Capitolo 2

Strumenti e tecnologie

All'interno di questo capitolo verranno descritte le tecnologie e gli strumenti utilizzati per lo sviluppo della web application *Rapporti di Lavoro*.

2.1 Tecnologie utilizzate

2.1.1 Web Application - Angular

Web Application

Una *Web Application* è un'applicazione distribuita, ovvero un'applicazione accessibile tramite il web. In particolare una web application può essere resa accessibile su *internet*, e quindi resa accessibile a tutti, oppure all'interno di una *intranet*[1], e quindi accessibile solo all'interno di un sistema informatico. Per la realizzazione dell'applicativo *Rapporti di Lavoro* è stato deciso di sviluppare una web application, in modo da renderla accessibile semplicemente con l'utilizzo di un browser installato su un semplice computer o un tablet.

[1] Una rete aziendale privata e isolata dalla rete esterna, che offre servizi accessibili solo dall'interno della rete aziendale.

Angular

Per la realizzazione della web application *Rapporti di Lavoro* si è deciso di utilizzare *Angular*. *Angular* è un framework opensource[2] dedicato appunto allo sviluppo di web application e sviluppato principalmente da Google. L'intenzione originaria era quella di creare uno strumento semplice e rapido per lo sviluppo di applicazioni in grado di girare su qualunque piattaforma e facilmente fruibili da qualsiasi tipo di dispositivo, come pc desktop, tablet o smartphone. Il più grande punto di forza di Angular è che le applicazioni vengono eseguite interamente dal browser. In questo modo anche le applicazioni più complesse e sofisticate risultano comunque veloci, leggere e facilmente fruibili.
Quando si parla di Angular non si può non parlare di *Single Page Application* o *SPA*, ovvero applicazioni web il quale utilizzo avviene tramite una singola pagina web. Questo garantisce una user experience molto più fluida e dinamica, comparabile a quella di un'applicazione desktop.
Angular non solo può essere utilizzato per le web application, ma si dimostra efficace anche in applicazioni mobile e/o desktop (tramite strumenti esterni come Ionic e Electron). Inoltre sfrutta anche moltissime librerie per l'integrazione con ulteriori servizi terzi con estrema facilità come Google o Amazon Web Services.

2.1.2 Linguaggi utilizzati

Nel paragrafo seguente verranno descritti i linguaggi utilizzati per la realizzazione della web application *Rapporti di Lavoro*.

HTML

HTML, abbreviazione di *Hypertext Markup Language*, è lo standard usato per formattare e impaginare documenti ipertestuali[3] ed è un linguaggio di markup. Un linguaggio di markup è un insieme di regole che descrivono i meccanismi di rappresentazione di un testo. Il linguaggio HTML consente quindi di descrivere semanticamente la struttura di un documento web attraverso tag. HTML rappresenta dunque la struttura portante delle pagine web: su questa struttura si possono aggiungere modifiche grafiche, grazie ai fogli di stile CSS, che verranno descritti successivamente, ed elementi dinamici. L'HTML è un linguaggio di

[2]Un software è definito opensource se il codice sorgente viene reso disponibile per la modifica, lo studio e l'utilizzo.
[3]Un ipertesto è un insieme di documenti correlati tra di loro tramite parole chiave.

pubblico dominio e la sua sintassi è stabilita dal *W3C* (*World Wide Web Consortium*), che si occupa di favorire lo sviluppo di tutte le potenzialità del World Wide Web. Il funzionamento del linguaggio HTML è quindi quello di gestire i contenuti associandone la struttura grafica grazie all'utilizzo di tag diversi. Infatti ogni tag identifica un diverso ruolo dei contenuti.

Un documento HTML inizia con una dichiarazione del tipo di documento, utilizzata per dare al browser l'indicazione per interpretare e visualizzare il documento stesso. Dopo la dichiarazione del tipo di documento, questo presenta una struttura ad albero annidato composta da sezioni delimitate da tag che a loro volta contengono altre sezioni sempre delimitate da tag. La struttura più esterna del documento è compresa tra i tag *<html>* e *</html>*. All'interno di questo tag lo standard prevede la definizione di due sottosezioni distinte: la sezione di intestazione (o *header*), delimitata tra i tag *<head>* e *</head>*, che contiene informazioni di controllo solitamente non visualizzate dal browser come collegamenti verso file esterni (file CSS o di script), e la sezione del corpo (o *body*), delimitata tra i tag *<body>* e *</body>*, che contiene la parte informativa, ovvero il testo, le immagini e tutto ciò che costituisce la parte che deve essere visualizzata dal browser. Oltre all'indicazione sulla struttura del documento lo standard prevede che le sezioni non debbano essere sovrapposte, ovvero che ogni sezione deve essere chiusa prima di iniziare la sezione successiva.

