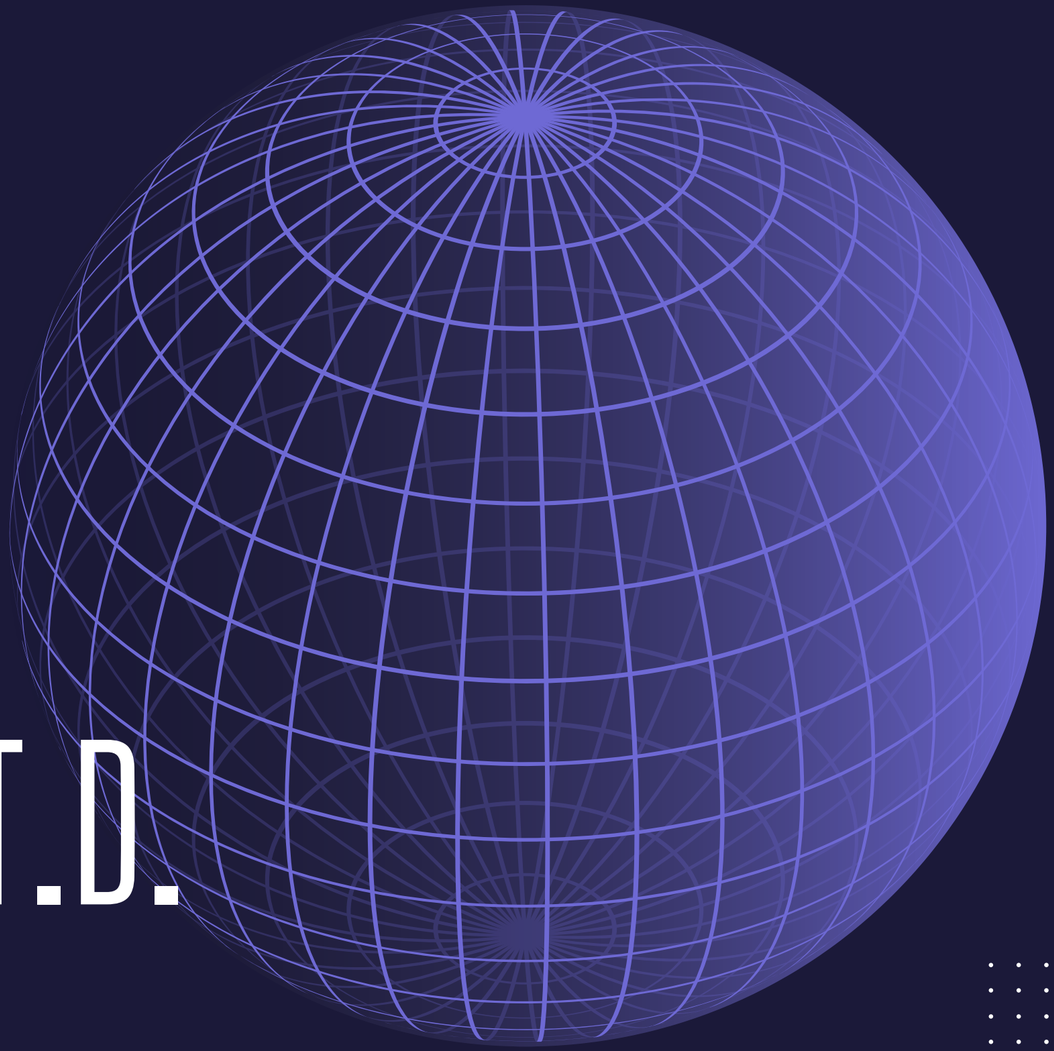
21/12/2022

# PRESENTAZIONE BRIEF PIPENETWORK L.T.D.



MUCCIACITO MARIA

ISTITUTO INFOBASIC

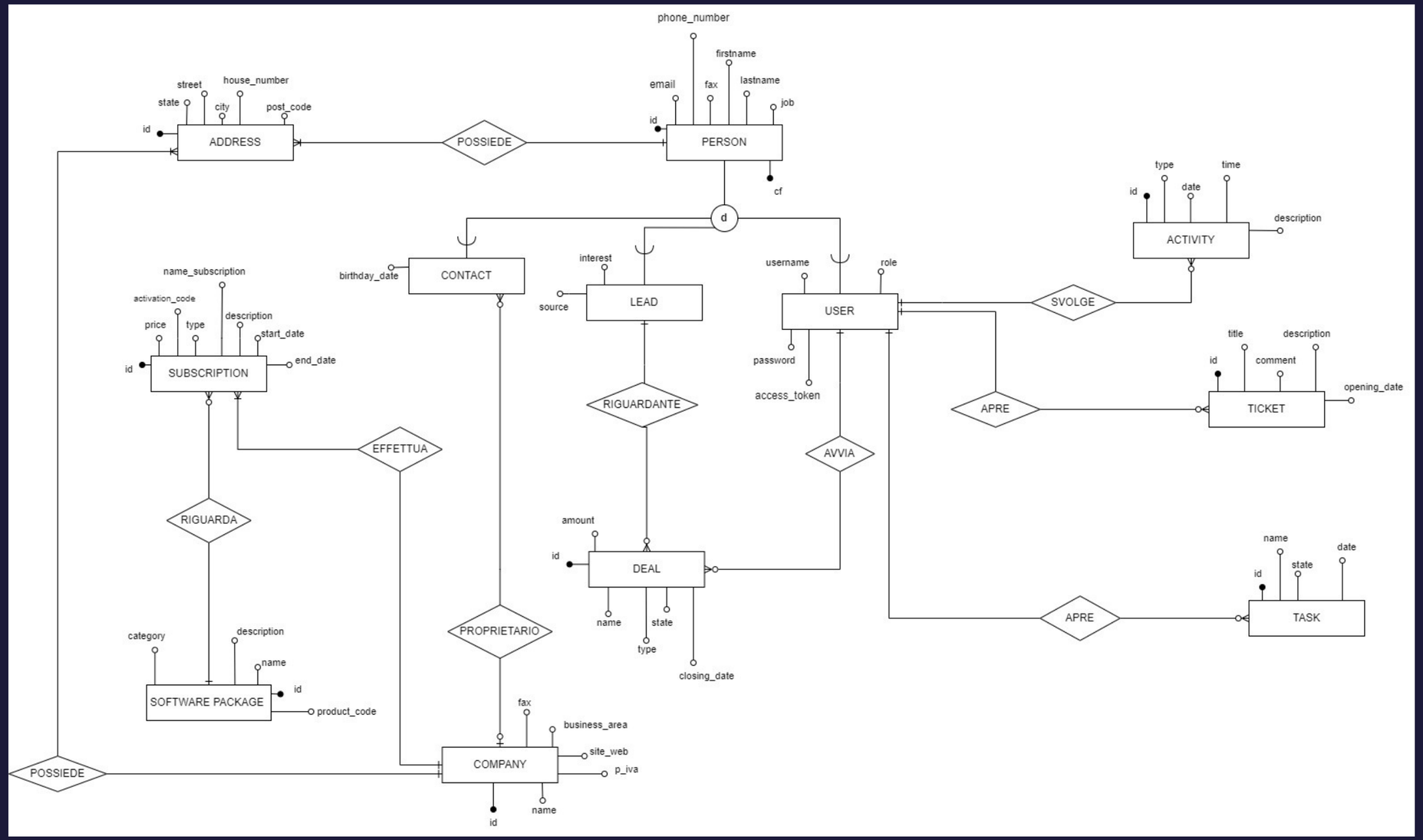
# INTRODUZIONE


L'obiettivo di questa presentazione è quello di documentare il progetto della PipeNetwork l.t.d. , azienda specializzata nella vendita di software ingegneristico in ambito meccanico. Il prodotto realizzato è un'applicazione CRM (Customer Relationship ), che permetta di gestire i clienti e i lead, migliorare la redditività e mostrare l'andamento delle attività.
Le funzionalità dell'applicazione sono quindi le seguenti:

