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

Tesi.

Indice

Capitolo 1

Introduzione

Nel lavoro che si descrive...

Il resto del documento è organizzato come segue. Infine, il capitolo 9 conclude il documento.

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. 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[5]. Il sistema è stato riprogettato al momento della transizione da AngularJS[6] con velocità e performance come punti cardine, rendendo qualunque applicazione realizzata con esso estremamente rapida e professionale già con le semplici impostazioni di default.

[3]https://www.typescriptlang.org/

[4]https://www.python.org/

[5]https://cordova.apache.org/

[6]https://angularjs.org/

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[7] e Vue[8] 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[9]. 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 Angular Material[10], sfruttata per il duplice obiettivo di evitare la creazione di ogni singolo componente da zero e di dare un'idea di coesione stilistica all'in-

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

[8] https://vuejs.org/

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

[10] https://material.angular.io/

tera 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[11], punto di riferimento per i programmatori JavaScript per tale compito. Signature Pad[12] è stata usata per gestire la firma di un dipendente o di un cliente in modo semplice e chiaro. Infine, il package uuid[13] 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[14]. Spring[15], da cui quest'ultimo deriva, è un framework di sviluppo per applicazioni che saranno eseguite sulla JVM, acronimo di Java Virtual Machine[16]. 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. Questa caratteristica ha reso Java e il relativo ecosistema estremamente diffusi e prolifici, tanto che anche progetti enormi, come Android[17] di Google,

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

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[18], 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 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.

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

Nel prossimo capitolo si presentano le funzionalità implementate in Report-Manager 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.

Capitolo 6

Le funzionalità implementate

ReportManager è stato realizzato con un approccio basato su singole unità di lavoro da realizzarsi in tre mesi. Durante il primo mese, 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 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 sempre stata 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 interamente prodotte dal server dell'applicativo che tiene traccia anche della sessione utente. Sciolti i dubbi architetturali, il secondo mese ha visto la realizzazione di mockup iniziali che schematizzassero i flussi informativi da instaurare e della parte più consistente del progetto. Il terzo mese è stato dedicato infine a ultimare le caratteristiche volute e ad aggiustare dettagli di usabilità e di interazione tra frontend e backend. Al termine

del periodo, 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.

Il primo scoglio da superare è stato integrare il sistema aziendale di autenticazione dei dipendenti con la piattaforma. 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 e i permessi di lettura e scrittura sulle cartelle della rete locale condivisa internamente. 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 che si utilizza quotidianamente 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 dai parametri standardizzati dalla specifica denominata X.500; in particolare:

- OU: unità organizzativa
- DC: componente di dominio

- DC: sottocomponente di dominio

Messi a fuoco questi tre punti, 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 sul componente 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 root 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. Terminata la configurazione di questo componente, diventato il cosiddetto AuthenticationProvider del progetto e cioè quel componente che gestisce in automatico l'autenticazione di un utente al momento della richiesta verso il backend, sono stati necessari ulteriori passi per risolvere alcune problematiche di diversa natura. Anzitutto è stata abilitata la gestione del 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 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. 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). 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: le claim. Oltre a queste, è specificata una data di 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
- 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 poc'anzi, 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 davvero in INFOLOG, è stata implementata infatti una chiamata a un'API REST esposta da Jira stesso che consente di verificare proprio 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 è stata dunque completata.

Capitolo 7

Le integrazioni

Capitolo 8

Il debito tecnico

Capitolo 9

Conclusioni

In questo lavoro è stato discusso...

Bibliografia

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