CSS

CSS, abbreviazione di *Cascading Style Sheets*, è il linguaggio usato per definire la formattazione dei documenti *HTML*. Così come per l'*HTML*, le regole per comporre il *CSS* sono contenute in un insieme di direttive emanate dal *W3C*. L'introduzione del *CSS* è stata utile per separare i contenuti delle pagine HTML dalla loro formattazione. Questa struttura garantisce anche il riutilizzo del codice, ovvero consente di richiamare parti di codice già scritte senza la necessità di doverle riscrivere. In particolare consente di assegnare lo stesso stile a più componenti all'interno del documento HTML. L'inserimento del codice CSS nelle pagine web può essere effettuato in tre modi diversi:

- inserimento nel tag *<head>* di un collegamento ad un foglio di stile esterno, come nell'esempio seguente:

```
<head>
  <link rel="stylesheet" type="text/css"
    href="stylesheet.css" />
```

```
</head>
```

- inserimento nel tag <*head*> delle dichiarazioni css tra i tag <*style*> e </*style*>, come nell'esempio seguente:

```
<head>
  <style type="text/css">
    /*codice css*/
  </style>
</head>
```

- inserimento del codice CSS all'interno degli elementi, come nell'esempio seguente:

```
<tag style="css"></tag>
```

Generalmente un foglio di stile CSS è strutturato come una sequeza di regole, ovvero coppie costituite da un selettore e un blocco di dichiarazioni. La struttura standard è la seguente:

```
selettore{
  p1: v1,
  p2: v2
}
```

Il selettore può essere di diverso tipo. Ad esempio il selettore può fare riferimento a un tipo o tag (come ad esempio *body*) o a una classe o a un identificatore. Il blocco di dichiarazioni invece è costituito da una proprietà e un valore assegnato a quella proprietà. Alcuni esempi di proprietà sono *width* e *height*, che vengono utilizzate per impostare altezza e larghezza di un determinato elemento, *color*, che viene utilizzata per definire il colore del testo di un elemento, e *text-align*, che viene utilizzato per definire l'allineamento degli elementi.

TypeScript

TypeScript è un linguaggio di programmazione fortemente tipizzato open source sviluppato da Microsoft. A differenza di JavaScript, TypeScript presenta tipi, classi e interfacce. Il linguaggio estende la sintassi di JavaScript in modo che qualunque programma scritto in quel linguaggio possa funzionare con TypeScript senza alcuna modifica. TypeScript presenta quindi un sistema di annotazione dei tipi, che consente il controllo di questi tipi durante la fase di

compilazione, anche se è comunque possibile utilizzare la tipizzazione dinamica di JavaScript. I tipi primitivi utilizzabili in Typescript sono *number*, *string* e *boolean*.

C#

C# è un linguaggio di programmazione orientato a oggetti e indipendente dai tipi. La sintassi di C# prende spunto da vari linguaggi nati precedentemente, come C++ o Java. C# solitamente viene utilizzato per sviluppare molti tipi di applicazioni eseguite in .NET, la piattaforma di sviluppo ideata e sviluppata da Microsoft. La sintassi di base è molto simile a quella dei linguaggi C, C++ e Java. C#, per la web application *Rapporti di Lavoro*, viene utilizzato per lo sviluppo della parte server, con i Controller che verranno descritti nel prossimo capitolo.

2.1.3 Database - Microsoft SQL Server

Database

Un *Database* è un insieme di dati strutturati memorizzati elettronicamente in un sistema informatico. I database solitamente vengono controllati e amministrati da un sistema DBMS (*Database Management Sysyem*). Esistono diversi tipi di database, che vengono descritti di seguito.

