PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
C.I.A.C.

G_TSS_C1-146-2021-0
TECNICO DI SVILUPPO SOFTWARE

Studente:
Paolo Antonio Grosso

Tutor Formazione:
Miriana Vendola

Docenti:
Luca Guerrini, Alfonso Domenici, Claudio Bovo, Paola Morbelli, Laura Ronchietto, Anna Sole Marta, Giulia Iossa

UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

PRESENTAZIONE

Il mondo dell'informatica mi ha sempre affascinato molto. La scintilla è nata con il mio primo smartphone con una connessione a Internet. Ricordo che rimasi affascinato dalle infinite possibilità che offriva: avevo un dubbio o una domanda su un argomento di scuola? Internet sapeva la risposta e sapeva fornirla in un istante. Guardare un video su YouTube? Detto fatto. Social? Acquisti online? Ristorante nei dintorni e possibilmente ben recensito? Insomma, da quel momento in poi, Internet è diventato parte integrante della mia quotidianità.

Il mio percorso di studi riflette la mia passione per l'informatica. Ho frequentato il liceo scientifico Aldo Moro di Rivarolo C.se, con opzione scienze applicate, ovvero con più matematica e informatica al posto del latino e di qualche ora di letteratura. Ho scelto poi di iscrivermi alla facoltà di Ingegneria Informatica presso il Politecnico di Torino. Tuttavia, dopo 3 anni di sacrifici e risultati non proprio eccellenti, ho deciso di cambiare percorso e iscrivermi a questo corso di sviluppo software. Questa scelta, che inizialmente mi spaventava parecchio, si è poi rivelata vincente. Il corso mi ha appassionato fin da subito e mi ha permesso di vedere e approfondire i linguaggi attualmente più utilizzati.

IL CORSO

JavaSE

Ho apprezzato molto il fatto che il focus principale del corso fosse Java. Dopo una panoramica generale sulla programmazione a oggetti e i vantaggi che questa offre, abbiamo trattato le basi di questo linguaggio a oggetti. Java nasce con l'idea di essere semplice, sicuro e multipiattaforma, ovvero di dare la possibilità di eseguire lo stesso codice su macchine con caratteristiche hardware e software differenti.

Avendo una base pregressa del linguaggio C, Java mi è sembrato un linguaggio familiare, intuitivo (anche grazie al fatto di non dover gestire nessun puntatore) ma con il grande potenziale della programmazione a oggetti.

Gli editor che abbiamo utilizzato sono stati Visual Studio Code, di Microsoft, e NetBeans, di Apache.

RDBMS

Per quanto riguarda la parte di database relazionali, abbiamo anche qui fatto una panoramica generale sulla loro struttura e funzionalità. Il termine Relational DataBase Management System, indica un sistema per la gestione di basi di dati rappresentabili come relazioni e manipolabili con gli operatori dell'algebra relazionale (unione, intersezione, differenza, prodotto cartesiano, selezione, ecc...).

Il linguaggio standard per manipolare i dati in un database è SQL (Structured Query Language). Nonostante sia uno standard, esistono diverse versioni di SQL che tuttavia supportano la maggiorparte dei compandi di base. In particolare noi ci siamo interfacciati a database MySQL e MariaDB utilizzando MySQL Workbench.

I comandi più usati sono:

- SELECT, usato per selezionare i cambi e delle tabelle da prendere in considerazione;
- UPDATE, usato per modificare un record già esistente su una tabella;
- DELETE, usato per cancellare un record su una tabella;
- INSERT, usato per aggiungere un record su una tabella;
- WHERE, usato per filtrare dei record secondo una particolare condizione;
- JOIN, usato per combinare le colonne di tabelle diverse.

Abbiamo poi visto che quasi tutte le tabelle possiedono un campo **Primary Key**, un valore numerico, unico e non nullo, per ogni record, usato per accedere ad un record in modo univoco.

