso è anche un investimento su un mondo che è aperto a opportunità continue e di conseguenza guardare al futuro.

Flutter e Xamarin

Nonostante mi ritenessi soddisfatto dell'impressione di Ionic, ho preso in cosiderazione un'ultima alternativa, Flutter.
Flutter è un framework che si occupa della creazione di interfacce grafiche native per iOS, Android e desktop. Sviluppato da Google e rilasciato nel 2018, utilizza un linguaggio proprietario Dart, presentato come alternativa a Javascript. L'engine, sviluppata in C++, si occupa di renderizzare i widget, oggetti grafici scritti dallo sviluppatore. Le applicazioni in Flutter quindi possono essere eseguite su tutti i dispositivi nei quali è presente la VM di Dart e il suo motore grafico. Attualmente questi sono implementati in iOS e Android e su alcuni browser.

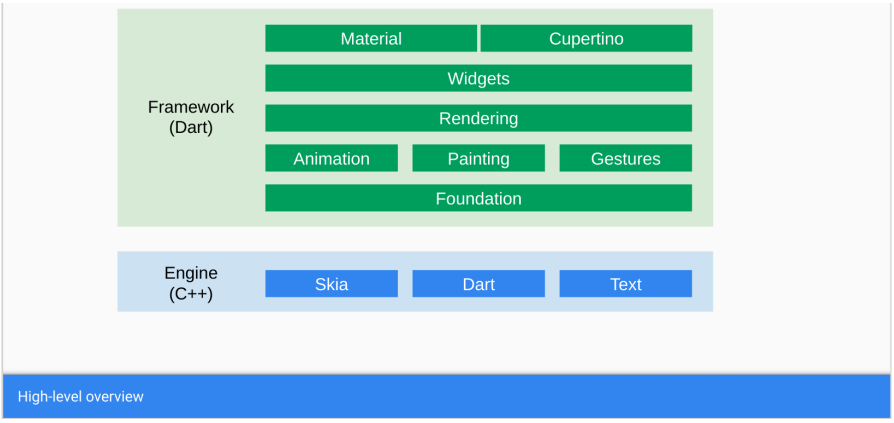


Figura 2.4: Architettura di Flutter

L'ovvia conseguenza di questo approccio sono le migliori prestazioni rispetto alle applicazioni web-based, eseguite su una WebView. Tuttavia il problema riscontrato in questo tipo di tecnologia è il fatto che si dovesse utlizzare un linguaggio proprietario, Dart, che fosse utile solo al campo dello sviluppo mobile. Inoltre tutti i lati positivi di un framework web non valgono anche in questo caso. Di fatto l'unica cosa che Flutter condivide con Ionic è il concetto di crossplatform.
Cito anche Xamarin, framework sviluppato da Microsoft, precedente a Flutter che utilizza come linguaggio C#. Le caratterisitiche sono molto simili alla proposta di Google e perciò non mi dilungherò oltre nel descriverlo.

La scelta

Dopo aver valutato le varie scelte per i motivi su citati, io e il mio tutor abbiamo optato per l'utilizzo di Ionic e Capacitor. Abbiamo pensato infatti che fosse un giusto compromesso tra innovazione e semplicità d'uso, non rinunciando però al bagaglio di funzionalità che venivano offerte da esse.
Una volta scelte le componenti da utilizzare non c'è rimasto che studiarle per valutarne una eventuale architettura. Sono andato così a realizzare un piccolo progetto con Ionic per capire bene il suo funzionamento.

Angular

L'interfaccia di Ionic è una Command Line Interface (CLI), che permette tramite un comando di inizializzazione di creare un progetto, scegliendo il framework web da utilizzare e un template di progetto dal quale partire. Per comprendere la progettazione front-end del progetto bisogna prima soffermarsi sullo studio di Angular.

C'è da precisare che quando si parla di Angular ci si riferisce alla versione 2.0+, che è differente dal suo predecessore AngularJS, storicamente individuata come versione 1.0. I due framework infatti non sono retrocompatibili, dato che nella nuova versione è stata completamente ridisegnata e riscritta.
Ci sono dei concetti fondamentali di Angular che ho utilizzato per l'implementazione dell'applicazione.

* Il primo tra questi è quello di **module**. Un'applicazione Angular è infatti costituita da *NgModules* che costituiscono il blocco fondamentale dal quale partire. Essi sono definiti come il contesto di compilazione nel quale vengono eseguite tutte le altre componenti. Esiste un modulo principale che solitamente è chiamato AppModule e costituisce la *root* dell'applicazione web.
* Le viste sono definite dai **components**. Ognuno di essi sarà infatti composto da tutti gli elementi grafici e il codice correlato (HTML5, SCSS), e dai loro comportamenti gestiti da quello che Angular chiama template. Il template sono una serie di direttive che collegano la logica dell'applicazione alle viste stesse.
* Il lato back-end dell'applicazione è rappresentato dai **services**. Sono definiti per contenere la logica, inoltre sono *Injectable*, cioè esiste un'istanza univoca di ogni servizio che può essere inserita all'interno dei *components*

Il punto di forza di Angular rimane comunque la sua modularità. Se debitamente progettate, tutte le parti dell'architettura sono studiate per lavorare in maniera indipendente una dall'altra. L'immagine Figura 2.5 descrive graficamente l'iterazione tra i componenti di Angular.

