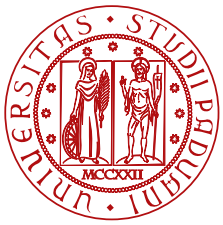


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Integrazione di servizi e strumenti di sviluppo per un'applicazione web

Relatore
Prof. Rodà Antonio

Laureando
Cognolatto Federico

ANNO ACCADEMICO 2023-2024

Data di laurea 15/07/2024

Sommario

Un settore che ha acquisito sempre più importanza negli ultimi anni, è quello dello sviluppo web, creando la necessità di utilizzare nuovi strumenti e metodi per lo sviluppo di applicazioni web con funzionalità complesse. Lo scopo di questa tesi è quello di mostrare un approccio moderno allo sviluppo di web app, trattando alcune delle tecnologie utilizzate attualmente, e un esempio di processo di sviluppo portando come esempio un'esperienza di tirocinio. In questa tesi vengono trattati i seguenti argomenti:

- Framework e tecnologie di sviluppo lato client.
- Framework e tecnologie di sviluppo lato server.
- Ambienti di sviluppo e DBMS(Database Management System).
- Esempio di sviluppo di una web app.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nel corso degli anni l'utilizzo di internet ha raggiunto una tale diffusione da entrare a far parte della vita quotidiana delle persone, infatti chiunque possegga almeno uno smartphone lo utilizza quotidianamente per lavoro o intrattenimento. Visto questo fenomeno le attività di ogni settore hanno iniziato a utilizzare piattaforme web per pubblicizzare i propri prodotti, oppure a creare il proprio sito. "Nel mondo quasi cinque miliardi di persone, poco più del 60% della popolazione mondiale, sono attive sui social. Oltre il 64% invece utilizza Internet globalmente."- [1] Oltre ai social si sono diffuse anche app di qualsiasi tipo e che forniscono svariati servizi, come home-banking e gestione delle mail. Ma oltre a questi, molte aziende di qualsiasi settore hanno deciso di creare proprie applicazioni, per gestire e fornire alcuni servizi, come per esempio supermercati, centri sportivi e molti altri. Questo ha portato alla crescita del settore dello sviluppo web, e alla nascita delle web app, che sono sempre più diffuse grazie alla compatibilità con quasi ogni dispositivo in commercio. Seguendo questo fenomeno, e grazie alla presenza di connessioni internet sempre più stabili e potenti, molte aziende che si occupano di sviluppo di software, hanno iniziato ad investire sempre più risorse per evolvere i propri applicativi in grado di essere usufruiti senza installare nulla sul dispositivo dell'utente, mettendo a disposizione applicativi come gestionali, video editor e molti altri, come se fossero dei semplici siti web. Questo permette di ridurre i costi di sviluppo e di assistenza da parte delle aziende, e rende accessibili funzioni che richiedono una considerevole potenza hardware anche agli utenti con dispositivi relativamente economici. L'obbiettivo di questo elaborato, e' quello di mostrare alcuni degli strumenti che vengono utilizzati nello sviluppo delle web app, e di raccontare di come e' possibile arricchire di funzioni aggiuntive un app , che di base e' nata come un gestionale, per trasformarla in una vera propria piattaforma in grado di interagire con altri servizi web esterni, tramite la mia esperienza di tirocinio presso l'azienda Sia Informatica.

Capitolo 2

Angular

Angular è un framework open source sviluppato da Google per la creazione di applicazioni web dinamiche. Si tratta di un framework front-end che facilita lo sviluppo di applicazioni single-page (SPA), in cui il contenuto viene caricato dinamicamente su una singola pagina, migliorando l'esperienza utente e riducendo il tempo di caricamento. Alcune caratteristiche chiave di Angular sono:

- **Two-way data binding:** Un sistema che sincronizza automaticamente i dati tra la vista (interfaccia utente) e il modello (dati). Questo semplifica la gestione dei dati e la loro visualizzazione dell'interfaccia utente.

- **Dependency Injection:** Angular utilizza un sistema di "dependency injection" che semplifica la gestione delle dipendenze tra i diversi componenti di un'applicazione.

- **Modularità:** Le applicazioni Angular sono organizzate in moduli, che consentono di suddividere il codice in blocchi funzionali e riutilizzabili.

- **Routing:** Angular fornisce un sistema di routing che permette la navigazione tra diverse pagine o viste senza dover ricaricare l'intera pagina.

- **Direttive:** Le direttive consentono di estendere o modificare il comportamento degli elementi HTML. Angular fornisce direttive predefinite e permette agli sviluppatori di crearne di personalizzate.

Angular è scritto in TypeScript, un superset di JavaScript che aggiunge tipizzazione statica al linguaggio. Angular è ampiamente utilizzato nell'industria per lo sviluppo di applicazioni web complesse e scalabili. Angular utilizza un compilatore per astrarre la complessità degli strumenti e ottimizzare il codice in modo da potersi concentrare sulla creazione dell'app.

2.0.1 Moduli e componenti

I componenti sono i principali elementi costitutivi per le applicazioni Angular. Ogni componente è composto da:

- Un modello HTML che dichiara ciò che viene visualizzato sulla pagina.
- Una classe TypeScript che ne definisce il comportamento.
- Un selettore CSS che definisce il modo in cui il componente viene utilizzato in un modello.
- Opzionalmente, gli stili CSS applicati al modello.

Ogni componente è rappresentato da una classe TypeScript. Questa classe contiene la logica del componente, gestisce i dati e definisce il comportamento. La classe del componente è decorata con il decoratore `@Component` che fornisce metadati su come il componente dovrebbe comportarsi. Questi metadati includono il selettore (selector) del componente, il template HTML associato e altre configurazioni.

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-mio-componente',
  templateUrl: './mio-componente.component.html',
  styleUrls: ['./mio-componente.component.css']
})
export class MioComponenteComponent {
  // Proprietà e metodi del componente
}
```

Figura 2.1: Scheletro classe TypeScript

Il template HTML definisce la struttura dell'interfaccia utente del componente. Può contenere direttive Angular, binding, logica condizionale e iterazioni. Il template è associato alla classe del componente tramite il decoratore @Component.

```html
<!-- mio-componente.component.html -->
<div>
  <h1>{{ titolo }}</h1>
  <p>{{ testo }}</p>
</div>
```

Figura 2.2: Esempio modello HTML

Ogni componente può avere uno o più file di stile associati per definire l'aspetto e lo stile del componente. Gli stili possono essere definiti direttamente nel file del componente o in file separati (ad esempio, CSS o SCSS).

Figura 2.3: Esempio file stile

I componenti devono essere registrati in un modulo Angular. Un modulo è un raggruppamento logico di componenti, direttive, servizi e altri moduli. La classe del modulo è decorata con `@NgModule` e contiene dichiarazioni di componenti, importazioni di moduli, provider di servizi e altri metadati.

2.0.2 Dependency Injection

La Dependency Injection in Angular viene utilizzata per consentire alle classi con decoratori Angular(es. `@Component`) di configurare le dipendenze di cui hanno bisogno, facilitando l'interazione tra i consumatori di dipendenza e i fornitori di dipendenza utilizzando un'astrazione chiamata Iniettore. Quando viene richiesta una dipendenza, l'iniettore controlla il suo registro per verificare se è già disponibile un'istanza. In caso contrario, una nuova istanza viene creata e archiviata nel Registro di sistema. Angular crea un iniettore a livello di applicazione (noto anche come iniettore "root") durante il processo di bootstrap dell'applicazione, così come qualsiasi altro iniettore, se necessario. Per creare una classe iniettabile basta decorarla con `@Injectable`.

```
@Injectable()
class HeroService {}
```

Successivamente per utilizzarla dobbiamo configurare il campo providers del componente che richiede la dipendenza con il nome della classe che la fornisce.

Figura 2.4: Esempio della configuarzione dei providers preso da [2]

In questo modo la classe esempio diventa disponibile per tutte le istanze del componente e dei componenti che lo utilizzano. Il modo più comune per iniettare una dipendenza è dichiararla in una classe costruttrice. Quando Angular crea una nuova istanza di un componente, una direttiva o una classe pipe, determina quali servizi o altre dipendenze di cui la classe ha bisogno guardando i tipi di parametri del costruttore.

```
@Component({ … })
class HeroListComponent {
  constructor(private service: HeroService) {}
}
```

Figura 2.5: Esempio di dependency injection preso da [2]

Oppure usando il metodo inject.

```
@Component({ … })
class HeroListComponent {
  private service = inject(HeroService);
}
```

Figura 2.6: Esempio di uso del metodo inject preso da [2]

Quando Angular rileva che un componente dipende da un servizio, controlla prima se l'iniettore ha istanze esistenti di tale servizio. Se non esiste un'istanza del servizio richiesto, l'iniettore ne crea uno utilizzando il provider registrato e lo aggiunge all'iniettore prima di restituire il servizio ad Angular. Quando tutti i servizi richiesti sono stati risolti e restituiti, Angular può chiamare il costruttore del componente con tali servizi come argomenti.

2.0.3 Direttive

Le direttive in Angular sono un meccanismo che consente di estendere o modificare il comportamento degli elementi HTML. Ci sono principalmente due tipi di direttive in Angular: le direttive incorporate (built-in directives) fornite dal framework e le direttive personalizzate create dagli sviluppatori. Le direttive incorporate in Angular sono predefinite e offerte dal framework. Alcune delle direttive incorporate più comuni includono:

- **NgIf:** Utilizzata per mostrare o nascondere un elemento in base a una condizione booleana. es: `<div *ngIf="mostraElemento">Contenuto visibile se 'mostraElemento' è true</div>`
- **NgFor:** Utilizzata per iterare su una lista di elementi. es:`<li *ngFor="let elemento of listaElementi">{{ elemento }}</li>`

- **NgSwitch:** Utilizzata per gestire condizioni di switch. es:

```
<div [ngSwitch]="valoreSwitch">
  <div *ngSwitchCase="'valore1'">Contenuto per valore1</div>
  <div *ngSwitchCase="'valore2'">Contenuto per valore2</div>
  <div *ngSwitchDefault>Contenuto predefinito</div>
</div>
```

Le direttive personalizzate sono create dagli sviluppatori per estendere la funzionalità di Angular. La classe della direttiva è scritta in TypeScript e contiene la logica della direttiva. La classe è decorata con il decoratore `@Directive` e può includere metodi, proprietà e hook del ciclo di vita. Dopo aver creato la direttiva personalizzata, è possibile utilizzarla in

```
@Directive({
  selector: '[appEsempioDirettiva]'
})
export class EsempioDirettivaDirective {
  @Input() colore: string = 'red';

  constructor(private el: ElementRef) {}

  @HostListener('mouseenter') onMouseEnter() {
    this.cambiaColore(this.colore);
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.cambiaColore(null);
  }

  private cambiaColore(colore: string | null) {
    this.el.nativeElement.style.backgroundColor = colore;
  }
}
```

un componente aggiungendo il selettore della direttiva all'elemento HTML desiderato. es: `<div appEsempioDirettiva [colore]="'blue'">Prova</div>`

2.0.4 Pipes e visualizzazione dei dati

Le pipes, sono classi che sono precedute dal decoratore `@Pipe{}` e che definiscono una funzione che trasforma i valori di input in valori di output per la visualizzazione in una vista. Una vista e il più piccolo gruppo di elementi che sono responsabili per la visualizzazione dei dati e la presentazione dell'interfaccia utente (come per esempio un componente e il suo template), in base allo stato e alle interazioni dell'utente. Le pipes possono essere usate per manipolare vari tipi di dati come stringhe, valute e date. Angular fornisce delle pipes di base, di seguito un elenco con alcune di quelle comunemente utilizzate:

- DatePipe: formatta le date seguendo le regole locali

- UpperCasePipe: trasforma tutte le lettere di un testo in maiuscole
- LowerCasePipe: trasforma tutte le lettere di un testo in minuscole
- CurrencyPipe: converte un numero in una stringa di valuta, secondo le regole locali(es. da euro a dollaro)

Per utilizzare una pipe, bisogna richiamarla dopo il simbolo '|' in un espressione di template come la seguente: The hero's birthday is birthday | date . Prima di poter usare una pipe in un componente bisogna importarlo all'interno della sua classe.

```
import { Component } from '@angular/core';
import { DatePipe } from '@angular/common';

@Component({
  standalone: true,
  selector: 'app-birthday',
  templateUrl: './birthday.component.html',
  imports: [DatePipe],
})
export class BirthdayComponent {
  birthday = new Date(1988, 3, 15); // April 15, 1988 -- since month parameter is
zero-based
}
```

Figura 2.7: Esempio di import di una pipe preso da [3]

Una pipe può anche avere dei parametri che possono essere specificati dopo il nome della pipe usando il simbolo ':'. es: amount | currency:'EUR' , amount | currency:'EUR':'Euros ' Inoltre possono essere concatenate usando '|' di seguito ad un'altra pipe. Oltre a quelle fornite di base, gli sviluppatori possono creare delle pipes personalizzate. Per farlo basta creare una classe decorata con @Pipe che implementa `PipeTransform`, che viene importata da angular/core. `import { Pipe, PipeTransform} from '@angular/core';`

```
@Pipe({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: any, args?: any): any {
    // Implementa qui la logica di trasformazione
    return transformedValue;
  }
}
```

Nel campo name di @Pipe, viene definito il nome con cui richiamare la classe pipe. Per poterla utilizzare bisogna registrarla nel suo modulo.

