Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

CUSTOMER RELATIONSHIP MANAGEMENT

0253591

Michele Tosi

Indice

1. Descrizione del Minimondo

Un sistema di Customer Relationship Management (o gestione delle relazioni con i clienti) è un sistema informativo che verte sulla fidelizzazione del cliente. Si vuole realizzare un sistema CRM per un'azienda marketing-oriented che intende realizzare relazioni durevoli di breve e lungo periodo con i propri clienti, massimizzando quindi il valore degli stessi.
La base di dati del sistema informativo dell'azienda di CRM deve poter memorizzare le informazioni su tutti i clienti di interesse dell'azienda, caratterizzati da nome, cognome, codice fiscale, data di nascita ed un insieme di contatti, sia in forma di indirizzi che di recapiti telefonici, email, fax. Alcuni dei clienti sono società che ricevono servizi dalla società di CRM. Di questi interessa anche mantenere il numero di partita IVA. Di tutti i clienti interessa sapere qual è la data di registrazione nel sistema di CRM.
L'azienda di CRM in questione è di dimensione elevata ed ha a disposizione vari funzionari che interagiscono con i clienti. A ciascun utente aziendale del sistema viene assegnato un sottoinsieme di clienti da gestire.
Su base periodica, gli operatori dell'azienda di CRM contattano i clienti mediante uno dei recapiti forniti. In questa fase operativa, l'utente deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali. Una risposta positiva di accettazione di una proposta commerciale può essere associata ad un appuntamento in sede. L'azienda ha più sedi, ciascuna caratterizzata da un indirizzo. In ciascuna sede sono presenti una o più sale riunione, in cui è possibile ricevere i clienti. Non è possibile assegnare una stessa sede, nello stesso giorno ed alla stessa ora, a più di un cliente. Agli appuntamenti partecipano i clienti e gli operatori dell'azienda.
L'azienda ha anche un gruppo di manager che definisce quali sono le proposte commerciali che l'azienda offre. Ogni proposta è identificata da un codice alfanumerico definito internamente dall'azienda. I manager hanno la possibilità di creare nuove proposte e di segnalare che alcune proposte già presenti nel sistema sono terminate, ovverosia che non possono più essere fornite ai clienti.
Infine, l'azienda ha un settore commerciale i cui membri reclutano nuovi clienti e li inseriscono all'interno del sistema.
In generale, il sistema informativo deve fornire le seguenti possibilità.

- ✶ Visualizzare il singolo cliente, eventualmente con i dati dell'azienda e del referente aziendale, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco dei servizi di consulenza acquistati.
- ✶ Possibilità di visualizzare l'elenco clienti a cui un utente è assegnato.
- ✶ Gestione delle note cliente: ogni volta che un cliente viene contattato deve essere possibile registrare/modificare/cancella
- ✶ Gestione delle opportunità: per ogni cliente deve essere possibile inserire una nuova opportunità, cioè una proposta commerciale.
- ✶ Gestione degli appuntamenti: deve essere possibile inserire un appuntamento con una nota descrittiva, una data/ora e un cliente a cui è riferito.
- ✶ Visualizzazione dell'agenda degli appuntamenti per un utente.
- ✶ Possibilità di inserire nuovi servizi di consulenza (riservata ai manager).
- ✶ Possibilità di inserire nuovi clienti (riservata al settore commerciale).
- ✶ Possibilità di inserire nuovi utenti dell'applicativo web (riservata ai manager).

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
5, 11, 14	di CRM	Marketing-oriented	perché il sistema è CRM l'azienda è marketing-oriented
9	società di CRM	Azienda marketing-oriented	perché il sistema è CRM l'azienda è marketing-oriented e società è ambiguo rispetto allo stesso termine usato per le società clienti dell'azienda
9	Anche mantenere il numero di partita IVA	mantenere i dati del referente incaricato di rappresentarla e la partita IVA della società	perché "anche" a riga 9 sottintende che per le società devo mantenere i dati elencati alla riga 6-8 del referente che vengono richiesti dall'operazione descritta a riga 31-33
12	utente aziendale del sistema	Funzionario	Utente aziendale del sistema può essere frainteso con cliente dell'azienda
14	operatori dell'azienda di CRM	Funzionario dell'azienda marketing-oriented	Operatori è vago
14	clienti	clienti loro assegnati	Per specificare che un cliente per essere contattato deve essere sotto la gestione del funzionario che lo contatta
15	recapiti	contatti	Recapiti richiama i recapiti telefonici (riga 8)
15	l'utente	Il funzionario	per uniformare i termini
17	Una risposta positiva di accettazione di una proposta commerciale	Una risposta positiva a una proposta commerciale	È più chiara
18	Sede	In una sala riunioni di una sede	L'incontro avviene in una sala riunioni
20	Sede	Sala riunioni	Sono le sale riunioni ad essere assegnate agli appuntamenti non le sedi
22	Gli operatori	I funzionari	Per uniformare i termini
31	Dell'azienda	Della società	Azienda può essere confusa con quella marketing-oriented
32	Aziendale	Societario	Perché il referente è quello della società cliente che fa da tramite con l'azienda marketing-oriented
33	Dei servizi di consulenza acquistati	Delle proposte commerciali accettate	Per uniformare i termini

Linea	Termine	Nuovo termine	Motivo correzione
34, 41, 44	Utente	Funzionario	Per uniformare i termini
40	Nota descrittiva	Proposta commerciale	L'appuntamento in sede ha luogo dopo che l'utente ha accettato una proposta commerciale
42	Nuovi servizi di consulenza	Nuove proposte commerciali	Per uniformare i termini

Specifica disambiguata

Un sistema di Customer Relationship Management (o gestione delle relazioni con i clienti) è un sistema informativo che verte sulla fidelizzazione del cliente. Si vuole realizzare un sistema CRM per un'azienda marketing-oriented che intende realizzare relazioni durevoli di breve e lungo periodo con i propri clienti, massimizzando quindi il valore degli stessi.
La base di dati del sistema informativo dell'azienda marketing-oriented deve poter memorizzare le informazioni su tutti i clienti di interesse dell'azienda, caratterizzati da nome, cognome, codice fiscale, data di nascita ed un insieme di contatti, sia in forma di indirizzi che di recapiti telefonici, email, fax. Alcuni dei clienti sono società che ricevono servizi dalla azienda marketing-oriented. Di questi interessa mantenere i dati del referente incaricato di rappresentarla e la partita IVA della società. Di tutti i clienti interessa sapere qual è la data di registrazione nel sistema di CRM.
L'azienda marketing-oriented in questione è di dimensione elevata ed ha a disposizione vari funzionari che interagiscono con i clienti. A ciascun funzionario viene assegnato un sottoinsieme di clienti da gestire.
Su base periodica, i funzionari dell'azienda marketing-oriented contattano i clienti loro assegnati mediante uno dei contatti forniti. In questa fase operativa, il funzionario deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali. Una risposta positiva a una proposta commerciale può essere associata ad un appuntamento in una sala riunioni di una sede. L'azienda ha più sedi, ciascuna caratterizzata da un indirizzo. In ciascuna sede sono presenti una o più sale riunione, in cui è possibile ricevere i clienti. Non è possibile assegnare una stessa sala riunioni, nello stesso giorno ed alla stessa ora, a più di un cliente. Agli appuntamenti partecipano i clienti e i funzionari dell'azienda.
L'azienda ha anche un gruppo di manager che definisce quali sono le proposte commerciali che l'azienda offre. Ogni proposta è identificata da un codice alfanumerico definito internamente dall'azienda. I manager hanno la possibilità di creare nuove proposte e di segnalare che alcune proposte già presenti nel sistema sono terminate, ovverosia che non possono più essere fornite ai clienti.
Infine, l'azienda ha un settore commerciale i cui membri reclutano nuovi clienti e li inseriscono all'interno del sistema.
In generale, il sistema informativo deve fornire le seguenti possibilità.