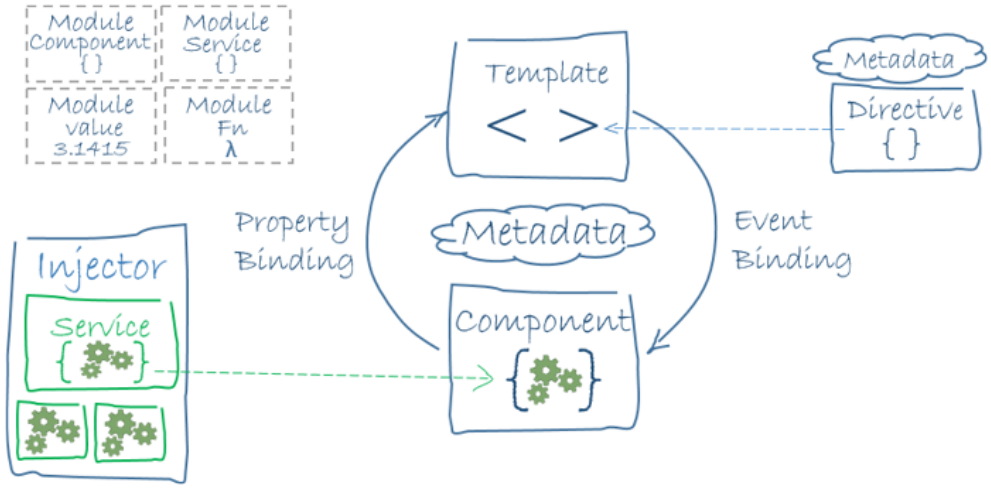


Figura 2.5: Architettura di Angular

Geolocalizzazione

Come definito nel requisito RF11O l'applicazione deve occuparsi di registrare la posizione dello smartphone. Su Ionic, questo è possibile attraverso l'uso dei plugin. Essendo coinvolta una risorsa hardware dello smartphone, questa non può essere gestita da Angular. Ho dovuto ricercare all'interno della documentazione di Ionic e di Capacitor un plugin adatto al nostro scopo. La ricerca non è stata difficile, Ionic infatti possiede nella sua documentazione una lista di plugin nativi, cioè plugin testati sul framework e compatibili con Capacitor, che derivano o non sono altro che plugin della community sviluppati per Cordova e adattati.
All'interno della documentazione si trovano due plugin che si occupano della localizzazione, Geolocation e Background Geolocation. Per decidere io e il mio tutor abbiamo constatato che secondo i requisiti dell'applicazione precedente era necessario che l'applicazione consumasse meno batteria possibile, ma che prendesse continuamente la posizione. Abbiamo quindi optato per Background Geolocation, che permette di localizzare il telefono anche in modalità background, cioè quando l'applicazione non è aperta nella schermata, o il telefono ha lo schermo spento.
L'approccio Android per quanto riguarda i processi in background è molto severo. Tutte le applicazioni che avviano processi in background, infatti, devono avere una notifica che avvisa l'utente che qualcosa sta eseguendo senza che se ne accorgano. Lato sviluppatore l'applicazione ha bisogno di molti permessi appositi, da inserire in quello che si chiama Manifest, file XML che dichiara tutte le proprietà principali dell'applicazione, tra i quali nome e package.
Altro rischio rilevato da questo plugin è l'ottimizzazione messa in atto dalle distribuzioni dei vari brand di Android stesso. Sul mio attuale smartphone, ad esempio, è installato una versione del sistema operativo che si chiama Oxygen OS. Questa particolare personalizzazione di Android di default ottimizza tutte le applicazioni installate. In questo frangente mi è tornato utile il sito *Don't Kill My App*. URL: https://dontkillmyapp.com/, il quale per ogni brand indica quali procedure seguire per disattivare queste ottimizzazioni lato utente.

Salvataggio offline

Le applicazioni mobile utilizzano per l'allocazione dati **SQLite**, una versione ridotta del noto SQL, per la gestione di database relazionali. SQLite permette la creazione di tabelle, form e report e l'interrogazione di questi, tutto all'interno dello stesso file. Per piccole basi di dati è molto più performante di SQL server e quindi perfetto per il salvataggio di dati di un sistema mobile.
Anche per questa funzionalità mi sono dovuto affidare a un plugin nativo di Ionic per la gestione di questi database. Per il debugging e la visualizzazione di questi database ho utilizzato un estensione di Google Chrome, chiamata Stetho.
Nonostante il difficile accesso al database con questo sistema, si è deciso anche che la maggior parte dei dati sensibili, quindi informazioni personali e chiavi per l'accesso non dovevano essere salvati in locale.

Comunicazione con le API

Come specificato dal requisito RF12O i dati raccolti dall'applicazione dovranno essere inviati al gestionale attraverso le sue API. Angular offre alcune delle sue librerie per comunicare con il server, ma si è preferito utilizzare axios, perché già conosciuto e di facile utilizzo.

Axios è una libreria di Javascript/Typescript che permette di eseguire chiamate HTTP di tipo POST/GET/PUT. Il metodo POST è una tipologia di chiamata utilizzata dal protocollo che prevede che i dati siano incapsulati all'interno del corpo del messaggio ed è utilizzata principalmente per caricare informazioni su un server. Il metodo GET contiene dati i nella risposta del server, mentre il PUT punta una risorsa nell'URI, se questa esiste nel server viene creata. Supporta il sistema di Promise di Javascript, quindi la possibilità di fare chiamate asincrone e trasforma tutte le risposte in oggetti JSON, facilmente leggibili anche non conoscendo la struttura del server. Permette inoltre di impostare un berear token: permette all'istanza di axios dell'applicazione di autenticarsi al server una singola volta, alle chiamate successive non sarà necessario l'invio dei codici per il login utente.

Capitolo 3

Progettazione e realizzazione

3.1 Progettazione

Una volta completato lo studio e analizzato in dettaglio le tecnologie scelte abbiamo proseguito verso la progettazione delle nostre componenti. Di seguito presenterò in sintesi come sono state modellate tutte le varie parti dell'applicazione, divise per funzionalità.

3.1.1 Login

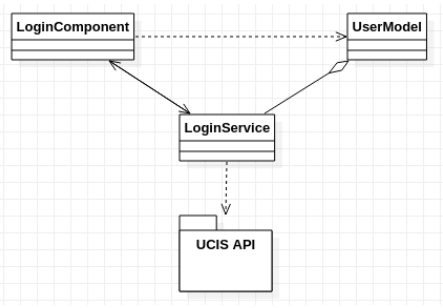


Figura 3.1: Diagramma login semplificato

La prima parte dell'applicazione della quale ci siamo occupati è quella di login. Per fare questo abbiamo optato per una semplice Model View Controller, con una classe *User* come modello, che rappresenta le informazioni di un utente, una *View* composta da un semplice form con un template Angular e un *controller* che si occupa di contattare il server e mantenere l'utente autenticato gestito da un *service*.

3.1.2 Struttura delle API del gestionale UCIS

Le Application Program Interface (API) esistenti mettono a disposizione quattro tipi di richiesta:

* l'autenticazione avviene tramite il metodo POST, dove nel body è inserita la username e la password. Il risultato ritorna la chiave, utilizzabile nelle richieste future. In caso le credenziali siano sbagliate ritornerà una pagina HTML con il seguente errore:

```
<div id="errorboxheader">0 - Class &#039;
    APIUnauthorisedException&#039; not found</div>
```

* è possibile con una richiesta GET richiedere la lista delle attività di un utente. La risposta contiene una classe JSON con la struttura delle categorie e lista delle attività.

* tramite richiesta PUT è possibile creare una nuova attività sul server associata all'account. Il client dovrà inviare la data in cui è stata creata l'id corrispondente alla categoria e l'id del cane con cui partecipa. Il server restituirà il codice associato all'attività.