```
import { CustomPipe } from './custom.pipe';

@NgModule({
  declarations: [
    // ... altre dichiarazioni
    CustomPipe
  ],
  // ... altre configurazioni del modulo
})
export class AppModule { }
```

E infine posso utilizzarla come una normale pipe. es: someData | customPipe

2.0.5 Data Binding

Il data binding è un meccanismo che stabilisce una connessione tra il modello (dati nell'applicazione) e la vista (interfaccia utente), e quindi permette di utilizzare proprietà della classe TypeScript del componente nel suo template, o viceversa rilevare determinati eventi che interessano parti specifiche del template, come cliccare un bottone. Angular gestisce questo meccanismo in diversi modi. Uno di questi è l'interpolazione, che viene utilizzato in un'espressione template per visualizzare il valore di una variabile o il risultato di un metodo o un istruzione TypeScript. Il codice TypeScript è contenuto nelle parentesi '' e può essere utilizzato nel modo seguente all'interno di qualsiasi parte del template: `<p>{{nomeVariabile }}</p>`. Si possono utilizzare anche array di variabili per generare elementi della vista come nell'esempio seguente.

```
<ul>
  <li *ngFor="let customer of customers">{{customer.name}}</li>
</ul>
```

Figura 2.8: Esempio di ngfor preso da [4]

Di seguito un esempio di come utilizzare il data binding con variabili di input.

```
<label for="customer-input">Type something:
  <input id="customer-input" #customerInput>{{customerInput.value}}
</label>
```

Figura 2.9: Esempio di input preso da [4]

Capitolo 3

C#

C# è un linguaggio di programmazione moderno, orientato a oggetti e indipendente dai tipi, che consente agli sviluppatori di compilare molti tipi di applicazioni eseguite in .NET. C# ha le sue radici nella famiglia di linguaggi C. C# è un linguaggio di programmazione orientato a oggetti e orientato ai componenti, e fornisce costrutti di linguaggio per supportare direttamente questi concetti, rendendo C# un linguaggio naturale in cui creare e usare componenti software. Alla base, C# è un linguaggio orientato a oggetti, con il quale è possibile definire i tipi e il relativo comportamento. Alcune di funzionalità C# sono:

- **Il Garbage Collector** recupera automaticamente la memoria occupata da oggetti inutilizzati e/o non raggiungibili.
- **I tipi nullable** sono sorvegliati dalle variabili che non fanno riferimento a oggetti allocati.
- **La gestione delle eccezioni** offre un approccio strutturato ed estendibile al rilevamento e al ripristino degli errori.
- **Le espressioni lambda** supportano tecniche di programmazione funzionale.
- **La sintassi LINQ (Language Integrated Query)** crea un modello comune per l'uso dei dati da qualsiasi origine. Il supporto del linguaggio per le operazioni asincrone fornisce la sintassi per la compilazione di sistemi distribuiti.
- **Sistema di tipi unificato**. Tutti i tipi C#, inclusi i tipi di primitiva quali int e double, ereditano da un unico tipo object radice. Tutti i tipi condividono un set di operazioni comuni. I valori di qualsiasi tipo possono essere archiviati, trasportati e gestiti in modo coerente. Supporta inoltre sia i tipi riferimento definiti dall'utente che i tipi valore, e consente l'allocazione dinamica di oggetti e l'archiviazione in linea di strutture leggere. Supporta tipi e metodi generici.

3.0.1 Garbage collector

Il Garbage Collector di C# gestisce l'allocazione e il rilascio di memoria per l'applicazione. Ogni volta che si crea un nuovo oggetto, Common Language Runtime alloca memoria per l'oggetto dall'heap gestito. Lo spazio per i nuovi oggetti verrà allocato in questo modo dal runtime fino all'esaurimento dello spazio degli indirizzi nell'heap gestito. La memoria, tuttavia, non è infinita. Alla fine il Garbage Collector deve eseguire una raccolta per liberare memoria. Il modulo di ottimizzazione del Garbage Collector consente di determinare il momento migliore per l'esecuzione di una raccolta sulla base delle allocazioni in corso. Durante l'esecuzione di una raccolta, il Garbage Collector verifica la presenza di oggetti non più usati dall'applicazione nell'heap gestito ed esegue le operazioni necessarie per reclamare la relativa memoria.

3.0.2 Tipi nullable

I tipi riferimento nullable fanno riferimento a un gruppo di funzionalità abilitate in un contesto con riconoscimento nullable che riduce al minimo la probabilità che il codice generi il runtime System.NullReferenceException. I tipi riferimento nullable includono tre funzionalità che consentono di evitare queste eccezioni, inclusa la possibilità di contrassegnare in modo esplicito un tipo di riferimento come nullable:

- Analisi del flusso statica migliorata che determina se una variabile può essere `null` prima di dereferenziarla.
- Attributi che annotano le API in modo che l'analisi del flusso determini lo stato Null.
- Annotazioni variabili usate dagli sviluppatori per dichiarare in modo esplicito lo stato Null previsto per una variabile.

L'analisi dello stato Null e le annotazioni delle variabili sono disabilitate per impostazione predefinita per i progetti esistenti, ovvero tutti i tipi di riferimento continuano a essere nullable.

3.0.3 Espressioni lambda

Un'espressione lambda è un tipo di scrittura che permette di creare funzioni anonime all'interno del codice tramite le sintassi:

- (input-parameters) => expression
- (input-parameters) => <sequence-of-statements>

Nel primo caso dati i parametri di input l'espressione lambda restituisce il risultato dell'espressione, mentre nel secondo caso vengono eseguite una serie di istruzioni come in una vera propria funzione.

Figura 3.1: Esempio di espressione lambda con una serie d'istruzione preso da [5]

Se si usa un solo parametro non sono richieste le parentesi tonde. I tipi dei parametri viene solitamente dedotto dal compilatore, ma possono anche essere specificati.

3.0.4 Language Integrated Query

LINQ (Language-Integrated Query) è il nome di un set di tecnologie basate sull'integrazione delle funzionalità di query direttamente nel linguaggio C#. In genere, le query sui dati vengono espresse come stringhe semplici senza il controllo dei tipi in fase di compilazione. Le espressioni di query vengono scritte con una sintassi di query dichiarativa. Tramite la sintassi di query è possibile eseguire operazioni di filtro, ordinamento e raggruppamento sulle origini dati usando una quantità minima di codice. Vengono usati gli stessi modelli di espressioni di query di base per eseguire una query e trasformare i dati in database SQL, set di dati ADO .NET, documenti e flussi XML e raccolte .NET. L'esempio seguente include: la creazione di un'origine dati, la definizione dell'espressione di query e l'esecuzione della query in un'istruzione `foreach`.

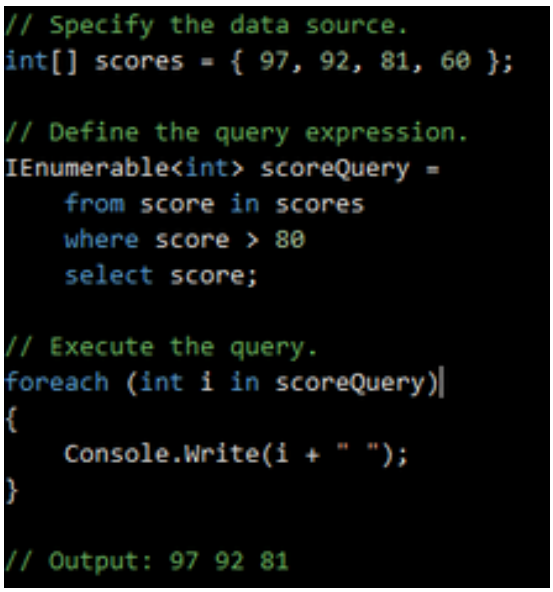


Figura 3.2: Esempio di esecuzione di una query all'interno di un ciclo foreach[6]

3.0.5 Sistema di tipi

C# è un linguaggio fortemente tipizzato. Ogni variabile e costante ha un tipo, così come ogni espressione che restituisce un valore. Ogni dichiarazione di metodo specifica un nome, e il tipo per ogni parametro di input e per il valore restituito. Quando si dichiara una variabile o una costante in un programma, è necessario specificare il tipo o usare la parola chiave `var` per consentire al compilatore di dedurre il tipo.

```
// Declaration only:
float temperature;
string name;
MyClass myClass;

// Declaration with initializers (four examples):
char firstLetter = 'C';
var limit = 3;
int[] source = { 0, 1, 2, 3, 4, 5 };
var query = from item in source
            where item <= limit
            select item;
```

Figura 3.3: Esempio di uso della parola chiave var preso da[7]

C# offre un set standard di tipi predefiniti. Questi rappresentano valori interi, valori a virgola mobile, espressioni booleane, caratteri di testo, valori decimali e altri tipi di dati. Si usano i costrutti struct, class, interface,enum e record per creare tipi personalizzati. Ci sono due punti fondamentali da comprendere sul sistema di tipi in C#:

- Supporta il principio di ereditarietà. I tipi possono derivare da altri tipi, denominati tipi di base. Il tipo derivato eredita (con alcune limitazioni) metodi, proprietà e altri membri del tipo di base, che a sua volta può derivare da un altro tipo. In questo caso, il tipo derivato eredita i membri di entrambi i tipi di base nella gerarchia di ereditarietà. Tutti i tipi, inclusi i tipi numerici predefiniti, ad esempio System.Int32 (parola chiave C#: `int`), derivano in definitiva da un unico tipo di base, ovvero System.Object (parola chiave C#: `object`). Questa gerarchia di tipi unificata prende il nome di Common Type System (CTS).

- Nel CTS ogni tipo è definito come tipo valore o tipo riferimento. Questi tipi includono tutti i tipi personalizzati nella libreria di classi .NET e anche i propri tipi definiti dall'utente. I tipi definiti usando la parola chiave `struct` sono tipi di valore. Tutti i tipi numerici predefiniti sono structs. I tipi definiti usando la parola chiave `class` o `record` sono tipi di riferimento. I tipi di riferimento e i tipi di valore hanno regole diverse e un comportamento diverso in fase di esecuzione.

La figura seguente illustra la relazione tra tipi di valore e tipi di riferimento nel CTS.

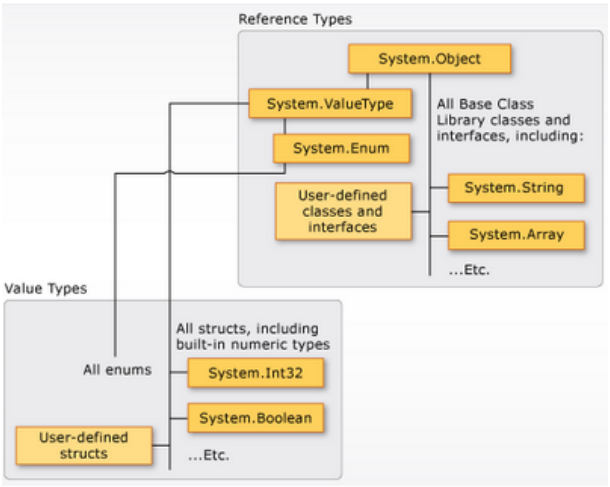


Figura 3.4: Relazione tra tipi di valore e tipi di riferimento nel CTS [7]

3.0.6 Programmazione asincrona

Se si dispone di esigenze associate a I/O , ad esempio la richiesta di dati da una rete, l'accesso a un database o la lettura e la scrittura in un file system, è consigliabile usare la programmazione asincrona. Il nucleo della programmazione asincrona è costituito da oggetti Task e Task<T> che modellano le operazioni asincrone. Sono supportati dalle parole chiavi `async` e `await`. Per il codice associato a I/O, si attende un'operazione che restituisce un Task oggetto o Task<T> all'interno di un metodo async. La parola chiave `await` è l'elemento cruciale. Restituisce il controllo al chiamante del metodo che esegue `await` ed è questo che in ultima analisi consente a un'interfaccia utente di essere reattiva o a un servizio di essere flessibile.

Capitolo 4

.NET

4.1

.NET è una piattaforma per sviluppatori open source gratuita e multipiattaforma per la creazione di molti tipi di applicazioni. .NET fa uso di vari linguaggi tra cui C#, F# e Visual Basic, riprendendo le loro caratteristiche e funzionalità. Ci concentreremo sul suo utilizzo con C#, più nello specifico della parte ASP .NET Core che tratta dello sviluppo di applicazioni web. ASP.NET Core è un framework open source multipiattaforma ad alte prestazioni per la creazione di app abilitate per il cloud e connesse a Internet.

4.1.1 Servizi

Alla base della struttura di una web app con lato server sviluppato con ASP.Net Core, ci sono delle classi chiamate servizi. Ognuna di queste classi viene creata in modo da gestire un determinato compito, come per esempio registrare i dati di un utente. Un servizio può utilizzare altri servizi per usare le loro funzioni. Per esempio: creo un servizio per registrare un utente, che ad un certo punto dovrà salvare i dati nel DB, ma per farlo impiega un servizio che gestisce le operazioni sul DB per le tabelle degli utenti.

4.1.2 Hosted Services

In ASP.NET Core le attività in background possono essere implementate come servizi ospitati. Un servizio ospitato è una classe con logica di attività in background che implementa l'interfaccia IHostedService. L'interfaccia IHostedService definisce due metodi per gli oggetti gestiti dall'host:

- StartAsync(CancellationToken)

- StopAsync(CancellationToken)

StartAsync(CancellationToken) contiene la logica per avviare l'attività in background. StartAsync viene chiamato prima:

- La pipeline di elaborazione delle richieste dell'app è configurata.
- Il server viene avviato e viene attivato IApplicationLifetime.ApplicationStarted.

StartAsync deve essere limitato alle attività a esecuzione breve perché i servizi ospitati vengono eseguiti in sequenza e non vengono avviati altri servizi fino al suo completamento. StopAsync(CancellationToken) viene attivato quando l'host esegue un arresto normale. StopAsync contiene la logica per terminare l'attività in background. Implementa IDisposable e i finalizzatori (distruttori) per eliminare tutte le risorse non gestite. BackgroundService è una classe di base per l'implementazione di un oggetto a esecuzione IHostedService prolungata. ExecuteAsync(CancellationToken) viene chiamato per eseguire il servizio in background. L'implementazione restituisce un oggetto Task che rappresenta l'intera durata del servizio in background. Non vengono avviati altri servizi fino a quando ExecuteAsync non diventa asincrono, ad esempio chiamando await. Il codice seguente crea un'attività in background timed asincrona:

```
namespace TimedBackgroundTasks;

public class TimedHostedService : BackgroundService
{
    private readonly ILogger<TimedHostedService> _logger;
    private int _executionCount;

    public TimedHostedService(ILogger<TimedHostedService> logger)
    {
        _logger = logger;
    }

    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        _logger.LogInformation("Timed Hosted Service running.");

        // When the timer should have no due-time, then do the work once now.
        DoWork();

        using PeriodicTimer timer = new(TimeSpan.FromSeconds(1));

        try
        {
            while (await timer.WaitForNextTickAsync(stoppingToken))
            {
                DoWork();
            }
        }
        catch (OperationCanceledException)
        {
            _logger.LogInformation("Timed Hosted Service is stopping.");
        }
    }

    // Could also be a async method, that can be awaited in ExecuteAsync above
    private void DoWork()
    {
        int count = Interlocked.Increment(ref _executionCount);

        _logger.LogInformation("Timed Hosted Service is working. Count: {Count}", count);
    }
}
```

Figura 4.1: Esempio di un'attività in background timed asincrona preso da[8]

4.1.3 Dependency injection

Un'interfaccia definisce un contratto. Qualsiasi classe o struttura che implementa tale contratto deve fornire un'implementazione dei membri definiti nell'interfaccia. Come in tutti i linguaggi che ne fanno uso possono contenere funzioni, ma anche costanti e altre proprietà che fanno da scheletro alla classe o struttura che la implementa. Queste sono le caratteristiche di base in C#, ma .NET utilizza le interfacce anche per un processo chiamato 'Dependency injection', ovvero una tecnica per ottenere l'inversione del controllo (IoC) tra classi e relative dipendenze. La direzione delle dipendenza all'interno dell'applicazione deve puntare verso l'astrazione, non verso i dettagli di implementazione. La maggior parte delle applicazioni viene scritta in modo che i flussi di dipendenza in fase di compilazione siano nella stessa direzione di quelli presenti nel runtime, generando un grafico di dipendenza diretta, dove una classe che richiama una funzione di un'altra classe dipenderà da essa in fase di compilazione, come mostrato nell'immagine sottostante.

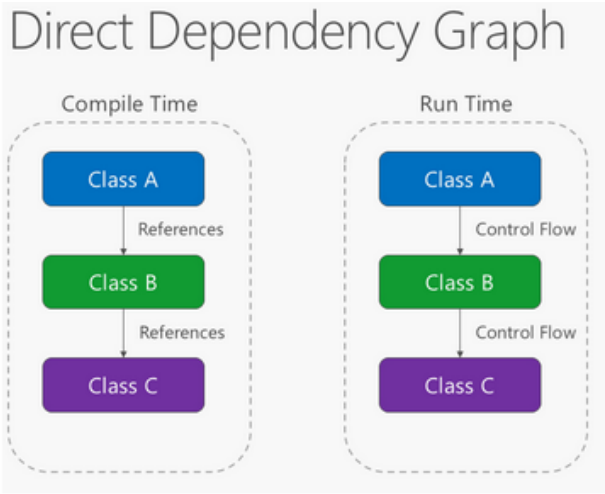


Figura 4.2: Grafo delle dipendenze dirette[9]

L'applicazione del principio di inversione delle dipendenze consente a A di chiamare i metodi su un'astrazione implementata da B, rendendo possibile chiamare A in fase di esecuzione, ma per B dipendere da un'interfaccia controllata da A in fase di compilazione (pertanto , invertendo la tipica dipendenza in fase di compilazione). In fase di esecuzione, il flusso di esecuzione del programma rimane invariato, ma l'introduzione di interfacce significa che sarà possibile collegare facilmente diverse implementazioni di tali interfacce.

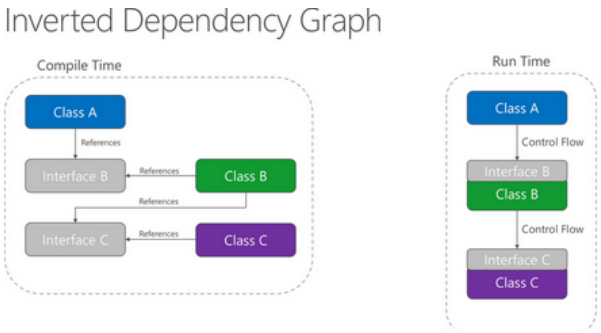


Figura 4.3: Grafo delle dipendenze inverse[9]

In ASP .NET Core, questa pratica viene utilizzata per creare oggetti relativi a servizi, repository e api, che vengono utilizzati dalla classe che ne ha bisogno. Per esempio definisco un'interfaccia e la sua implementazione.

```
public interface IMyDependency
{
    void WriteMessage(string message);
}
```

```
public class MyDependency : IMyDependency
{
   public void WriteMessage(string message)
   {
       Console.WriteLine($"MyDependency.WriteMessage Message:{message}");
   }
}
```

Nell'esempio seguente il servizio IMyDependency viene associato con il tipo concreto MyDependency. La funzione AddScoped registra il servizio con una durata con ambito, ovvero la durata di una singola richiesta.

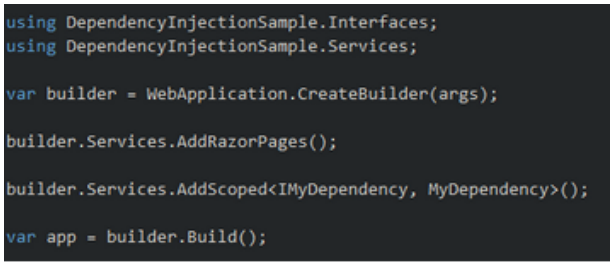


Figura 4.4: Esempio registrazione di un servizio[9]

Infine possiamo utilizzare il nostro servizio in una nuova classe nel modo seguente.

Figura 4.5: Esempio di uso di un servizio registrato[9]

4.1.4 API

API web, acronimo di "Application Programming Interface web" (Interfaccia di Programmazione delle Applicazioni web), si riferisce a un insieme di regole e protocolli che consentono a diverse applicazioni software di comunicare tra loro attraverso la rete. Le API web permettono

a diverse applicazioni di scambiare dati e funzionalità, consentendo l'integrazione e l'interoperabilità tra di esse. ASP.NET Core supporta la creazione di API Web usando controller o API minime, ma tratteremo solo dell'approccio tramite controller. I controller in un'API Web sono classi che derivano da ControllerBase. I controller vengono attivati ed eliminati per ogni richiesta. La classe ControllerBase offre molti metodi e proprietà utili per la gestione delle richieste HTTP, tra le quali quelle elencate nella tabella seguente.

Method	Note
BadRequest	Restituisce il codice di stato 400.
NotFound	Restituisce il codice di stato 404.
PhysicalFile	Restituisce un file.
TryUpdateModelAsync	Richiama l'associazione di modelli.
TryValidateModel	Richiama la convalida dei modelli.

Figura 4.6: Lista metodi supportati[10]

Lo spazio dei nomi Microsoft.AspNetCore.Mvc include attributi che possono essere usati per configurare il comportamento dei controller di API Web e i metodi di azione, tra i quali:

Attributo	Note
[Route]	Specifica il modello di URL per un controller o un'azione.
[Bind]	Specifica il prefisso e le proprietà da includere per l'associazione di modelli.
[HttpGet]	Identifica un'azione che supporta il verbo dell'azione HTTP GET.
[Consumes]	Specifica i tipi di dati accettati da un'azione.
[Produces]	Specifica i tipi di dati restituiti da un'azione.

Figura 4.7: Lista metodi d'azione[10]

Una classe controller può essere decorata con l'attributo [ApiController] per abilitare i comportamenti seguenti:

- Requisiti del routing degli attributi
- Risposte HTTP 400 automatiche
- Inferenza del parametro di origine di associazione
- Inferenza di richieste multipart/form-data
- Dettagli del problema per i codici di stato di errore

4.1.5 NuGet

Come in altri casi, a volte è necessario utilizzare librerie sviluppate da terzi che non fanno parte di quelle presenti in .NET. Questo processo, in .NET, viene gestito con i pacchetti NuGet. Un

pacchetto NuGet è un singolo file ZIP con l'estensione .nupkg che contiene codice compilato (DLL), altri file correlati a tale codice e un manifesto descrittivo che include informazioni come il numero di versione del pacchetto. Gli sviluppatori con codice da condividere creano pacchetti e li pubblicano in un host pubblico o privato. I consumer di pacchetti ottengono tali pacchetti dagli host appropriati, li aggiungono ai loro progetti e quindi chiamano le funzionalità di un pacchetto nel codice dei progetti. NuGet gestisce tutti i dettagli intermedi. In breve, un pacchetto NuGet è un'unità di codice condivisibile, ma non richiede né implica un modo particolare di condivisione. Nel ruolo di host pubblico, NuGet gestisce il repository centrale in nuget.org. NuGet consente anche di ospitare i pacchetti in modo privato nel cloud, in una rete privata o anche solo nel file system locale, rendendo i pacchetti in questo modo disponibili solo per gli sviluppatori che hanno accesso all'host.

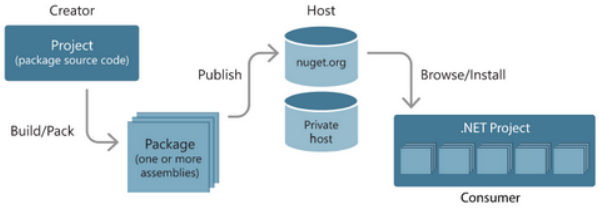


Figura 4.8: Catena pubblicazione pacchetti NuGet[11]

Per gestire i pacchetti NuGet relativi ad un progetto da Visual Studio, basta aprire l'apposita finestra dove si possono gestire i pacchetti installati o scaricarne di nuovi.

Capitolo 5

T-SQL

Transact-SQL (T-SQL) è un'estensione del linguaggio SQL (Structured Query Language) utilizzato principalmente per la gestione dei dati in database relazionali. È stata sviluppata da Microsoft ed è utilizzata principalmente su piattaforme come SQL Server, Azure SQL Database e Azure Synapse Analytics.

5.0.1 Sintassi SQL Estesa

Transact-SQL estende la sintassi standard di SQL con funzionalità aggiuntive. Queste includono istruzioni per il controllo del flusso (come IF...ELSE e LOOP), definizione di variabili, definizione di procedure e funzioni e gestione degli errori, che rendono più semplice l'esecuzione di comandi complessi, come query parametriche e operazioni sui dati che coinvolgono più tabelle.

5.0.2 Stored Procedures e Funzioni

Transact-SQL consente la creazione di stored procedures e funzioni, che sono blocchi di codice SQL riutilizzabili che possono essere chiamati da altre parti del codice o direttamente dalle applicazioni.Possono contenere una o più istruzioni SQL e possono anche includere logica di programmazione, come istruzioni di controllo del flusso, loop e gestione degli errori. Possono ricevere anche dei parametri in ingresso da usare come variabili all'interno del codice della stored procedure(es. l'id di un utente). Le stored procedure possono essere create e alterate con la sintassi:

```
CREATE [ OR ALTER ] { PROC | PROCEDURE } [ schema_name.] procedure_name
[ { @parameter data_type } [ OUT | OUTPUT ] ] [ ,...n ]
AS
{
```

```
  [ BEGIN ] sql_statement [;][ ,...n ] [ END ]
}
[;]
```

Le stored procedure vengono utilizzate per gestire operazioni complesse sui dati delle tabelle, come flussi di dati generati da un'applicazione web, spostamenti di record da una tabella temporanea ad una tabella principale, e nel caso di operazioni come la cancellazione di dati che coinvolgono più tabelle dello schema utilizzando delle politiche personalizzate.

5.0.3 Gestione degli Errori

T-SQL fornisce un sistema di gestione degli errori tramite l'uso di blocchi TRY...CATCH, come nella gestione delle eccezioni in linguaggi come java e c++. Questo consente di gestire in modo appropriato gli errori che si verificano durante l'esecuzione del codice SQL, garantendo una maggiore resilienza alle applicazioni.

5.0.4 Transazioni

Transact-SQL supporta le transazioni, che consentono di raggruppare una serie di operazioni SQL in un'unica unità atomica. Questo garantisce la coerenza dei dati, consentendo di eseguire un commit o un rollback delle operazioni in base al successo o al fallimento dell'intera transazione, evitando di eseguire solo le operazioni che andrebbero a buon fine.

5.0.5 Ottimizzazione delle Query

Transact-SQL offre un set di strumenti per ottimizzare le query, compresi piani di esecuzione, hint di query e statistiche di tabella. Questi strumenti consentono di migliorare le prestazioni delle query, garantendo che vengano utilizzati gli indici corretti e vengano scelti i metodi di accesso più efficienti.

5.0.6 Gestione dei Ruoli e delle Autorizzazioni

T-SQL consente di gestire i ruoli e le autorizzazioni per controllare l'accesso ai dati e alle risorse del database. Questo include la creazione di ruoli personalizzati, l'assegnazione di autorizzazioni specifiche e la gestione della sicurezza a livello di database. In sintesi, Transact-SQL è un linguaggio potente e flessibile per la gestione dei dati in ambienti di database relazionali Microsoft, offrendo una vasta gamma di funzionalità per la manipolazione dei dati, la gestione degli errori, l'ottimizzazione delle prestazioni e la sicurezza. Il vantaggio principale è di velocizzare la scrittura delle query e di tutta la struttura del database grazie alla compatibilità di t sql con i

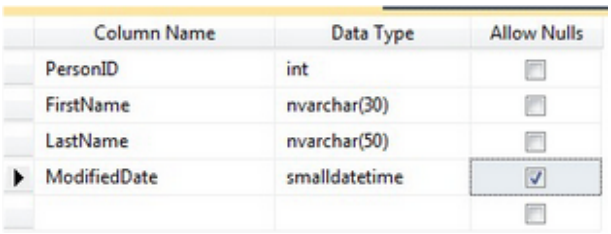


Figura 5.1: Esempio di table designer [12]

dbms come SQL Server. Sql server mette a disposizione strumenti grafici per facilitare la creazione e modifica di tabelle,viste,schema e stored procedures, e manipolare i dati associati ad essi. Di seguito un esempio di table designer di SQL Server Management Studio.

Da questa finestra è possibile effettuare diverse operazioni, come aggiungere o eliminare colonne, modificare le relazioni che coinvolgono la tabella e verificarne le proprietà.

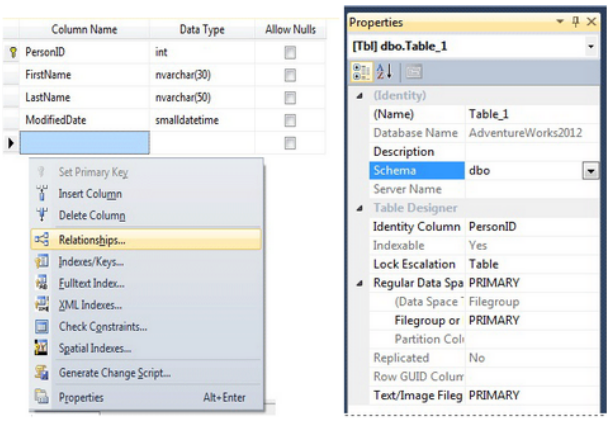


Figura 5.2: Esempio finestra del dettaglio di una tabella [12]

Inoltre, questo dbms offre una serie di comandi che permettono di generare delle istruzioni SQL relative a tabelle, viste, stored procedures e schemas che fungono da scheletro per la creazioni di istruzioni più complesse, che possono essere utili anche per testare rapidamente le funzioni basilari del nostro DB.

Capitolo 6

IDE

Un ambiente di sviluppo integrato(integrated development environment), è un'applicazione che agevola lo sviluppo di software, fornendo strumenti che velocizzano processi come la scrittura di codice e il debugging. Esistono molti IDE e alcuni dei più usati secondo il TOP IDE index[13] sono:

- **Visual Studio**, utilizzato principalmente per lo sviluppo di grossi progetti in C# e C++.
- **Visual Studio Code**, utilizzato per progetti più piccoli sviluppati principalmente utilizzando JavaScript, TypeScript e Node.js.
- **Eclipse**, utilizzato per progetti Java.
- **PyCharm**, utilizzato per progetti Python.

La scelta di un IDE può essere influenzata dal tipo di linguaggio utilizzato, ma grazie ai plugin(funzionalità aggiuntive) quasi ogni IDE supporta la maggior parte dei linguaggi più diffusi. Questo non implica che tutti gli IDE siano adatti a svolgere lo stesso compito, infatti ogni IDE è specializzato in determinati settori. Per esempio, Eclipse vanta strumenti di debugging molto efficaci per software scritti in Java, che permettono la creazione di apposite test suite in modo semplice e rapido. Di seguito verranno approfonditi gli IDE Visual Studio e Visual Studio Code, che sono stati gli IDE impiegati nel progetto.

6.0.1 Visual Studio

Visual Studio è un IDE creato da Microsoft, utilizzato per progetti come lo sviluppo di app per dispositivi mobile, desktop e anche videogiochi. Questo IDE viene spesso utilizzato per gestire grossi progetti, grazie ai suoi strumenti per il debugging, con i quali è possibile monitorare ogni aspetto del programma durante l'esecuzione dei test. Durante l'installazione, è possibile selezionare quali componenti installare, che variano dalle librerie del framework .NET a strumenti

per la scrittura di codice come Copilot, che si tratta di un IA che assiste gli sviluppatori nella scrittura del codice completando parti di esso. Oltre ai pacchetti forniti di base da Microsoft, è possibile installare dei plugin, che permettono di utilizzare linguaggi non supportati di base o di utilizzare librerie create da altri sviluppatori, come per esempio Google. Lo svantaggio principale di Visual Studio è che richiede l'utilizzo di molte risorse rispetto ad altri IDE.

6.0.2 Visual Studio Code

Visual Studio Code(VS Code) è un IDE creato da Microsoft, ed è molto diffuso per sviluppare siti web e progetti non troppo complessi. Di base VS Code supporta JavaScript, TypeScript e Node.js, ma grazie all'installazione di diversi plugin, può essere compatibile con la maggior parte dei linguaggi più diffusi. Rispetto alla sua controparte, VS Code non utilizza molte risorse, ma è limitato nelle funzioni di base. La mancanza di strumenti per il debugging, può essere in parte compensata dall'installazione di plugin pensati per questo scopo, ma allo stato attuale non offrono la stessa quantità di funzioni presenti in Visual Studio.

6.0.3 Quale usare

Come visto finora, ogni IDE è pensato per soddisfare varie necessità degli sviluppatori. Nel caso del progetto trattato in questa tesi, la parte back end viene sviluppata utilizzando Visual Studio, mentre la parte front end viene sviluppata su VS Code. La parte back end è scritta utilizzando .NET, e comprende molti progetti che devono essere eseguiti contemporaneamente, quindi è stato scelto Visual Studio per la sua compatibilità con il framework .NET e le sue funzioni di debugging e gestione dei progetti. La parte frontend è stata sviluppata utilizzando Angular, infatti tutti i test effettuati su questa parte vengono fatti direttamente sul browser, utilizzando gli strumenti presenti su di esso.