Database Relazionali I database relazionali sono stati introdotti negli anni '80. All'interno di un database relazionale i dati sono organizzati sotto forma di insiemi di tabelle composte da colonne e record. Questo tipo di tecnologia offre la soluzione più efficiente e flessibile per accedere alle informazioni. Le tabelle del database relazionale spesso includono un identificatore univoco per ogni riga all'interno della tabella, chiamato chiave primaria. Questa chiave primaria consente di indicizzare i dati e può anche essere utilizzata per condividere valori tra più tabelle all'interno di un database. I dati che fanno riferimento ad altri e che vengono utilizzati per mettere in relazione tra di loro le tabelle sono chiamati chiavi esterne. Questa tipologia di database è quella utilizzata per la web application *Rapporti di Lavoro*. Il linguaggio utilizzato per la gestione di questa tipologia di database è *SQL* e verrà approfondito nei prossimi paragrafi.

Database orientati agli oggetti All'interno di un database orientato agli oggetti, le informazioni vengono rappresentate sotto forma di oggetti, come nel tipo di programmazione omonima. In un database orientato agli oggetti un set

di dati viene associato con tutti i suoi attributi ad un unico oggetto. Oltre agli attrubuti, negli oggetti vengono memorizzati anche i metodi, infatti, come nel metodo di programmazione, ogni oggetto ha determinate attività che può svolgere. A ogni oggetto viene assegnato automaticamente un'identificazione univoca, in modo da indirizzare e richiamare facilmente gli oggetti salvati.

Database distribuiti Un database distribuito è composto da almeno due file presenti in due sedi diverse. Il database può quindi essere essere memorizzato su più computer all'interno della stessa rete o anche in reti diverse.

Data warehouse Un data warehouse è un tipo di database progettato espressamente per query e analisi veloci e consiste in un repository centralizzato per i dati. In particolare questo tipo di database viene utilizzato per il supporto di attività di business intelligene, in particolare gli analytics.

Database NoSQL Un database NoSQL, detto anche non relazionale, consente di archiviare e manipolare dati non strutturati e semi-strutturati, a differenza di quanto accade con un database relazionale che definisce come devono essere composti tutti i dati inseriti nel database. Il punto di forza di questo tipo di database è quindi la possibilità di avere modelli di dati flessibili e a differenza dei database relazionali i dati correlati tra loro non devono essere suddivisi tra tabelle, ma i dati correlati sono annidati all'interno di un unica struttura dati.

Database grafici Un database grafico memorizza i dati in termini di entità e relazioni tra le entità. Questo tipo di database quindi è mappato su grafici, tramiti i quali le informazioni e le relative relazioni sono visualizzate chiaramente e memorizzate come un insieme di dati ampio e coerente.

Database OLTP OLTP è un database di analisi dei dati veloce progettato per far fronte a un gran numero di transazioni eseguite da più utenti.

SQL

SQL (*Structured Query Language*) è un linguaggio di programmazione utilizzato dalla maggior parte dei database relazionali per l'esecuzione di query e la manipolazione e la definizione dei dati. SQL è stato progettato per eseguire le seguenti operazioni:

- *DDL* (*Data Definition Language*), ovvero creare e modificare gli schemi del database;

- *DML* (*Data Manipulation Language*), ovvero inserire, modificare e gestire dati memorizzati;
- *DQL* (*Data Query Language*), ovvero interrogare i dati memorizzati;
- *DCL* (*Data Control Language*), ovvero creare e gestire strumenti di controllo e accesso ai dati.

Data Definition Language *DDL* viene utilizzato per la creazione, la modifica o l'eleminazione degli oggetti in un database. Sono quindi i comandi che definiscono la struttura del database. Per eseguire questi comandi, l'utente che li esegue deve avere i permessi necessari, che vengono assegnati tramite il *DCL*. Il linguaggio è quindi utilizzato in fase di progettazione del database. Questi comandi possono agire su *Domini* (per creare altri tipi da assegnare agli attributi), *Schemi* (per creare una collezione di tutti gli oggetti che faranno parte della base di dati), *Database*, *Tabelle* e *Indici*. Di seguito vengono mostrati alcuni dei comandi del *DDL*.