- Gestione dei contatti
- Gestione dei lead
- Grafici illustrativi

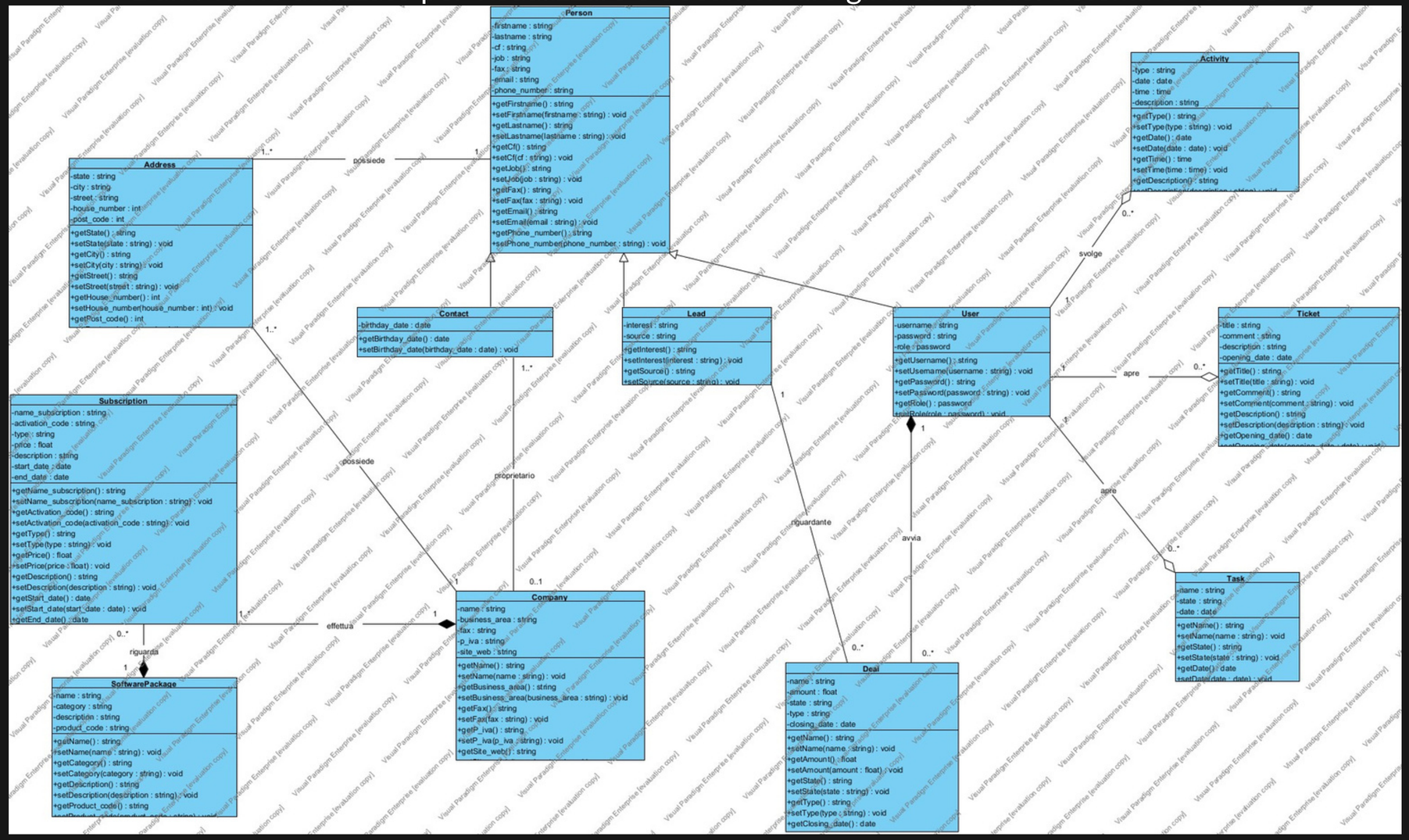
# DIAGRAMMA ER

Il modello ER consente di visualizzare il modo in cui le entità all'interno di un database si relazionano tra loro e con gli attributi di ciascuna entità.