Capitolo 7

Sales Connect

7.1 CRM

7.1.1 Che cos'è?

Un sistema CRM(Customer Relationship Management) riunisce, collega e analizza tutti i dati dei clienti raccolti, tra cui contatti,interazioni con i rappresentanti dell'azienda, gli acquisti, le richieste di assistenza, le risorse e i preventivi/proposte. Questo permette all'azienda che utilizza un sistema CRM di elaborare questi dati per conoscere le preferenze del cliente, permettendo lo sviluppo di strategie di marketing mirate ad acquisire nuovi clienti e fidelizzare quelli gia' esistenti. Inoltre consente anche ai rappresentanti dell'azienda di relazionarsi in modo piu' efficace con i clienti e di facilitare il raggiungimento degli obbiettivi seguenti:

- Offrire e vendere nuovi prodotti aggiuntivi, al momento giusto, nel modo giusto e al prezzo giusto
- Aiutare i team di assistenza clienti a risolvere i problemi più rapidamente
- Aiutare i team di sviluppo a creare prodotti e servizi migliori

7.1.2 Diffusione

Come emerge dai dati forniti da Oracle: "Le soluzioni di Customer Relationship Management sono una delle categorie maggiori e in più rapida crescita di software di applicazioni aziendali. Il mercato del CRM è stata valutato pari a 41,93 miliardi di dollari nel 2019 e si prevede che raggiungerà 96,39 miliardi di dollari entro il 2027, con un tasso di crescita annuale dell'11,1% dal 2020 al 2027." Oracle [14], i sistemi CRM sono sempre piu' utilizzati da aziende di ogni settore, ma questo comporta un investimento da parte di quest'ultime in hardware che consente

l'utilizzo di questi software, in manuntenzione dei dispositivi e interventi da parte di tecnici che si occupano di configurare e aggiornare il sistema CRM sui dispositivi aziendali. Negli ultimi anni, in molti hanno iniziato a sostituire i classici CRM con i Cloud CRM, che non so altro che delle piattaforme web da cui e' possibile utilizzare tutte le funzioni di un normale CRM, con il vantaggio di potervi accedere dovunque e con qualunque dispositivo.

7.2 Sales Connect

7.2.1 Introduzione

Sales Connect e' un Cloud CRM sviluppato dall'azienda Sia Informatica, che permette all'utente di gestire tutti i dati che riguardano le interazioni tra la propria azienda ed i clienti, tra cui elenco degli ordini, fatture e dati anagrafici. Gli utenti in possesso di un account Sales Connect possono utilizzarlo da qualsiasi dispositivo, che sia uno smartphone o un PC, che supporta i web browser piu' diffusi come Edge, Chrome o Firefox. Inoltre e' possibile utilizzare funzioni piu' complesse, grazie al fatto che tutte le operazioni vengono effetuate dal server, lasciando solo il compito di visualizzare il risultato al dispositivo dell'utente.

7.2.2 Cosa lo distingue?

Oltre alla personalizzazzione dell'interfaccia, Sales Connect supporta molte funzioni aggiuntive rispetto ad un normale CRM. Grazie alla sua natura scalabile, Sales Connect e' in continua espansione, aggiungendo funzioni e integrando servizi esterni, per creare un'unica piattaforma da cui l'utente e' in grado di gestire ogni aspetto della propria azienda. Infatti e' gia' possibile organizzare le attivita' che devono essere svolte dai propri dipendenti, con un apposito calendario e una timeline in cui e' possibile visualizzare tutti i dipendenti registrati sul portale. I dipendenti vengono registrati manualmente come risorse dall'utente. Inoltre grazie all'integrazione con Creditsafe, un sistema che valuta l'affidabilita' di un'azienda grazie ai suoi movimenti finanziari, e' possibile monitorare lo score(rappresentala probabilità di default, ovvero la probabilità che un'impresa diventi insolvente nei 12 mesi successivi), il limite di credito e i dati finanziari dei propri clienti o partner. Un'altra funzione molto comoda e' il lettore business card, con la quale è possibile acquisire dati di nuovi contatti ed inserirli in automatico in anagrafica semplicemente fotografando il biglietto da visita.

Capitolo 8

Il progetto

8.1 Introduzione

8.1.1 Di cosa si tratta?

Come discusso nel capitolo precedente, Sales Connect e in continua espansione, e uno degli obiettivi è di connettere servizi esterni utilizzati dagli utenti mirati a gestire ogni aspetto dell'organizzazione e della gestione della propria azienda. Questo ha portato a creare il progetto di cui mi sono occupato durante il periodo di tirocinio. Il progetto consiste nell'integrare nell'applicazione web principale i servizi di posta elettronica e di calendario di google, modificando opportunamente le funzioni già presenti relative ai servizi già citati. Nel caso del calendario, l'utente deve essere in grado di visualizzare, ma non modificare, gli appuntamenti e i calendari della piattaforma su google calendari, mentre potrà visualizzare la lista di tutti i calendari google sulla nostra applicazione, selezionando quelli di cui visualizzare gli appuntamenti. Per le mail invece l'utente deve essere in grado di visualizzare e scaricare gli allegati delle mail di gmail.

8.1.2 Progettazione

La fase di progettazione si è svolta principalmente in due parti:

- Studio del framework aziendale e degli standard utilizzati.
- Studio del funzionamento e utilizzo delle API dei servizi Google.

Inizialmente sono stato affiancato ad un tutor aziendale che mi ha spiegato come utilizzare gli strumenti forniti dall'azienda, e come integrare le funzioni richieste nel progetto principale. Dopo aver preso familiarità con i metodi di sviluppo aziendali, insieme al team di sviluppo, è stata stilata una lista di requisiti funzionali da soddisfare. Da qui ho potuto procedere alla

ricerca e lo studio delle funzioni utili al raggiungimento dell'obiettivo del progetto messe a disposizione da Google. Prima di definire i requisiti non funzionali, sono stati necessari svariati test che hanno portato alla creazione di applicazioni di test per riuscire ad autenticare un utente utilizzando il suo account Google. Per finire, è stato progettato il DB basandosi su quello già esistente associato all'app principale, adattandolo alla struttura dei dati scambiati con le API di Google.

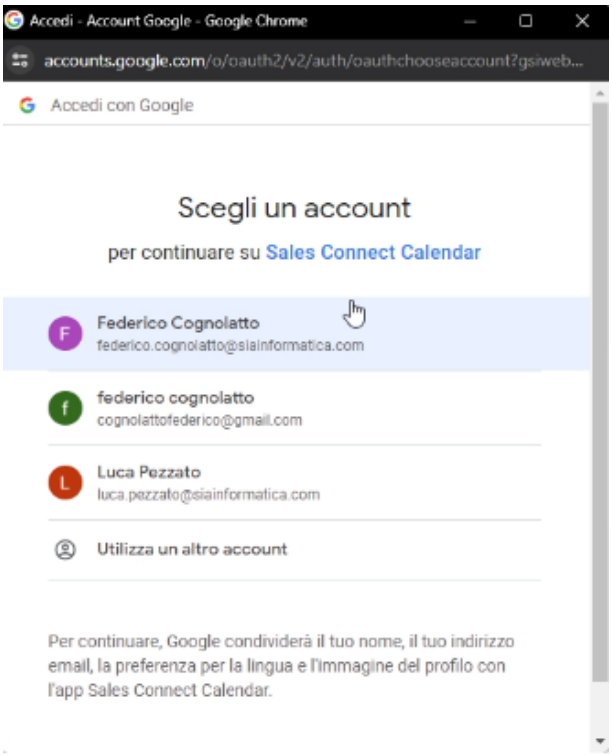
8.1.3 Implementazione

Terminata la fase di progettazione, è stata creata una sezione apposita nel progetto principale che ospita le implementazioni Google del progetto e quelle future. In seguito è stata creata una struttura base che può' essere riutilizzata per tutte le implementazioni future tramite l'uso delle API di Google, evitando di riscrivere grosse parti di codice per ogni funzione. Insieme alla base, è stata implementata la nuova parte del DB che si occupa di associare i dati ricevuti da Google all'account Sales Connect dell'utente. Infine sono state implementate le funzioni di calendario e di casella di posta di Google, partendo dalla logica della parte server. Per l'implementazione lato client, è stata necessaria una ristrutturazione generale dell'interfaccia utente, per renderla più semplice e intuitiva.