- ✶ Visualizzare il singolo cliente, eventualmente con i dati della società e del referente societario, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco delle proposte commerciali accettate.
- ✶ Possibilità di visualizzare l'elenco clienti a cui un funzionario è assegnato.
- ✶ Gestione delle note cliente: ogni volta che un cliente viene contattato deve essere possibile registrare/modificare/cancella.
- ✶ Gestione delle opportunità: per ogni cliente deve essere possibile inserire una nuova opportunità, cioè una proposta commerciale.
- ✶ Gestione degli appuntamenti: deve essere possibile inserire un appuntamento con una proposta commerciale, una data/ora e un cliente a cui è riferito.
- ✶ Visualizzazione dell'agenda degli appuntamenti per un funzionario.
- ✶ Possibilità di inserire nuove proposte commerciali (riservata ai manager).
- ✶ Possibilità di inserire nuovi clienti (riservata al settore commerciale).
- ✶ Possibilità di inserire nuovi funzionari dell'applicativo web (riservata ai manager).

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Cliente	Persona o referente societario che riceve proposte commerciali dall'azienda di CRM		Funzionario, Interazione, proposta commerciale
Funzionario	Dipendente dell'azienda che gestisce il suo gruppo di clienti		Clienti, Interazione, appuntamento
Interazione	Resoconto del funzionario riguardo interazione con il cliente ed eventuali risposte affermative alle proposte commerciali	Nota testuale, nota descrittiva	Funzionario, Cliente
Sede	Filiale dell'azienda di CRM identificata da un indirizzo		Sala riunioni
Sala riunioni	Sale delle sedi dove vengono ricevuti i clienti		Sede, appuntamento
Proposta commerciale	Offerta sottoposta ai clienti da parte dell'azienda		Cliente, Appuntamento
Appuntamento	Avviene in una sala riunione tra un funzionario e un cliente in una determinata data e ora a seguito di una risposta positiva del cliente a una proposta commerciale		Cliente, funzionario, sala riunione, proposta commerciale

Raggruppamento dei requisiti in insiemi omogenei

Frasi relative a cliente
Tutti i clienti di interesse dell'azienda, caratterizzati da nome, cognome, codice fiscale, data di nascita ed un insieme di contatti, sia in forma di indirizzi che di recapiti telefonici, email, fax.
Alcuni dei clienti sono società che ricevono servizi dall'azienda marketing-oriented. Di questi interessa mantenere i dati del referente incaricato di rappresentarla e la partita IVA della società.
Di tutti i clienti interessa sapere qual è la data di registrazione nel sistema di CRM.
A ciascun funzionario viene assegnato un sottoinsieme di clienti da gestire.
In questa fase operativa, il funzionario deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali
L'azienda ha più sedi, ciascuna caratterizzata da un indirizzo. In ciascuna sede sono presenti una o più sale riunione, in cui è possibile ricevere i clienti.
Non è possibile assegnare una stessa sede, nello stesso giorno ed alla stessa ora, a più di un cliente.
Visualizzare il singolo cliente, eventualmente con i dati della società e del referente societario, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco delle proposte commerciali accettate.
Possibilità di visualizzare l'elenco clienti a cui un funzionario è assegnato.
Possibilità di inserire nuovi clienti (riservata al settore commerciale).

Frasi relative a funzionario
L'azienda marketing-oriented in questione è di dimensione elevata ed ha a disposizione vari funzionari che interagiscono con i clienti.
A ciascun funzionario viene assegnato un sottoinsieme di clienti da gestire.
I funzionari dell'azienda marketing-oriented contattano i clienti mediante uno dei recapiti forniti.
Il funzionario deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali.
Agli appuntamenti partecipano i clienti e i funzionari dell'azienda.
Possibilità di visualizzare l'elenco clienti a cui un funzionario è assegnato.
Possibilità di inserire nuovi funzionari dell'applicativo web (riservata ai manager).

Frasi relative a interazione
Il funzionario deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali.
Visualizzare il singolo cliente, eventualmente con i dati della società e del referente societario, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco delle proposte commerciali accettate.
Gestione delle note cliente: ogni volta che un cliente viene contattato deve essere possibile registrare/modificare/cancella.
Gestione degli appuntamenti: deve essere possibile inserire un appuntamento con una nota descrittiva, una data/ora e un cliente a cui è riferito.

5

Frasi relative a sede
Una risposta positiva a una proposta commerciale può essere associata ad un appuntamento in una sala riunioni di una sede.
L'azienda ha più sedi, ciascuna caratterizzata da un indirizzo.
In ciascuna sede sono presenti una o più sale riunione, in cui è possibile ricevere i clienti.
Non è possibile assegnare una stessa sede, nello stesso giorno ed alla stessa ora, a più di un cliente.

Frasi relative a sala riunioni
Una risposta positiva a una proposta commerciale può essere associata ad un appuntamento in una sala riunioni di una sede.
In ciascuna sede sono presenti una o più sale riunione, in cui è possibile ricevere i clienti.
Non è possibile assegnare una stessa sala riunioni, nello stesso giorno ed alla stessa ora, a più di un cliente.

Frasi relative a proposta commerciale
In questa fase operativa, il funzionario deve inserire una nota testuale in cui viene riportato un breve resoconto dell'interazione con il cliente, annotando anche possibili risposte affermative alle proposte commerciali.
Una risposta positiva a una proposta commerciale può essere associata ad un appuntamento in sede.
L'azienda ha anche un gruppo di manager che definisce quali sono le proposte commerciali che l'azienda offre.
Ogni proposta è identificata da un codice alfanumerico definito internamente dall'azienda.
I manager hanno la possibilità di creare nuove proposte e di segnalare che alcune proposte già presenti nel sistema sono terminate, ovverosia che non possono più essere fornite ai clienti.
Visualizzare il singolo cliente, eventualmente con i dati della società e del referente societario, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco delle proposte commerciali accettate.
Gestione delle opportunità: per ogni cliente deve essere possibile inserire una nuova opportunità, cioè una proposta commerciale.
Possibilità di inserire nuove proposte commerciali (riservata ai manager).

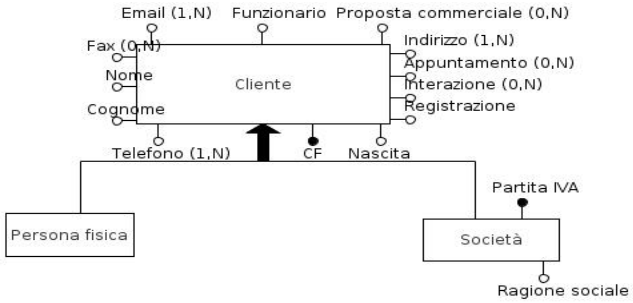
Frasi relative a appuntamento
Una risposta positiva a una proposta commerciale può essere associata ad un appuntamento in sede.
Agli appuntamenti partecipano i clienti e i funzionari dell'azienda.
Gestione degli appuntamenti: deve essere possibile inserire un appuntamento con una nota descrittiva, una data/ora e un cliente a cui è riferito.
Visualizzazione dell'agenda degli appuntamenti per un funzionario.