* la gestione della registrazione avviene tramite richieste POST, nelle quali si specifica se sono di tipo *start*, *pause*, *end* e *log*. Ognuno di essi deve inviare al server la posizione e rispettivamente indicheranno se l'attività e ricominciata, è in pausa, è terminata o è un semplicemente un log intermedio.

3.1.3 Creazione e visualizzazione delle attività

Secondo il requisito RF8O l'applicazione deve permettere all'utente la creazione di un'attività. Il concetto da attività è stato ripreso da quello già esistente nelle API. Ognuna di esse è formata da:

* una categoria individuata da:
  - una *macrocategoria* (es. esercitazioni, addestramento) individuata da un id;
  - una *sottocategoria* (es. addestramento macerie, ricerca su superficie) individuata da un proprio id e da quello della macrocategoria.

* un nome scelto dall'utente;

* il cane con cui si svolge l'attività.

Inoltre l'applicazione tiene traccia dei seguenti dati:

* data di creazione;

* data di inizio;

* data di fine (se ancora in corso impostata a valore di default);

* i nomi delle categorie assiociati agli id;

* colore associato alla macrocategoria.

Anche in questo caso si è seguito il modello di Angular, molto simile al Model View Controller. Il *modello* è rappresentato da una classe attività che possiede come attributi tutte le caratterisitiche precedenti. La *vista* è un *components* Angular compreso di un form e il *controller* da un servizio chiamato *ActivityService*. Quest'ultimo non gestisce solo la creazione di nuove attività, ma è un facade per tutti components che gestiscono le attività.
Un'altra vista importante è quella che permette la visualizzazione della lista delle attività. Qui l'architettura è più complicata e necessità la spiegazione di cosa sta dietro al facade. Infatti secondo il requisito RQ3O l'applicazione deve funzionare anche se offline, è per questo che la lista delle attività deve essere disponibile anche in assenza di connessione al server del gestionale. All'apertura dell'applicazione *ActivityService* inizializza un altro servizio con un comportamento simile al proxy,

ActivityStore, che si occupa di leggere nella tabella del database SQLite se sono salvate delle attività e caricarle nei dati dell'applicazione. Nel frattempo *ActivityStore* inizializza *ActivityFetcher* il quale si occupa invece dell'interrogazione dell'API, che restituirà una lista di attività se la connessione va a buon fine, oppure un errore. Nel caso ci sia risposta *ActivityStore* confronta il risultato con ciò che ha letto dal database e sistema le differenze. Nel caso contrario *ActivityFetcher* rimane in ascolto della rete dello smartphone (tramite plugin di Ionic) e una volta trovata, rilancia l'interrogazione al server.

3.1.4 Registrazione Attività

La parte di registrazione delle attività risulta essere molto più complicata per alcuni motivi:

* continuo invio di dati al gestionale per requisito RF11O;
* registrazione anche se l'app è in background;
* registrazione anche se lo smartphone è offline.

Questi punti hanno richiesto molto per essere modellati e più volte l'architettura di questa componente è stata modificata.
Ad alto livello quello che si tenta di fare è mettere in comunicazione il plugin di Ionic che riceve i dati sulla posizione e inviarli al gestionale. Qui entrano in gioco due servizi di Angular: uno si occupa di ricevere, registrare e salvare i log nel database locale dell'applicazione; un altro che si occuperà di verificare la rete periodicamente, e, nel caso fosse presente, procederà a caricare i log che trova all'interno della memoria. La caratteristica che devono avere i due servizi, che chiameremo *ActivityRecorder* e *uploader*, è di dover essere eseguibili anche in background. Inoltre l'uploader dovrà occuparsi di sincronizzare i propri processi, dato che potrà inviare una sola richiesta alla volta.

3.1.5 Progettazione delle viste

Uno dei problemi rilevanti dell'applicazione UCIS Report Tool era la mal progettazione delle viste. Io e il mio tutor abbiamo deciso di andare a creare un software il più usabile possibile e quindi di studiare un modo per organizzare nel miglior modo le viste.
La struttura generale dell'applicazione è una barra superiore con il nome della vista e a sinistra il classico pulsante per il menù "a sandwich". Il menù è a comparsa e contiene il link a tutte le viste, un pulsante per il logout e uno per le impostazioni.

Home La vista principale alla quale si viene reindirizzati all'apertura dell'applicazione è composta dalle informazioni sull'applicazione in generale, da una sezione dedicata all'account, dove sarà possibile visualizzarlo o effettuare un login se non ne è presente alcuno. Le viste successive saranno accessibili solamente se è stato eseguito un login valido.

Login La schermata di login è un semplice form con la possibilità di inserire il nome dell'utente e la sua password e un bottone per la conferma. Per il debugging è stato inserita la possibilità di fare richiesta anche al server di test e uno switch per scegliere di farle in HTTP o in HTTPS. Eventuali messaggi di errore, saranno elaborati e

visualizzati nella stessa schermata al di sotto del pulsante di conferma. Da questa schermata non sarà visibile il menù ma sarà possibile accedere alle impostazioni.

Lista delle attività Per accedere alla registrazione di un'attività è prima necessario selezionarne una da quelle create o in corso. È presente dunque una vista che elenca tutte le attività disponibili. Qui è da inserire un bottone per l'indirizzamento alla vista per creare una nuova attività.

Nuova attività Questa vista consiste in un form da compilare il quale richiede la selezione della macrocategoria e della categoria associata, del nome personalizzato e dal cane con il quale si inizia l'attività. Il form sarà ovviamente provvisto di un pulsante di conferma per l'invio dei dati.

Attività Premendo su uno dei record nella lista si aprirà la vista dedicata alla singola attività. Da qui si potrà avviare, mettere in pausa o terminare. Saranno visualizzate, inoltre, delle informazioni relative ad essa, come il numero di log registrati e inviati al server, il tempo trascorso durante l'attività e quelle inserite nella nuova attività.

3.2 Realizzazione

La fase di codifica è stata una semplice conseguenza di quello progettato in precedenza. Ormai potevo considerarmi sicuro degli strumenti che utilizzavo. Per la realizzazione ho proceduto in maniera modulare, prendendo in considerazione un modulo per volta.

3.2.1 Difficoltà riscontrate

Possiamo dividere le difficoltà in due categorie: quelle derivanti dalla progettazione e quelle legate al linguaggio. Riguardo alle prime sono state risolte e superate grazie al metodo agile che mi ha permesso assieme al tutor di andare a correggere anche durante la codifica errori o parti dell'architettura che sono risultate irrealizzabili così come progettate inizialmente. Da questo punto di vista non sono stati effettuati grandi cambiamenti.
Per quanto riguarda le difficoltà derivanti dal linguaggio abbiamo adottato diverse soluzioni che descriverò nelle sezioni successive.