8.2 Integrazioni

8.2.1 Autenticazione

Il processo è composto da una parte lato client e una parte lato server. Nella parte client viene fatta prima di tutto una richiesta http alle API di google per richiedere i permessi di accesso ai dati dell'utente. In seguito al login dell'utente con il suo account google, il client riceve un token con i dati di autenticazione e la mail dell'utente, che è necessaria per utilizzare le funzioni delle librerie di Google. Per ottenere un token da utilizzare per un servizio specifico bisogna includere nella chiamata alle API di Google degli scope, che specificano il tipo di permessi e le informazioni che si vogliono ricevere, presi dalla lista ufficiale di Google (https://developers.google.com/identity/protocols/oauth2/scopes?hl=it), in modo da ottenere tutti i permessi e i dati necessari per utlizzare le librerie di Google lato server. Nel lato client è stato creato un nuovo modulo , che contiene due componenti Angular.Il primo si occupa esclusivamente di ottenere il token dalle API, e di inviarlo tramite una richiesta http ad un apposito endpoint, che invia i dati necessari come parametri per la funzione richiesta, ad un servizio lato server. Il tutto si traduce in una semplice schermata di login.



Infine un servizio lato server, si occupa della parte finale dell'autenticazione, dove viene effettuata una semplice richiesta post alle api di google che oltre ai tokens restituisce anche un id token, che si tratta di un jwt token che contiene la mail dell'utente oltre a le informazioni relative al token stesso. L'id token viene decifrato tramite l'uso della libreria di jwt. Al termine del processo, il servizio chiama un repository, che si occupa di salvare i dati nel DB, e un Hosted Service si occupa di rigenerare il token quando questo sta per scadere. Anche nella parte server è stato creato un progetto apposito nella sezione google dell'app, per rendere i servizi riutilizzabili per implementazioni future di altri servizi Google.

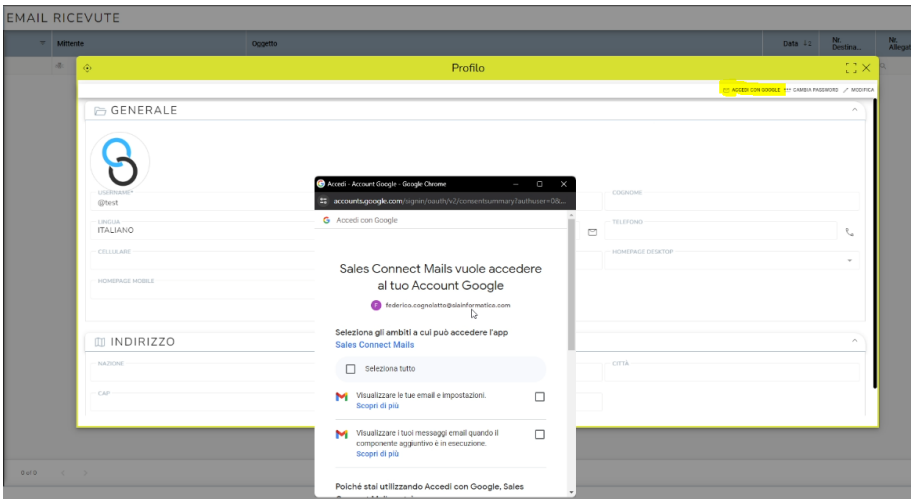
8.2.2 Calendario

Le chiamate alle api di google calendar vengono gestite con la libreria messa a disposizione da google, utilizzando la versione in C# per applicazioni .NET, infatti le informazioni ottenute precedentemente vengono poi unite alla mail dell'utente e a dei parametri aggiuntivi per generare un oggetto CalendarService, che si occupa di gestire tutte le chiamate alle API di Google calendar. In base ai permessi richiesti con la prima chiamata del client, l'applicazione è in grado di leggere e/o modificare e inserire nuovi dati nei calendari Google dell'utente. In questo caso l'applicazione è in grado di gestire interamente tutti i dati del calendario di un utente. Quando un access token sta per scadere, viene richiamata una funzione che scambia il refresh token con un nuovo access token e refresh token. Durante la procedura di login, vengono creati su Google

Calendar i calendari presenti nella web app, con tutti gli appuntamenti dell'utente, e viene scaricata la lista di tutti i calendari creati dall'utente su Google. Nel lato client, era già presente un componente Angular che si occupa del calendario, quindi è stato modificato in modo da essere compatibile con la struttura dei calendari e degli eventi di Google, creando anche un apposito pannello, da cui utilizzare tutte le funzioni aggiuntive del calendario. Nella parte server, come per la gestione dei token, è stato creato un nuovo progetto nella sezione Google, che si occupa di gestire tutte le operazioni relative a eventi e calendari Google. In seguito al login da parte dell'utente, è possibile scegliere dall'apposito pannello di controllo quali calendari sincronizzati, che in seguito ad una chiamata http all'endpoint relativo al servizio che gestisce gli eventi, verranno scaricati e salvati nel DB gli eventi relativi ai calendari selezionati. I calendari sincronizzati vengono segnati con un flag nel DB, e un Hosted Service si occupa di aggiornare gli eventi di quest'ultimi, in modo da visualizzare le modifiche effettuate su Google Calendar.

8.2.3 Mails

Nel caso delle mail, è stata gestita solamente la visualizzazione e lo scaricamento degli allegati. Nella parte client sono stati utilizzati i componenti Angular già presenti, creando un pulsante di login a Google specifico per Gmail nella pagina del profilo dell'utente, e una pagina dedicata alla visualizzazione delle mail.



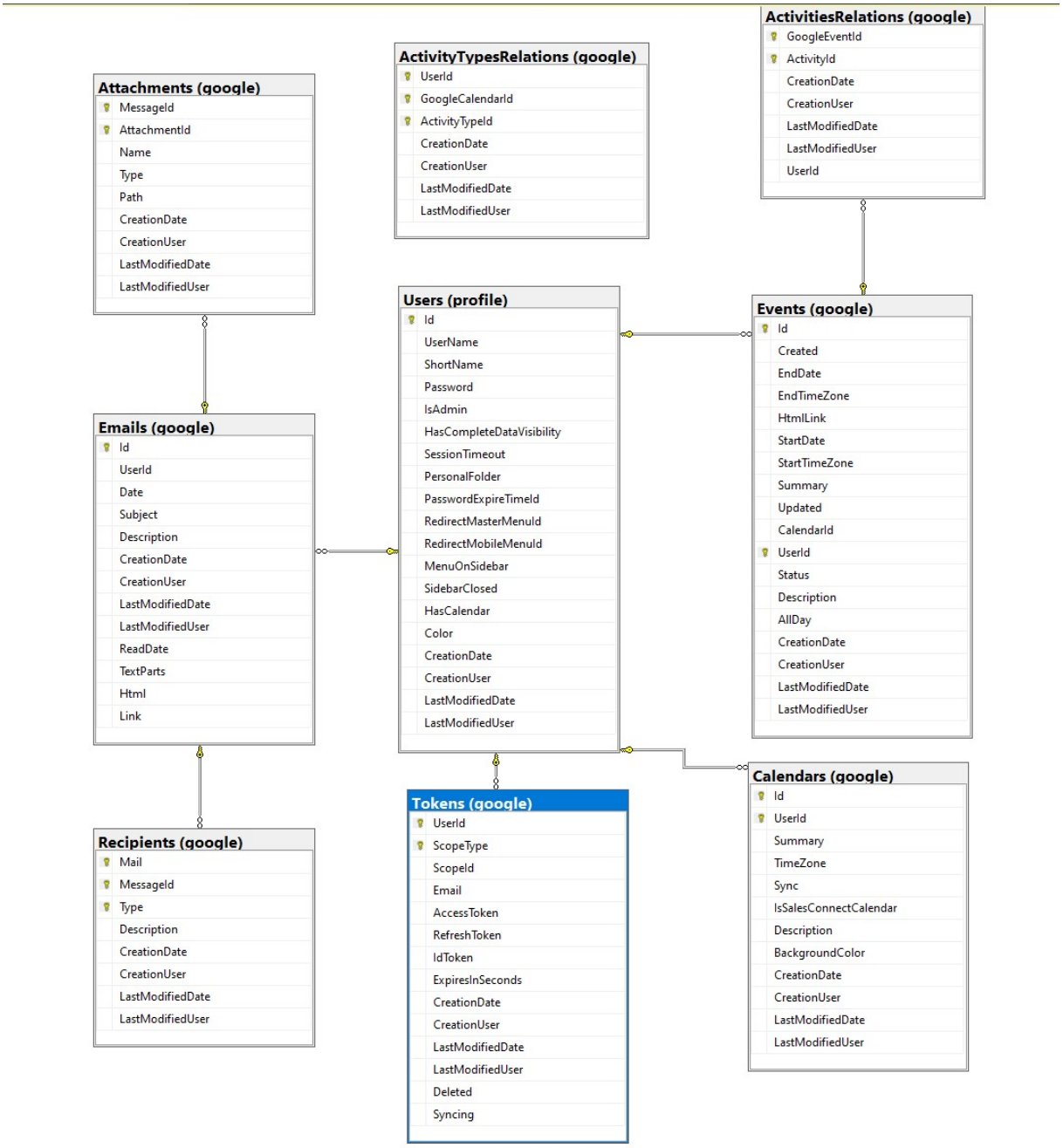
Nella parte server, è stato creato un progetto apposito nella sezione Google, che si occupa di recuperare le mail e gli allegati dell'utente, utilizzando le API di Gmail. Durante la procedura di login, vengono scaricate tutte le mail ricevute dall'utente su Gmail negli ultimi trenta giorni. In questo caso, gli oggetti utilizzati dalle classi delle librerie di Gmail, contenevano i contenuti delle mail suddivisi in vari componenti. è stato necessario creare un parser, per decodificare il corpo delle mail utilizzando la codifica base64 e leggere gli header per trovare informazioni quali:

data e ora dell'invio, mittente, destinatario e altri dello standard RFC. In seguito, i dati delle mail vengono salvati nel database, e il client viene notificato che sono disponibili nuove mail da visualizzare. Un Hosted Service si occupa di scaricare le nuove mail in arrivo nella casella di posta dell'utente ad intervalli di tempo regolari, utilizzando la stessa modalità impiegata nella procedura di login.

Capitolo 9

Database

Il DB e‘ stato realizzato ampliando quello gia‘ esistente per la gestione delle altre parti della web app. Per prima cosa e‘ stato creato un apposito schema che può essere utilizzato per l’implementazione di altri servizi Google in futuro. Essendo l’estensione di un DB già esistente, le nuove tabelle sono state create in modo da poterle associare alle tabelle già esistenti, convertendo i tipi di dati ricevuti da Google in quelli utilizzati dalla web app. Durante la fase di implementazione, sono state tralasciate alcuni vincoli d’integrità referenziale, per evitare conflitti con i tipi di dato di alcune tabelle che non permettevano il corretto funzionamento di alcune parti del framework lato server, che verranno aggiornate in futuro per risolvere i problemi di compatibilità. Di seguito lo schema che rappresenta le tabelle impiegate dai servizi di Google Calendar e Gmail.