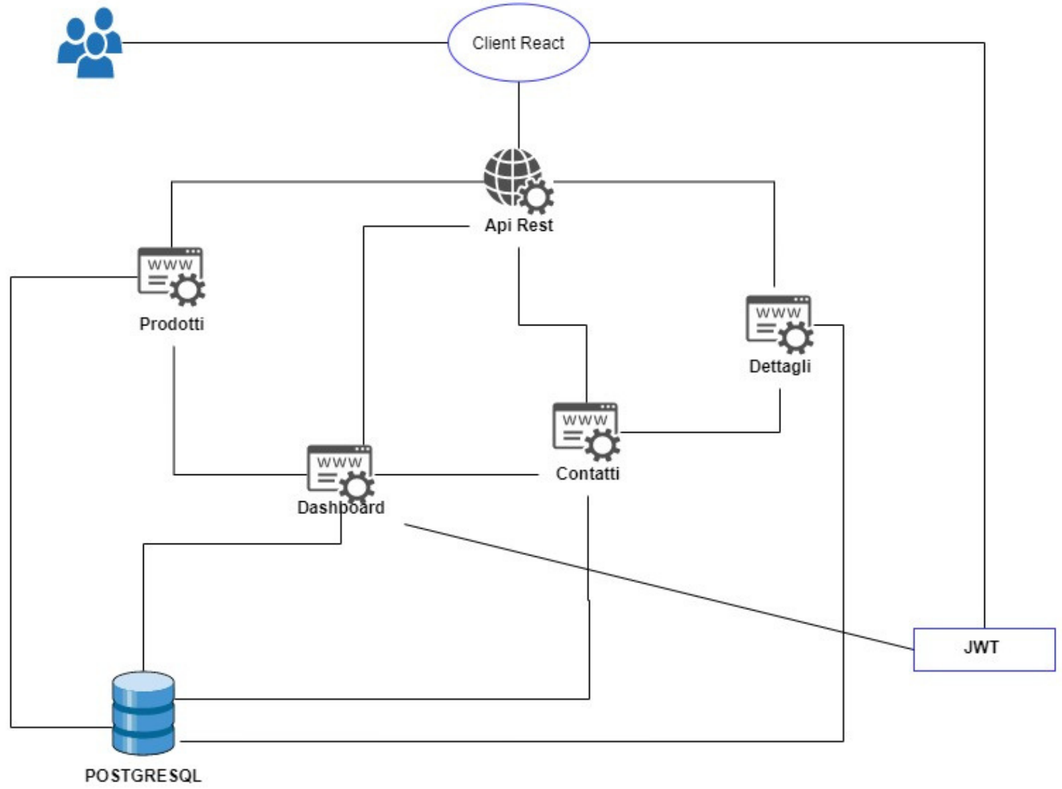
# UMl Classi

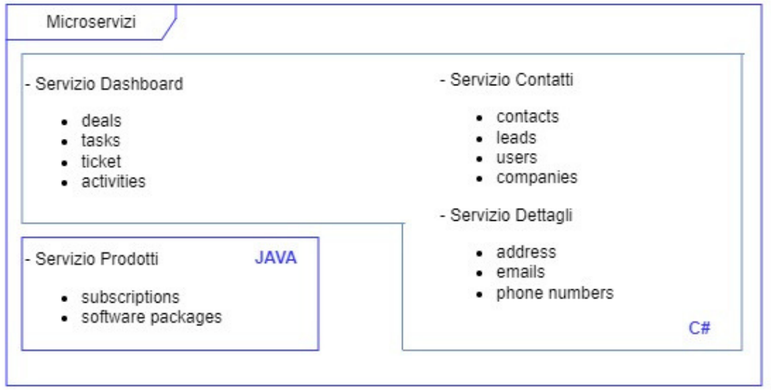
L'uml viene utilizzato per visualizzare e documentare gli artefatti del sistema software

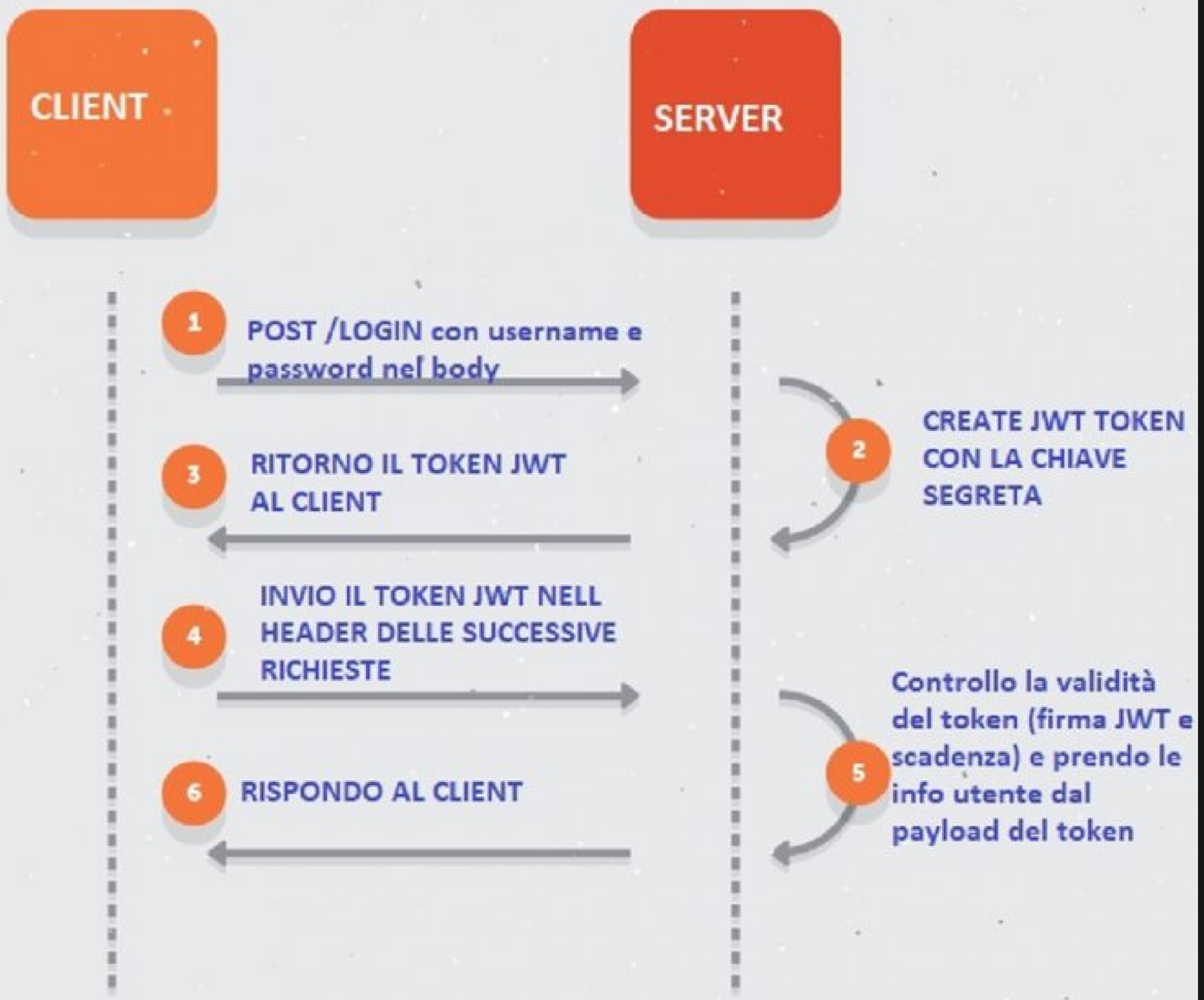


# STRUTTURA APPLICAZIONE

La struttura dell'applicazione, come possiamo vedere dal grafico, si presenta nel seguente modo:
I visitatori del sito web si interfacciano con esso grazie a React Js, effettuando così l'autenticazione che permette di visualizzare e gestire il crm (tramite jwt). Essi potranno quindi visualizzare e interagire con la daashboard, i contatti, i lead e le loro attività. Tutto ciò è possibile grazie alle api rest, realizzate nel backend in java e c#, che permettono il collegamento con i dati salvati in un database Postgresql