Asincronia dei processi

Forse la fonte di errori è stata quella derivante dalla gestione della sincronia delle funzioni in Typescript. In questo linguaggio, come anche in Javascrpt, questo aspetto è delegato alle *Promise*, promesse in italiano. Le *Promise* sono degli oggetti che possono assumere tre stati: risolta, respinta, in attesa. Ai primi due stati è assegnata una funzione, chiamata callback, da eseguire nel caso la promessa li assuma. Tendenzialmente lo stato di *rejected*, respinta in italiano, ha associata una funzione per gestire l'errore derivato da una qualche chiamata.
Le promesse solitamente sono ritornate da funzioni asincrone, che non danno la disponibilità dell'oggetto in modo immediato. Infatti, la funzione una volta chiamata non viene aspettata di modo che ritorni un risultato, ma l'esecuzione continua subito dopo la chiamata. Una promessa non gestita è molto pericolosa, rischia di desincronizzare lo stack delle chiamate e rischia di creare concorrenza sulle risorse. D'altra parte queste

sono uno strumento molto potente. Permettono infatti a processi molto lunghi, come può essere una chiamata alle API, che può richiedere anche secondi, di non bloccare l'esecuzione dell'applicazione. Ed è per questo che le librerie e i plugin utilizzati nella maggior parte dei casi utilizzano le *Promise*.
Per gestirle a livello di codifica ho adattato una tecnica comune in questo linguaggio, che riguarda l'utilizzo del *then*. Questo è una funzione della Promise che può essere chiamata, ad esempio, da una chiamata di funzione e richiede una funzione inline (chiamate *lambda function* o *arrowfunction* in gergo) da eseguire non appena la promessa è stata risolta. Questo permette di inserire all'interno di quest'ultima funzione del codice da eseguire subito dopo che l'oggetto di ritorno è disponibile e utilizzabile. Di seguito un esempio di codice di una funzione che gestisce il salvataggio di un log.

```
saveBacklog(location: GeolocationResponse, item: LocationData)
    {
  await this.uploader.log(item)
      .then( (success) => {
          console.log('Inserted in Room Database', success);
          this.uploader.sync();
      })
      .catch( error => {
          console.log('Error converting to locationdata');
      });
  await this.saveALog(location, 'log')
      .then(() => {
          console.log('Log Saved');
      });
}
```

In questa porzione di codice possiamo notare che sull'oggetto uploader è chiamata la funzione log che restituisce una promessa. All'interno della promessa viene ritornato un valore booleano, quindi o vero o falso. Il *then* cattura questo valore all'interno della variabile *success* e solo successivamente chiama sull'pggetto uploader la funzione sync che verifica se è presente connessione e in caso positivo inizia a caricare i log in coda. Da notare due parole chiavi che ancora non sono state delucidatate. La prima è il *catch*, che esattamente come nel *then* gestisce il caso la promessa venga rifiutata. La seconda è *await* sconsigliata da utilizzare, ma in certi casi essenziale, chiede alla funzione di aspettare il risultato in arrivo dalla Promise prima di continuare l'esecuzione.

Observables, *Promise* e Angular

La gestione di componenti grafici in cambiamento legati al modello sottostante, come può essere l'indicatore numerico del numero dei log registrati, è stato per me un problema relativamente macchinoso da risolvere. Non c'è una tecnica condivisa nel gestire il legame tra un componente grafico e la struttura informativa sottostante in cambiamento. In Angular si usano molto spesso gli *Observables*, oggetti che corrispondo a un noto *design pattern*. Questo tipo di oggetti si dicono sottoscrivibili, cioè un soggetto può attraverso il comando subscribe reagire a un suo cambiamento. In pratica se su un oggetto di tipo Observables viene chiamata la sua funzione *next*, che cambia il valore di quest'ultimo e avvisa tutti i soggetti *subscriber* (cioè sottoscritti). Viene quindi implementato in questo modo: nella struttura sottostante, gli oggetti vengono definiti come *Observable*se all'interno del componente Angular vengono creati dei soggetti che rappresentano l'elemento grafico i quali si sottoscrivono ad esso.

Un'altra tecnica, quella che ho personalmente utilizzato di più, è quella dell'utilizzo delle *Promise*. Infatti questi oggetti, descritti nella sezione precedente, sono gestiti in automatico da Angular all'interno del component. Basterà quindi chiamare una funzione che restituisce un valore marcata *async* all'interno del file HTML, per renderla sensibile al cambiamento della stessa. Di seguito un esempio in un file HTML.

```
<ion-card>
  <ion-card-header>
      <ion-card-title>Stato</ion-card-title>
  </ion-card-header>
  <ion-card-content>
      <p>Stato: {{getStatus() | async}}</p>
      <p>Log: {{getLogNum() | async}}</p>
  </ion-card-content>
</ion-card>
```

3.3 Verifica

Il processo di verifica e validazione è stato avviato dopo la codifica di una Product Baseline funzionante.

3.3.1 Analisi statica

Dato che la maggior parte dell'applicazione è stata codificata in Typescript gli strumenti utilizzati per la verifica sono quelli standard del linguaggio, integrati anche nell'Integrated Development Environment (IDE). Per l'analisi statica si è utilizzato TSLint, reperibile al link:

TSLint. URL: https://palantir.github.io/tslint/

3.3.2 Analisi Dinamica

Per i test dinamici sull'interfaccia si è usato un framework apposito di Angular, Protractor anch'esso reperibile al link:

Protractor. URL: https://www.protractortest.org/#/

I test sono definiti come *behaviour driven*, vengono definiti degli input sull'interfaccia e dei comportamenti aspettati. L'applicazione viene poi lanciata sul browser per verificare che l'output dell'applicazione sia corretto.

3.3.3 Analisi umana

Alcune delle componenti dell'applicazione, purtroppo, non sono stati testati in modo automatico, perchè non esistente alcuno strumento per farlo. È il caso del test per il plugin di geolocalizzazione, non era possibile testarlo con nessun tool e questo si è tradotto in prove empiriche sul campo, durante gli spostamenti per andare in ufficio, ad esempio.