Oltre alle tabelle, sono state create anche viste, che facilitano lo scambio d'informazioni tra lato server e lato client, quando devono essere eseguite delle operazione specifiche come visualizzare le mail di un utente. Sono state create anche delle stored procedures che popolano delle tabelle d'appoggio per gestire il flusso di dati tra i server di Google e l'applicazione, che vengono svuotate una volta terminate le operazioni di aggiornamento dei dati sulle tabelle principali, in modo da diminuire il numero di operazioni che vengono eseguite dal DB, e per gestire la cancellazione dei dati di un utente in caso di logout da Google.Nello schema google sono state

gestiti 3 servizi principali: Tokens, Gmail e Calendar.

9.0.1 Tokens

Per la gestione dei token e‘ stata creata una sola tabella collegata a quella gia‘ esistente degli utenti, dato che ogni token viene identificato da un id utente e da uno scope type, che non e‘ altro che l’ambito per cui e‘ stato rilasciato quel token(es. calendario), dato che ad ogni utente e‘ associato un solo token per servizio. In caso di token scaduto o di errori, il token viene rigenerato e i dati del record associato vengono aggiornati con quelli del nuovo token.

9.0.2 Calendar

Nel caso del calendario sono state create 4 tabelle: Calendars, Events, ActivitiesRelations e ActivityTypesRelations. Calendars tiene traccia dei dati relativi ai calendari Google dell’utente, che possono essere privati o creati dall’applicazione, e vengono individuati con il flag IsSalesConnectCalendar. Ogni calendario e‘ individuato da un utente e un id generato da google, e nel caso si tratti di un calendario non privato, viene associato ad una tabella che gestisce le tipologie di eventi del calendario dell’app tramite la tabella ActivityTypesRelations. Nella tabella Events vengono scritti solo i dati degli eventi dei calendari google con il flag Sync con valore 1 che deve essere selezionato dall’utente all’interno dell’app. Nel caso di calendari non privati, gli eventi vengono sempre registrati e associati ad una tabella che gestisce le attività dei calendari dell’app tramite la tabella ActivitiesRelations.

9.0.3 Gmail

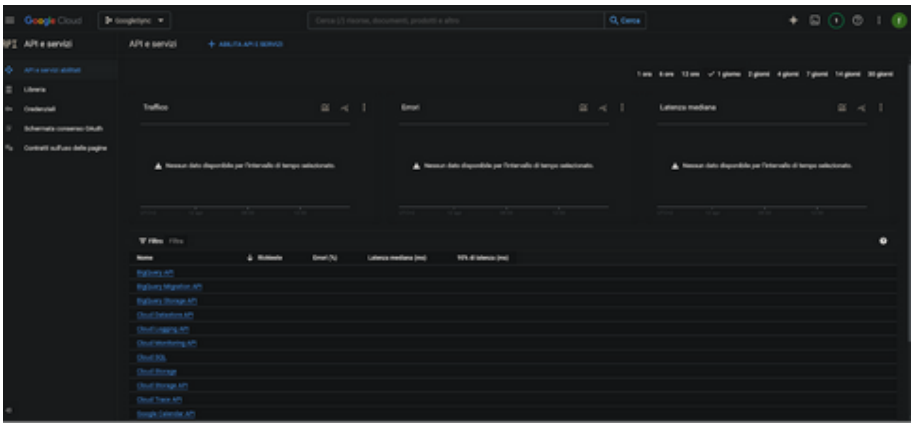
Per la gestione delle mail sono state create 3 tabelle: Emails, Recipients e Attachments. Ogni mail ha un id univoco generato da Google e nella tabella Emails, vengono salvate le informazioni generali della mail come l’oggetto e il corpo. Ogni mail e‘ associata ad un utente. In Recipients vengono registrati i dati del mittente e dei destinatari(se ce ne sono altri oltre all’utente), e vengono identificati da un tipo, una mail e l’id del messaggio a cui fanno riferimento. In Attachments vengono salvati i dati relativi agli allegati dei messaggi che vengono identificati da un id è l’id del messaggio con cui sono stati ricevuti. Viene salvato anche un path che permette all’utente di scaricare i file dal server dell’app, che viene generato temporaneamente su richiesta di quest’ultimo, dato che dopo un certo lasso di tempo il file viene eliminato dal server per evitare sprechi di memoria.

Capitolo 10

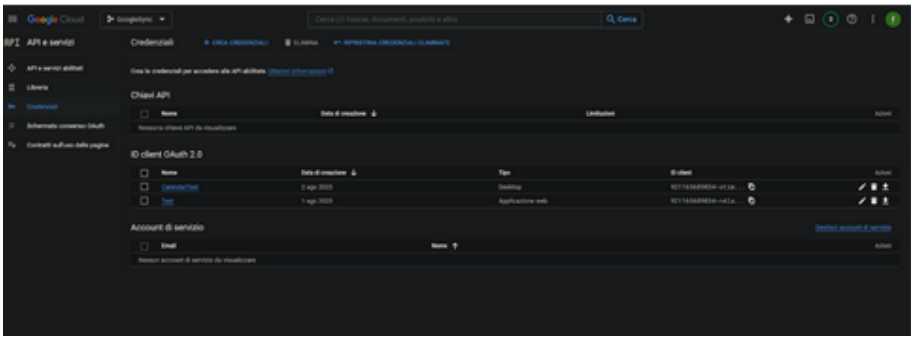
Google API

Lo scopo principale del progetto era quello di integrare alle funzionalità dell'applicazione web, la sincronizzazione della casella di posta di Gmail e i calendari Google degli utenti che associano il loro account Google a quello dell'app. Per fare questo è necessario comunicare con i servizi Google utilizzando degli strumenti chiamati API, messi a disposizione da Google stesso. Le API di Google sono strumenti che consentono agli sviluppatori di integrare le funzionalità offerte dai servizi Google nei propri progetti e applicazioni. Queste API forniscono un insieme di strumenti e protocolli che consentono agli sviluppatori di accedere e utilizzare dati e funzionalità di servizi come Google Maps, YouTube, Gmail, Google Drive e molti altri. Le API di Google consentono agli sviluppatori di creare applicazioni che sfruttano le potenti funzionalità dei servizi Google, come l'accesso ai dati, l'elaborazione delle informazioni e l'interazione con gli utenti. Nel caso di Google, l'abilitazione e i permessi relativi alle API, vengono gestiti sulla piattaforma Google Cloud. Per utilizzare i servizi bisogna creare un progetto con il proprio account.

E in seguito dalla pagina API e Servizi è possibile configurare il tipo di servizi richiesto e monitorare il traffico generato dall'uso delle API.



Per utilizzare le librerie google per ricevere dei dati è necessario autenticare l'app generando un client O Auth2.0 nella sezione credenziali, a cui sono associati un client id ed un client secret che vengono utilizzati per l'autenticazione.



Inoltre nella sezione schermata di consenso O Auth, è possibile gestire lo stato di pubblicazione dell'app(che deve essere approvata da Google tramite un processo descritto nella pagina di pubblicazione), gli utenti di prova per la fase di sviluppo, e la finestra di login a Google con la richiesta per l'autorizzazione da parte dell'utente per utilizzare i servizi richiesti.

In seguito si possono implementare le librerie fornite da Google utilizzando uno dei linguaggi compatibili tra i quali: Java, Python, C# e JavaScript. Per ogni tipologia di servizio è

presente una documentazione con una guida basilare sul suo funzionamento nella piattaforma developers.google. Per il progetto sono state utilizzate le librerie per Node.js e i pacchetti nugget per .NET.

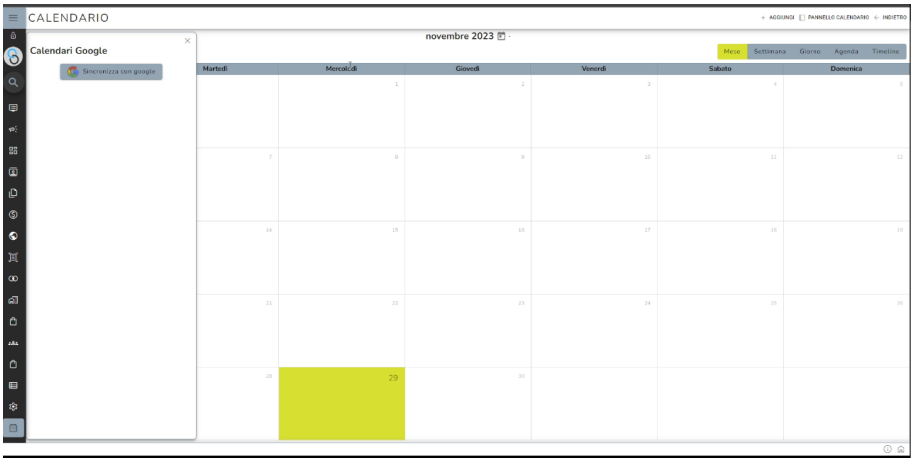
Capitolo 11

Use Cases

11.1 Google Calendar

11.1.1 Login

Partendo dalla pagina del calendario, l'utente deve aprire il pannello del calendario usando l'apposito bottone.



Dopo l'apertura del pannello, l'utente deve premere il bottone "Sincronizza con google", che aprirà una schermata di login, dove verrà richiesto l'uso di un account Google e di autorizzare l'app all'uso dei servizi richiesti.

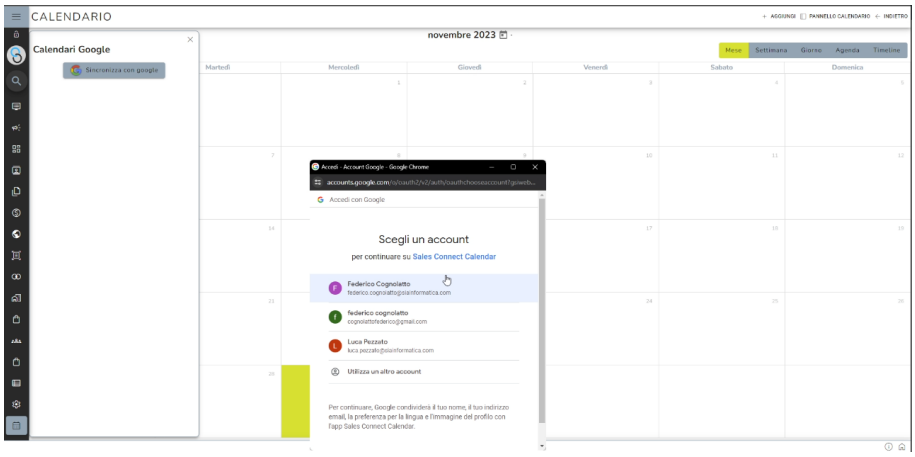


Figura 11.1: Login google

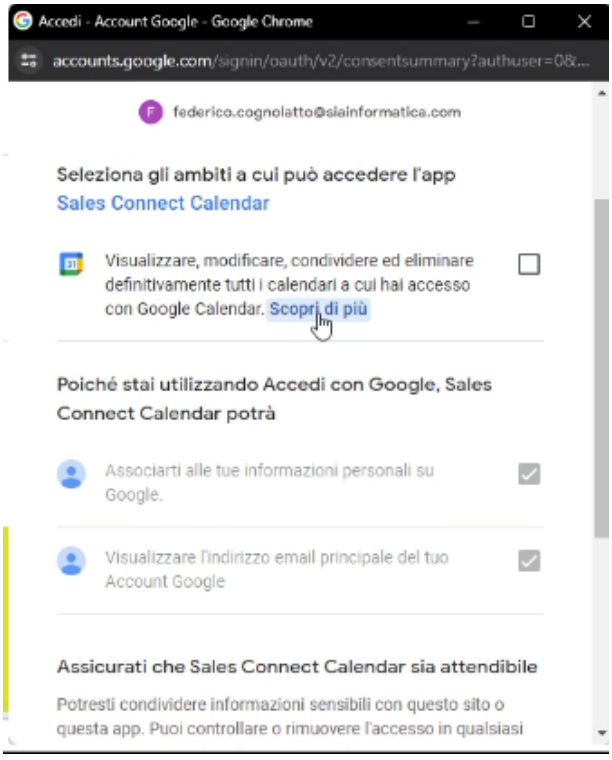
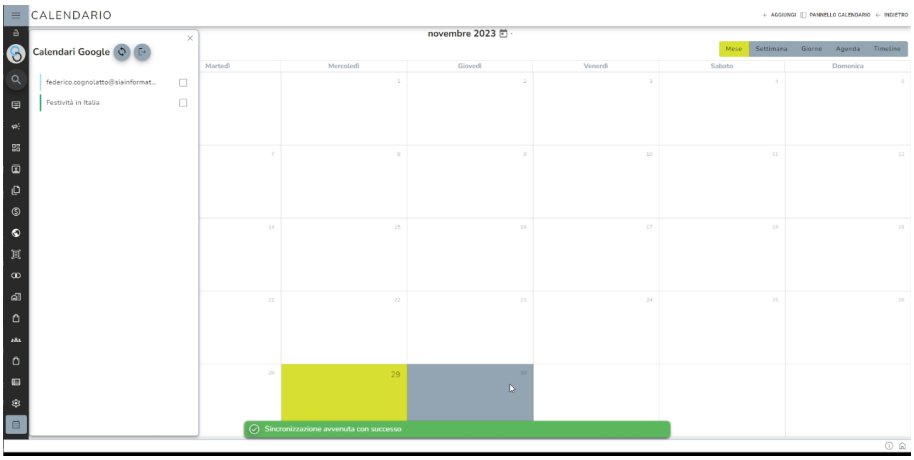


Figura 11.2: Autorizzazioni

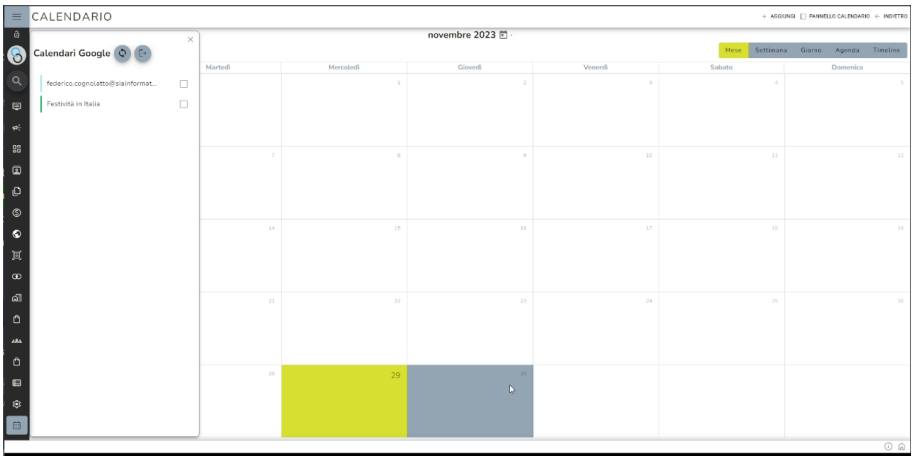
Dopo aver cliccato continua, l'utente dovrà attendere un caricamento, che una volta terminato mostrerà la lista dei suoi calendari Google, il tasto di logout e di sincronizzazione nel pannello.



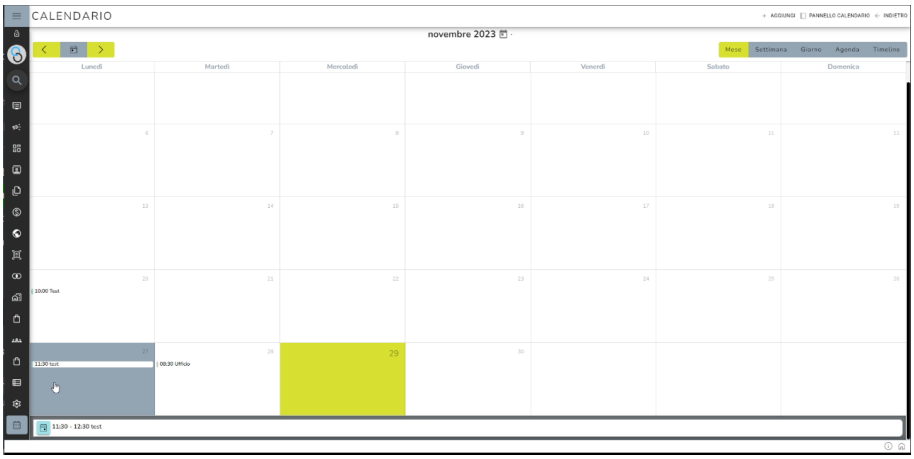
Nel caso il login non andasse a buon fine, l'utente sarà notificato con un messaggio a schermo.

11.1.2 Importare gli eventi di un calendario

Per importare gli eventi di uno o più calendari, l'utente deve aprire il pannello del calendario dalla pagina del calendario. Una volta aperto, se l'utente ha già effettuato il login, sarà visibile una lista di calendari da selezionare, un tasto di logout e un tasto per avviare la procedura d'importazione.