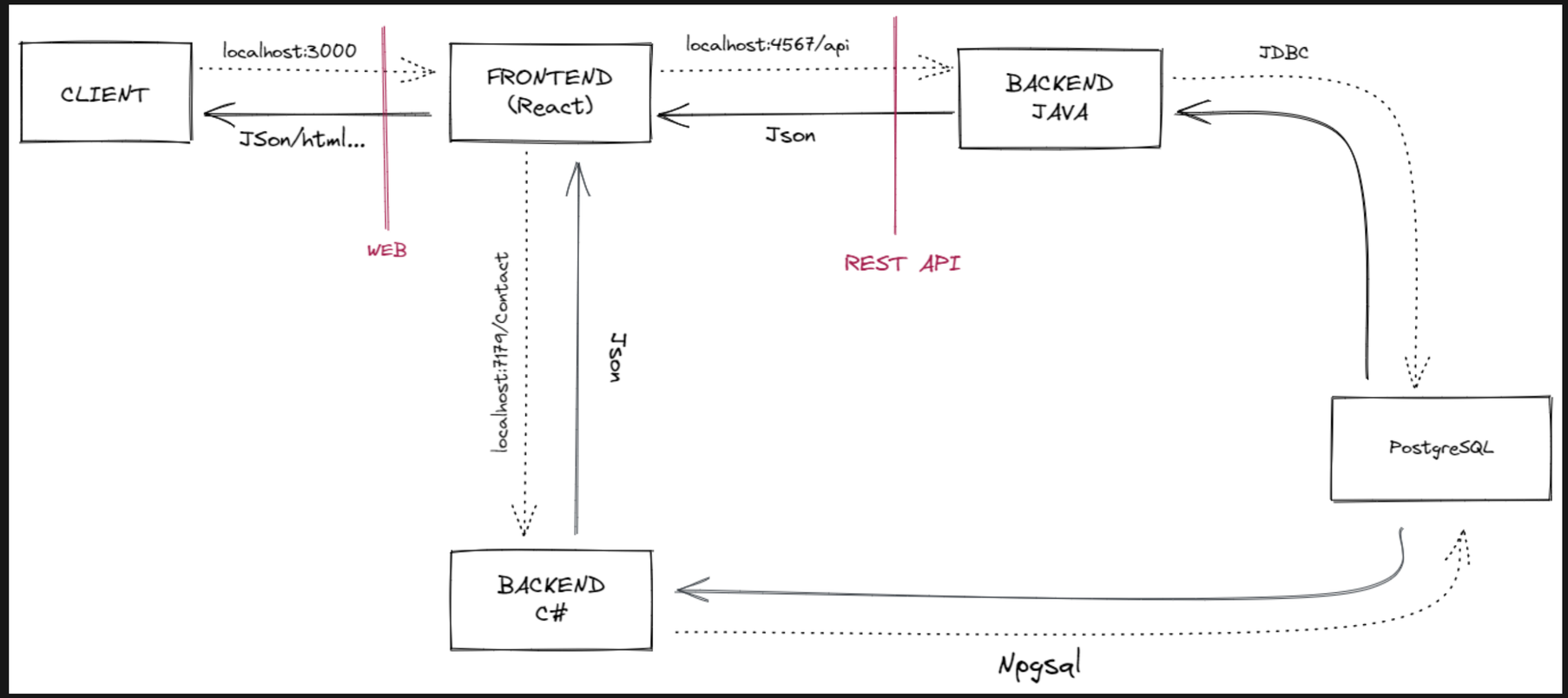



# JWT

JWT, acronimo di JSON Web Token, è un sistema di cifratura e di contatto in formato JSON per lo scambio di informazioni tra i vari servizi di un server. Si genera così un token che può essere cifrato e firmato tramite una chiave disponibile solo a colui che lo ha effettivamente generato.
Infatti l'autenticazione dell'utente nell'applicazione avviene tramite questo sistema, ovvero: l'utente fa una richiesta ad un api realizzata in c#, se possiede le credenziali esatte, gli viene inviato un token, che viene salvato nel localstorage, grazie a react js, e che gli permette di accedere alle diverse pagine protette .
Successivamente , si può effetuare il logout, che elimina il token salvato e riporta alla pagina di login, disconnettendo quindi l'utente dall'applicazione

# SCHEMA API REST

Il REST è un sistema di trasmissione dei dati che utilizza l'HTTP, ricorrendo alle funzioni GET, POST, PUT, PATCH, DELETE .
Con il termine API (Application Programming Interface) si intendono un insieme di procedure volte a portare a termine un ben preciso e determinato compito.
All'interno dell'applicazione esse vengono utilizzate per effettuare le funzioni CRUD (Create Read Update Delete) sui modelli del database, per la realizzazione dei grafici e per la comunicazione tra backend e frontend.
Sono state realizzate in java con Apache Spark e in c# con Entity Framework.



# Pattern MVC

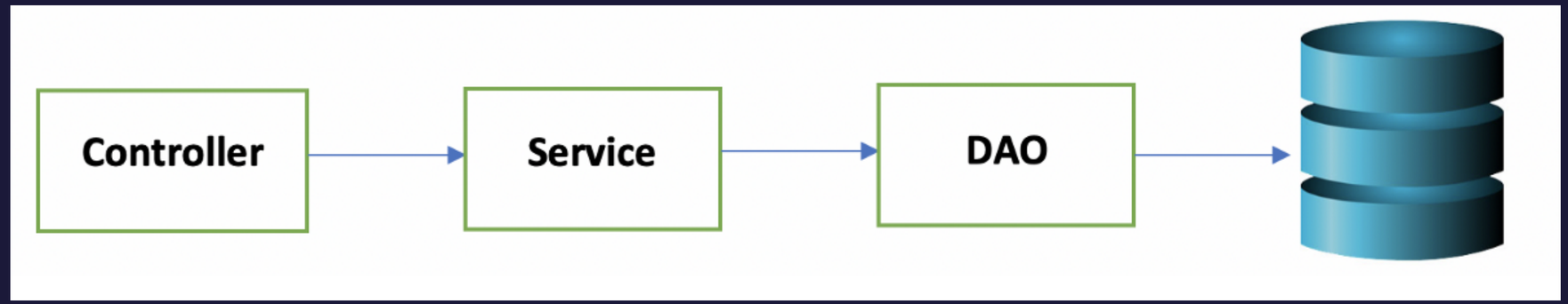
Model-View-Controller (MVC) è un pattern utilizzato in programmazione per dividere il codice in blocchi dalle funzionalità ben distinte.
Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- Model: si occupa della gestione dei dati;
- View: visualizza i dati contenuti nel Model e si occupa dell'interazione con gli utenti
- Controller: riceve i comandi dell'utente (in genere attraverso il View) e li attua modificando lo stato degli altri componenti.

Stiamo parlando di questo pattern, in quanto è stato utilizzato per la costruzione delle api in java, insieme al pattern DAO.
Il pattern DAO è usato per separare la logica di business dalla logica di acceso ai dati. I concetti principali sono