Capitolo 4

Conclusioni

4.1 Obiettivi raggiunti

Prima di cominciare lo stage assieme al relatore Paolo Baldane al mio tutor aziendale Francesco Costa è stato messo a punto un Piano di Lavoro, cioè un documento dove si è definito, in linea generale, gli obiettivi e la pianificazione dello stage. Di seguito la lista degli obiettivi programmati trattati in esso:

Tabella 4.1: Tabella degli obiettivi

Obiettivo	**Risultato**
Obbligatori	
Progettazione e realizzazione di applicazione mobile	Soddisfatto
Implementazione servizio di geolocalizzazione preciso nell'applicazione	Soddisfatto
Versione beta dell'applicazione da pubblicare sullo store	Soddisfatto
Desiderabili	
Applicazione cross-platform, combatibile anche su dispositivi datati	Soddisfatto
Messa in produzione dell'applicazione	Soddisfatto
Facoltativi	
Implementazione delle notifiche push	Non Soddisfatto
Integrazione di una chat	Non Soddisfatto
Applicazione funzionante anche su dispositivi senza servizi Google	Non Soddisfatto

In generale sono stati rispettati tutti gli obiettivi, tranne quelli ritenuti facoltativi. Quest'ultimi posso dire di averli trattati solo parzialmente e non averli completati. Durante lo stage ho infatti studiato i metodi per le notifiche push su Android e su come sviluppare applicazioni in dispositivi senza servizi Google, come ad esempio alcuni modelli recenti *Huawei*. Non nascondo un po' di rammarico nel non essere riuscito a completare tutti gli obiettivi, ma a posteriori posso dire che forse nel piano di lavoro siamo stati un po' troppo ambiziosi e non si sono considerati tutti i rischi. Per questo quelli facoltativi sono stati posti nell'ultima settimana, dando ovvia priorità agli altri. Considerando ciò posso comunque ritenermi soddisfatto del risultato ottenuto. Credo che parte il successo sia dovuto al metodo agile integrato nel nostro metodo di lavoro che ci ha permesso di settimana in settimana di definire chiaramente cosa sarei andato a fare, correggendo il piano se qualcosa avesse richiesto più tempo.

4.1.1 Valutazione della scelta

Per capire correttamente, perchè ho scelto la proposta di Alternative Stuidio, è bene specificare alcune cose.
Quello di cui ero alla ricerca era un contesto piccolo con ampi margini di crescita e di formazione personale sulle tecnologie moderne, più che sul metodo di lavoro, che invece tende a essere molto più definito in aziende grandi, con un modello affermato e utilizzato da tempo. Quello che mi interessava era sperimentare vari ruoli, comprendere le varie responsabilità che questi comportano e come interfacciarsi con altri dipendenti. Oltre al contesto aziendale ero attratto anche dall'argomento del progetto. Volevo sperimentare nuove tecnologie e nuovi linguaggi, disegnare e creare del software, piuttosto che mantenerlo o estenderlo. Da questo punto di vista l'Università di Padova, attraverso Stage-IT mi ha messo in contatto con molte aziende interessanti. Inoltre la mia ricerca del progetto di stage si è basata sul desiderio di lavorare in ambito frontend e questo ha ristretto di molto le varie opportunità.
Il fattore decisivo sulla scelta dello stage è stato il fatto che da tempo volevo sperimentare le tecnologie per lo sviluppo di applicazioni mobile. Io stesso, precedentemente, avevo provato ad avviare qualche piccolo progetto in questo campo e per questo ho voluto scegliere una proposta che avesse questo come tema principale.
In questo senso l'offerta di Alternative Studio ha subito catturato la mia attenzione e nonostante fosse la loro prima esperienza con uno studente laureando, ho preso contatti per avviare con loro il mio stage.

Obiettivi personali

Le esperienze lavorative precedenti a questa mi avevano insegnato cosa vuol dire lavorare in una squadra e cos'è prendersi carico di responsabilità, ma con nessuna di queste avevo messo in pratica ciò che avevo studiato finora. Sentivo quindi il bisogno di consolidare il bagaglio di nozioni acquisite durante gli anni del corso e lo stage ne è stata l'occasione perfetta. Certo, ci sono stati i molteplici progetti didattici, ma nulla è come entrare nel contesto lavorativo e avere contatto diretto con questa realtà. Di fatto lo stage ha unito due mondi che già conoscevo, quello lavorativo, ma sentendomi sicuro di me stesso, sapendo di essere qualificato per affrontare quello che mi aspettava. Riguardo lo stage in generale avevo aspettative simili che si possono riassumere nei seguenti punti.

* collaborare con colleghi con molta più esperienza e conoscenza di me;
* imparare nuove tecnologie e distinguere quelle buone da quelle cattive;

* migliorare la mia gestione del tempo e organizzarmi con le scadenze dei vari compiti;

* trovare proposte e contatti per progetti futuri.

Da questo punto di vista, a posteriori, credo sia stato un successo. Lo stage è riuscito a non deludermi e a soddisfare tutti gli obiettivi in maniera esaustiva. Di seguito spiego in che modo:

* Alternative Studio mi ha dato la possibilità di collaborare a stretto contatto con professionisti nel settore che mi hanno formato e hanno plasmato il mio metodo di lavoro. Sono stato contento di aver lavorato in un contesto giovane e in espansione, dove un giovane stagista può sentirsi a proprio agio crescendo assieme alla realtà in cui è immerso. Nonostante questo mi sento di aver imparato a distinguere ciò che è *professionale* da ciò che non lo è.

* Nella sezione successiva descriverò in dettaglio quali tecnologie ho imparato. In generale penso che la lunga analisi avviata nello studio di fattibilità mi abbia aiutato e fatto capire cosa mi propone il mondo informatico. Sono riuscito a sviluppare un metodo per esaminare e sviscerare le caratteristiche di ognuna delle tecnologie, che consiste nelle leggere la documentazione ufficiale, costruire una mia opinione e solo in seguito leggere opinioni di altri, cercando ovviamente le più autorevoli. Ho capito anche che un buon software non è composto solo da una buona codifica, una buona interfaccia o un buon supporto, ma da una grande community che fornisce un troubleshooting adeguato a tutti i problemi che puoi incorrere.

* Sono abbastanza soddisfatto di come sia riuscito a gestire il tempo. Ero molto preoccupato che la gestione elastica dettata dal telelavoro mi avrebbe penalizzato, date le distrazioni "casalinghe", o al contrario non avere orari di ufficio mi avrebbe portato a fare più ore di quelle stabilite. Non nascondo che è ovviamente capitato, quando uno sviluppatore si intestardisce nel trovare una soluzione a un problema può rimane anche ore su di esso, ma spesso il tutor è intervenuto a regolarizzare il ritmo lavorativo.