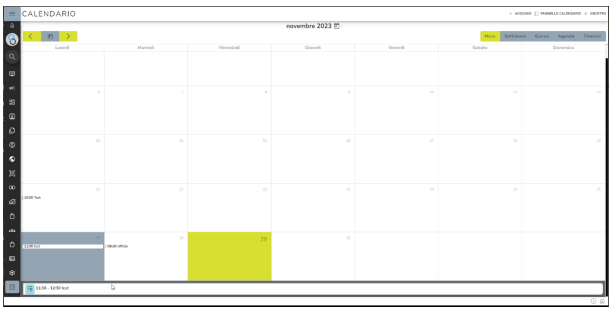
Prima di premere il bottone per la sincronizzazione, l'utente deve selezionare i calendari dei quali vuole importare gli eventi. Una volta terminata la sincronizzazione, apparirà una notifica a schermo e gli eventi saranno visibili nel calendario.



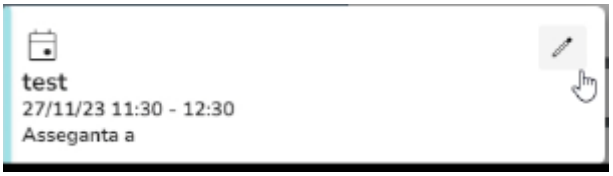
Ogni calendario è associato ad un colore, che è visibile a fianco al suo nome nella lista dei calendari, e gli eventi importati sono segnati con il colore del calendario di appartenenza. Nel caso un calendario venga selezionato, gli eventi legati ad esso verranno cancellati la prossima volta che l'utente avvierà la sincronizzazione.

11.1.3 Dettaglio evento

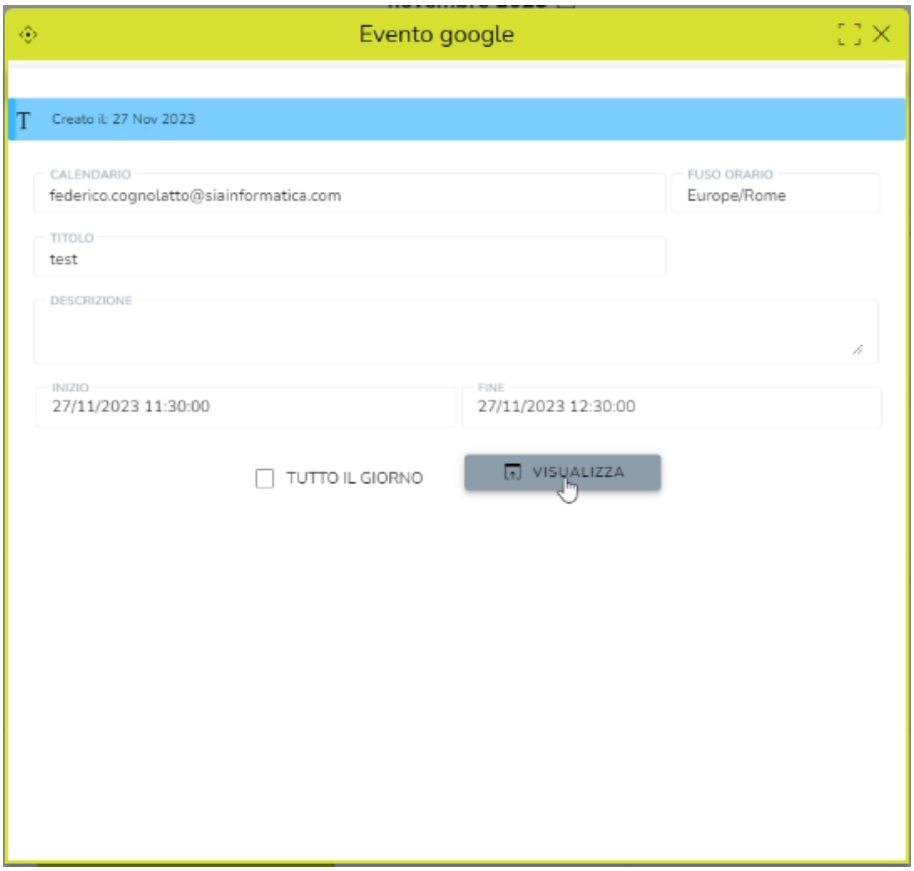
L'utente, dopo aver effettuato il login con google e aver sincronizzato almeno un calendario, deve selezionare, dalla pagina principale del calendario, la casella del giorno in cui è stato fissato l'evento che vuole visualizzare. Cliccando la casella, si aprirà in basso la lista degli eventi assegnati a quel giorno.



Dalla lista, l'utente deve selezionare l'evento che vuole visualizzare e cliccare il bottone che raffigura una penna nel popup che verrà aperto.

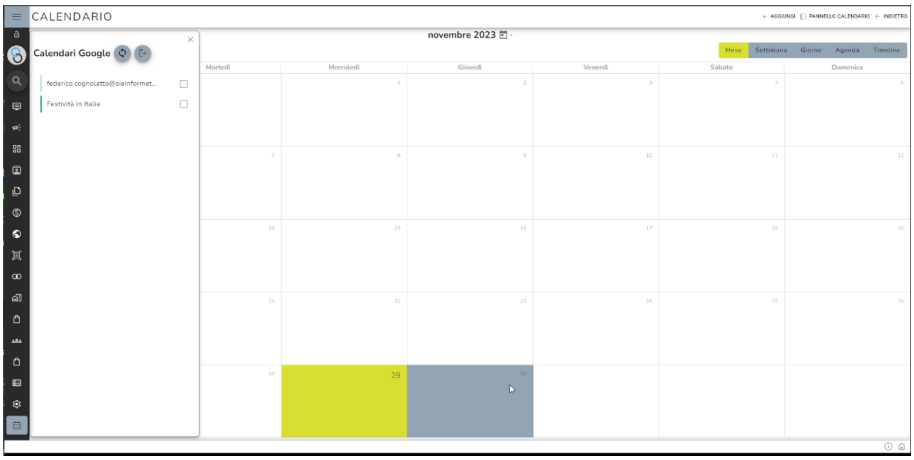


In seguito, si aprirà la finestra dedicata alla dettaglio dell'evento, dove è possibile visualizzare tutte le informazioni relative ad esso e visualizzarlo su Google Calendar.



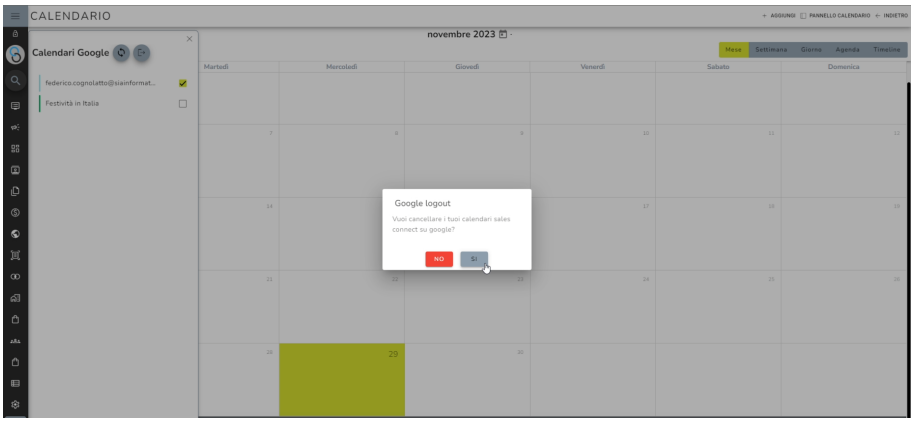
11.1.4 Logout

Per effettuare il logout, l'utente deve aprire il pannello del calendario usando l'apposito bottone.



Una volta aperto, vicino alla scritta "Calendari Google", saranno presenti il bottone di sincronizzazione e quello di logout. Una volta premuto il bottone, si aprirà un popup con un messaggio che chiede all'utente se vuole cancellare i calendari creati dall'app su Google Calendar. Nel ca-

so in cui l'utente non clicchi fuori dal popup prima di selezionare la risposta, l'operazione di logout viene annullata.

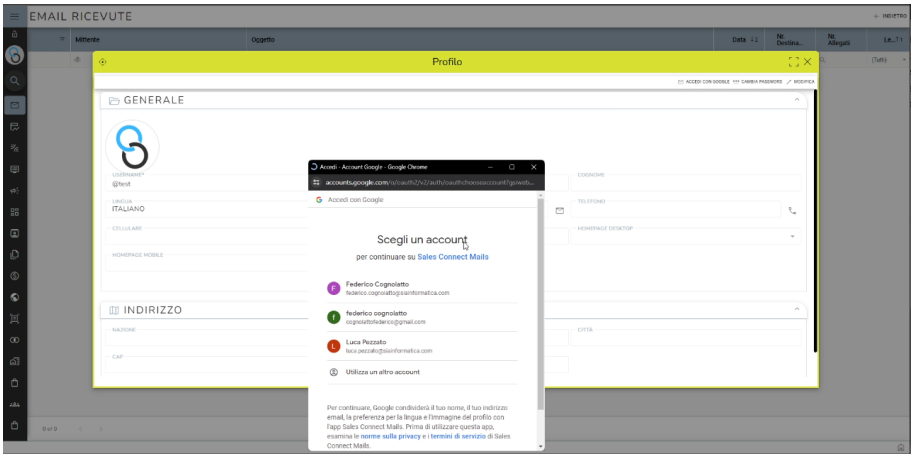


Dopo aver scelto se cancellare o meno i calendari su Google Calendar, inizierà il processo di logout che una volta terminato, riporterà il pannello allo stato precedente al login, inoltre tutti gli eventi legati ai calendari importati da Google Calendar vengono cancellati dal calendario dell'app.

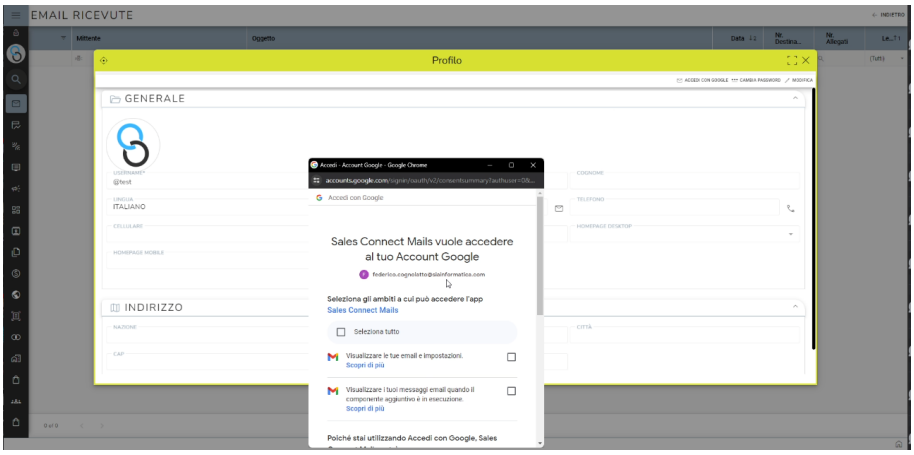
11.2 Gmail

11.2.1 Login

Partendo da un punto qualsiasi dell'app, l'utente deve cliccare sull'icona del suo profilo presente sulla sidebar a sinistra, e in seguito premere il bottone "accedi con google", presente nella finestra che è stata aperta in alto a destra.



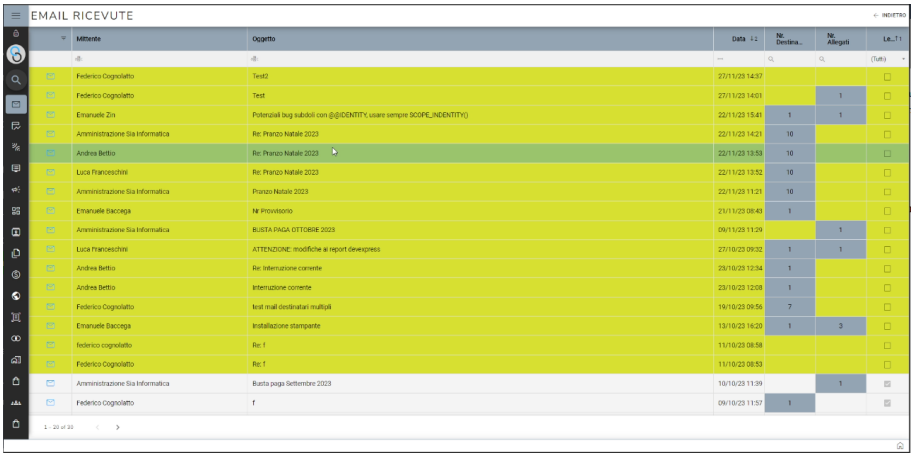
Dopo aver selezionato l'account Google da utilizzare, dalla finestra di login, l'utente deve dare i permessi all'app per utilizzare i servizi di Gmail dalla stessa finestra.



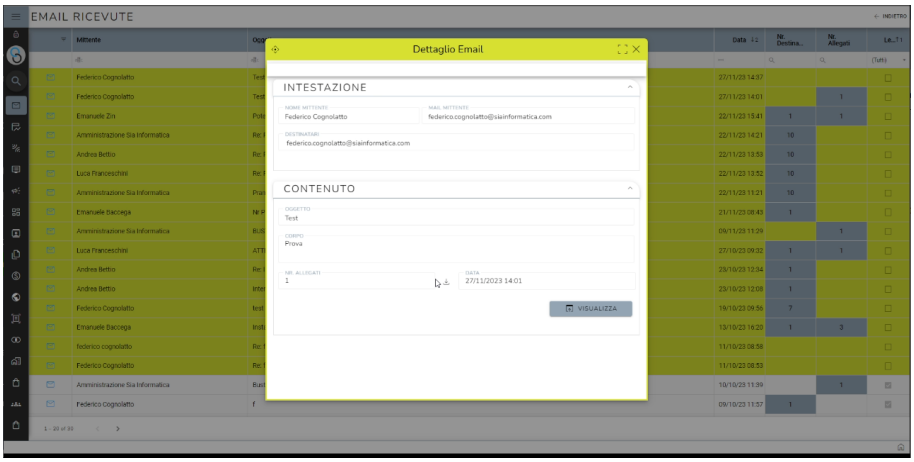
Una volta terminato il login, l'utente verrà notificato da un messaggio a schermo, e sull'icona delle mail nella sidebar a sinistra, apparirà un contatore che segnala il numero di mail non lette.

11.2.2 Visualizzazione mail

Per visualizzare una mail, l'utente deve accedere alla pagina delle mail, cliccando l'apposito bottone nella sidebar.



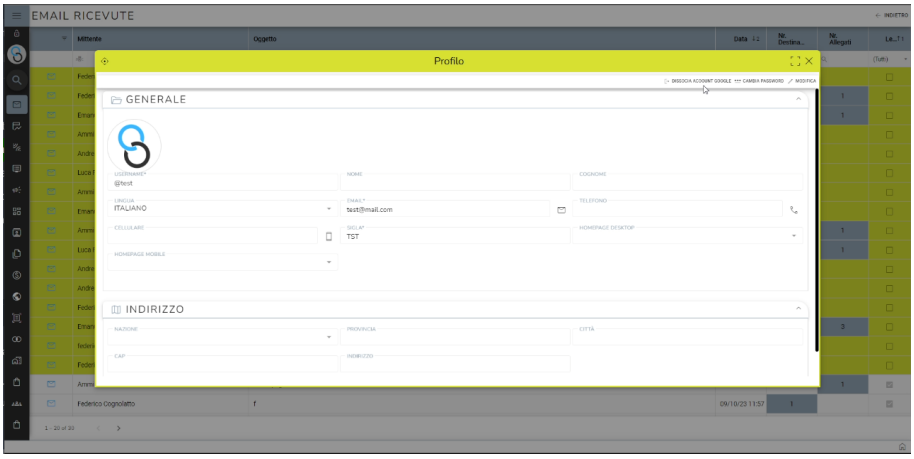
In seguito, l'utente deve cliccare sull'icona della mail che vuole visualizzare, aprendo la finestra con tutti i dettagli della mail.



Da questa finestra, l'utente può oltre a visualizzare i dettagli della mail, scaricare gli allegati , se presenti, con il bottone a fianco al numero degli allegati e aprire la mail nella casella di posta su Gmail con il bottone in basso a destra.

11.2.3 Logout

Per effettuare il logout, l'utente deve cliccare sull'icona del suo profilo presente sulla sidebar a sinistra, e in seguito premere il bottone "dissocia account google", presente nella finestra che è stata aperta in alto a destra.



Una volta premuto il bottone, inizierà la procedura di logout, e una volta terminata l'utente verrà notificato con un messaggio a schermo.

Capitolo 12

Conclusioni

Grazie a questa esperienza di tirocinio, ho avuto modo di applicare in parte le conoscenze che ho acquisito durante il mio percorso di studi, e di apprendere l'uso di nuovi linguaggi e strumenti. Infatti ho avuto modo di imparare ad utilizzare i framework Angular e .NET, e a come applicare alcuni dei metodi di progettazione appresi durante il percorso di studi. Per quanto riguarda il risultato del progetto, sono state implementate con successo tutte le funzioni richieste, e in seguito rilasciate nella versione corrente di Sales Connect. Il rilascio non è stato immediato, dopo la fine della fase di test, a causa delle politiche di Google sul trattamento dei dati dell'utente, dato che la funzione di casella di posta richiede l'accesso alle mail della casella di Gmail dell'utente. Infatti è stato necessario adattare l'applicazione in modo che soddisfacesse i requisiti richiesti da Google per l'uso delle API richieste.

Bibliografia

[1] T. Santamato, «Nel mondo più del 60% delle persone è sui social media,» *Ansa*, 2023.

[2] Angular, «Dependency injection in Angular,» *angular.io*, 2022.

[3] Angular, «Understanding pipes,» *angular.io*, 2023.

[4] Angular, «Understanding binding,» *angular.io*, 2023.

[5] A. T. Bill Wagner Genevieve Warren, «Espressioni lambda e funzioni anonime,» *microsoft.com*, 2024.

[6] G. W. Bill Wagner Zev Spitz, «LINQ (Language-Integrated Query),» *microsoft.com*, 2024.

[7] K. S. Bill Wagner, «Sistema di tipi C#,» *microsoft.com*, 2024.

[8] J. M. Tom Dykstra Luke Latham, «Attività in background con servizi ospitati in ASP.NET Core,» *microsoft.com*, 2024.

[9] R. A. Tom Dykstra Zlatin Stanimirov, «Dependency injection in ASP.NET Core,» *microsoft.com*, 2023.

[10] A. B. Tom Dykstra Rick Anderson, «Creare API Web con ASP.NET Core,» *microsoft.com*, 2024.

[11] M. J. Jon Douglas Alex Buck, «Introduzione a NuGet,» *microsoft.com*, 2023.

[12] C. M. Mike Ray William Assaf, «Informazioni di riferimento su Transact-SQL (motore di database),» *microsoft.com*, 2023.

[13] pypl, «Top IDE index,» *pypl.github.io*, 2024.

[14] Oracle, «Che cos'è il CRM? La guida completa al CRM,» *oracle.com*, 2024.

Ringraziamenti

Vorrei ringraziare il professore Antonio Rodà per avermi seguito durante l'esperienza di tirocinio, e di aver contribuito alla stesura di questo elaborato in quanto relatore. Ringrazio i miei tutor aziendali Luca Pezzato e Emanuele Baccega, che mi hanno aiutato nello svolgimento del progetto di tirocinio e per avermi fatto integrare nell'ambiente aziendale. Ringrazio l'azienda Sia Informatica s.r.l. per la disponibilità e l'accoglienza. Infine ringrazio la mia famiglia e i miei amici per il sostegno ricevuto durante questo percorso.