Università degli Studi di Modena e Reggio Emilia

DIPARTIMENTO DI SCIENZE FISICHE, INFORMATICHE E MATEMATICHE
Corso di Laurea in Informatica

Tesi di Laurea

Una web application per la gestione dei rapporti d'intervento

Candidato:
Luca Bonetti

Relatore:
Chiar.mo Prof. Giacomo Cabri

Anno Accademico 2020-2021

A tutti coloro che mi supportano e sopportano (ardua impresa), per avermi sostenuto e per non avermi mai abbandonato.

Indice

Capitolo 1

Introduzione

ReportManager è una web application per la gestione della compilazione, dell'archiviazione e della stampa dei rapporti d'intervento dei dipendenti di INFOLOG SpA presso i propri clienti.
Nel prosieguo saranno illustrate la realtà aziendale, le esigenze che hanno portato alla concezione del software e i requisiti scaturiti, l'architettura scelta per la realizzazione del progetto e le tecnologie impiegate; saranno dunque illustrate e discusse le funzionalità implementate e si concluderà con alcune riflessioni su quanto è stato realizzato, su ciò che è stato preso in considerazione e su ciò che manca.
Nel vasto panorama odierno della programmazione web, si sono affermati negli anni diversi metodi di scrittura delle applicazioni che vengono eseguite sui browser: dapprima tutto era gestito dal server, compresa la renderizzazione delle pagine da mostrare, poi si è diffuso il paradigma REST, fino ad arrivare a oggi, dove a far da padrone sono le tecnologie serverless e le lambda-function in ambienti distribuiti.
ReportManager nasce sulla base del consolidato paradigma REST. Ogni dettaglio sarà ampiamente trattato nei capitoli seguenti.

Capitolo 2

L'azienda

INFOLOG SpA[1] è una grande realtà aziendale del territorio modenese che si occupa da almeno 25 anni di progettazione e sviluppo di soluzioni software. L'attività nasce negli anni '80 e si incentra fin da subito sul settore gestionale e su quello logistico. Con l'avvento del nuovo millennio, nei laboratori viene adottato il linguaggio di programmazione Java[2] come standard e inizia una progressiva e costante crescita che porta la ditta alla trasformazione in S.P.A., avvenuta nel 2005. Questa espansione continua nel 2012 con l'acquisizione di Netfabrica srl, attiva nei settori di assistenza e consulenza informatica. Nel 2020, infine, la presenza sul mercato dell'azienda si rafforza ulteriormente grazie all'ingresso in Var Group[3] del Gruppo Sesa[4].
Internamente, INFOLOG SpA (da questo punto in poi chiamata semplicemente INFOLOG per brevità) è strutturata in 5 reparti operativi:

- L'amministrazione si occupa della gestione del personale, della sovrintendenza e della fatturazione

[1] https://www.infolog.it/
[2] https://www.java.com/it/
[3] https://www.vargroup.it/
[4] https://www.sesa.it/

- Il reparto tecnico comprende operatori specializzati per la gestione della strumentazione e delle reti interne e per l'installazione dei sistemi presso i clienti

- AS400 è la divisione che si occupa dello sviluppo e della manutenzione degli omonimi sistemi legacy (oggi IBM i[5])

- Unitegy è la branca che si occupa della soluzione di gestione delle attività commerciali, basata sul software open source Compiere[6]

- Logistics è il reparto più numeroso e attivo negli ambiti di ricerca e sviluppo ed è responsabile della produzione, dell'integrazione e della customizzazione di INTELLIMAG, il gestionale di magazzino proposto in ambito nazionale e internazionale

La realizzazione del progetto oggetto di questa tesi è stata svolta presso INFOLOG con l'aiuto del team Logistics.

[5]https://www.ibm.com/it-it/it-infrastructure/power/os/ibm-i
[6]http://www.compiere.com/

Capitolo 3

Il progetto

In un contesto aziendale come quello descritto nel capitolo 2, è necessaria un'organizzazione del lavoro precisa, puntuale e soprattutto condivisa da parte dei diversi team. È infatti impensabile che differenti unità di lavoro non condividano mezzi efficaci per le attività interdipartimentali come la pianificazione e la gestione dei compiti. Un ulteriore importante aspetto è legato alla compilazione dei rapporti d'intervento effettuati presso i clienti. Questi costituiscono un rendiconto delle operazioni svolte su software e hardware, devono essere appositamente controfirmati da un responsabile dell'azienda presso la quale viene svolto il lavoro e vanno consegnati all'amministrazione per l'emissione delle fatture di pagamento.
INFOLOG mette a disposizione dei propri dipendenti diverse tecnologie a supporto della stesura di tali documenti:

- Un software locale offline
- Un modulo stampabile e compilabile a mano
- Una parte di Unitegy che consente di esportare un file in formato PDF

Il progetto proposto si pone l'obiettivo di unificare le diverse modalità con cui vengono compilati i rapporti d'intervento, integrandosi coi sistemi aziendali esistenti. Un'unica piattaforma che riunisca tutti i documenti emessi consentirà all'amministrazione una maggiore autonomia e faciliterà l'emissione delle fatture, consentendo una progressiva migrazione dei dati esistenti di diversa natura, una progressiva dismissione dei vecchi servizi e fornirà altresì un singolo punto dal quale reperire le informazioni.

3.1 I requisiti

Le esigenze di modernità e di fruibilità del sistema hanno portato alla decisione di realizzare un portale web fruibile da browser su differenti dispositivi, eventualmente convertibile in app per smartphone in un secondo momento. Dovranno essere messe in campo diverse integrazioni per consentire a tutti gli utenti interni all'azienda di poter fruire dei servizi a loro dedicati. A tale scopo, dovrà essere integrato il servizio di autenticazione LDAP fornito da Microsoft Active Directory[1], presente all'interno dei server aziendali e gestito dal reparto IT. I programmatori del reparto Logistics, inoltre, dovranno essere in grado di utilizzare il software Atlassian Jira[2] tramite il portale; questo particolare programma consente una migliore gestione dello sviluppo attraverso la frammentazione delle caratteristiche da implementare in unità semplici e permette, attraverso un vasto panorama di plug-in installabili, di gestire aspetti avanzati come le ore uomo impiegate su un singolo compito. Gli utenti Logistics avranno la possibilità di inserire direttamente i task di Atlassian Jira nei rapporti d'intervento, annettendo automaticamente le informazioni relative alle ore uomo. Per il resto degli utenti, sarà invece

[1] https://docs.microsoft.com/it-it/azure/active-directory/fundamentals/auth-ldap
[2] https://www.atlassian.com/it/software/jira

possibile l'inserimento manuale di suddetti dati. Dovrà essere consentita la registrazione delle firme dei clienti e dei dipendenti tramite tablet o dispositivo idoneo; un utente della piattaforma potrà inserire una firma di default da poter applicare direttamente al rapporto d'intervento senza avere la necessità di immetterla di volta in volta. Dovrà essere possibile l'invio del rapporto compilato a un particolare indirizzo email, sia esso quello del cliente o dell'amministrazione interna o esterna, tramite un account di Microsoft Office 365[3] opportunamente configurato e attraverso il protocollo SMTP. I dati dei clienti per la compilazione dei rapporti dovranno essere letti direttamente da una particolare vista messa a disposizione dai programmatori di Unitegy; allo stesso modo, i rapporti compilati dovranno essere scritti e dunque inseriti direttamente in un'altra vista dello stesso database Oracle[4]. Infine, il progetto dovrà supportare un proprio database PostgreSQL[5] integrato, per memorizzare i rapporti d'intervento compilati in modo centralizzato. Questo offrirà ridondanza all'intero processo di gestione dei report, perché se uno tra i sistemi Unitegy e quello del progetto dovesse subire danneggiamenti ai dati, l'altro offrirebbe comunque un backup allineato degli stessi.
Lo stack tecnologico scelto per la produzione del software rispecchia le moderne tendenze in ambito enterprise. Data la pregressa esperienza di INFOLOG in merito, è stata scelta la realizzazione di una API sul modello RESTful con Spring Boot per esporre i servizi ed elaborare i dati e di un applicativo Single Page Application con Angular per la parte grafica.