* Grazie a Stage-IT sono riuscito a raccogliere molti contatti. Mi piacerebbe continuare a collaborare con Alternative Studio nei suo progetti futuri.

4.2 Valutazione degli strumenti utilizzati

Gli strumenti che mi ha messo a disposizione Alternative Studio sono stato un notevole supporto al mio lavoro.

Gitlab Avendo avuto esperienza con altri software, come Github, posso dire che mi è sembrato completo in tutte le sue funzionalità. Il sistema di issues è stato utilissimo a notificare e salvare idee. Purtroppo alcune delle sue parti non sono state utilizzate a dovere durante il progetto, come quella che si occupa della Continous Intengration (CI) o le *operations*, piccoli script che vengono eseguiti a ogni push, che possono calcolare, ad esempio alcuni dei valori di verifica del codice.

Codifica WebStorm è un'IDE potentissimo, completo sotto ogni punto di vista. Ha moltissime funzionalità e supporta moltissimi linguaggi. Tuttavia l'ho trovato molto pesante per le risorse del computer, soprattutto la RAM, e con molti errori e bug, che spesso interrompono il workflow.

Postman Il software è risultato molte utile e perfetto per lo scopo. Mi ha permesso di testare le API, prima di procedere con la codifica. La creazione di un *environment*, cioè una collezione di richeste salvate e ordinate, si è rivelato ottimo per la collaborazione con gli altri componenti del team.

4.3 Valutazione delle tecnologie scelte

Ionic Sono rimasto molto soddisfatto dal framework scelto. Il supporto e le soluzioni proposte della community sono sempre state adeguate al problema. Ci sono alcuni punti negativi, quali una documentazione a tratti troppo sintetica o incompleta e la scelta limitata dei plugin nativi disponibili. Questo ci ha portato durante il progetto a sviluppare un plugin apposito per la gestione degli account tramite il manager a disposizione dei dispositivi Android. In generale l'effettiva versatilità del sistema e la disponibilità di librerie grafiche moderne e piacevoli sono state motivo per apprezzare sempre di più questo potente framework.

Angular È stato un eccellente piattaforma nel quale sviluppare l'applicazione. Ho avuto molte difficoltà nel gestire le risorse asincrone, penso che esistano sistemi meno macchinosi. Ho apprezzato invece l'architettura che garantisce una modularità quasi obbligata e quindi uno sviluppo eccellente da parte di un team.

Plugin utilizzati Nonostante Ionic offrisse dei plugin cosiddetti *nativi*, questi non erano altro che adattatori per Capacitor, di plugin sviluppati per Cordova. Sono rimasto a lungo a leggere la loro documentazione che spesso non si è rivelata professionale, si parla, comunque di progetti open source. Non nego inoltre di essere spesso andato a leggere il loro codice per capire come dovessero essere utilizzati. Tuttavia questa scelta è stata fatta in maniera quasi obbligata date le risorse a disposizione del progetto.

4.4 Considerazioni sul prodotto

Quello che ho ottenuto alla fine del progetto, non è certamente un prodotto che può considerarsi completo. Rispetta effettivamente i requisiti imposti inizialmente, ma possiede ancora alcune imperfezioni che sicuramente verrano corrette con il tempo. L'applicazione non è ancora stata resa disponibile ai membri di UCIS, ma è stata testata da alcuni. Le impressioni sono ottime ed è saltato all'occhio il confronto con l'applicazione precedente. È vero che l'impatto su un utente inesperto di un pacchetto grafico moderno e accattivante, è sicuramente forte, ma sono stati notate le soluzioni ai problemi dell'applicazione precedente.
Sono state definite comunque delle posibili estensioni al progetto. Si è programmato di portare a termine gli obiettivi facoltativi, implementazione delle notifiche push, integrazione della chat e l'applicazione funzionante senza serivizi Google. Inoltre si stanno prendendo accordi per sviluppare la versione iOS, alla quale spero di partecipare.

4.5 Conoscenze e competenze acquisite

4.5.1 Collaborazione con i colleghi

Il lavoro di squadra era uno dei punti che più mi preoccupava. Prima dello stage ero convinto che iniziare un progetto in più persone senza dividere bene i ruoli e le mansioni in modo definito fosse impossibile. Dopo questa esperienza mi sento di dire che lavorare in un team cercando il confronto continuo e avendo sempre qualcuno a supporto è il modo migliore per raggiungere gli obiettivi.
Ho capito che il fallimento è contemplato in questo settore e che se una cosa non riesce puoi sempre rivolgerti a qualcun altro che ti darà una visione diversa del problema. Questo approccio mi è tornato utile molto spesso durante il progetto quando mi trovavo fermo in punto, senza trovare vie d'uscita, rivolgermi al tutor è sempre stata la soluzione migliore. Un'altro aspetto al quale tengo molto e la gestione delle relazioni tra colleghi che deve andare oltre al semplice rapporto lavorativo, per creare un ambiente rilassato e confortevole, dove è più semplice lavorare. Ho imparato quindi a essere professionale, ma al contempo non troppo rigido.

4.5.2 Analisi e Studio di Fattibilità

Prima di intraprendere il percorso di stage la consideravo una fase noiosa e poco utile di un progetto. Ho completamente cambiato idea affrontandola con il tutor. In precedenza, probabilmente avevo un approccio meno interattivo e un po' retrogrado, ma con il tutor sono riuscito a cambiarlo radicalmente.
Dato che i requisiti erano già stati quasi tutti fissati, lo studio delle tecnologie da utilizzare è quello che mi ha coinvolto di più. Capire le caratteristiche di ognuna di esse e apprezzarne le differenze mi ha appassionato portandomi ad apprezzare questo processo che prima ritenevo superfluo. È da sottolineare che questo progetto partendo da zero, aveva bisogno di un analisi completa e questo mi ha dato molta libertà nella ricerca.

4.5.3 Tecnologie apprese

Oltre ad affinare l'utilizzo di Git e degli strumenti di controllo di versione, ho appreso meglio il concetto di CI attraverso gli strumenti di GitLab. Ho conosciuto il mondo dello sviluppo mobile e di tutte le tecnologie che ci stanno attorno, ordinando le conoscenze acquisite in precedenza. È stato affascinante passare da utente di applicazioni per smartphone a programmatore, iniziando a notare tutti gli stessi errori e le imperfezioni che altri sviluppatori hanno commesso come me su alcune cose.
Ho imparato a programmare con Ionic e Angular, che sicuramente mi saranno utili in futuro nel mondo del web development. Ora sono in grado di gestire in modo efficiente la programmazione concorrente attraverso le librerie di Node.js con le *Promise*. So interfacciarmi con delle API su un server web. Ho imparato, inoltre, a avviare un progetto in Android tramite il framework nativo e modificare le sue parti in XML e Java.