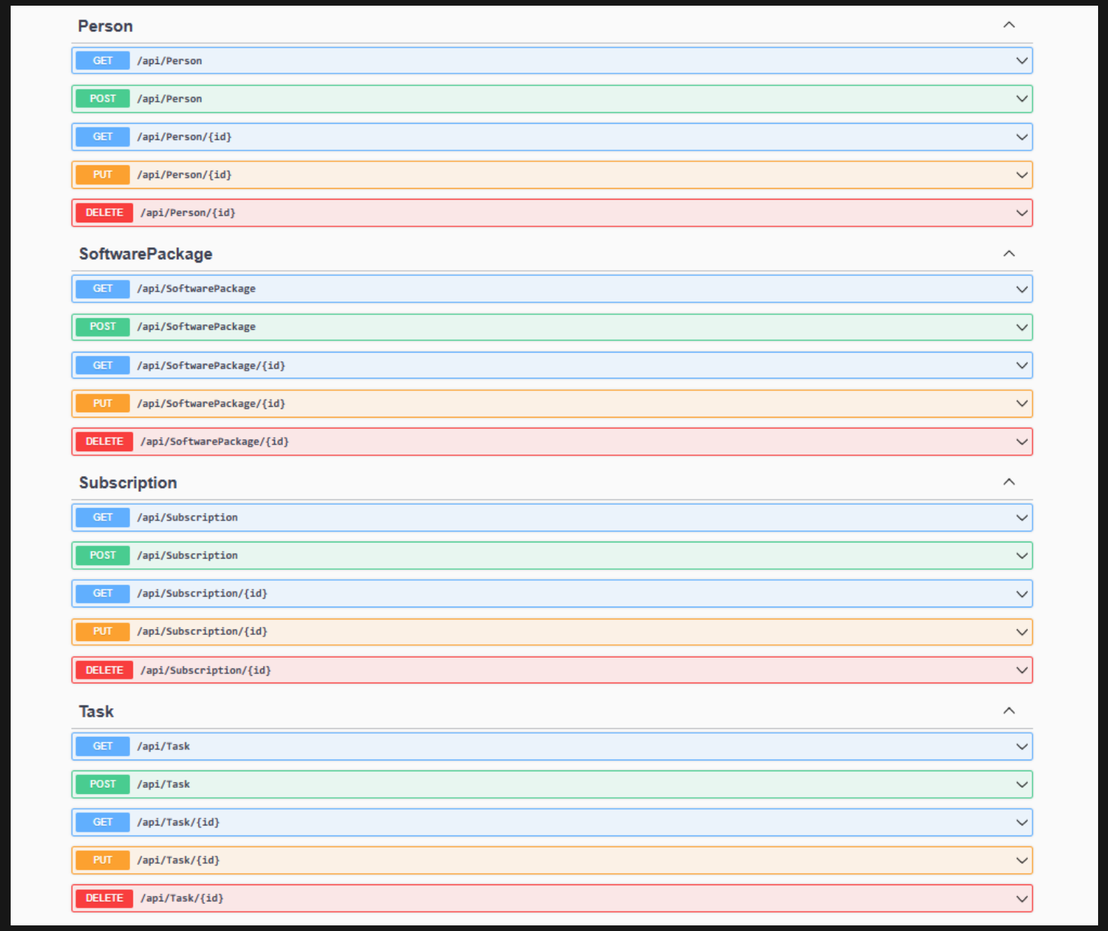
- una classe (model) per ogni tabella
- una interfaccia (detta Dao) per ogni tabella contenente tutti i metodi Crud relativi a quella tabella
- una implementazione per ogni interfaccia Dao

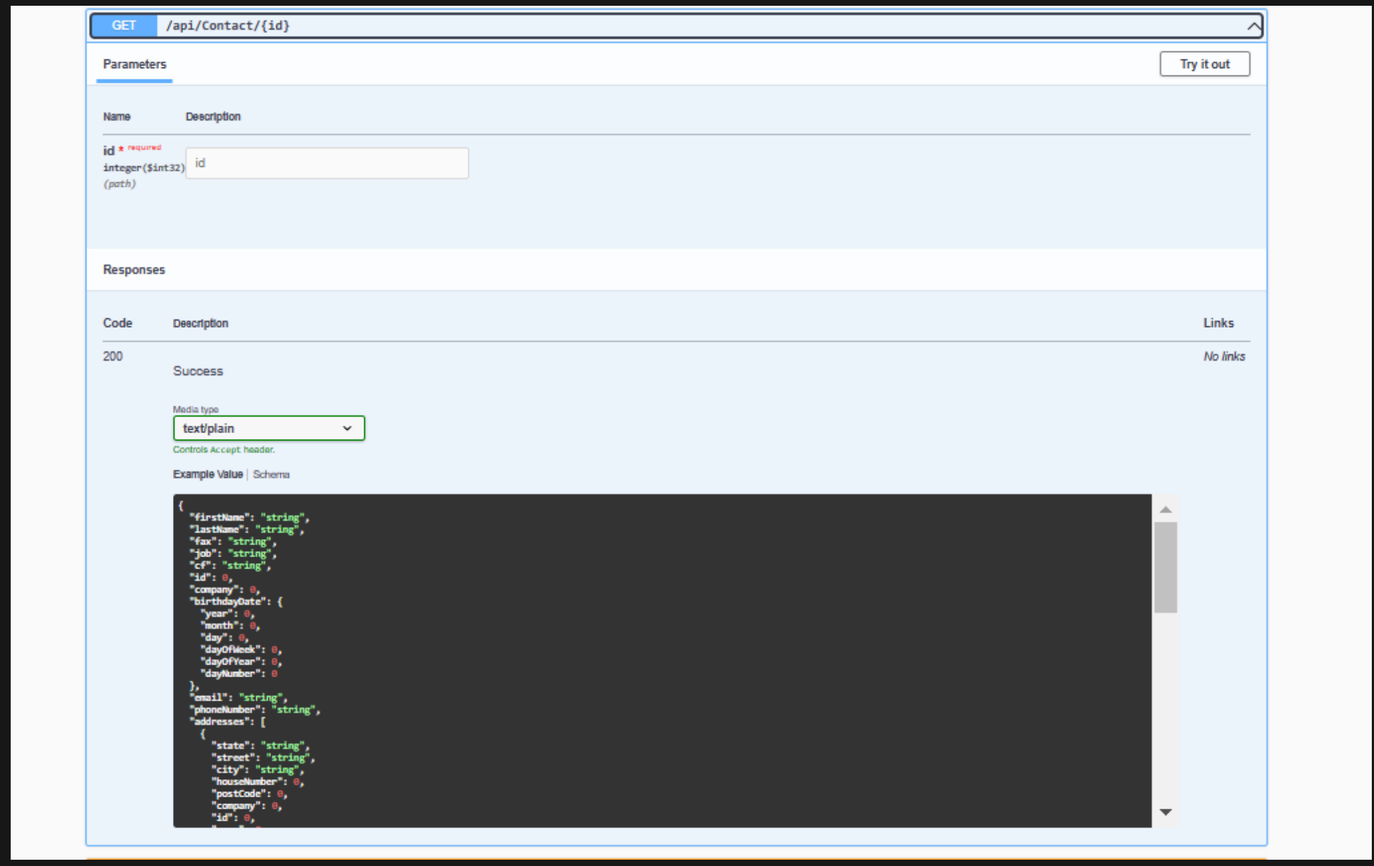
L'obiettivo principale del livello DAO è gestire i dettagli del meccanismo di persistenza, mentre il livello di servizio si trova al di sopra di esso per gestire i requisiti aziendali. Il controller invece gestirà la navigazione tra le diverse viste (le api).