3. Progettazione concettuale

Costruzione dello schema E-R

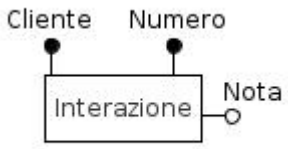
Per lo sviluppo dello schema ER è stato eseguito un approccio bottom-up modellando dapprima i concetti basici che sono stati ricavati dall'analisi della specifica e poi le loro associazioni concettuali, di seguito sono riportati i passi seguiti (alcuni dei quali per evitare ridondanza già rappresentati in modo più specifico):

1. Per il concetto di cliente si è scelto di specificarlo in 2 tipologie di clienti: i clienti fisici e le società. I dati mantenuti per la società che si riferiscono a persone fisiche si riferiscono al suo referente. Inoltre un cliente è strettamente legato al funzionario a cui è assegnato, agli appuntamenti a cui partecipa, alle proposte commerciali che accetta e alle interazioni che ha con il funzionario che sono tutti riportati come attributi e verranno reificati in seguito.

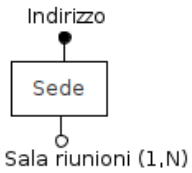


2. Si è scelto di identificare il funzionario attraverso la matricola piuttosto che il codice fiscale in quanto in genere in un luogo di lavoro ad ogni dipendente viene associata una matricola. Per il funzionario è importante sapere quale insieme di clienti gli è stato assegnato e a quali appuntamenti partecipa, in una prima fase sono stati perciò messi come attributi che verranno reificati in seguito.

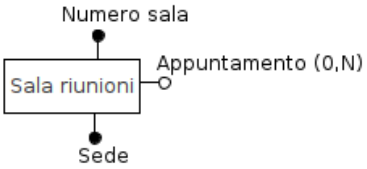
3. Per l'interazione è importante sapere tra quale funzionario e quale cliente si è svolta e quale nota viene scritta. In una prima fase funzionario e cliente vengono messi come attributi che verranno poi reificati.



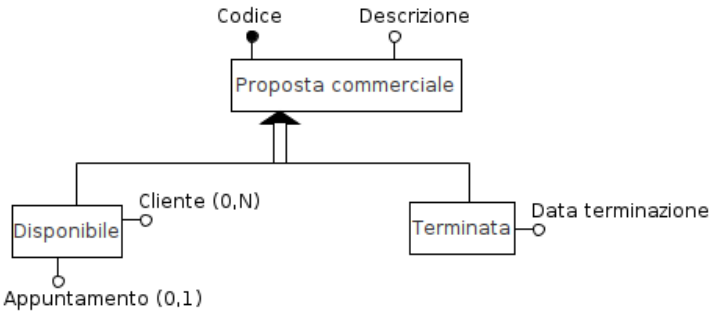
4. Alla sede è stato associato l'indirizzo che funge da identificatore e un insieme di sale riunioni rappresentate come attributo che verrà in seguito reificato.



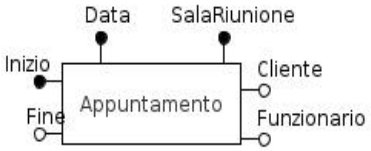
5. Per la sala riunioni è importante sapere a quale sede appartiene e il numero di sala riunioni poiché una sede può avere diverse sale riunioni. L'attributo sede verrà reificato in seguito.



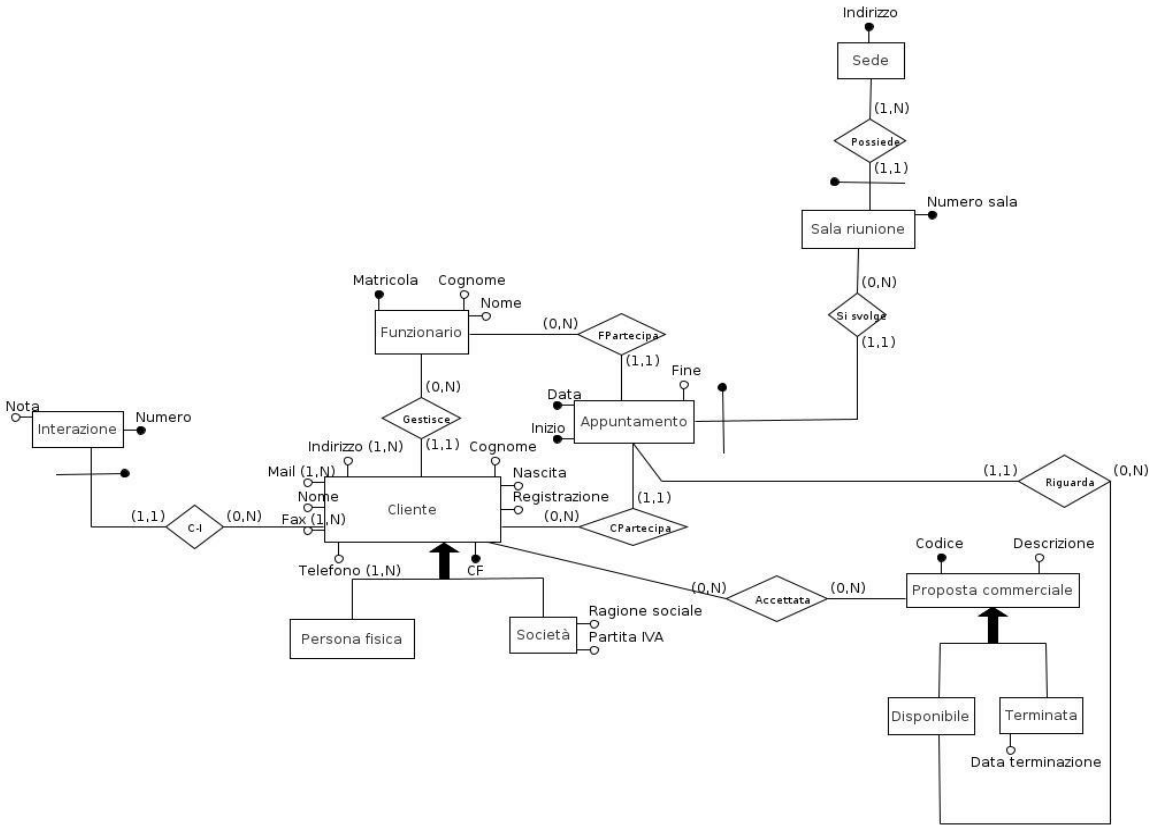
6. Alla proposta commerciale come riportato nella specifica è stato associato un codice e la sua descrizione. Si è scelto inoltre di specificarlo in 2 tipologie: attive e terminate. Queste ultime presentano anche la data di terminazione. Gli attributi cliente e appuntamento verranno reificati in seguito.



7. Per l'appuntamento è importante mantenere la data e l'ora in cui si svolge che insieme alla sala riunioni coinvolta servono a identificarlo. Inoltre un appuntamento è strettamente legato al cliente e il funzionario che vi partecipano, questi sono riportati come attributi che verranno in seguito reificati.



Integrazione finale



Nota: Il tool di disegno utilizzato non permette la rappresentazione di identificatori formati da un identificatore esterno più altri eventuali attributi di un'entità, perciò per le entità che si identificano in questo modo gli attributi che formano l'identificatore primario insieme a quello esterno sono rappresentati tramite pallino nero. La stessa cosa vale per le entità il cui identificatore è formato da più attributi (come per l'entità interazione), in quanto non vi è a possibilità di utilizzare la notazione "barretta con pallino nero" che li unisce.

Inoltre il tool di disegno non permette la rappresentazione di attributi composti che sono stati rappresentati come attributi normali questi sono “Indirizzo” di “Sede” e di “Cliente” che sono composti dagli attributi Via, Civico e Città.