[3]https://www.microsoft.com/it-it/microsoft-365/
[4]https://www.oracle.com/it/database/
[5]https://www.postgresql.org/

Capitolo 4

L'architettura

Come anticipato nel capitolo 3, l'architettura del sistema progettato segue il modello API RESTful. Questa tipologia di paradigma parte dal concetto di interazione client/server e comprende una netta suddivisione tra gli aspetti legati alla logica di presentazione dei dati e quelli legati alla loro elaborazione e memorizzazione. Nel complesso, questa separazione si concretizza tipicamente nella produzione di due progetti quasi completamente indipendenti. L'unico nesso tra queste due parti, chiamate frontend per la presentazione e backend per elaborazione e memorizzazione, è l'interfaccia di comunicazione. Il protocollo HTTP funge da unico collante per la trasmissione dei dati in entrambi i sensi.

4.1 Il paradigma REST

La struttura teorica prevede la costruzione di differenti punti di contatto, detti endpoint, che il frontend richiama quando necessita di uno o più dati. Questi punti di contatto possono restituire informazioni, come nel caso di chiamate HTTP di tipo GET, possono memorizzarne, come nel caso di

chiamate HTTP di tipo POST, possono modificare la situazione esistente, come nel caso di chiamate HTTP di tipo PUT o PATCH e possono eliminare entità persistenti, attraverso chiamate HTTP di tipo DELETE. Va precisato che il mapping tra funzionalità e verbi HTTP non rende di per sé una API RESTful.

In un'architettura di tipo client/server generica, un server espone servizi verso uno o più client che lo interrogano per usufruirne. Il server è responsabile della manutenzione della sessione di ogni client che comunica con esso. La sessione rappresenta sostanzialmente l'insieme dei dati che servono a identificare l'utente e a distinguere le richieste consentite in base a quelle precedenti. REST, acronimo di REpresentational State Transfer, non sfugge a questa definizione ma ne deriva e ne modifica i vincoli. Il paradigma è stato introdotto nella tesi di dottorato di Roy Fielding [1], celebre informatico statunitense contemporaneo. Rispetto al tradizionale concetto di stato persistente mantenuto all'interno della parte server dell'applicativo, questo nuovo approccio definisce la comunicazione verso esso come stateless, ovvero senza stato. In quest'ottica, un client che intenda usufruire di un particolare servizio del server dovrà fornire ad esso tutti i dati necessari perché la propria richiesta sia soddisfatta. Il server, d'altro canto, non avrà alcuna memoria rispetto alle differenti richieste giunte dal client, da cui la definizione di stateless (senza stato). Il principale vantaggio di non avere una sessione mantenuta nel backend è un enorme alleggerimento nella gestione di client differenti e/o non omogenei. Il ruolo del client, tuttavia, subisce un netto cambiamento: la sessione deve ora essere immagazzinata, infatti, in questa porzione dell'applicativo. A partire dalla dissertazione con cui Fielding introduce nel 2000 questo nuovo stile architetturale, REST è diventato uno standard de-facto per i più moderni servizi internet. Il progressivo arricchimento delle risorse

disponibili ai client ha reso la manutenzione di una sessione progressivamente meno esosa in termini di costi di elaborazione e ha sicuramente favorito la diffusione del pattern. Un ulteriore aspetto che ha permesso a diverse aziende di adottare REST come metodologia di sviluppo standard per sistemi distribuiti e non è l'indipendenza tra frontend e backend. Fintanto che l'interfaccia tra le due parti non viene modificata, infatti, i due progetti possono evolvere indipendentemente l'uno rispetto all'altro e possono essere realizzati da team differenti o perfino da società diverse.
Il nome scelto per l'applicazione è ReportManager. L'architettura API RESTful è stata implementata attraverso due distinti componenti: backend e frontend.

4.2 Il backend

Il backend è costituito da una web application, ovvero un software fruibile attraverso il web e in particolare tramite protocollo HTTP. Java è il linguaggio di programmazione di riferimento per questa parte del software e per tale ragione è stato impiegato il servlet container Tomcat per poter debuggare e poi mettere in produzione quanto realizzato. Il servlet container è un particolare strato che si frappone tra l'utenza del servizio e il compilato Java che esegue i comandi; è necessario perché fornisce già la traduzione e il mapping delle chiamate HTTP, reindirizzandole agli endpoint definiti. In questo modo, prendendo l'esempio di una chiamata di tipo GET, verranno richiesti dati al giusto URL definito per questa interrogazione. Come anticipato, il framework di riferimento per questa porzione del progetto è Spring Boot.

4.3 Il frontend

Il frontend è costituito da una Single Page Application sviluppata, invece, con il framework Angular. Una Single Page Application, comunemente abbreviata con l'acronimo SPA, è fondamentalmente una pagina web caricata totalmente al primo accesso. Nel momento in cui un utente utilizza una SPA, nell'accedere all'URL corrispondente a essa otterrà immediatamente l'intero applicativo sul proprio browser. Il sistema si differenzia rispetto a una normale pagina internet poiché non c'è alcun rallentamento dovuto al caricamento di pagine intere nella navigazione; infatti, grazie all'interattività fornita da linguaggi di scripting come JavaScript, i contenuti sono completamente dinamici e vengono aggiornati quasi in tempo reale. L'esperienza di fruizione risulta notevolmente appesantita solamente in una fase di caricamento iniziale, diventando pressochè istantanea nel prosieguo. L'impiego di tecniche asincrone come AJAX, inoltre, consente di raggiungere risultati che fino a qualche tempo addietro si pensavano impossibili. AJAX, acronimo di Asincronous JavaScript and XML, è una tecnica che consente di disaccoppiare l'invio e la ricezione di dati verso e da server dalla logica di presentazione delle pagine web. AJAX non è una libreria e non è un linguaggio di programmazione; consente, tuttavia, di effettuare chiamate asincrone anziché sincrone. La sostanziale differenza sta nel fatto che una chiamata sincrona deve attendere un esito prima di proseguire con l'elaborazione mentre una asincrona no. La logica sincrona impedisce di procedere con il flusso dei dati e, eventualmente, di mostrare a video i risultati delle operazioni richieste dall'utente. Quella asincrona, viceversa, constente alle pagine di visualizzare le informazioni una volta pronte, una volta ricevute.
Nel prossimo capitolo saranno approfonditi gli aspetti tecnologici legati ai framework utilizzati sia per il backend che per il frontend.

