Università degli Studi di Modena e Reggio Emilia

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

Tesi di Laurea

Titolo della tesi...

Candidato:
Luca Bonetti

Relatore:
Chiar.mo Prof. Giacomo Cabri

Anno Accademico 2021-2022

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 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 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 del Gruppo Sesa.
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
- 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
- Unitegy è la branca che si occupa della soluzione di gestione delle attività commerciali, basata sul software open source Compiere
- 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.

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.

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. I programmatori del reparto Logistics, in particolare, dovranno essere in grado di utilizzare il software Atlassian Jira 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 possibile l'inserimento manuale di suddetti dati.

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.

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.

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.
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.
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

Capitolo 6

Le funzionalità implementate

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.