- *CREATE DOMAIN/SCHEMA/DATABASE/TABLE/INDEX*, utilizzato per la creazione di nuovi domini, schemi, database, tabelle e indici;
- *DROP DOMAIN/SCHEMA/DATABASE/TABLE/INDEX*, utilizzato per la rimozione di domini, schemi, database, tabelle e indici esistenti;
- *ALTER DATABASE/TABLE*, utilizzato per la modifica di database e tabelle esistenti.

Nella creazione delle tabelle è possibile aggiungere anche dei vincoli, descritti di seguito:

- *NOT NULL*, vincolo che impone che l'attributo al quale è assegnato non può assumere valori nulli;
- *UNIQUE*, vincolo che impone che per l'attributo al quale è assegnato non possa comparire lo stesso valore su righe diverse;
- *PRIMARY KEY*, vincolo che indica qual è la chiave primaria della tabella, ovvero qual è l'attributo che identifica in modo univoco una riga all'interno di una tabella;
- *FOREIGN KEY*, vincolo che impone che uno o più attributi fanno riferimento a una chiave primaria di un'altra tabella. In questo modo, per l'attributo che viene chiamato chiave esterna (appunto, foreign key) è possibile inserire solo valori presenti nella tabella referenziata.

Data Manipulation Language *DML* viene utilizzato per l'inserimento, la modifica e l'eliminazione dei dati all'interno delle tabelle di un database. La struttura di questi dati viene definita con l'utilizzo del DDL. Per eseguire i comandi del DML, l'utente che li esegue deve avere i permessi necessari, assegnati tramite il *DCL*. Di seguito vengono mostrati alcuni dei comandi del *DML*:

- *INSERT*, che consente di inserire dati nelle tabelle. Le colonne di destinazione dei valori possono essere dichiarate oppure no nel comando. La sintassi del comando Insert è la seguente:

  ```
  INSERT INTO nomeTabella (elencoColonne)
  VALUES (elencoValori)
  ```

  dove *elencoColonne* è l'elenco delle colonne in cui inserire i valori e *elencoValori* è l'elenco dei valori da inserire nella tabella, che devono essere inseriti rispettando l'ordine dei campi dichiarati nell'elenco colonne;

- *UPDATE*, che consente di modificare i dati presenti nelle tabelle. E' possibile effettuare degli update generici o con condizione. La sinstassi del comando Update è la seguente:

  ```
  UPDATE nomeTabella
  SET nomeCampo = nuovoValore
  WHERE condizione
  ```

  dove dopo la parola chiave *SET* bisogna inserire l'assegnamento del nuovo valore al campo desiderato, mentre dopo la parola chiave *WHERE* vanno inserire le condizioni per effettuare una selezione sulle righe da modificare;

- *DELETE*, che consente di eliminare i dati nelle tabelle. Il comando può effettuare operazioni in modo generico cancellando tutte le righe della tabella oppure ponendo delle condizioni. La sisntassi del comando Delete è la seguente:

  ```
  DELETE FROM nomeTabella
  WHERE condizione
  ```

  dove dopo la parola chiave *WHERE* vanno inserire le condizioni per effettuare una selezione sulle righe da eliminare.

Data Query Language *DQL* viene utilizzato per la lettura e l'elaborazione dei dati presenti in un database. Questi dati vengono inseriti tramite il *DML*

con strutture che sono state create con il *DDL*. Per eseguire questi comandi, l'utente che li esegue deve avere i permessi necessari, che vengono assegnati tramite il *DCL*. Il comando principale per l'estrazione dei dati è *SELECT*. La sintassi è la seguente:

```
SELECT listaElementi
FROM listaTabelle
WHERE condizione
[GROUP BY listaColonne]
[ORDER BY listaColonne]
```

dove *listaElementi* è l'elenco dei campi da estrarre, *listaTabelle* è l'elenco delle tabelle da cui estrarre i dati, *condizione* è l'elenco delle condizioni che un campo deve rispettare per potere essere selezionato dalla query e *listaColonne* è l'elenco di colonne di riferimento per l'ordinamento dei dati. La clausola *GROUP BY* viene utilizzata per raggruppare i risultati mentre la clausola *ORDER BY* viene utilizzato per ordinarli.