Capitolo 5

Le tecnologie

5.1 Angular

Il frontend del software è stato realizzato con Angular[1]. Angular è un framework di sviluppo per interfacce e applicazioni web sviluppato e mantenuto da Google. Viene distribuito tramite licenza simile a quella MIT e ciò rende possibile visionare, modificare e redistribuire il codice o parte di esso senza alcun limite. La prima versione della piattaforma risale al 2010 e prende il nome di AngularJS; l'avvento di questo strumento ha consentito a moltissimi programmatori di avvicinarsi allo sviluppo di interfacce grafiche e pagine web sempre più dinamiche grazie all'uso massivo di JavaScript, linguaggio di programmazione implementazione dello standard ECMAScript[2], per l'interattività. Nel 2016, il software viene completamente riscritto per adattarsi alle esigenze di modernità del panorama di sviluppo emergente. Il nome viene cambiato in Angular e nasce il progetto di cui ReportManager

[1] https://angular.io/

[2] https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

fa uso. Attualmente, questo strumento è scritto in TypeScript[3], un superset di JavaScript. Un superset, nel gergo informatico, indica una tecnologia che ne estende una già esistente, arricchendone le caratteristiche e migliorandola. Nel caso di TypeScript, la differenza sostanziale rispetto a JavaScript è legata alla tipizzazione. Se uno dei maggiori punti a sfavore di linguaggi di natura dinamica come JavaScript e Python[4] è l'uso di meccanismi complessi per la comprensione dei tipi delle variabili, soprattutto per progetti di medie e grandi dimensioni, TypeScript offre una soluzione a questo presunto difetto fornendo agli sviluppatori la sintassi per introdurre la tipizzazione forte. In realtà i vantaggi sono visibili tipicamente durante la fase di sviluppo, perché TypeScript viene poi tradotto in JavaScript nella fase di compilazione e interpretazione del codice. L'attività dello sviluppatore risulta migliore soprattutto grazie alla possibilità di definire per ogni oggetto di tipo JSON la natura di ogni parametro espresso; questo dà l'enorme vantaggio di non doversi più affidare unicamente al nome di una variabile o al suo uso nel ciclo di vita del software per comprenderne l'effettivo impiego.

Angular offre numerose altre caratteristiche, oltre al TypeScript. Prima di tutto, è basato su Node.js[5], il runtime JavaScript basato sul motore V8 di Chrome (il browser più diffuso al mondo). Nasce come framework multipiattaforma, rendendo possibile lo sviluppo di applicativi che possano essere eseguiti correttamente su browser diversi, su sistemi desktop e su smartphone dotati di sistemi operativi Android o iOS grazie a interfacce terze come il progetto Apache Cordova[6]. Il sistema è stato riprogettato al momento della transizione da AngularJS[7] con velocità e performance come punti cardine,

[3]https://www.typescriptlang.org/
[4]https://www.python.org/
[5]https://nodejs.org/it/
[6]https://cordova.apache.org/
[7]https://angularjs.org/

rendendo qualunque applicazione realizzata con esso estremamente rapida e professionale già con le semplici impostazioni di default. Moltissimi ambienti di sviluppo integrati (o IDE) offrono un'ottima compatibilità con esso e con i tool specifici che aiutano nel programmare e nello sfruttare al meglio le sue caratteristiche. Trattandosi di un framework sviluppato fin da 2016, infine, è estremamente diffuso in tutto il mondo e ha alle spalle numerosi team che lo supportano, che ne segnalano problematiche nella repository pubblica ufficiale e che lo utilizzano per progetti in ambiti che spaziano dal privato all'enterprise. La community è una delle più attive tra i numerosi progetti dello stesso tipo presenti al giorno d'oggi e si contende con React[8] e Vue[9] lo scettro di miglior ambiente per lo sviluppo di interfacce web. Offre, infine, di default il pattern Model-View-Controller (o MVC), nel quale un applicativo si suddivide in tre parti principali che comunicano ma si suddividono i ruoli; la parte di Model identifica le entità che saranno gestite e offre funzioni per la loro gestione, la parte View rappresenta la vista dei dati e la loro presentazione all'utente mentre la parte Controller implementa le logiche di business, cioè il vero e proprio motore del software.

All'interno di ReportManager, Angular è fiancheggiato da altre librerie che meritano menzione. Una di queste è sicuramente RxJS[10]. Essa viene impiegata per la gestione delle chiamate asincrone e dei flussi di dati basati su eventi. In estrema sintesi, RxJS implementa il design pattern observer nel quale un oggetto osservato funge da fulcro e tutti gli ascoltatori interessati a tale oggetto e ai dati ad esso correlati scelgono di abbonarsi per ricevere aggiornamenti puntuali rispetto a qualsivoglia modifica. Un'altra libreria è

[8] https://it.reactjs.org/

[9] https://vuejs.org/

[10] https://rxjs.dev/guide/overview

Angular Material[11], sfruttata per il duplice obiettivo di evitare la creazione di ogni singolo componente da zero e di dare un'idea di coesione stilistica all'intera interfaccia utente pur non avendo alcuna competenza grafica pregressa. Per quanto riguarda la gestione ottimizzata delle date e degli orari tra le viste applicative e il backend, è stata impiegata la libreria Moment.js[12], punto di riferimento per i programmatori JavaScript per tale compito. Signature Pad[13] è stata usata per gestire la firma di un dipendente o di un cliente in modo semplice e chiaro. Infine, il package uuid[14] è stato usato per generare id univoci quando necessario; gli identificativi ottenuti in questo modo hanno una bassissima chance di collisione e dunque si prestano bene a distinguere oggetti o entità che devono essere univocamente determinate e determinabili.

5.2 Spring Boot

Il backend di ReportManager è stato realizzato con Spring Boot[15]. Spring[16], da cui quest'ultimo deriva, è un framework di sviluppo per applicazioni che saranno eseguite sulla JVM, acronimo di Java Virtual Machine[17]. La Java Virtual Machine è uno strumento che consente di eseguire codice scritto con l'omonimo linguaggio di programmazione su differenti dispositivi; questo perché costituisce un ponte tra le chiamate di sistema e le API descritte e utilizzate dalle librerie Java stesse. Basta dunque che un sistema sia compatibile con la JVM perché un programma Java possa essere eseguito su esso.

[11] https://material.angular.io/
[12] https://momentjs.com/
[13] https://github.com/szimek/signature_pad
[14] https://www.npmjs.com/package/uuid
[15] https://spring.io/projects/spring-boot
[16] https://spring.io/
[17] https://www.java.com/it/download/manual.jsp

Questa caratteristica ha reso Java e il relativo ecosistema estremamente diffusi e prolifici, tanto che anche progetti enormi, come Android[18] di Google, ne fanno uso. In quest'ottica, Spring si presta molto bene per la creazione di servizi web basati su Servlet. I Servlet sono oggetti Java che operano in un server web, come ad esempio Apache Tomcat[19], che li espone via internet. Spring nasce nel 2003 in risposta alle complessità delle specifiche emergenti di J2EE (oggi Jakarta EE), cioè un insieme di specifiche volte a estendere le capacità di Java verso un mondo sempre più orientato a internet e ai servizi di livello enterprise a esso connessi. Assume ben presto una posizione che si affianca a queste specifiche, finendo con l'adozione di alcune di esse come, ad esempio, le già citate Servlet API e JPA, specifica di persistenza dei dati che determina come un programma deve interagire con un database. Spring è fin dagli albori completamente modulare e, proprio grazie a questa caratteristica, nel corso degli anni sono nati diversi progetti che lo integrano e lo estendono; ogni singolo modulo mira a un preciso caso d'uso: Spring Security offre funzionalità di sicurezza, Spring Data implementa le specifiche JPA, Spring HATEOAS fornisce metodi per rendere una API completamente RESTful aggiungendo la navigazione ipertestuale tra le diverse entità gestite, ...