Regole aziendali

(RV1) Il funzionario che interagisce con un cliente deve avere il cliente sotto la sua gestione

(RV2) Il funzionario per organizzare un appuntamento con un cliente deve avere il cliente sotto la sua gestione

(RV3) Un cliente per essere aggiunto al CRM deve essere maggiorenne

(RV4) Il funzionario può fissare un appuntamento con un cliente solo se non ha altre riunioni nello stesso intervallo e la sala riunioni selezionata è libera

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Cliente	Persona fisica o società cliente dell’azienda marketing-oriented	CF, Nome, Cognome, DataNascita, Contatti, DataRegistrazione	CF
Persona Fisica	Cliente dell’azienda	CF, Nome, Cognome, DataNascita, Contatti, DataRegistrazione	CF
Società	Società cliente dell’azienda con un referente	CF,Partita IVA, Nome, Cognome, DataNascita, Contatti, DataRegistrazione	CF
Funzionario	Operatore dell’azienda che contatta i clienti	Matricola, Nome, Cognome	Matricola
Interazione	Colloquio tra cliente e operatore	Nota, numero	Numero, cliente
Sede	Sede dell’azienda	Indirizzo	Indirizzo
Sala riunioni	Sala della sede dove si svolgono le riunioni	Numero	Numero, Sede
Proposta commerciale	Proposta commerciale fatta dall’azienda ai clienti	Codice, descrizione	Codice
Disponibile	Proposta commerciale fatta dall’azienda ai clienti ancora disponibile	Codice, descrizione	Codice
Terminata	Proposta commerciale fatta dall’azienda ai clienti terminata	Codice, descrizione, DataTerminazione	Codice
Appuntamento	Appuntamento tra cliente e operatore	Data, ora	Data, ora, Sala riunione

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
Cliente	E	218880
Persona Fisica	E	168960
Società	E	49920
Funzionario	E	1920
Interazione	E	264960
Sede	E	60
Sala riunioni	E	960
Proposta commerciale	E	80
Disponibile	E	40
Terminata	E	40
Appuntamento	E	218880
C-I	R	264960
Gestisce	R	218880
CPartecipa	R	218880
FPartecipa	R	218880
Accettata	R	175104
Riguarda	R	40
Si svolge	R	218880
Possiede	R	960

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
1	Visualizzare il singolo cliente, eventualmente con i dati della società e del referente societario, con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco delle proposte commerciali accettate.	2016/giorno
2	Possibilità di visualizzare l'elenco clienti a cui un funzionario è assegnato.	1920/giorno
3	Registrazione nota	1104/giorno
4	Modifica nota	110/giorno
5	Cancella nota	110/giorno
6	Gestione delle opportunità: per ogni cliente deve essere possibile inserire una nuova opportunità, cioè una proposta commerciale.	730/giorno
7	Gestione degli appuntamenti: deve essere	912/giorno

1 Indicare con E le entità, con R le relazioni

	possibile inserire un appuntamento con una proposta commerciale, una data/ora e un cliente a cui è riferito.	
8	Visualizzazione dell'agenda degli appuntamenti per un funzionario.	2016/giorno
9	Possibilità di inserire nuove proposte commerciali (riservata ai manager).	10/anno
10	Possibilità di decidere che alcune proposte sono terminate (riservata ai manager)	10/anno
11	Possibilità di inserire nuovi clienti (riservata al settore commerciale).	10944/giorno
12	Possibilità di inserire nuovi funzionari dell'applicativo web (riservata ai manager).	96/giorno

Costo delle operazioni

Op.	Accessi	Tipo	Costo
1	1 accesso a Cliente 1 accesso a Società 2 accesso a C-I 2 accesso a Interazione 1 accesso a Accettata 1 accesso a Proposta commerciale	L L L L L L	(1+0,2*1+2*1+2*1+1+1)*2016= 14515 accessi al giorno.
2	100 accesso a Gestisce 100 accesso a Cliente	L L	(100+100)*1920= 384000 accessi al giorno.
3	1 accesso a Interazione 1 accesso a C-I	S S	(2+2)*1104= 4416 accessi al giorno.
4	1 accesso a Interazione 1 accesso a Interazione	L S	(1+2)*110= 330 accessi al giorno.
5	1 accesso a Interazione 1 accesso a C-I	S S	(2+2)*110= 440 accessi al giorno.
6	1 accesso a Accettata 1 accesso a Proposta Commerciale	S L	(2+1)*730= 2190 accessi al giorno.
7	1 accesso a Appuntamento 1 accesso a Cliente 1 accesso a CPartecipa 1 accesso a Fpartecipa 1 accesso a Proposta Commerciale 1 accesso a Riguarda 1 accesso a Sala Riunione 1 accesso a SiSvolge	S L S S L S L S	(2+1+2+2+1+2+1+2)*912= 910 accessi al giorno.
8	150 accesso a FPartecipa 150 accesso a Appuntamento	L L	(150+150)*2016= 604800 accessi al giorno.
9	1 accesso a Proposta Commerciale 1 accesso a Disponibile	S S	(2+2)*10= 40 accessi all'anno.
10	1 accesso a Disponibile 1 accesso a Disponibile	L S	(1+2+2)*10= 50 accessi all'anno

	1 accesso a Terminata	S	
11	1 accesso a Cliente 1 accesso a Gestisce 1 accesso a Persona o Società	S S S	(2+2+2*0,2+2*0,8)*10944= 65664 accessi al giorno.
12	1 accesso a Funzionario	S	(2)*96= 192 accessi al giorno.

Ristrutturazione dello schema E-R

- L'associazione "FPartecipa" è ridondante in quanto il funzionario che partecipa all'appuntamento può essere ricavato dal cliente che ci partecipa. In particolare:
  * Togliendo l'associazione "FPartecipa" gli accessi sono i seguenti in caso di visualizzazione dell'agenda degli appuntamenti di un funzionario:
    1. 114 accessi a FPartecipa
    2. 114 accessi a Appuntamento
  - Lasciandolo gli accessi sono:
    1. 114 accesso a Gestisce
    2. 114 accesso a Cliente
    3. 114 accesso a CPartecipa
    4. 114 accesso a Appuntamento

    C'è quindi un aumento di 228 accessi al costo di 10*218880=2,2 Mbyte.

- Per la generalizzazione sull'entità "Cliente" si è scelto di eliminare l'entità "Persona fisica" e mettere una associazione tra "Cliente" e "Società" chiamata "Riferisce" con quest'ultima che diventa entità debole rispetto all'associazione. In questo modo si evitano ridondanze per i clienti che sono persone fisiche e si evita di mantenere in memoria diversi valori NULL dovuti al fatto che solo $\frac{1}{5}$ dei clienti sono società.

- Per la generalizzazione sull'entità "Proposta commerciale" si è scelto di eliminare la generalizzazione e di aggiungere un attributo opzionale chiamato "Terminata", questo permette di evitare ridondanze.

Trasformazione di attributi e identificatori

L'attributo composto "Indirizzo" di "Sede" viene sostituito dagli attributi "Via", "Civico", "Città".