# EndPoint API

Swagger viene utilizzato per documentare le API REST, poiché è in grado di mappare quasi tutti i servizi web e le informazioni relative all'interfaccia



# EndPoint API

http://localhost:4567/dealbystate

GET http://localhost:4567/dealbystate

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not hav

ody Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "clousure lost": 1,
    "clousure win": 1,
    "sales ready": 1
}
```


http://localhost:4567/leadbysource

GET http://localhost:4567/leadbysource

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results

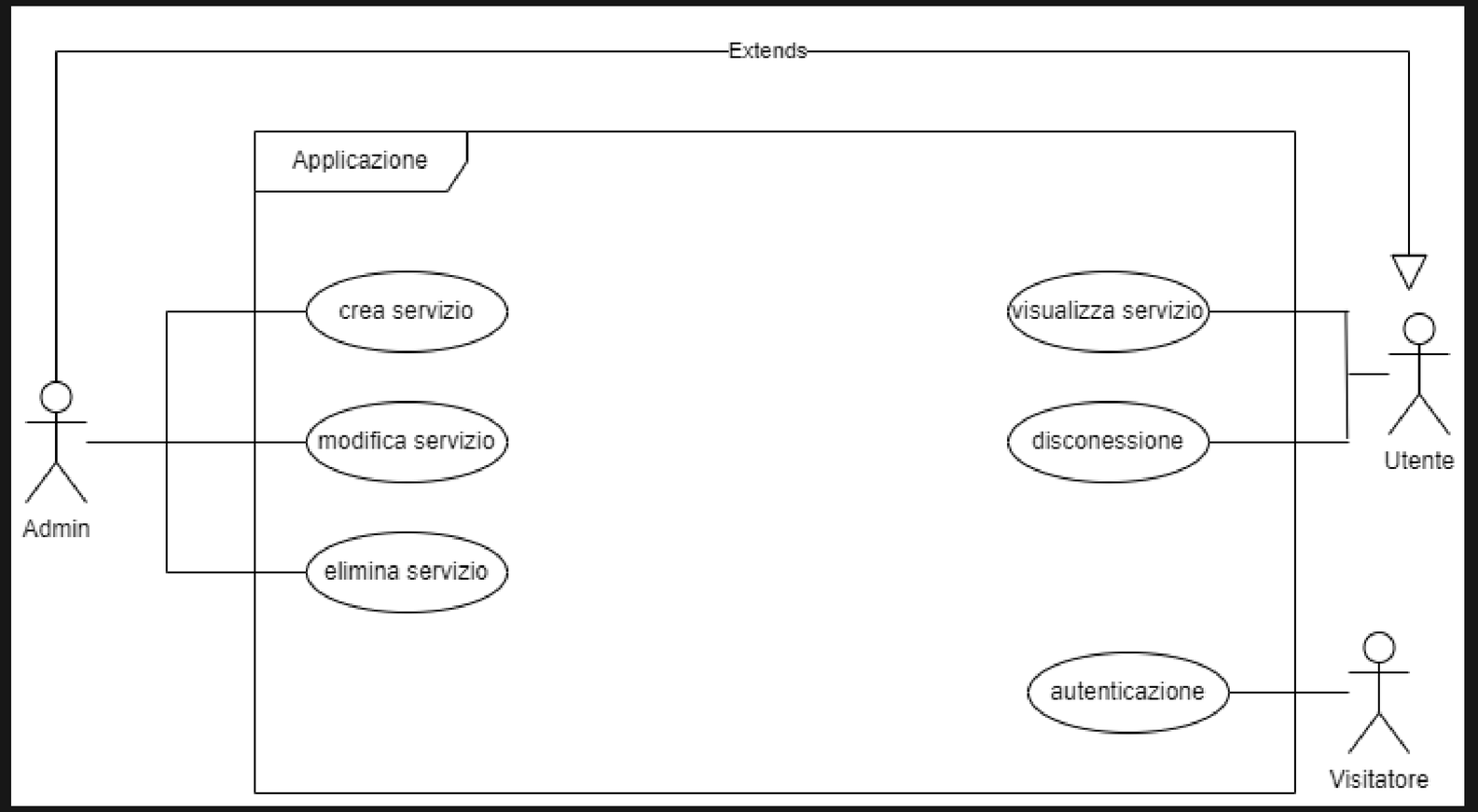
Pretty Raw Preview Visualize JSON