Un importante progetto legato a questo ecosistema è Spring Boot. La prima versione di questo ramo di sviluppo del framework Spring risale al 2014 e prende subito piede perché offre una visione opinionata di Spring stesso, consentendo a molti sviluppatori di avvicinarsi alla piattaforma senza doverne conoscere da subito a pieno ogni logica. Spring è infatti uno strumento che si dimostra da un lato molto efficiente e robusto ma dall'altro difficilmente

[18] https://www.android.com/intl/it_it/

[19] http://tomcat.apache.org/

configurabile. Il progetto del backend di ReportManager è stato realizzato con questa versione proprio per diminuire le difficoltà iniziali che si incontrano inevitabilmente quando si approccia questa tecnologia. Per inizializzare il progetto è stato utilizzato Spring Initializr[20] che offre una comoda interfaccia per cominciare da zero, scegliendo le dipendenze che sono necessarie e sfruttando Apache Maven[21], tool di gestione dei progetti Java e Kotlin.
Nel prossimo capitolo si presentano le funzionalità implementate in ReportManager e si mettono in evidenza le caratteristiche di Angular e Spring Boot che hanno consentito di raggiungere almeno parzialmente i risultati voluti in fase di stesura dei requisiti.

[20]https://start.spring.io/
[21]https://maven.apache.org/

Capitolo 6

Le funzionalità implementate

ReportManager è stato realizzato con un approccio basato su singole unità di lavoro da realizzarsi in tre fasi successive. Durante la prima, a seguito dell'introduzione in azienda e alla raccolta delle esigenze di progetto, ci si è concentrati sulla scelta e sul successivo apprendimento delle basi delle due tecnologie principali impiegate: Angular e Spring Boot. È stato necessario un cambio di paradigma fin da subito perché durante i corsi frequentati all'università non ci si era mai soffermati sull'uso delle API RESTful per il backend; la metodologia vista durante le lezioni era quella che viene oggi definita come Server Side Rendering (SSR), nella quale un framework di backend si occupa di elaborare anche la parte di presentazione grafica dei dati tramite pagine HTML che vengono poi eventualmente aggiornate dinamicamente, pur restando prodotte dal server dell'applicativo che tiene traccia anche della sessione utente. Sciolti i dubbi architetturali, la seconda fase ha visto la realizzazione di mockup iniziali che schematizzassero i flussi informativi da instaurare e della parte più consistente del progetto. Il terzo step è stato dedicato, infine, a ultimare le caratteristiche richieste e ad aggiustare dettagli di usabilità e di interazione tra frontend e backend. Al termine

del percorso è stato eseguito un deploy locale di test su macchina virtuale per verificare l'effettivo funzionamento delle dinamiche costruite. Il deploy è la fase di distribuzione del software, l'ultimo passo per rendere produttivo quanto realizzato. Solitamente viene eseguito da reparti specializzati in collaborazione con gli sviluppatori ma date le finalità del lavoro in oggetto è stato scelto di simularlo.

6.1 L'autenticazione e l'autorizzazione

Il primo scoglio da superare è stato integrare il sistema aziendale di autenticazione e autorizzazione dei dipendenti con la piattaforma. L'autenticazione è il processo mediante il quale si verifica che un determinato soggetto sia chi dice di essere effettivamente. Questa procedura differisce, ma è spesso confusa, con l'autorizzazione che determina invece i permessi di accesso a determinate risorse da parte del soggetto stesso. In un primo momento sono stati organizzati incontri con i tecnici del settore IT che si occupano della manutenzione interna di tale infrastruttura; dalle riunioni, è emerso che la tecnologia di riferimento utilizzata è Microsoft Active Directory. Questo sistema utilizza il protocollo Lightweight Directory Access Protocol (LDAP) per gestire e memorizzare i dati di tutto il personale di INFOLOG. Vengono immagazzinati in una struttura gerarchica i nomi degli utenti, le password di accesso crittografate e i permessi di lettura e scrittura sulle cartelle della rete locale condivisa. Si tratta dunque di uno standard per gestire completamente e in modo centralizzato quali operazioni sono consentite alle diverse figure e ai diversi ruoli tra gli impiegati. Per certi versi, un sistema di naming (così viene definito) come LDAP condivide caratteristiche col sistema

di risoluzione dei nomi DNS[1] che si utilizza quotidianamente in implicito per associare un indirizzo IP a una stringa URL di una risorsa sul web durante la navigazione. Nel caso specifico di INFOLOG, le risorse di interesse sono gli utenti che faranno parte di ReportManager e l'identificazione degli stessi si ottiene dal nodo della gerarchia identificato da tre parametri standardizzati dalla specifica denominata X.500[2]; in particolare:

- OU: unità organizzativa
- DC: componente di dominio
- DC: sottocomponente di dominio

Messi a fuoco questi tre punti fondamentali e richiesto l'aiuto dei tecnici per i corretti valori di configurazione, è stato ricercato un componente di Spring Boot che fornisse una API chiara e semplice per integrare questo tipo di autenticazione. La scelta è ricaduta su ActiveDirectoryLdapAuthenticationProvider di Spring Security, già incluso in fase di inizializzazione del progetto backend. Attraverso il costruttore di questa classe, sono stati specificati i parametri di cui sopra come nodo radice degli utenti e sono state incluse le informazioni relative all'indirizzo pubblico del sistema Active Directory della rete da contattare per verificare le credenziali e la presenza degli utenti, oltre al nome di dominio. Terminate le impostazioni di base di questo componente, diventato il cosiddetto AuthenticationProvider del progetto che gestisce in automatico l'autenticazione dell'utente al momento della richiesta verso il backend, sono stati necessari ulteriori passi per gestire situazioni correlate ma di diversa natura. Anzitutto è stata abilitata la gestione del

[1] https://it.wikipedia.org/wiki/Domain_Name_System
[2] https://en.wikipedia.org/wiki/X.500

CORS, acronimo di Cross-Origin Resource Sharing. Il CORS è un meccanismo di difesa realizzato per impedire a qualunque origine non conosciuta di utilizzare le funzionalità esposte dal backend. Un'origine, nel contesto, è una qualsiasi combinazione di tre parametri dello stack TCP/IP[3] standard: dominio, schema e porta. Nel caso di ReportManager, l'unica applicazione in grado di comunicare e condividere risorse con la parte server deve essere l'applicazione Angular del frontend. Una configurazione corretta del CORS è necessaria anche in fase di sviluppo perché le origini del frontend e del backend sono diverse: la prima è tipicamente *localhost:4200* mentre la seconda *localhost:8080*; come è noto, due servizi anche se gestiti contemporaneamente sullo stesso sistema operativo, devono essere connessi a porte diverse e questo li rende due origini CORS diverse. Ci si è poi posti il problema di come mantenere lo stato autenticato per un utente nel frontend. Nel backend questa caratteristica è delegata a LDAP; questa parte dell'applicazione concettualmente lavora per singole richieste e in quanto RESTful non ha il concetto di stato. Lo stato, infatti, è mantenuto nel frontend e dunque si rende necessario un meccanismo efficace di gestione di login e logout. Come da best practice diffusasi con l'avvento di REST stesso, la scelta in questo caso è ricaduta sui JSON Web Tokens (JWT)[4]. Un JSON Web Token è un insieme di dati sotto forma di oggetto JavaScript che il frontend custodisce e invia in un particolare header HTTP con ogni chiamata al backend. Se l'utente è riconosciuto e ha il permesso di eseguire operazioni è perché nella chiamata ha annesso questo token. Al suo interno sono infatti presenti le informazioni specifiche su ciò che può o non può fare un determinato soggetto e chi esso identifica: le claim. Oltre a queste, è specificata una data di