Esistono poi particolari relazioni di dipendenza tra i record di tabelle diverse, come ad esempio quella tra una tabella anagrafica e una di impiegati: un impiegato dovrà sempre essere presente nella tabella anagrafica. I RDBMS permettono di salvaguardare l'integrità dei dati grazie all'utilizzo delle **Foreign Key**, campi di una tabella che si riferiscono a campi di altre tabelle.

Infine abbiamo visto gli indici, i quali permettono di velocizzare enormemente la ricerca di un record in tabelle di grandi dimensioni.

HTML/CSS/JavaScript

Sono i linguaggi standard del web:

- HTML (Hyper Text Markup Language) è usato per descrivere la struttura di una pagina e gli elementi che questa contiene, come un menu, un'immagine, un paragrafo o un link;
- CSS (Cascading Style Sheets), definisce lo stile grafico con cui gli elementi di HTML devono essere visualizzati;
- JAVASCRIPT, o ECMAScript, da quando nel 1997 è diventato standard ECMA, è il linguaggio di programmazione più usato al mondo, viene spesso usato per inserire funzioni alle pagine web, ad esempio per ascoltare il click su un tasto e fare una determinata azione di conseguenza.

JavaEE/Jakarta

Sono un insieme di specifiche che estendono le funzionalità di base del linguaggio Java. Vedere tutte le implementazioni di Jakarta non è stato ovviamente possibile all'interno del corso tuttavia abbiamo trattato quelle più utilizzate.

Specifiche web service
I web service sono servizi che vengono messi a disposizione in una rete, privata o pubblica. L'obbiettivo di questi servizi è quello di mettere a disposizione dei dati, a prescindere dal linguaggio con cui sono ricavati e dalla macchina che li sta richiedendo. Richieste e risposte sono formattati secondo lo standard XML e trasportati tramite i protocolli del web, in genere HTTP. In genere questi servizi sono utilizzati per separare la parte "back-end" di un'applicazione (come i dati forniti vengono effettivamente calcolati), da quella "front-end" (come questi dati vengono rappresentati dal client). Si dice che le modifiche che vengono effettuate da una o dall'altra applicazione, avvengono in modo "trasparente", ovvero apportate delle modifiche a una delle parti, non implica di dover modificare anche l'altra parte.

In particolare, durante il corso, abbiamo realizzato un servizio REST che rispondeva a chiamate HTTP restituendo dei record di un db come json.

Specifiche enterprise
Specifiche proprie di Jakarta. Abbiamo visto l'iniezione delle dipendenze (Dependecy Injection) che permettono di gestire l'inversione del controllo (IoC), ovvero di cedere la gestione delle istanze delle classi all'application server.

Specifiche database
Specifiche che permettono la persistenza dei dati all'interno di un database, in

particolare noi abbiamo usato Java Persistence API (JPA, package javax.persistence). Uno dei vantaggi di JPA è quello di permettere allo sviluppatore di interfacciarsi con qualsiasi RDBMS senza dover necessariamente conoscere SQL. JPA consente infatti di mappare lo schema del RDBMS tramite le Entity, classi Java, appositamente annotate, che rappresentano le tabelle del db.

Specifiche web
Aggiungo questa sezione, la quale anche se è stata trattata marginalmente all'interno del corso, ho approfondito prima e durante il percorso di stage.

Sono delle componenti legate alla visualizzazione delle pagine web. Tra queste ho usato Java Server Faces (JSF), un framework basato sul design pattern Model View Controller (MVC), ovvero che separa la presentazione dei dati, dalla logica dell'applicazione:

- *Model* è rappresentato dalle entity, fornisce metodi getter e setter per accedere ai dati;
- *View* visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- *Controller* riceve ed esegue i comandi inseriti dall'utente attraverso il view.

JSF semplifica la creazione di interfacce utente (UI) per applicazioni web lato server fornendo un insieme di componenti UI standard, riutilizzabili e personalizzabili, collegando eventi lato client con la relativa azione lato server.