Traduzione di entità e associazioni

FUNZIONARIO(Matricola, Nome, Cognome)

SEDE(Via, Civico, Città)

PROPOSTACOMMERCIALE(Codice, Descrizione, Terminata)

ACCETTATA(Cliente, Proposta commerciale)

SALARIUNIONE(Numero, SedeVia, SedeCivico, SedeCittà)

CLIENTE(CF, Nome, Cognome, Nascita, Registrazione, Funzionario)

SOCIETÀ(Cliente, RagioneSociale, PartitaIVA)

MAIL(Cliente, Mail)

TELEFONO(Cliente, Numero)

INDIRIZZO(Cliente, Via, Civico, Città)

FAX(Cliente, Fax)

INTERAZIONE(Cliente, Numero, Nota)

APPUNTAMENTO(SalaNumero, SedeVia, SedeCivico, SedeCittà, Data, Inizio, Fine, Cliente, Proposta Commerciale)

Con vincoli di integrità referenziale:

CLIENTE(Funzionario)⊆**FUNZIONARIO**(Matricola)

SALARIUNIONE(SedeVia, SedeCivico, SedeCittà)⊆**SEDE**(Via, Civico, Città)

ACCETTATA(Cliente)⊆**CLIENTE**(CF)

ACCETTATA(PropostaCommerciale)⊆**PROPOSTACOMMERCIALE**(Codice)

TELEFONO(Cliente)⊆**CLIENTE**(CF)

INDIRIZZO(Cliente)⊆**CLIENTE**(CF)

MAIL(Cliente)⊆**CLIENTE**(CF)

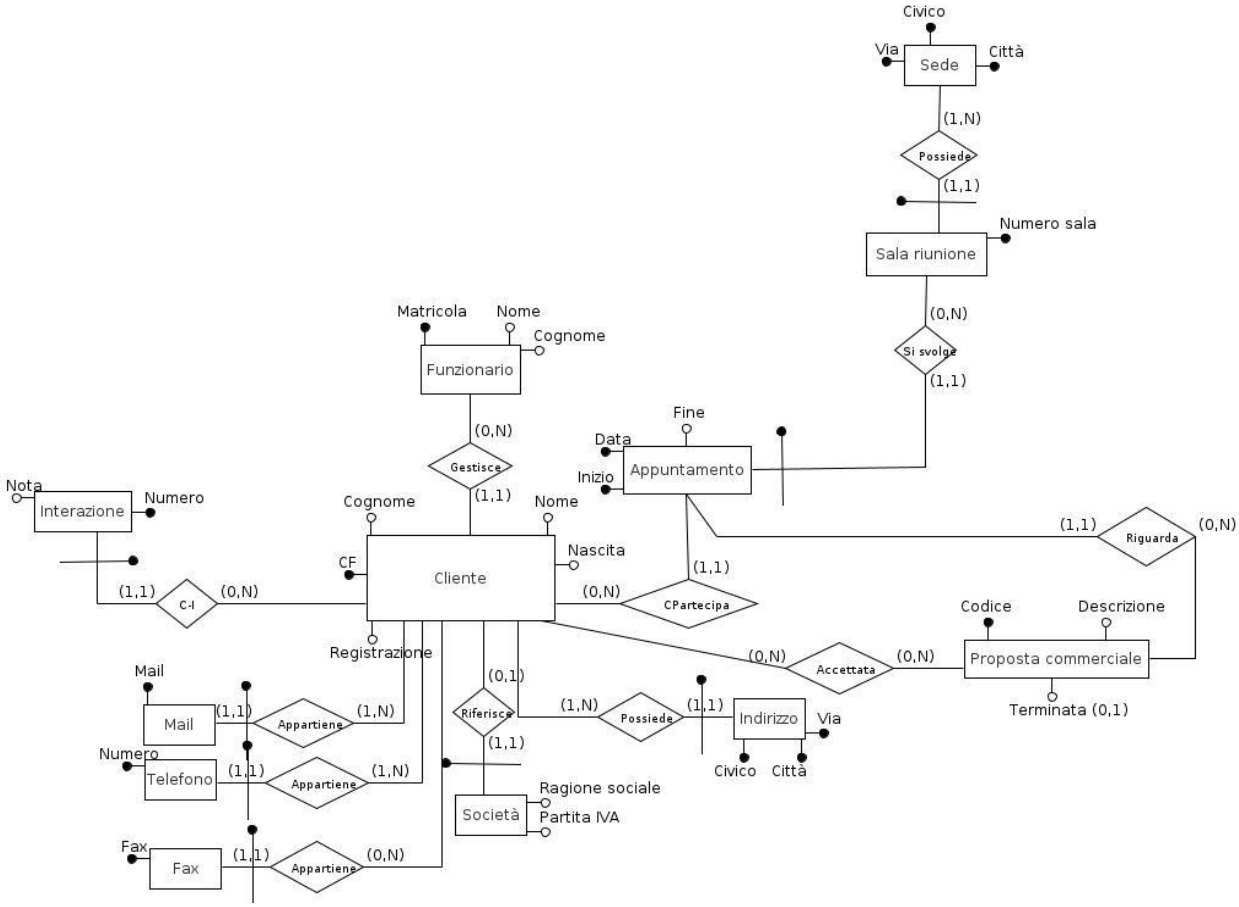
FAX(Cliente)⊆**CLIENTE**(CF)

INTERAZIONE(Funzionario)⊆**FUNZIONARIO**(Matricola)

INTERAZIONE(Cliente)⊆**CLIENTE**(CF)

APPUNTAMENTO(SalaNumero)⊆**SALARIUNIONE**(Numero)

APPUNTAMENTO(SedeVia, SedeCivico, SedeCittà)⊆**SEDE**(Via, Civico, Città)



Normalizzazione del modello relazionale

Forma 1NF:

Il modello è in 1NF infatti ogni attributo è definito su un dominio con valori atomici (indivisibili) e ogni attributo contiene un singolo valore da quel dominio.

Forma 2NF:

Il modello è in 2NF infatti è in forma 1NF e tutti gli attributi non chiave dipendono funzionalmente dall'intera chiave e non solo da una parte di essa.

Forma 3NF:

Il modello è in 3NF infatti è in forma 2NF e tutti gli attributi non-chiave dipendono dalla chiave soltanto.

5. Progettazione fisica

Utenti e privilegi

- Funzionario

  Il funzionario ha necessità di vedere i clienti a cui è assegnato per questo ha bisogno di accedere in lettura a “Cliente”, inoltre, ha necessità di vedere tutti i dati relativi a ognuno dei suoi clienti, i suoi recapiti per contattarlo e alle proposte accettate da quest’ultimo in particolare per questo può accedere in lettura anche a “Società”, “Fax”, “Telefono”, “Mail” e “Indirizzo”, “Accettata”. A seguito di un’interazione con il cliente può anche aggiungere note, modificarle o cancellarle quindi ha bisogno dei privilegi per leggere, modificare e cancellare note nella tabella “Interazione”. Inoltre, il funzionario può aggiungere appuntamenti con i clienti e vedere la sua agenda di appuntamenti già fissati per fare ciò ha bisogno dei privilegi in lettura e scrittura sulla tabella “Appuntamento”. Può inoltre vedere gli appuntamenti fissati dai suoi colleghi per evitare conflitti nel momento in cui cerca di aggiungere un nuovo appuntamento con un suo cliente per questo ha bisogno anche del privilegio in lettura sulla tabella “SalaRiunione”.

  Infine, ha bisogno anche del privilegio in scrittura sulla tabella “Accettata” per inserire nuove proposte accettate dai suoi clienti.

  Per fare questo ha bisogno dei privilegi di esecuzione sulle seguenti stored procedures:

  - Aggiungi_appuntamento
  - Aggiungi_nota
  - Elimina_nota
  - Inserisci_proposta_accettata
  - Modifica_nota
  - Nuovi_appuntamenti
  - Proposte_attive
  - Tutti_appuntamenti
  - Visualizza_accettate
  - Visualizza_cliente
  - Visualizza_fax
  - Visualizza_indirizzo
  - Visualizza_lista_clienti

- Visualizza_mail
- Visualizza_sale
- Visualizza_societa
- Visualizza_telefono

- Manager

  Il manager deve avere la possibilità di inserire nuove proposte e dichiarare che qualche proposta è terminata quindi deve avere il privilegio di lettura e scrittura sulla tabella "PropostaCommerciale" in modo da poter visualizzare il codice e la descrizione della proposta che vuole terminare.