[3]https://it.wikipedia.org/wiki/Suite_di_protocolli_Internet

[4]https://jwt.io/

scadenza che serve come ulteriore misura di sicurezza per il caso in cui un utente malintenzionato riuscisse a impossessarsi del token. Perché il meccanismo stia in piedi e non ci siano forzature o compromissioni, è necessario che ogni JWT sia correttamente verificato e approvato dal backend. Nel caso del progetto ReportManager, la sicurezza è garantita dalla firma digitale apposta sul JWT stesso; questa firma è ottenuta e verificabile tramite una coppia di chiavi (pubblica e privata) che costituiscono un'implementazione del concetto di cifratura asimmetrica. Riassumendo, un dipendente che voglia effettuare il login col proprio account Active Directory (LDAP) aziendale compirà implicitamente i seguenti step:

- Inserirà le proprie credenziali e sottometterà le stesse tramite form presentata dal progetto Angular di frontend all'avvio
- Il backend le riceverà a un particolare endpoint che verificherà in primis la correttezza della sorgente CORS
- Il backend verificherà poi la presenza del JWT e la sua validità sulla base delle chiavi definite e della scadenza impostata
- Le informazioni saranno inviate a LDAP per l'effettiva validazione dell'utente tramite i parametri configurati di cui sopra
- Se non presente nel database locale di ReportManager, l'utente sarà aggiunto a esso per non richiedere i dettagli a LDAP a ogni richiesta successiva
- Verrà prodotto un nuovo JWT per il richiedente e sarà restituito al frontend che lo salverà localmente per rendere finalmente operativo il soggetto

Nel capitolo 3 è stato anticipato che i dipendenti di INFOLOG che utilizzeranno ReportManager si dividono in due categorie: quelli che fanno parte della piattaforma Atlassian Jira e quelli che non ne fanno parte. A seguito di un'attenta analisi del flusso descritto precedentemente, può sembrare che manchi un dettaglio importante: come distinguere un utente Jira da uno Unitegy "semplice"? È infatti vero che un qualunque impiegato di INFOLOG fa parte di Unitegy mentre solo la porzione di questi che lavora in Logistics utilizza il software di Atlassian. In realtà il token JWT fornirà al frontend anche questa informazione; dopo l'interrogazione a LDAP per sapere se presente, è stata implementata infatti una chiamata a un'API REST esposta da Jira stesso che consente di verificare appunto l'esistenza di un certo utente sulla base dell'indirizzo email. È stato quindi verificato con il reparto IT che come regola ogni utente Jira di INFOLOG accedesse effettivamente con la stessa email aziendale presente su Unitegy. Esclusa la possibilità che i due indirizzi non coincidessero, l'integrazione era dunque completata.

6.2 La dashboard

Il termine dashboard indica una sezione di controllo tramite la quale è possibile monitorare tutte le informazioni generali relative alle funzionalità e ai dati di una piattaforma. Una volta effettuato il login, il frontend reindirizza l'utente a questa schermata tramite Angular router. Angular router è un componente integrato in Angular che consente di gestire la navigazione tra le diverse viste del software di frontend. Il suo uso si rende necessario perché rispetto a un sito web di stampo tradizionale, come spiegato nel capitolo 4, una Single Page Application offre una sola pagina HTML. Tutti i dati che compongono o servono per comporre le diverse schermate sono richiesti al

momento del caricamento iniziale e sono in seguito disponibili nella cache. La cache è un concetto fondamentale dell'informatica e serve sostanzialmente come deposito di dati per un rapido accesso e per limitare la richiesta di risorse; in quanto a velocità, è di gran lunga superiore al database che è invece destinato a moli consistenti di byte e costituisce una memoria persistente, a lungo termine. Il fatto che un progetto di frontend venga realizzato come Single Page Application implica dunque tecniche specifiche per simulare l'interazione che l'utente avrebbe con un sito tradizionale. Queste sono gestite direttamente da Angular router e tramite API concise e chiare è possibile configurare i diversi percorsi che corrispondono alla navigazione cercata. Nella dashboard di ReportManager vengono mostrate due sezioni:

- I dettagli dell'utente, comprensivi di nome, email, nome di dominio e, eventualmente, nome su Atlassian Jira
- Le azioni consentite per l'utente autenticato

Uno dei grandi vantaggi di sfruttare un framework di questa portata è la possibilità di adottare la filosofia DRY, cardine delle applicazioni informatiche di successo. DRY è l'acronimo di *Don't Repeat Yourself*, espressione inglese che si traduce in *non ripeterti*. L'idea è quella di evitare, laddove possibile, di replicare parti di codice e logiche applicative in diversi punti. L'intero frontend è basato su questo concetto e il fatto di gestire due utenti di tipo diverso con un singolo componente dashboard ne è uno dei diversi esempi. Se un dipendente di INFOLOG che non fa parte di Logistics accedesse a ReportManager, le azioni consentitegli sarebbero solamente due:

- *Nuovo report* - per la generazione di un nuovo report
- *I miei report* - per la visualizzazione dei report registrati a proprio nome

Se invece il dipendente fosse parte di Logistics, allora vedrebbe automaticamente anche il bottone *Gestione task* - per l'inserimento delle ore su Atlassian Jira e la gestione dei task di lavoro. Premendo uno dei bottoni della dashboard, si accede alle sottosezioni specifiche.

6.3 La gestione dei task e delle ore su Atlassian Jira

Jira è il programma di Atlassian per la gestione dello sviluppo con metodologia Agile. Il manifesto Agile[5] è un elenco di principi atti a favorire la produzione di software di qualità in un tempo stabilito e soprattutto a ritmi continui, senza rilasci casuali o al termine di un intero percorso di lavoro. I fondamenti principali che lo caratterizzano sono:

- Individui e interazioni sono più importanti di processi e strumenti
- Il software funzionante è più importante di una documentazione esaustiva
- La collaborazione col cliente è più importante della negoziazione del contratto
- La risposta al cambiamento è più importante del seguire un piano prestabilito alla lettera

Le più grandi aziende e quelle che ottengono i migliori risultati, ad oggi, adoperano Agile su piccola, media o larga scala per la gestione dei team di programmatori e del rapporto con le altre figure dell'organizzazione. INFOLOG, scegliendo questo modus operandi, ha adottato il software in questione

[5]https://agilemanifesto.org/iso/it/manifesto.html

perché di ottima fattura e perché è risultato perfetto per le caratteristiche operative del reparto Logistics. Il pacchetto fornito dal produttore è fruibile attraverso due modalità:

- In cloud, senza dunque installare alcunché e sfruttando l'infrastruttura fornita senza doverla gestire direttamente
- In locale, installandolo sui server aziendali per poter avere il pieno controllo sul database, sugli aggiornamenti e sulle estensioni

Data l'esperienza dei suoi tecnici, INFOLOG ha optato per la seconda opzione.
Jira suddivide il lavoro in progetti differenti. La definizione di ciò che viene rappresentato da un progetto può variare a seconda dell'uso che si fa del programma, anche se l'accezione comune è che si tratti di un singolo progetto software o di un modulo di esso. INFOLOG ha scelto di creare un progetto generico per INTELLIMAG, nel quale vengono sviluppate le funzionalità comuni del gestionale di magazzino. Per ognuna delle versioni modificate per i clienti, poi, sono stati creati altri progetti ai quali tutti i membri del team Logistics hanno accesso e su cui possono creare task operativi. Per ogni progetto si procede allo sviluppo per sprint, ovvero periodi della durata tipica di un paio di settimane. Uno sprint è gestibile attraverso una sezione apposita che mostra colonne diverse per i task da fare, per quelli in corso e per quelli completati. Anche in questo caso, in realtà, l'intestazione e il numero di colonne si possono adeguare alle proprie esigenze. All'interno di ogni progetto nascono compiti di grandi dimensioni, che raggruppano diverse operazioni più piccole; questi prendono il nome di epiche. Un'epica deve essere necessariamente creata da un project manager, figura che assolve funzioni di coordinamento del team. Arrivando all'anello più basso della catena dello

sviluppo, i task rappresentano le singole attività operative di programmazione. Questi possono essere creati, modificati e cancellati da un qualunque sviluppatore di Logistics; ognuno è collegato a un'epica di riferimento e a un programmatore che lo esegue materialmente.