Il ciclo di vita di una JSF consiste in 6 fasi principali:

- ***Restore View***: inizializza la vista. Durante questa fase si procede alla costruzione della vista della pagina, dei gestori degli eventi, dei validatori e dei componenti presenti nella vista.
- ***Apply Request Values***: estrae i valori dalla request e li converte nei tipi previsti dal managed bean.
- ***Process Validation***: controlla i valori inseriti dall'utente secondo le logiche di validazione definite.
- ***Update Model Values***: aggiorna le proprietà del managed bean con i valori inseriti dall'utente se questi hanno superato la fase di validazione.
- ***Invoke Application***: gestisce gli eventi dell'applicazione (quali ad esempio il click su un bottone).
- ***Render Response***: renderizza la view.

JSF utilizza inoltre dei ***managed bean***, delle semplici classi Java contenenti gli

attributi dell'entità che si vuole gestire e i relativi metodi setter e getter. La loro funzione è quella di validare i dati ricevuti e gestire gli eventi causati dai componenti.

NoSQL DB

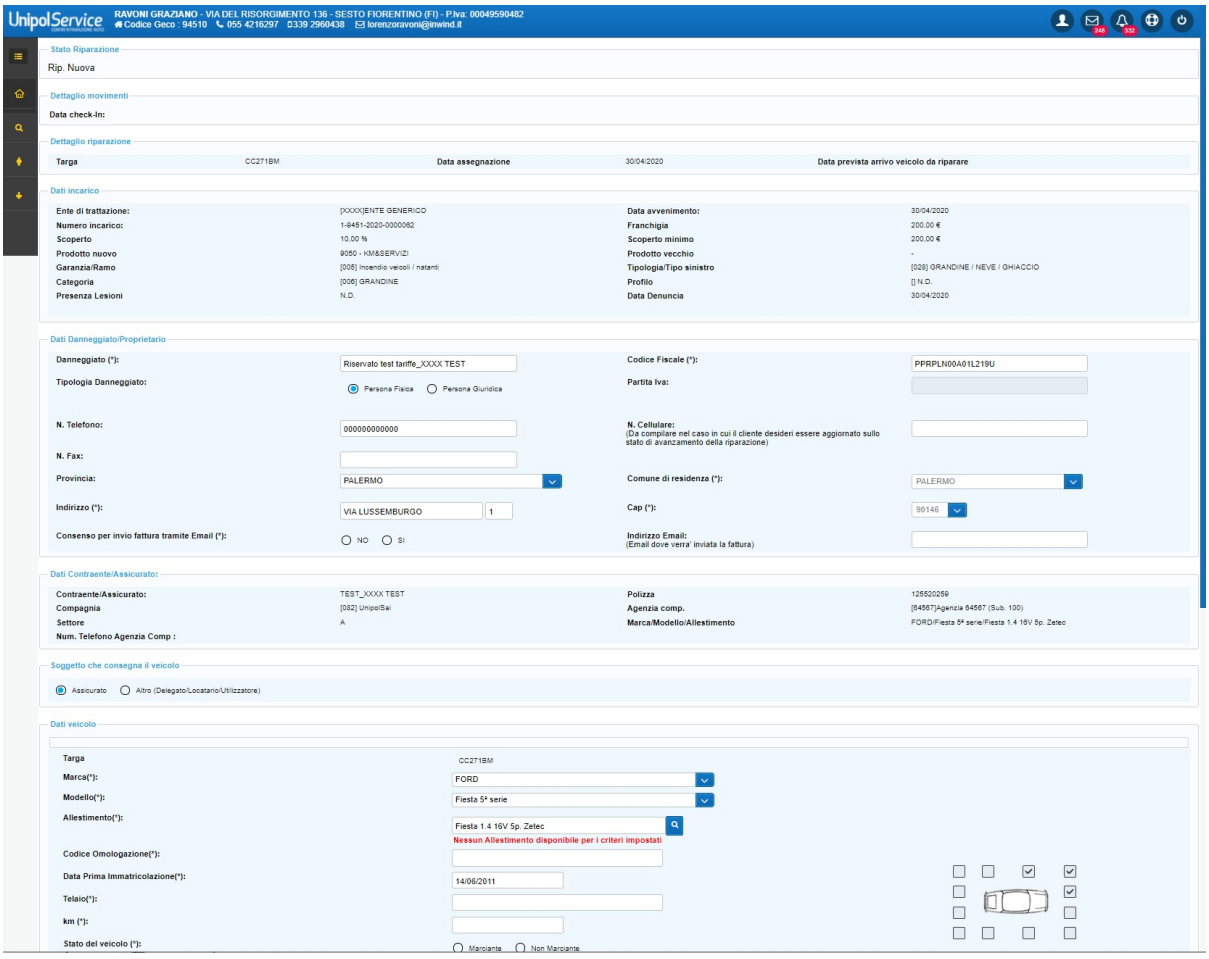
Infine abbiamo visto una breve panoramica su una diversa tipologia di database, detti *non relazionali*. La principale differenza che li contraddistingue dai più classici RDBMS è l'assenza di relazioni tra i dati che vengono inseriti. Il che significa che nella stessa tabella possono entrare record con campi diversi:

Il vantaggio di questo tipo di database sta nella velocità con cui i dati possono essere letti o inseriti, a discapito però di una minore sicurezza ed affidabilità (anche se in parte aggirabile tramite controlli a priori dell'inserimento dei dati nel db).

STAGE

Il mio stage è stato effettuato presso la Estiatechnology, azienda di consulenza informatica, che quindi ci ha messo in contatto con una seconda azienda, loro cliente, Corvallis S.r.l. . Questa azienda offre serivizi in una miriade di settori tra cui quello bancario, assicurativo, industriale, IoT, realtà aumentata, sicurezza informatica e ricerca in vari laboratori di università italiane.

Il progetto su cui ho lavorato riguarda il settore assicurativo stradale. Sotto commissione di Auto Presto & Bene, Corvallis ha realizzato un software che mette in contatto il centro assicurativo AP&B con le carrozzerie e i ricambisti di tutta Italia. L'obbiettivo è quello di ridurre tempi e costi delle riparazioni, oltre che garantire una maggiore trasparenza e affidabilità al cliente finale. Nella prima settimana sono stato affiancato da un collaudatore che mi ha fatto vedere, tramite le interfacce lato AP&B e lato carrozzeria, il ciclo di vita di un sinistro stradale, dal momento in cui viene registrato a quando l'auto viene riconsegnata al cliente. La gestione dei sinistri varia in base alla tipologia del danno: generico (parti da riparare/sostituire), vetri (MyGlass) o grandine (tirabolli/sosituzione pezzi). Nota interessante è che per selezionare le parti dannegiate, sfruttano un servizio REST offerto da Quattroruote per gestire gli esplosi di qualsiasi auto in commercio.



Dopo questa introduzione iniziale su come si presenta attualmente il prodotto finale, sono stato contattato dal team di svilupo che, dopo avermi dato accesso tramite VPN al loro server e fatto settare opportunamente l'ambiente di lavoro (Rational Application Developer, workspace del progetto e Websphere 8.5), mi ha dato delle task di prova da portare a termine. La prima, portata a termine dopo aver seguito un ragazzo del team che ha mostrato i vari passaggi, consisteva nel creare una pagina web con il format di AP&B , con un menu a tendina a sinistra dalla quale poter scegliere un servizio per mostrare una tabella con i dati presi dal DB (Oracle). Essendo il progetto già stato creato da ormai più di 10 anni, la vera difficoltà consisteva nel comprendere la struttura del progetto e in particolare cosa devono fare le varie classi o .xml da utilizzare. Per questa "semplice" task infatti, abbiamo sfruttato:

- iBator, una tecnologia che permette di creare le tabelle le classi per accedervi in automatico;
- un DataAccesObject, per Spring-MVC, mappare la tabella sql su un apposito file .xml;
- creare un BusinessObject (simil DTO) della classe che si vuole visualizzare;
- un ServiceHelper con i metodi per gestire la tabella;
- un WorkingBean (simil EJO), ulteriore passaggio tra DAO e controller (ServiceHelper);
- creare due pagine .xhtml, con PrimeFaces, una per il menu e l'altra con la tabella vera e propria;
- settare action e outcome nel dialog-config.xml.

Terminata questa task, mi è poi stato chiesto di implementare alla tabella creata, la possibilità di modificare i record nel DB da front-end, usando PrimeFaces (v6.5) e questa volta senza "tutorial" iniziale. Fortunatamente però PrimeFaces ha una documentazione molto ricca e dopo qualche tentativo sono riuscito a portare a termine la task.

In parallelo a questo progetto, con Estia abbiamo intrapreso un percorso di formazione interno, seguiti da uno sviluppatore senior. Il progetto che ho realizzato è simile a quello fatto per Corvallis, tuttavia abbiamo utilizzato esclusivamente tecnologie open source.

L'applicazione consiste nella gestione degli impiegati di diverse aziende. L'idea iniziale era quella di creare l'applicazione usando Spring-boot e JSF, in modo da prendere dimestichezza con il progetto del cliente.

Ci è stato riservato uno schema su un database di Estia, questa volta MySQL, dove, tramite JPA, ho creato 3 tabelle: Users, Companies ed Employees. Ho poi aggiunto una parte di codice per fare le CRUD tramite Postman, ovvero crare, modificare o eliminare dei record senza passare da MySQLWorkbench.

Per fare partire l'applicazione, Spring-boot possiede un application server interno, Tomcat. Il tutor, per completezza, ci ha chiesto di provare a fare partire il progetto usando un Tomcat esterno. Per farlo, ho aggiunto la dipendenza di Tomcat e cambiato un'impostazione nel pom.xml, specificando di costruire un pacchetto war invece di un jar.

Dopo questa parte iniziale, il tutor ci ha chiesto di aggiungere la parte di frontend usando JSF e Primefaces. Qui sono nati i primi problemi in quanto i managed beans di Spring-boot, non sono rappresentabili direttamente con pagine JSF. Ho quindi aggiunto delle classi intermedie, ovvero i beans gestiti da CDI, e con questi sono riuscito a passare i dati alle pagine jsf.

Altra difficoltà che ho avuto è stata la gestione delle dipendenze da parte di Joinfaces, progetto non più portato avanti dagli sviluppatori e che è rimasto a una versione vecchia di Primefaces. Per aggiungere le crud alla pagina di frontend ho quindi usato, come anche nel progetto di Corvallis, sempre per ragioni simili, un DataTabel edit, ovvero dando la possibilità di modificare un record semplicemente cliccandoci sopra. Ho quindi aggiunto "manualmente" i tasti "NEW" e "DELETE" per permettere di creare o di cancellare i record e ho poi collegato la pagina con il backend per salvare i dati nel db.

Concludendo, sono davvero soddisfatto di avere seguito questo corso, non solo per la formazione ricevuta, ma ha anche per l'opportunità che mi ha offerto di entrare direttamente nel mondo del lavoro. Lo stage, anche se iniziato con qualche ritardo, trattava gli stessi argomenti visti a lezione ma in un contesto più ampio. Quello del programmatore non è un lavoro per tutti, bisogna avere passione ma per fortuna quella non mi manca. In futuro spero di trovare un posto di lavoro che mi permetta sempre di migliorare e magari un giorno, di poter realizzare qualcosa di mio.

2 Giugno 2021 CIAC Ivrea (TO)