  Inoltre, deve poter aggiungere nuovi funzionari, quindi, ha il privilegio di scrittura sulla tabella "Utente".

  Per fare questo ha bisogno del privilegio di esecuzione sulle seguenti stored procedures:

  - Aggiungi_funzionario
  - Inserisci_proposta
  - Termina_proposta

- Membro settore commerciale

  Il membro del settore commerciale deve avere la possibilità di inserire nuovi clienti nel sistema sia come persone fisiche che come società oltre a tutti i loro dati e contatti, quindi, deve avere il privilegio di scrittura sulle tabelle "Cliente", "Societa", "Mail", "Telefono", "Fax", "Indirizzo".

  Per fare questo ha bisogno del privilegio di esecuzione sulle seguenti stored procedures:

  - Aggiungi_fax
  - Aggiungi_indirizzo
  - Aggiungi_mail
  - Aggiungi_telefono
  - Inserisci_cliente
  - Inserisci_societa

Strutture di memorizzazione

Tabella Accettata		
Attributo	**Tipo di dato**	**Attributi**[2]
proposta_commerciale	VARCHAR(10)	PK, NN
cliente	VARCHAR(16)	PK,NN

Tabella Appuntamento		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	NN
SalaRiunione_numero	INT	PK, NN
SalaRiunione_sede_via	VARCHAR(45)	PK, NN
SalaRiunione_sede_civico	VARCHAR(4)	PK, NN
SalaRiunione_sede_citta	VARCHAR(45)	PK, NN
data	DATE	PK, NN
ora_inizio	TIME	PK, NN
ora_fine	TIME	NN
PropostaCommerciale_codice	VARCHAR(10)	NN

Tabella Cliente		
Attributo	**Tipo di dato**	**Attributi**
cf	VARCHAR(16)	PK, NN
nome	VARCHAR(45)	NN
cognome	VARCHAR(45)	NN
nascita	DATE	NN
registrazione	DATE	NN
Funzionario_matricola	VARCHAR(6)	NN

Tabella Fax		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
fax	INT	PK, NN

5

2 PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella Indirizzo		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
via	VARCHAR(45)	PK, NN
civico	VARCHAR(5)	PK, NN
citta	VARCHAR(45)	PK, NN

Tabella Interazione		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
data	DATETIME	PK, NN
nota	VARCHAR(128)	NN

Tabella Mail		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
mail	VARCHAR(45)	PK, NN

Tabella PropostaCommerciale		
Attributo	**Tipo di dato**	**Attributi**
codice	VARCHAR(10)	PK, NN
descrizione	VARCHAR(45)	NN
terminata	DATE	

Tabella SalaRiunione		
Attributo	**Tipo di dato**	**Attributi**
numero	INT	PK, NN
Sede_via	VARCHAR(45)	PK, NN
Sede_civico	VARCHAR(5)	PK, NN
Sede_citta	VARCHAR(45)	PK, NN

5

Tabella Sede		
Attributo	**Tipo di dato**	**Attributi**
via	VARCHAR(45)	PK, NN
civico	VARCHAR(5)	PK, NN
citta	VARCHAR(45)	PK, NN

Tabella Societa		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
ragione_sociale	VARCHAR(45)	NN
partita_iva	VARCHAR(16)	NN

Tabella Telefono		
Attributo	**Tipo di dato**	**Attributi**
Cliente_cf	VARCHAR(16)	PK, NN
numero	INT	PK, NN

Tabella Utente		
Attributo	**Tipo di dato**	**Attributi**
matricola	VARCHAR(6)	PK, NN
nome	VARCHAR(20)	NN
cognome	VARCHAR(20)	NN
password	CHAR(32)	NN
ruolo	ENUM(‘manager’, ‘settorecommerciale’, ‘funzionario’)	NN

Indici

Tabella Accettata	
Indice PRIMARY	**Tipo[3]:**
proposta_commerciale, cliente	PR
Indice Accettata_FK_Cliente_idx	**Tipo:**
cliente	IDX
Indice Accettata_FK_Codice_idx	**Tipo:**
proposta_commerciale	IDX

Tabella Appuntamento	
Indice PRIMARY	**Tipo:**
SalaRiunione_numero, SalaRiunione_sede_via, SalaRiunione_sede_civico, SalaRiunione_sede_citta, data, ora_inizio	PR
Indice Appuntamento_FK_Cliente_idx	**Tipo:**
cliente	IDX
Indice Appuntamento_FK_Codice_idx	**Tipo:**
PropostaCommerciale_codice	IDX

Tabella Cliente	
Indice PRIMARY	**Tipo:**
cf	PR
Indice Cliente_FK_Funzionario_idx	**Tipo:**
Funzionario_matricola	IDX

5

Tabella Fax	
Indice PRIMARY	**Tipo:**
Cliente_cf, fax	PR

Tabella Indirizzo	
Indice PRIMARY	**Tipo:**
Cliente_cf, via, civico, citta	PR

3 IDX = index, UQ = unique, FT = full text, PR = primary.

Tabella Interazione	
Indice PRIMARY	**Tipo:**
Cliente_cf, data	PR

Tabella Mail	
Indice PRIMARY	**Tipo:**
Cliente_cf, mail	PR

Tabella PropostaCommerciale	
Indice PRIMARY	**Tipo:**
codice	PR

Tabella SalaRiunione	
Indice PRIMARY	**Tipo:**
numero, Sede_via, Sede_civico, Sede_citta	PR
Indice SalaRiunione_FK_Sede_idx	**Tipo:**
Sede_via, Sede_civico, Sede_citta	IDX

5

Tabella Sede	
Indice PRIMARY	**Tipo:**
sede, civico, citta	PR

Tabella Societa	
Indice PRIMARY	**Tipo:**
Cliente_cf	PR

Tabella Telefono	
Indice PRIMARY	**Tipo:**
Cliente_cf, numero	PR

Tabella Utente	
Indice PRIMARY	**Tipo:**
matricola	PR

Trigger

- Questo trigger serve a controllare che il funzionario crei nuovi appuntamenti solo con clienti sotto la sua gestione, che l'appuntamento inserito non sia precedente a data e ora corrente, che la proposta di cui si discuterà durante l'appuntamento non sia terminata e che il funzionario, o la sala selezionata, non siano già occupati in un'altra riunione nello stesso intervallo di tempo.

```
CREATE DEFINER = CURRENT_USER TRIGGER `CRM-db`.`Appuntamento_BEFORE_INSERT` BEFORE INSERT ON `Appuntamento` FOR EACH ROW
BEGIN
   declare var_matricola varchar(6);
   select Funzionario_matricola into var_matricola from Cliente where cf=new.Cliente_cf;
if exists(select * from PropostaCommerciale where new.PropostaCommerciale_codice=codice and terminata is not null) then
        signal sqlstate '45001' set message_text="Proposal closed";
   end if;

   if exists(select * from Appuntamento where `data`=new.`data` and new.ora_inizio>=ora_inizio and new.ora_inizio<ora_fine and new.Cliente_cf in(select cf from Cliente where Funzionario_matricola=var_matricola)) then
        signal sqlstate '45001' set message_text="You have another appoinment at the same time";
end if;


if exists(select * from Appuntamento where `data`=new.`data` and new.ora_fine>ora_inizio and new.ora_fine<=ora_fine and new.Cliente_cf in(select cf from Cliente where Funzionario_matricola=var_matricola)) then
        signal sqlstate '45001' set message_text="You have another appointment at the same time";
end if;

   if exists(select * from Appuntamento where SalaRiunione_numero=new.SalaRiunione_numero and SalaRiunione_sede_via=new.SalaRiunione_sede_via and SalaRiunione_sede_civico=new.SalaRiunione_sede_civico and SalaRiunione_sede_citta=new.SalaRiunione_sede_citta and `data`=new.`data` and new.ora_inizio>ora_inizio and new.ora_inizio<ora_fine) then
        signal sqlstate '45001' set message_text="There is another appoinment in the same data and in the same meeting room";
```

```
    end if;

    if exists(select * from Appuntamento where
    SalaRiunione_numero=new.SalaRiunione_numero and
    SalaRiunione_sede_via=new.SalaRiunione_sede_via and
    SalaRiunione_sede_civico=new.SalaRiunione_sede_civico and
    SalaRiunione_sede_citta=new.SalaRiunione_sede_citta and `data`=new.`data` and
    new.ora_fine>ora_inizio and new.ora_fine<=ora_fine) then
        signal sqlstate '45001' set message_text="There is another appoinment in the same
    data and in the same meeting room";
    end if;

      if new.ora_inizio>new.ora_fine then
        signal sqlstate '45001' set message_text="Ending time must be later than starting
    time";
    end if;

    END
```