Jira, inoltre, è altamente estensibile. Offre infatti un vero e proprio ecosistema di plugin di terze parti che integrano le più disparate funzioni: dalla messaggistica fino al deploy automatizzato sulla base della chiusura di un'epica. È addirittura presente online uno store virtuale[6] nel quale cercare l'estensione che soddisfa maggiormente le proprie esigenze. INFOLOG ha bisogno di tenere monitorati i tempi che ogni sua figura del team Logistics passa nello sviluppo di un determinato task. Questo consente ai superiori di stabilire un rendiconto dei costi e di quanto prodotto per fini gestionali e commerciali. Per fare questo, è stato annesso a Jira il plugin WorklogPRO - Timesheets for Jira[7]. WorklogPRO offre una comoda interfaccia per inserire un riassunto delle operazioni svolte e dell'arco temporale impiegato su un task specifico. Sia Jira che WorklogPRO offrono un'API RESTful per la comunicazione esterna. Grazie alla documentazione ben scritta e completa di entrambi, la loro integrazione risulta particolarmente agevole.

La pagina di gestione dei task di ReportManager offre all'utente del reparto Logistics tutto il necessario per creare, aggiornare e cancellare non solo i singoli task di Jira ma anche le ore di WorklogPRO. Chiaramente l'inserimento delle ore è possibile solo se il task in questione esiste su Jira stesso, e questo controllo è stato implementato fin da subito. All'apertura della pagina, viene mostrato un caricamento perché viene fatta una richiesta al backend per ottenere una lista di task assegnati all'utente; una volta caricati i dati,

[6] https://marketplace.atlassian.com/

[7] https://marketplace.atlassian.com/apps/1212626/worklogpro-timesheets-for-jira?tab=overview&hosting=datacenter

cioè quando lo scambio di informazioni è stato ultimato, viene nascosto il componente e vengono mostrati i dati ricevuti. Di ogni task viene mostrato quanto segue:

- Chiave univoca del progetto
- Chiave univoca dell'epica cui è relativo
- Descrizione
- Data di inizio
- Data termine

I campi relativi all'epica e alla data di inizio sono un altro esempio dell'enorme flessibilità offerta dal software di Atlassian quando gestito internamente; infatti si tratta di informazioni che normalmente non sono presenti e sono state aggiunte dal team di IT di INFOLOG, forzando anche le connessioni al momento della creazione di un nuovo elemento. Nel backend, per evitare problematiche di sicurezza, sono state ristrette le chiamate alle REST API esposte da Jira sulla base dell'utente identificato dal token che viene prodotto sulla base delle credenziali del dipendente. In questo modo, un malintenzionato dovrebbe essere in possesso di queste o del token stesso per poter agire in sua vece.

All'atto del caricamento dei task, per verificare l'eventuale presenza di ore segnate, viene effettuata una chiamata all'API di WorklogPRO alla quale vengono fornite due informazioni: l'identificativo del task (definito *issue* nella documentazione di Jira) e quello dell'utente. In base alla risposta, viene poi eventualmente mostrata una tabella in fondo a ogni sezione dedicata alle unità di lavoro; questa comprende, per ogni worklog inserito, le seguenti informazioni:

- Data di creazione
- Data di inizio
- Tempo del log
- Riassunto del lavoro svolto

Il tempo del log è mostrato secondo un particolare formato che permette di inserire le ore seguite da una h che indica il termine inglese *hour* e i minuti seguiti da m che indica il termine inglese *minute*. Ogni riga della tabella dei worklog presenta un bottone d'azione che consente di modificare quanto inserito attraverso un apposito modal. Un modal è una schermata che compare sopra gli elementi della pagina ricoprendola parzialmente (o addirittura totalmente) e catturando l'attenzione dell'utente. Può essere usato per l'inserimento di dati o per notificare importanti messaggi che l'utente non deve assolutamente ignorare. Ulteriori funzionalità della sezione in oggetto sono elencate di seguito:

- Ogni task consente di aggiungere attraverso un altro bottone d'azione un worklog per volta
- È possibile modificare i dettagli del singolo task attraverso un bottone d'azione specifico
- È possibile cancellare un task con le relative ore inserite se l'utente che compie l'azione è abilitato a farlo
- Si può cercare il dettaglio di un task inserendone nell'apposita barra di ricerca la chiave univoca; in questo modo, verrà mostrato unicamente questo al posto dell'intero elenco

6.4 La struttura del report

La pagina di visualizzazione dei propri report è accessibile dalla dashboard attraverso il bottone *I miei report*.
Questa funzionalità è disponibile a tutti gli utenti, indipendentemente dalla tipologia. All'atto del caricamento della schermata, dopo i comuni controlli per verificare il login dell'utente, viene mostrato lo spinner di caricamento a indicare che è in corso un'operazione potenzialmente lenta. Lo spinner è un componente grafico costituito da un elemento rotante (come possono essere un semicerchio o una clessidra) che viene solitamente usato per questi scopi. L'operazione lenta in questione è la richiesta al backend per ottenere i dettagli dei report generati dal richiedente. Ottenuta la risposta, lo spinner viene nascosto e viene mostrata la lista ottenuta o un errore in caso di problemi occorsi durante la chiamata HTTP. La struttura dati che fa da base nel backend per i rapporti d'intervento è costituità da due entità distinte ma connesse:

- *ReportHeaderModel* - contenente i dati di testata
- *ReportLineModel* - contenente i dati di dettaglio, o righe

La distinzione nasce dal fatto che un singolo report presenta caratteristiche comuni, come ad esempio il cliente per cui è compilato, ma può avere diverse commesse cui è riferito. Per tale ragione, è stato scelto di inserire informazioni come il cliente nella testata e altre, come appunto la commessa presente sul database di Unitegy, sulle righe. Per mantenere l'integrità dei dati, è stato inoltre scelto che l'esistenza di una riga sia sempre subordinata a quella di una testata. Per questo motivo, non è possibile avere un oggetto di tipo ReportLineModel senza un riferimento al relativo ReportHeaderModel e tale evenienza genera un errore in fase di salvataggio nel backend che viene

propagato di conseguenza al frontend e dunque segnalato all'utente.
I dati della **testata** del report a livello di modello sono i seguenti:

- Un riferimento all'utente che lo crea
- Un numero progressivo cui è anteposto un prefisso univoco per dipendente costituito da lettere e numeri
- Una data principale di riferimento: ogni riga deve essere successiva a questa che è da intendersi come "inizio dei lavori"
- Il totale delle ore lavorate, somma dei valori delle righe
- Il totale delle ore da fatturare, somma dei valori delle righe
- Il riferimento al cliente per come indicato su Unitegy
- Un flag che indica se il rapporto d'intervento è già stato inviato al database di Unitegy
- Un campo note
- La firma del cliente, serializzata come stringa sul database

I dati della **riga**, invece, sono i seguenti:

- Un riferimento alla testata che indica il report
- La data di riferimento per l'intervento della riga
- Un campo testo in cui scrivere in sintesi le operazioni svolte
- Le ore lavorate
- Le ore da fatturare

- Il riferimento alla commessa presente su Unitegy, chiaramente riferita allo stesso cliente della testata cui la singola riga è riferita

La presenza di due totali per le ore, uno riguardante quelle operative e uno quelle da fatturare, è dovuto a un'altra esigenza di stampo commerciale; può infatti capitare che per un determinato lavoro venga aperta una commessa e il cliente paghi per un pacchetto predeterminato oltre il quale INFOLOG andrebbe in perdita. Sebbene questo non rappresenti un problema, essere consci della situazione permette di correggere laddove possibile le eventuali criticità.