```
{
    "call": 2,
    "email": 1
}
```

# CASO D'USO

Si tratta di un diagramma o di una descrizione che esamina un sistema o una sua parte. Individua chi ha a che fare con il sistema (attori) e che cosa gli attori possono fare (casi d’uso). Spiega quindi il funzionamento desiderato dal sistema.
Nello schema seguente abbiamo tre tipi di attori:

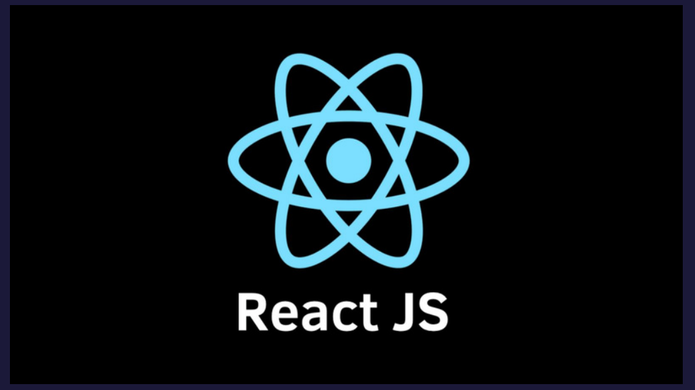
- l’admin che ha i permessi per modificare, eliminare e creare microservizi, oltre ad avere quelli dell’utente
- l’utente che ha i permessi per visualizzare i servizi
- il visitatore che effettua l’autenticazione



# TECNOLOGIE UTILIZZATE

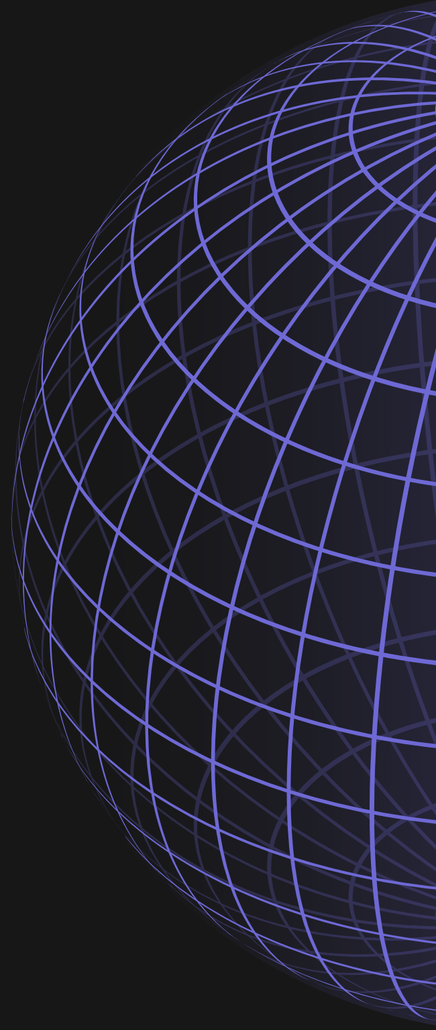
- Java
- C#
- Entity Framework
- Postman
- JDBC
- Apache Spark
- React js
- Axios
- JWT
- Swagger
- Postgresql
- ElephantDB



# FUNZIONI DEL SITO?

Di seguito le funzionalità che l'applicazione offre



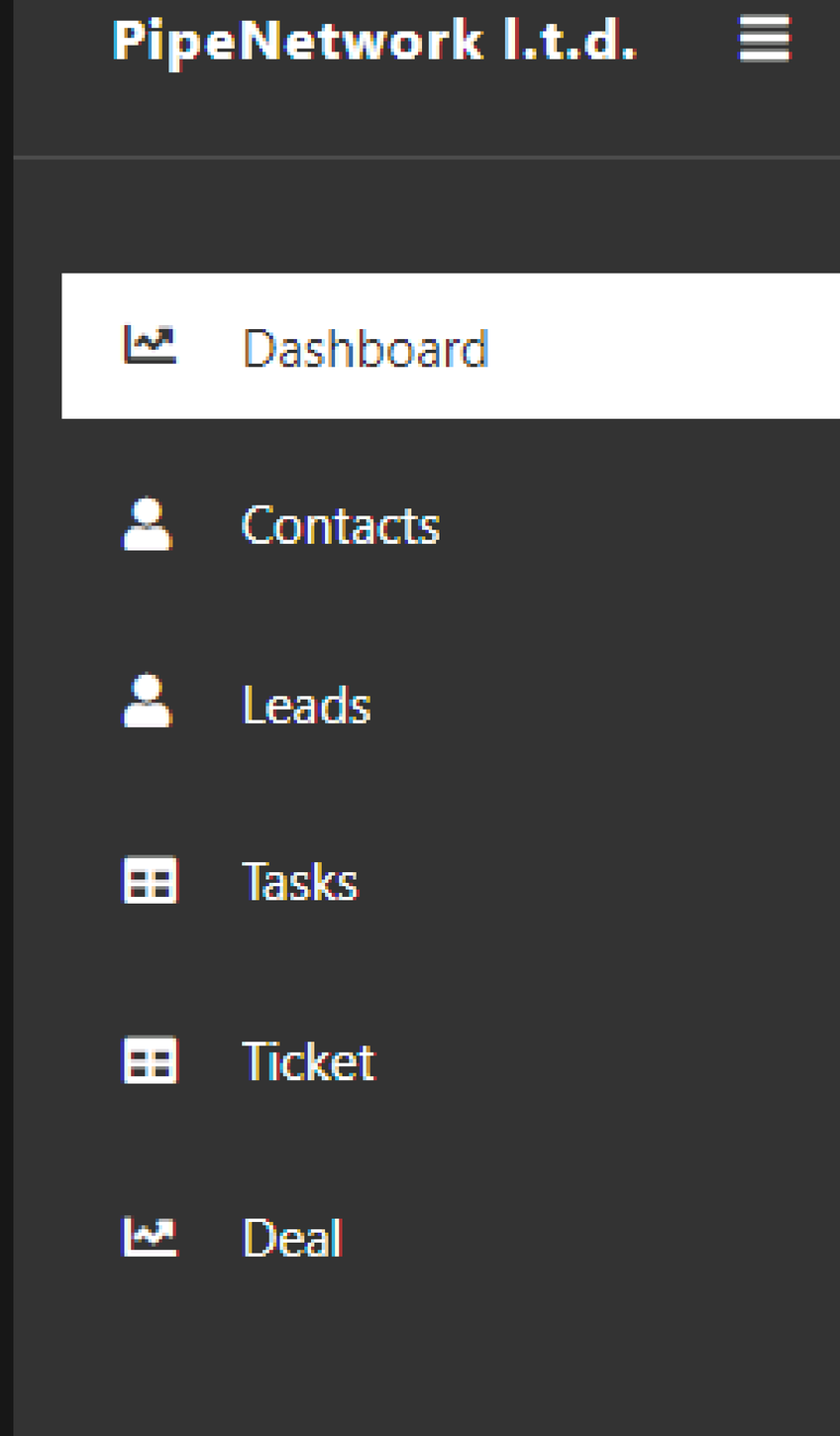

# Pagina di Login

La pagina del login permette ai visitatori dell'applicativo di inserire le credenzialli, che verranno poi verificate dal backend tramite api e se corrette , l'utente verrà rimandato alla pagina di dashboard(grazie al token), dove potrà spostarsi a suo piacimento ed effettuare vare operazioni , in base alle sue esigenze.

# Sidebar

La sidebar permette di spostarsi tra i vari servizi del crm. Nel nostro caso abbiamo:

- Dashboard
- Contatti
- Lead
- Task
- Ticket
- Deal

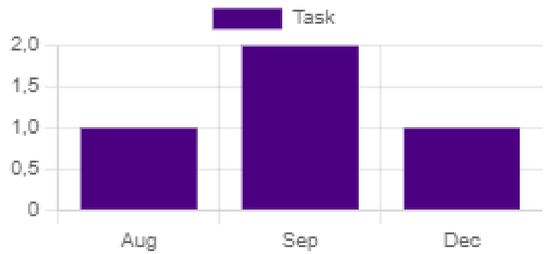


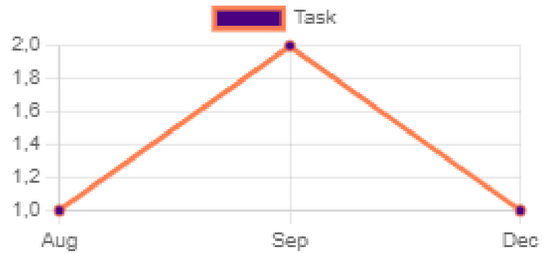
# Dashboard

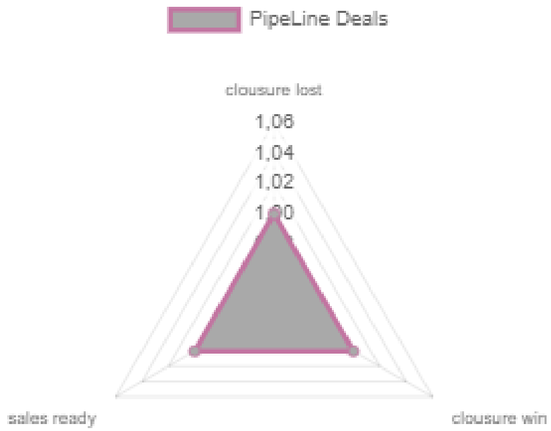
WAITING TASK
1 task

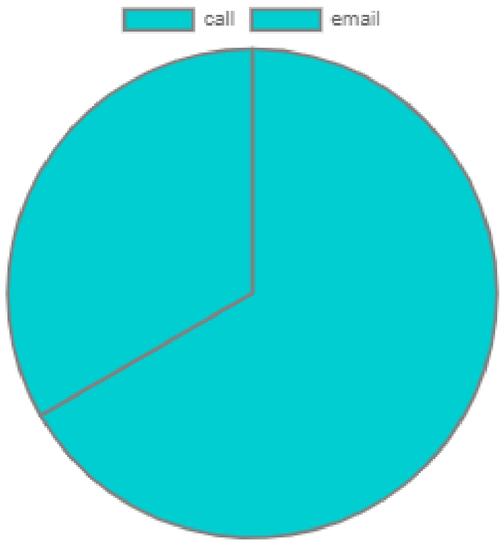
IN PROGRESS TASK
0 task

COMPLETED TASK
3 task



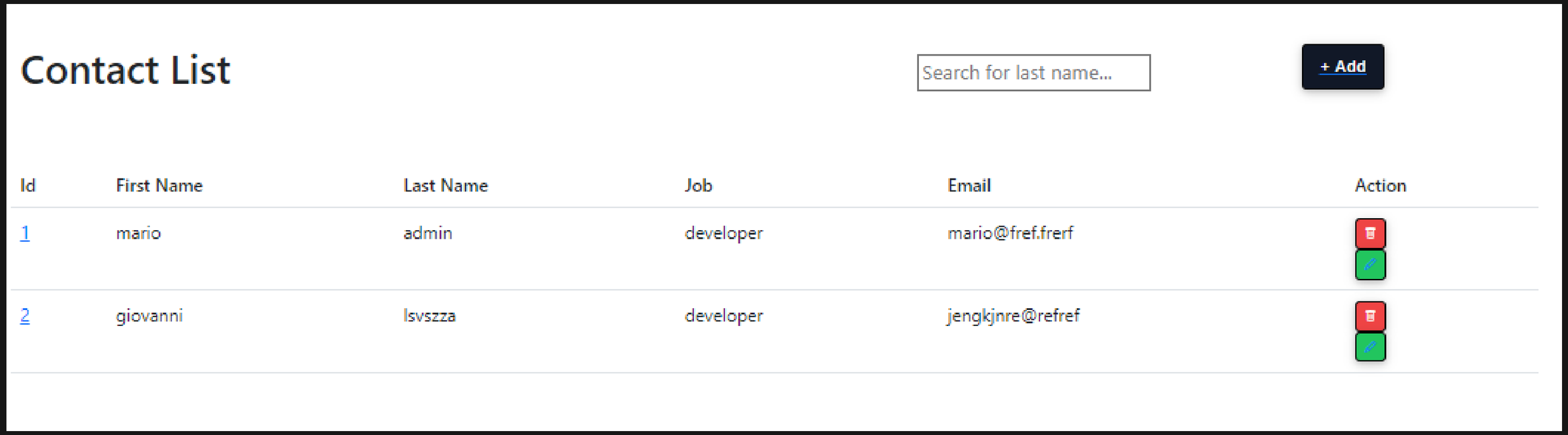



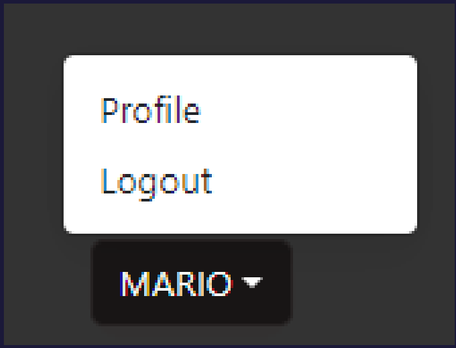

# DASHBOARD

La dashboard permette di visualizzare i grafici fatti in base a dati reali e effetuare poi eventuali statistiche o strategie di business.

# Contact e Lead

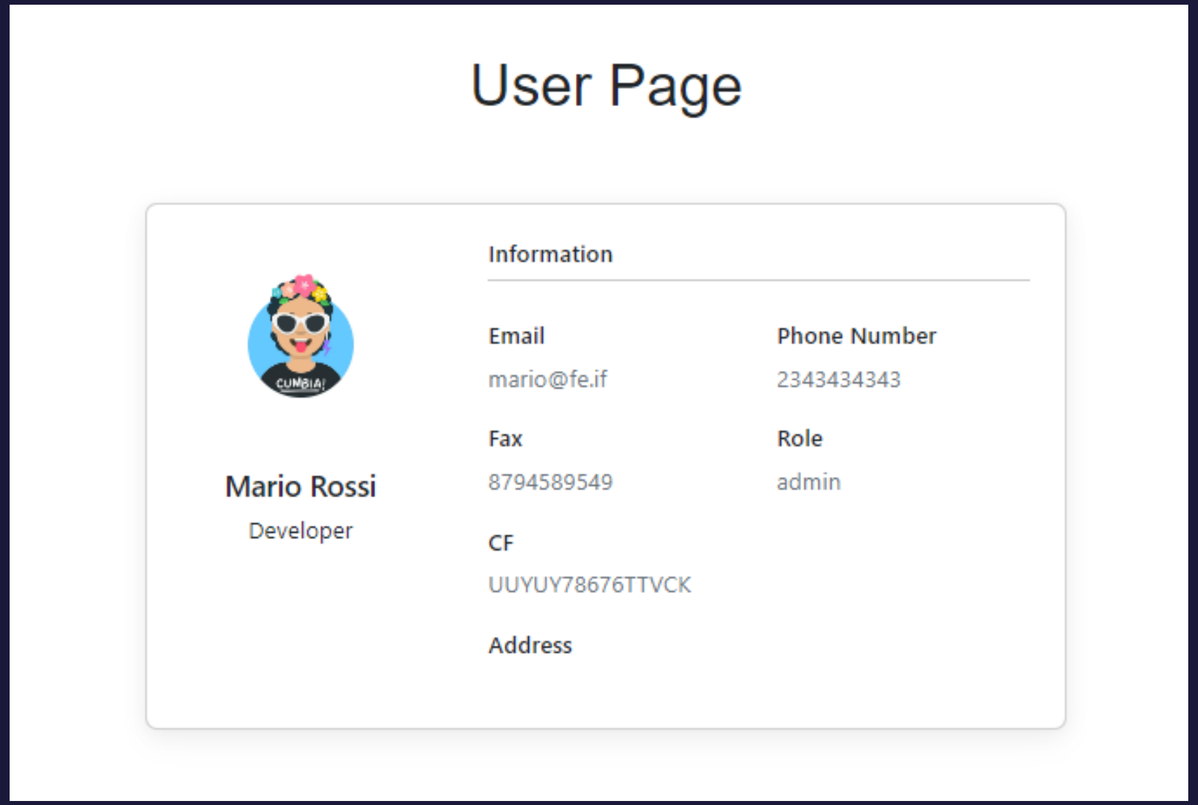
I servizi di contacts e leads permettono di visualizzare la lista dei rispettivi, di aggiungere un nuovo oggetto, di modificare quelli già presenti o eventualmente eliminarli






# Utente

In basso, nella sidebar, abbiamo l'username dell'utente che si è loggato, con le opzioni di logout (che rimanda alla pagina di login) e profile, che manda ai dettagli dell'utente



# CODICE REACT