- Questo trigger serve a controllare che la mail inserita abbia un formato corretto.

```
    CREATE DEFINER = CURRENT_USER TRIGGER `CRM-db`.`Mail_BEFORE_INSERT`
    BEFORE INSERT ON `Mail` FOR EACH ROW
    BEGIN
    if new.mail not regexp'^[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9._-]@[a-zA-Z0-9][a-zA-Z0-
    9._-]*[a-zA-Z0-9]\\.[a-zA-Z]{2,63}$' then
        signal sqlstate '45001' set message_text="Incorrect mail";
      end if;
    END
```

- Questo trigger controlla che una proposta commerciale aggiunta a un cliente non sia terminata.

```
  CREATE DEFINER = CURRENT_USER TRIGGER `CRM-
  db`.`Accettata_BEFORE_INSERT` BEFORE INSERT ON `Accettata` FOR EACH ROW
  BEGIN
    if (select `terminata` from `PropostaCommerciale` where
  `codice`=NEW.proposta_commerciale) is not null then
        signal sqlstate '45000'= "Selected proposal is closede";
    end if;
  END
```

- Questo trigger controlla che tutti gli utenti inseriti siano maggiorenni e che il codice fiscale inserito abbia un formato corretto:

```
  CREATE DEFINER = CURRENT_USER TRIGGER `CRM-db`.`Cliente_BEFORE_INSERT`
  BEFORE INSERT ON `Cliente` FOR EACH ROW
  BEGIN
    set new.registrazione = curdate();
```

```
    if not (YEAR(NOW())-YEAR(new.nascita))>18 OR (DAY(NOW())>DAY(new.nascita)
AND MONTH(NOW())>=MONTH(new.nascita) AND YEAR(NOW())-
YEAR(new.nascita)=18) then
        signal sqlstate '45001' set message_text="Customer is a minor";
  end if;

    if new.cf not regexp'^[A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]$' then
        signal sqlstate '45001' set message_text="Incorrect tax code";
    end if;
END
```

Eventi

Non sono stati previsti eventi.

Viste

Non sono state previste viste.

Stored Procedures e transazioni

```
-- -----------------------------------------------------
-- procedure login
-- -----------------------------------------------------

CREATE PROCEDURE `login` (in var_matricola varchar(6), in var_pass varchar(45), out var_role
INT)
BEGIN
    declare var_user_role ENUM('manager', 'settorecommerciale', 'funzionario');

    select `ruolo` from `Utente`
        where `matricola` = var_matricola
and `password` = md5(var_pass)
into var_user_role;

-- See the corresponding enum in the client
        if var_user_role = 'manager' then
            set var_role = 1;
        elseif var_user_role = 'settorecommerciale' then
            set var_role = 2;
        elseif var_user_role = 'funzionario' then
            set var_role = 3;
        else
            set var_role = 4;
        end if;
END

-- -----------------------------------------------------
-- procedure aggiungi_funzionario
-- -----------------------------------------------------
```

```
CREATE PROCEDURE `aggiungi_funzionario` (IN var_matricola VARCHAR(45), IN var_nome
varchar(45), IN var_cognome VARCHAR(45), IN var_password VARCHAR(45))
BEGIN
    insert into `Utente` values (var_matricola, var_nome, var_cognome, MD5(var_password),
'funzionario');
END

-- -----------------------------------------------------
-- procedure modifica_nota
-- -----------------------------------------------------
Prima di modificare la nota controllo che il funzionario che vuole eseguire l'operazione ha il cliente
sotto la sua gestione.
CREATE PROCEDURE `modifica_nota` (in var_cf_cliente varchar(16), in var_data datetime, in
var_nota varchar(128), in var_funzionario varchar(6))
BEGIN

    declare exit handler for sqlexception
  begin
        rollback;
    resignal;
    end;

  set transaction isolation level read committed; #evito letture sporche

    if (select Funzionario_matricola from Cliente where cf=var_cf_cliente)<>var_funzionario
then
        signal sqlstate '45001' set message_text="Uncorrect customer";
  end if;
  if not exists(select * from `Interazione` where `Cliente_cf`= var_cf_cliente and `data`= var_data)
then
        signal sqlstate '45001' set message_text="There is no note on the selected date";
    end if;
    update `Interazione`
  set `nota`= var_nota
  where `Cliente_cf`= var_cf_cliente and `data`= var_data;
  commit;
END

-- -----------------------------------------------------
-- procedure aggiungi_nota
-- -----------------------------------------------------

CREATE PROCEDURE `aggiungi_nota` (in var_cliente_cf varchar(16), in var_nota varchar(128))
BEGIN
    declare var_date datetime;
      begin
        rollback;
        resignal;
    end;
```

```
    set transaction isolation level read committed; #evito letture sporche

        if (select Funzionario_matricola from Cliente where cf=var_cliente_CF)<>var_funzionario
5   then
            signal sqlstate '45001' set message_text="Uncorrect customer";
    end if;

    set var_date =DATE_FORMAT(NOW(), '%Y/%m/%d %H/%i');
10      insert into `Interazione` (`Cliente_CF`,`data`, `nota`) values (var_cliente_cf,var_date,
    var_nota);
    END

    -- ----------------------------------------------------
15  -- procedure termina_proposta
    -- ----------------------------------------------------
    Aggiorno solo le proposte che non sono già state terminate da altri manager.
    Prima di segnare come terminata una proposta controllo che essa sia effettivamente presente
    all'interno del sistema.
20
    CREATE PROCEDURE `termina_proposta` (in var_codice varchar(10))
    BEGIN

        declare exit handler for sqlexception
25  begin
            rollback;
      resignal;
        end;

30  set transaction isolation level read committed; #evito letture sporche

        if(select count(*) from `PropostaCommerciale` where `codice`= var_codice and terminata is
    null)=0 then
            signal sqlstate '45001' set message_text="Wrong proposal ID";
35  end if;
        update `PropostaCommerciale` set `Terminata`=curdate() where `codice`= var_codice;
    END

    -- ----------------------------------------------------
40  -- procedure elimina_nota
    -- ----------------------------------------------------

    CREATE PROCEDURE `elimina_nota` (in var_cliente_CF varchar(16), in var_data datetime, in
    var_funzionario varchar(6))
45  BEGIN

        declare exit handler for sqlexception
    begin
            rollback;
50      resignal;
```

```
        end;

    set transaction isolation level read committed; #evito letture sporche

        if (select Funzionario_matricola from Cliente where cf=var_cliente_CF)<>var_funzionario
then
            signal sqlstate '45001' set message_text="Uncorrect customer";
    end if;
        DELETE FROM `Interazione` where  `Cliente_CF`= var_cliente_CF and `data`= var_data;
        commit;
END

-- ----------------------------------------------------
-- procedure aggiungi_appuntamento
-- ----------------------------------------------------
Prima di inserire un appuntamento controllo che la data di esso sia successiva a quella corrente.

CREATE PROCEDURE `aggiungi_appuntamento` (in var_CF varchar(16), in var_numero_sala int,
in var_viasede varchar(45), in var_civicosede varchar(5), in var_cittasede varchar(45), in var_data
date, in var_inizio time, in var_fine time, in var_codice varchar(10))
BEGIN
        declare exit handler for sqlexception
    begin
            rollback;
      resignal;
        end;

        if var_data<CURDATE() or (var_data=CURDATE() and var_inizio<CURTIME()) then
            signal sqlstate '45001' set message_text="Enter a date after the current date";
        end if;
        INSERT INTO `Appuntamento` VALUES(var_CF, var_numero_sala, var_viasede,
var_civicosede, var_cittasede, var_data, var_inizio, var_fine, var_codice);
        commit;
END

-- ----------------------------------------------------
-- procedure inserisci_cliente
-- ----------------------------------------------------

CREATE PROCEDURE `inserisci_cliente` (in var_cf varchar(16), in nome varchar(45), in cognome
varchar(45), in var_nascita date, in var_Funzionario_matricola varchar(6), in var_mail varchar(45),
in var_telefono varchar(10), in var_via varchar(45), in var_civico varchar(45), in var_citta
varchar(45), in var_fax varchar(11))
BEGIN

    #Transazione perché insert collegati tra loro

        INSERT INTO `Cliente` (cf, nome, cognome, nascita, Funzionario_matricola)
        VALUES (var_cf, nome, cognome, var_nascita, var_Funzionario_matricola);
        INSERT INTO `Mail` (Cliente_cf, mail)
```

```
        VALUES (var_cf, var_mail);
        INSERT INTO `Telefono` (Cliente_cf, numero)
        VALUES (var_cf, var_telefono);
        INSERT INTO `Indirizzo` (Cliente_cf, via, civico, citta)
        VALUES (var_cf, var_via, var_civico, var_citta);
        INSERT INTO `Fax` (Cliente_cf, fax)
        VALUES (var_cf, var_fax);
        COMMIT;
END

-- ------------------------------------------------------
-- procedure inserisci_societa
-- ------------------------------------------------------
```