6.5 La creazione di un nuovo report

Per creare un nuovo rapporto d'intervento, dalla dashboard occorre fare click sul pulsante *Nuovo report*.
La schermata che si presenta all'utente è costituita da uno stepper[8], cioè da un componente grafico che suddivide un'azione in diverse sezioni che devono essere completate una dopo l'altra. È stato scelto di configurare la pagina in modo tale che gli step fossero obbligatoriamente successivi e che solo giunti al termine fossero possibili il salvataggio e l'invio dei dati al backend; è stato inoltre impedito di saltare uno o più step non opzionali al di fuori dell'ordine stabilito. Queste decisioni hanno semplificato di molto la logica che controlla la coerenza delle informazioni. Chiaramente, più che mai per questa sezione del software che rappresenta il principale caso d'uso, i controlli che sono stati implementati nel frontend sono riverificati anche nel backend. La prima richiesta che viene scatenata quando si vuole generare un nuovo report è quella che permette di ottenere un codice; tale requisito è indispensabilmente

[8]https://material.angular.io/components/stepper/overview

univoco per la base di dati e presenta un prefisso configurabile che varia da dipendente a dipendente e permette di distinguerlo in modo mnemonico. Insieme al prefisso, vengono forniti alla Single Page Application anche il numero progressivo da utilizzare, anch'esso univoco per utente, il nome e cognome dell'utente se non già raccolti in precedenza, la lista dei clienti per scegliere a quale di questi intestare il rapporto d'intervento e la firma del dipendente se salvata. I passaggi per la generazione del report sono 7 in totale:

1. Inserimento del codice univoco di cui sopra
2. Inserimento di cliente e data della testata
3. Inserimento dei dettagli delle righe del report (con l'opzione dell'inserimento dai worklog di Jira se utente Logistics)
4. Inserimento di eventuali note
5. Inserimento o modifica delle firme del cliente e del dipendente
6. Salvataggio e invio dei dati al backend per la convalida
7. Opzioni aggiuntive: richiesta di un file PDF, richiesta di invio in allegato a email, richiesta di invio dei dati a Unitegy

Particolare attenzione è stata riposta nella progettazione e nello sviluppo del punto 3 perché si fosse certi che i totali delle ore inserite fossero coerenti, che non ci fossero sovrapposizioni nell'inserimento di righe diverse per la stessa commessa e che le date delle righe non fossero precedenti a quella della testata di riferimento.

6.5.1 L'inserimento delle firme

L'inserimento delle firme che avviene al punto 5 della compilazione ha richiesto particolari accorgimenti.

Anzitutto, è stato scelto di salvare la firma stessa come stringa nel backend e per tale ragione, trattandosi di un'immagine, è stato scelto un formato di codifica. Per semplificare le operazioni di scambio dati, è stato deciso di serializzare l'immagine in Data URL[9].

Un Data URL è un particolare URL che contiene le informazioni codificate sul formato dell'immagine stessa, che nel caso di ReportManager è PNG, e sui pixel che la compongono; in estrema sintesi, si tratta di una stringa che contiene tutto ciò che serve a un interlocutore per ricostruire l'immagine di partenza. Non è sicuramente il miglior metodo con cui inviare a un server contenuti multimediali ma per oggetti di piccole dimensioni è più che sufficiente. Quando un utente accede a ReportManager, viene eseguito un controllo automatico per individuare quale tipo di dispositivo è in uso. Tramite le API fornite dai moderni browser web, scaturite dalla sempre più ampia disomogeneità dei dispositivi in circolazione, questa informazione è facilmente reperibile. Verificato questo dettaglio, viene settato un particolare flag che indica a tutte le schermate che ne hanno bisogno se si ha a disposizione un touchscreen. Il pannello in cui inserire la firma viene mostrato solo se questo flag è settato.

L'elemento della pagina in cui si può disegnare è un canvas, nello specifico un HTMLCanvasElement che fornisce già un metodo di serializzazione comodamente utilizzabile: toDataURL. Allo stesso modo, all'atto del caricamento della schermata, viene caricata la firma eventualmente salvata del dipendente che esegue il lavoro presso il cliente. Questa conversione è in senso opposto

[9] https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs

rispetto a quella descritta poc'anzi e sfrutta il metodo fromDataURL.

6.5.2 Le opzioni aggiuntive

Le opzioni aggiuntive vengono abilitate solo in conseguenza del salvataggio dei dati nel backend e solo in caso di esito positivo dello stesso.
È possibile richiedere lo scaricamento di un documento in formato PDF che riporti esattamente i dettagli del report generato. Per questa integrazione sono state proposte diverse soluzioni: la più completa e complessa, che prevedeva l'impiego di un file di template sorgente da compilare di volta in volta da codice con i parametri del rapporto d'intervento, una seconda soluzione che prevedeva la compilazione direttamente nel dispositivo richiedente e dunque lato frontend con una libreria JavaScript e una terza che prevedeva un compromesso tra versatilità e richiesta di risorse in quanto realizzata tramite richiesta al backend. In seguito a diverse valutazioni con il team Logistics, è stata scelta la terza opzione.
Al fine di realizzare tale processo, è stato annesso al backend il componente iText PDF; questa libreria consente in modo semplice e intuitivo di creare pagine esportabili definendo dettagli altrimenti complessi da gestire come l'impaginazione, i margini, i bordi, le immagini e i diversi font.
Grazie a questo strumento, è stato possibile creare un endpoint che prendesse in input un identificativo di un report e, se presente, restituisse il file creato dopo averlo salvato in un'apposita cartella che idealmente sarà poi creata su una repository specifica su server.
Un'altra opzione aggiuntiva è l'invio di suddetto file via email. Questa caratteristica prevede un algoritmo che cerca dapprima l'eventuale file PDF da allegare, in caso manchi lo genera con l'ausilio della libreria di cui sopra e poi effettua l'invio a un indirizzo email specificato sul database di Unitegy

direttamente sul contatto del cliente. In Spring Boot è stato usato il modulo JavaMailSender attraverso una specifica dipendenza che consente di inserire in maniera semplice i parametri di connessione da usare per l'invio nel file di properties dell'applicazione. Queste impostazioni sono state fornite direttamente dal reparto IT e consentono a ReportManager di sfruttare il protocollo SMTP per la posta elettronica.

Infine è stata annessa la possibilità di inviare i dati a Unitegy stesso per sincronizzare i due database. Questo ha richiesto un'attenta analisi delle caratteristiche di transazionalità tra le due piattaforme al fine di non creare inconsistenze.

La transazionalità è una caratteristica che deve essere sempre tenuta in considerazione quando si ha a che fare con un database. I moderni DBMS (acronimo di DataBase Management System) rispettano il cosiddetto paradigma ACID[10]. Questi indicano come deve essere gestita una transazione singola, intesa come operazione di qualsiasi tipo sui dati persistiti. Spring Boot offre un'enorme agevolazione, in questi termini: incorpora Hibernate.