```
const postDelete = (id, e) => {
  e.stopPropagation();
  axios
    .delete(`/Contact/${id}`)
    .then((result) => {
      setShow(false);
    })
    .catch((err) => console.log(err));
};
```

```
export const RequireAuth = () => {
  const auth = JSON.parse(localStorage.getItem("user"));

  return auth?.accessToken ? <Outlet /> : <Navigate to="/login" />;
};
```

```
const createContact = async (e) => {
  e.preventDefault();
  let json = JSON.stringify({
    firstName: firstName,
    lastName: lastName,
    fax: fax,
    job: job,
    birthdayDate: birthdayDate.toString(),
    cf: cf,
    company: company,
    email: email,
    phoneNumber: phoneNumber,
  });
  try {
    await axios
      .post(CREATE_CONTACT, json, {
        headers: { "Content-Type": "application/json" },
      })
      .then((result) => {
        if (result.status === 201) {
          setMsg("Il nuovo contatto è stato inserito con successo!");
        }
        console.log(msg);
      })
      .catch((error) => console.log(error));

    setLastName("");
    setFirstName("");
    setBirthdayDate("");
    setJob("");
    setFax("");
    setCf("");
    setCompany("");
    setEmail("");
    setPhoneNumber("");
  } catch (error) {
```

# GRAZIE PER AVER PARTECIPATO.

ISTITUTO INFOBASIC