4.6 Valutazioni personali

Ritengo di aver avuto parecchie aspettative sullo stage finale, considerandolo la conclusione di un percorso che mi avrebbe lanciato poi nel mondo del lavoro e testando

tutte le mie conoscenze apprese durante gli studi. Ho scoperto però che non è stato del tutto così. Avrei dovuto affrontarlo con aspettative più basse e un altro tipo di mentalità. Non è stata un'esperienza negativa, al contrario ho imparato molto e credo sia quella più utile proposta dal corso di laurea, ma da questa ho scoperto che mi mancano ancora moltissime competenze e conoscenze per essere del tutto pronto al mondo del lavoro.
Nel complesso mi sento soddisfatto e credo di aver fatto un ottimo lavoro, mettendo tutto me stesso nel progetto e impegnandomi nella sua realizzazione. Grazie al contesto aziendale nel quale mi sono trovato ho avuto tutti gli strumenti per crescere e spero di trovarne di simili nel prossimo futuro. So che dovrò spendere molto più delle 300 ore messe a disposizione dallo stage per sentirmi completamente realizzato e pronto per diventare un vero e proprio informatico.

Acronimi

API Application Programming Interface. 1, 4, 13, 15, 19, 27

ASD Alternative Studio. 1, 2, 4, 10, 22, 23, 27

CI Continous Integration. 23, 25, 27

CLI Command Line Interface. 12, 27

HTML HyperText Markup Language. 20, 27

HTTP HyperText Transfer Protocol. 3, 14, 17, 27

IDE Integrated Development Environment. 3, 20, 24, 27

MVC Model View Controller. 15, 16, 27

SQL Structured Query Language. 13, 27

UCIS Unità Cinofile da Soccorso. 1, 4, 24, 27

UML Unified Modeling Language. 27

URI Unified Resource Identifier. 14, 27

XML eXstensible Markupd Language. 25, 27

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 27

Continous Integration In ingegneria del software, la Continous Intengration è una pratica che consiste nell'allineare periodicamente tutti gli spazi di lavoro degli sviluppatori coinvolti nello stesso progetto. La conseguenza è che ogni sviluppatore lavorerà sempre su un progetto aggiornato da ognuno. . 27

HTML È il linguaggio di markup standard per i documenti visualizzabili nel web. Questi sono letti e interpretati da i browser web, i quali leggono i fogli di stile in scritti in CSS e gli script in JavaScript. . 27

HTTP È un protocollo standard per architetture client-server, per l'invio e la ricezione di pagine web, ora diffuso nella versione HTTPS. . 27

IDE Si definiscono IDE ambienti di sviluppo che integrano l'editor per il codice e tutti gli strumenti per il controllo e lo sviluppo. Gli IDE sono studiati per facilitare la codifica con, ad esempio, suggeritori, controlli sulla sintassi, controlli statici, compilazione e building, debugging. . 27

UCIS É un'Associazione Nazionale di Volontariato, inserita nell'Albo istituito presso il Dipartimento di Protezione Civile. Raggruppa, tutela e coordina i Soccorritori Cinofili presenti sul Territorio Nazionale. 27

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di lingua franca nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 27

XML è un linguaggio marcatore (di markup in inglese), cioè attraverso dei tag è possibile definire e caratterizzare gli elementi, solitemente del testo, contenuti al loro interno. La particolarità di XML è che rende possibile creare i propri tag e per questo viene chiamato estensibile (eXstensible). . 27

agile In ingegneria del software è un insieme di modelli di sviluppo, nate in contrapposizione alla rigidità di quelli precedenti. Questi si basano su quattro principi fondamentali:

* gli individui e l'interazione è più importante dei processi e degli strumenti
* il software funzionante è più importante della documentazione comprensibile
* la collaborazione con il cliente piuttosto che la negoziazione del contratto
* la risposta al cambiamento è più importante di seguire il piano

. 1, 5, 18, 22, 27

Android È un sistema operativo open source per dispositivi mobili scritto su kernel Linux. Sviluppato da Google, esce con il primo smartphone nel 2007, ora è il sistema operativo mobile più diffuso al mondo. Le applicazioni installabili sono fornite tramite pacchetti APK, che, per i dispositivi che hanno installato i servizi Google, sono scaricabili da Play Store.. 4, 13, 22, 27

back-end Nella progettazione software e sviluppo è la parte del sistema che gestisce le componenti non visibili all'utente. Semplificando è la logica che si occupa di elaborare i dati provenienti da un'interfaccia, producendo un output o modificando lo stato del sistema. . 12, 27

background In questo elaborato con il termine background si indica lo stato di un applicazione che sta eseguendo, ma che non può essere controllata dall'utente. Si oppone al termine foreground che definisce quando un utente esegue l'applicazione ma può interagire direttamente con essa. . 9, 13, 27

C++ È l'evoluzione di C, un linguaggio ad alto livello molto utilizzato negli anni '70-'80, che introduce la programmazione orientata agli oggetti. Le sue caratteristiche sono le perfomance, l'efficienza e la flessibilità. Permette infatti la gestione della memoria a basso livello, implementado il concetto di costruttore e distruttore. Il linguaggio e completamente compilato e non possiede nessun layer per l'interpretazione, come ad esempio Java.. 11, 27

C# È un linguaggio orientato agli oggetti sviluppato attorno al framework .NET di Microsoft per lo sviluppo nel suo sistema operativo, Windows. È sviluppato per supportare la creazione di componenti e gestirne l'interazione. L'obiettivo degli sviluppatori era quello di creare un linguaggio che potesse girare in sistemi complessi, come sistemi operativi, ma anche in sistemi integrati di piccoli dispositivi.. 11, 27

CLI Indica una particolare tipologia di interfaccia utente completamente a riga di comando. Solitamente le applicazioni utilizzano la bash (o shell testuale) del sistema operativo nel quale sta eseguendo. . 27

cross-platform Si definisce software cross-platform o multipiattaforma un applicazione implementata su più piattaforme. Il termine piattaforma può riferirsi all'hardware o al sistema operativo. Due piattaforme diverse possono quindi differire per l'architettura fisica o per quella software, come può essere nel caso di un dispositivo Apple, nel quale gira iOS, e Android. . 5, 27