Hibernate è un cosiddetto persistence layer che si frappone tra l'accesso alla base di dati e la logica applicativa. Come l'informatica intera, è un perfetto esempio di astrazione che leva dalla mente del programmatore dettagli implementativi di basso livello consentendogli di concentrarsi sulla logica di business e sulla progettazione. Grazie a Hibernate, le transazioni possono essere gestite esplicitamente marcando un metodo di una classe Java con *@Transactional*; quando un metodo risulta avere tale annotazione, se qualcosa dovesse andare storto e un'eccezione Java fosse lanciata verrebbe fatto il cosiddetto rollback automaticamente. Il rollback indica il ripristino di uno stato precedentemente consistente dei dati, lasciando il database in uno sta-

[10]https://it.wikipedia.org/wiki/ACID

to nuovamente utilizzabile. Ogni scrittura su Unitegy è stata implementata secondo questa metodologia.

6.6 Sicurezza e HTTPS

Il progetto ReportManager nasce per esigenze interne gestionali ma dovrà essere impiegato dai dipendenti anche al di fuori della rete locale perché spesso e volentieri i rapporti d'intervento sono compilati con il cliente stesso presso la sede in cui i lavori sono stati eseguiti.
A questo proposito, un futuro deploy dell'applicazione dovrà esporre i servizi del backend su internet. Per tale ragione, occorre proteggere gli eventuali dati sensibili con protocolli adeguatamente cifrati come HTTPS.
INFOLOG possiede certificati proprietari che rinnova periodicamente e che garantiscono ai propri servizi la cifratura dei dati comunicati in entrambe le direzioni: front-to-back e back-to-front.
Anche se durante la fase di sviluppo di un'applicazione tipicamente non si utilizza tale misura di sicurezza in quanto ambiente locale, per preparare e testare che tutto funzioni in modo adeguato è stato richiesto al reparto tecnico di fornire un certificato di prova. Ottenuto tale file, è stato configurato il progetto Spring Boot perché leggesse i dati e li scrivesse sfruttando suddetto protocollo di comunicazione. Sia le API di Jira che di WorklogPRO hanno endpoint esposti su internet e quindi questa accortezza riservata al progetto sarà sicuramente utile per sviluppi futuri.

Capitolo 7

Il debito tecnico e gli sviluppi futuri

Il debito tecnico[1] è un termine metaforico coniato da Ward Cunningham[2], programmatore statunitense di fama internazionale, per descrivere come l'attività di sviluppo di un software finalizzata unicamente al rilascio porti inevitabilmente con sé ripercussioni nelle fasi successive.
Prendendo in prestito un esempio[3] da Martin Fowler[4], altro famoso ingegnere informatico, si può paragonare tale concetto a un debito finanziario; il tempo aggiuntivo che si impiegherà per aggiungere una nuova caratteristica al progetto è da vedersi come l'interesse sul debito e idealmente bisogna cercare di farlo tendere a zero.
Sono stati condotti numerosi studi sulla questione, alcuni dei quali[2], propongono metodologie di carattere volutamente generale per comprendere il fenomeno e analizzarlo.

[1] https://martinfowler.com/bliki/TechnicalDebt.html
[2] https://it.wikipedia.org/wiki/Ward_Cunningham
[3] https://martinfowler.com/bliki/TechnicalDebt.html
[4] https://it.wikipedia.org/wiki/Martin_Fowler

Nonostante ciò, numerosissime aziende faticano a comprendere il concetto, a riconoscerlo e di conseguenza a fronteggiarlo.
ReportManager è stato realizzato tenendo ben presente questa possibile problematica. Ogni scelta è stata condivisa con il management, ogni dipendenza annessa è stata illustrata ai membri del team Logistics e si è cercato di documentare il più possibile le funzionalità principali per rendere la manutenzione agevole a chiunque dovrà approcciarvisi.
Nonostante l'impegno e la buona volontà, l'applicazione non è completa in ogni sua parte. Diverse sono le migliorie future che potranno essere programmate e messe in produzione e il seguente elenco, sebbene non esaustivo, riassume i punti individuati:

- Utilizzo di un controllo unico per verificare il login utente lato frontend anziché uno replicato su ogni pagina
- Miglioramento della descrizione degli errori sia nel logging del backend che nel frontend
- Miglioramento della posizione dei vari popup mostrati nelle differenti pagine, con codice colori rosso, giallo e verde per distinguere la gravità del messaggio
- Uso dello spinner di caricamento in conseguenza di ogni operazione potenzialmente lenta
- Eliminazione del logging di debug lato frontend
- Sostituzione dei parametri fissati nelle interrogazioni delle API di Atlassian Jira con i corrispondenti dinamici
- Miglioramento della gestione delle firme e passaggio da stringa a immagine reale

- Inserimento di un bottone per creare un nuovo report se attualmente non ne sono stati creati per quell'utente

Un ultimo aspetto che vale la pena di affrontare è quello relativo alle *best practices*, in italiano *buone pratiche*. Le best practices sono pattern di buona programmazione o comunque di organizzazione del software che vengono consigliate spesso dagli sviluppatori dei framework che si usano o dagli utenti più esperti. Chiaramente si tratta talvolta di finezze e altre volte invece di concetti basilari per un occhio esperto. Il neofita, tuttavia, trae beneficio da questi e impara la filosofia con cui i framework funzionano e come sono stati pensati, assumendo maggiore consapevolezza del lavoro che dovrà svolgere. ReportManager non fa uso esplicito di best practices e in una fase futura di refactoring dovrà assolutamente prevederne l'impiego.
Il mondo Angular prevede, ad esempio, di utilizzare i service per tutte le funzionalità di business logic del software; questo aderisce perfettamente al concetto di Model-View-Controller me nel software in oggetto se ne fa un uso solo parziale.
Nel mondo Spring Boot, invece, buone pratiche di esempio possono essere il principio di singola responsabilità e l'uso di Data Transfer Object (o DTO). Il primo indica l'idea secondo cui ogni metodo e più in generale ogni classe sia deputata a un solo compito; il secondo forza invece una netta separazione tra il backend e il frontend, affermando che non tutti i dettagli mappati su database debbano essere riportati all'interfaccia e che sia sufficiente un sottoinsieme di questi, o talvolta un insieme semplicemente differente di informazioni.
Concludendo, un'applicazione moderna non può ritenersi tale senza test automatizzati. Esistono diversi strumenti coi quali è possibile attuare tecniche di programmazione efficaci come il Test-Driven Development (TDD) e la

scrittura di codice avente per scopo il test di porzioni di programma. Sia Angular che Spring Boot offrono un supporto estremamente ampio a questa tematica anche se in una fase iniziale di progetto si tende spesso a tralasciare questo aspetto.

Capitolo 8

Conclusioni

In questo lavoro è stato presentato ReportManager, un software per la gestione dei rapporti d'intervento realizzato appositamente per INFOLOG SpA.
È stata presentata la storia dell'azienda, è stato spiegato ciò che esisteva prima, è stato spiegato ciò che era richiesto e come lo si è realizzato, riflettendo sugli inevitabili compromessi che sono stati fatti.
Ci si è soffermati sulle tecnologie utilizzate, sulle peculiarità delle stesse e su punti di forza e punti deboli di ognuna, per poi entrare nel dettaglio delle funzionalità implementate e accennare a ciò che è rimasto incompiuto e che sarà oggetto di sviluppi futuri, annettendo una breve riflessione sul debito tecnico.
Terminata l'attività di tirocinio, INFOLOG SpA ha espresso la volontà di adottare la soluzione realizzata come unico progetto per le finalità proposte.

Bibliografia

[1] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[2] Edith Tom, Aybüke Aurum, and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.