Data Control Language *DCL* viene utilizzato per concedere o revocare agli utenti i permessi necessari per poter utilizzare i comandi *DML* e *DDL*. I comandi *DCL* sono *Grant* e *Revoke*. *Grant* concede uno o più permessi ad un determinato utente, mentre *Revoke* li revoca.

Microsoft SQL Server

Come già anticipato in precedenza, i database vengono controllati e amministrati da un DBMS (*Database Management System*), ovvero un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione di database. In particolare, per la web application *Rapporti di Lavoro* viene utilizzato *Microsoft SQL Server* che è un *DBMS* relazionale prodotto da Microsoft. *Microsoft SQL Server* usa una variante del linguaggio SQL standard, ovvero il *Transact-SQL* o *T-SQL*. *T-SQL* è l'estensione proprietaria del linguaggio *SQL* sviluppata da Microsoft. In aggiunta all'SQL standard, *T-SQL* aggiunge funzioni per controllo di flusso, definizione di variabili locali e funzioni per la manipolazione di stringhe o date.
Per quanto riguarda il controllo di flusso *T-SQL* implementa alcune keyword come ad esempio *IF* e *ELSE*, che consentono l'esecuzione condizionale di blocchi di istruzioni, *BEGIN* e *END*, che delimitano un blocco di istruzioni, o *WHILE*, che implementa dei cicli.
Invece, per quanto riguarda la definizione di variabili locali, *T-SQL* implemen-

ta l'utilizzo di variabili locali, accessibili solo allo script che le utilizza, mentre invece non è possibile implementare variabili globali. Le parole chiave relative all'utilizzo di variabili locali sono *DECLARE*, utilizzata per dichiarare una variabile, e *SET*, utilizzata per attribuirle un valore. Di seguito viene mostrato un esempio di come utilizzare queste parole chiave:

```
DECLARE @nomeVariabile tipoVariabile
SET @nomeVariabile = valore
```

2.2 Stumenti utilizzati

2.2.1 Microsoft Entity Framework Core

Entity Framework Core è una versione open source e multipiattaforma della tecnologia di accesso ai dati *Entity Framework*, che consente agli sviluppatori di lavorare con i dati sotto forma di oggetti, senza doversi preoccupare della struttura del database in cui sono memorizzati questi dati e hanno quindi la possibilità di lavorare a un livello più alto di astrazione. Quindi, *Entity Framework Core*, o *EF Core*, può essere utilizzato come mapper relazionale a oggetti che consente agli sviluppatori .NET di usare un database usando semplicemente degli oggetti e elimina la necessità della maggior parte del codice di accesso ai dati.
Con *EF Core*, l'accesso ai dati viene eseguito tramite un modello, costituito da classi di entità e da un oggetto contesto (*Context*) che rappresenta una sessione con il database. L'oggetto *Context* consente l'esecuzione di query e il salvataggio dei dati. Tramite questo oggetto è quindi possibile eseguire query ai dati memorizzati nel database e salvarne di nuovi.
Per quanto riguarda l'eseguire query, è possibile effettuare i vari comandi appartenenti al *DQL*. Di seguito viene mostrato un esempio per l'esecuzione di una query:

```
using (var context = new DbContext())
{
  var dati = context.Entity
      .Where( x => x.id == 1).ToList();
}
```

All'interno di questo esempio la variabile *context* è il riferimento al database e *Entity* è la classe di rifermiento alla'entità desiderata. Con il metodo *Where* è possibile applicare dei filtri ai dati e con il metodo *ToList* i dati che soddisfano

le condizioni inserite nel metodo *Where* vengono restituiti in un oggetto *List* che rappresenta un elenco di oggetti di un determinato tipo.
Invece, per quanto riguarda il salvataggio di dati, è possibile eseguire i vari comandi appartenenti al *DML*. Di seguito viene mostrato un esempio:

```
using (var context = new DbContext())
{
  var nuovoDato = new Entity();

  context.Entity.Add(nuovoDato);
  context.SaveChanges();
}
```

Questo esempio mostra come inserire un nuovo elemento all'interno della tabella *Entity*. Infatti, con il metodo *Add* è possibile inserire l'oggetto *nuovoDato* nella tabella *Entity*. Invece, il metodo *SaveChanges* applica le modifiche al database. Senza l'utilizzo di questo metodo gli aggiornamenti apportati al database vengono annullati.