Daily Stand-up È una pratica che fa parte del metodo Scrum che prevede di fare una riunione giornaliera di massimo 15 minuti, nella quale ogni membro del team dice cos'ha fatto prima del meeting e pianifica cosa andrà a fare fino al successivo. . 2, 27

database Base di dati in italiano, è una collezione di dati organizzati secondo un modello e una struttura. I più diffusi sono i database relazionali, cioè i dati sono salvati in tabelle in relazione tra di loro. . 13, 27

diagramma di Gantt È una tipologia di diagramma usata principalmente nella pianificazione di progetto. L'asse orizzontale rappresenta il tempo, suddiviso in fasi e quello verticale le attività. Le barre nell'area del grafico descrivono la durata delle attività stesse, che si possono sovrapporre in verticale, indicando la possibilità di uno svolgimento parallelo di quest'ultime.. 27

facade In progettazione software, il proxy è un pattern architetturale. Tutte le funzionalità contentute in un sistema vengono richiamate da un'unica classe, chiamata appunto facade. Questa tecnica viene utilizzata in sistemi complessi per ridurre le dipendenze e modularizzare le componenti. . 16, 27

framework Nello sviluppo software indica un insieme di strumenti e un'architettura a supporto del prodotto sul quale potrai andare a progettare e realizzare il prodotto. Spesso viene utilizzato per indicare un'insieme di librerie corredate da software che le integrano in maniera facilitata a ciò che le utilizza . 27

front-end Nella progettazione software e sviluppo è la parte del sistema che va a interagire direttamente con l'utente. L'esempio più banale può essere quello dell'interfaccia grafica. . 27

Git È un software per il controllo di versione, nato ufficialmente nel 2005. Venne scritto inizialmente come strumento per lo sviluppo del kernel di Linux, sistema operativo open source molto diffuso, poi rilasciato al pubblico e allargato agli altri sistemi operativi, come Windows e Mac. L'interfaccia è a riga di comando (CLI) ed è sviluppato tramite C++, Python e altri linguaggi che lo rendono molto leggero e versatile. . 2, 3, 27

Go È un linguaggio open source sviluppato da Google, che si contraddistingue per essere particolarmente efficiente nel risolvere problemi legati alla concorrenza.. 3, 27

Google Google LLC è una delle più importanti aziende che opera nel settore informatico e offre servizi prevalentemente online. Ad essa è associato il nome dell'omonimo motore di ricerca dal quale la società è partita ad espandersi, diventando l'attuale colosso leader del settore a livello mondiale. . 27

HTML5 È l'ultima versione del noto linguaggio HTML, mantenuta da un consorzio formato dalle maggiori società che producono browser.. 27

incrementale È un concetto di Ingegneria del Software, che indica che il modello di sviluppo di un progetto viene ripetuto ad ogni incremento. In pratica il prodotto viene portato a una fase funzionante iniziale e da questo momento si potranno applicare incrementi, cioè i passi dello sviluppo software (tranne analisi, progettazione) verranno ogni volta rieseguti sul prodotto ottenuto. . 1, 27

iOS È un sistema operativo proprietario di Apple Inc. per i suoi dispositivi mobili. Le applicazioni disponibili per iOS si possono scaricare da App Store.. 9, 27

Issue Tracking System Sono dei particolari software che permettono la pubblicazione, la gestione e la rimozione di issues (in italiano problemi/questioni). Sono utilizzati in combinazione con VCS come Git per tenere traccia di alcune criticità del software, rendendole visibili agli altri programmatori. . 2, 27

Java Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti. Per essere il più indipendente possibile dalla piattaforma hardware di esecuzione si appoggia alla JVM (Java Virtual Machine) che interpreta il bytecode, cioè un file di output derivante dalla compilazione. Per questo viene considerato come un linguaggio semi-interpretato. . 3, 9, 25, 27

Node.js È un environment per l'esecuzione runtime di JavaScript. Nel web development permette, lato server, l'esecuzione di script in linguaggio, scaricando l'onere dal client.. 25, 27

open source Un software è definito open source, se il suo codice sorgente è accessibile pubblicamente. Questo comporta a sancire delle politiche di distribuzione che sono regolamentate da particolari licenze (MIT o Apache ad esempio). Un software open source è considerato libero, in contrapposizione al concetto di gratuito. Esso infatti è posseduto molto spesso da una azienda software che concede l'utilizzo, la modifica e la distribuzione libera regolamentate, però, dalle licenze succitate. . 8, 9, 27

Product Baseline In ingegneria del software indica gli attributi attribuili a un prodotto software in un punto definito nel tempo. . 20, 27

proxy In progettazione software, il proxy è un pattern architetturale. La classe che funge da proxy non è altro che un interfaccia per una componente che vuol essere nascosta al resto dell'architettura. . 16, 27

Scrum È un metodo di sviluppo che sposa il manifesto agile. Questo definisce com'è composto un team e quali sono i comportamenti da avere all'interno di esso. Il metodo Scrum si sposa bene con il modello incrementale, questo infatti prevede degli eventi tra i quali lo Sprint, il Daily Stand-up, la rethrospective, che vanno ripetuti ad ogni incremento. . 2, 27

Software Development Kit È una raccolta di strumenti per lo sviluppo software installabile in un singolo pacchetto. Per lo sviluppo con Java, ad esempio, è necessario un SDK che prevede le sue librerie di base, il suo debugger e il suo compilatore. . 9, 27

Sprint Lo sprint è un'intervallo di tempo, solitamente di 2-4 settimane, utilizzato nel metodo Scrum come unità di misura che intercorre tra una revisione e un'altra. Durante questo periodo ogni lavoro assegnato dovrà essere svolto e pronto per il successivo sprint. . 2, 27

Tipizzato Si riferisce a una caratteristica di un linguaggio di programmazione nel quale ogni programmatore deve associare ad ogni variabile il proprio tipo. Il compilatore associato deve garantire poi che le caratteristiche associato ad ogni tipo siano rispettate (tipizzazione statica). . 27

Bibliografia

Siti web consultati

Cordova vs Capacitor. URL: https://ionicframework.com/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development.

Don't Kill My App. URL: https://dontkillmyapp.com/ (cit. a p. 13).

Firebase Cloud Messaging. URL: https://firebase.google.com/docs/cloud-messaging.

Hybrid vs.Native. URL: https://cdn2.hubspot.net/hubfs/3776657/Ionic%20eBook%20-%20Hybrid%20vs%20Native.pdf.

Introduction to Angular concepts. URL: https://angular.io/guide/architecture.

Protractor. URL: https://www.protractortest.org/#/ (cit. a p. 20).

TSLint. URL: https://palantir.github.io/tslint/ (cit. a p. 20).