Prima di inserire una società e il suo referente controllo che non siano già all'interno del sistema.

```
CREATE PROCEDURE `inserisci_societa` (in var_cf varchar(16),in nome varchar(45), in cognome
varchar(45), in var_nascita date, in var_Funzionario_matricola varchar(6), in var_mail varchar(45),
in var_telefono varchar(10), in var_via varchar(45), in var_civico varchar(45), in var_citta
varchar(45), in var_fax varchar(11), in var_ragione_sociale varchar(45), in var_partita_iva
varchar(11))
BEGIN
        declare var_num int;

   declare exit handler for sqlexception
   begin
                rollback;
      resignal;
        end;

   set transaction isolation level read committed; #evito letture sporche

   select count(*) from Cliente where cf=var_cf into var_num;
   if var_num=0 then
                INSERT INTO `Cliente` (cf, nome, cognome, nascita, Funzionario_matricola)
                VALUES (var_cf, nome, cognome, var_nascita, var_Funzionario_matricola);
                INSERT INTO `Mail` (Cliente_cf, mail)
                VALUES (var_cf, var_mail);
                INSERT INTO `Telefono` (Cliente_cf, numero)
                VALUES (var_cf, var_telefono);
                INSERT INTO `Indirizzo` (Cliente_cf, via, civico, citta)
                VALUES (var_cf, var_via, var_civico, var_citta);
                INSERT INTO `Fax` (Cliente_cf, fax)
                VALUES (var_cf, var_fax);
        INSERT INTO `Societa` (Cliente_cf, ragione_sociale, partita_iva) VALUES (var_cf,
var_ragione_sociale, var_partita_iva);
        else
                select count(*) from Societa where Cliente_cf=var_cf into var_num;
                if var_num=0 then
                        INSERT INTO `Societa` (Cliente_cf, ragione_sociale, partita_iva) VALUES
(var_cf, var_ragione_sociale, var_partita_iva);
```

```
                else
                        signal sqlstate '45001' set message_text="The client is already registered in the
CRM system";
                end if;
        end if;
    commit;
END

-- ----------------------------------------------------
-- procedure inserisci_proposta
-- ----------------------------------------------------

CREATE PROCEDURE `inserisci_proposta` (in var_codice varchar(10), in var_descrizione
varchar(45))
BEGIN
        insert into `PropostaCommerciale` (codice, descrizione) values (var_codice, var_descrizione);
END

-- ----------------------------------------------------
-- procedure nuovi_appuntamenti
-- ----------------------------------------------------

CREATE PROCEDURE `nuovi_appuntamenti` (in var_matricola varchar(6))
BEGIN
        set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

        select `a`.Cliente_cf, `a`.SalaRiunione_sede_citta as citta, `a`.SalaRiunione_sede_via as via,
`a`.SalaRiunione_sede_civico as civico,`a`.SalaRiunione_numero as numero_sala,
DATE_FORMAT(`data`, '%d/%m/%Y') as `data`, DATE_FORMAT(ora_inizio, '%H:%i') as
ora_inizio, DATE_FORMAT(ora_fine, '%H:%i') as ora_fine, `a`.PropostaCommerciale_codice as
codice
    from `Appuntamento` as `a` join `Cliente` on Cliente_cf=cf
    where Cliente_cf in (select cf from `Cliente` where Funzionario_matricola=var_matricola) and
(DATE_FORMAT(`data`, '%Y/%m/%d')>CURDATE() or (DATE_FORMAT(`data`,
'%Y/%m/%d')=CURDATE() and ora_inizio>=CURTIME()))
        order by DATE_FORMAT(`data`, '%Y/%m/%d'),`ora_inizio`, `SalaRiunione_numero`;
    commit;
END

-- ----------------------------------------------------
-- procedure inserisci_proposta_accettata
-- ----------------------------------------------------

CREATE PROCEDURE `inserisci_proposta_accettata` (in var_codice_proposta varchar(10), in
var_cf_cliente varchar(16), in matricola varchar(6))
BEGIN
         declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
```

```
                rollback;
        resignal;
            end;

    set transaction isolation level read committed;

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf_cliente;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;
        INSERT INTO `CRM-db`.`Accettata` VALUES (var_codice_proposta, var_cf_cliente);
        commit;
END

-- -----------------------------------------------------
-- procedure visualizza_lista_clienti
-- -----------------------------------------------------

CREATE PROCEDURE `visualizza_lista_clienti` (in var_funzionario varchar(6))
BEGIN

    set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

        select `c`.CF, `c`.nome, `c`.cognome
    from Cliente as c
    where `Funzionario_matricola`=var_funzionario;
    commit;
END

-- -----------------------------------------------------
-- procedure aggiungi_mail
-- -----------------------------------------------------

CREATE PROCEDURE `aggiungi_mail` (in var_cf varchar(16), in var_mail varchar(45))
BEGIN
        insert into Mail (Cliente_cf, mail) values (var_cf, var_mail);
END

-- -----------------------------------------------------
-- procedure aggiungi_telefono
-- -----------------------------------------------------

CREATE PROCEDURE `aggiungi_telefono` (in var_cf varchar(16), in var_numero varchar(10))
BEGIN
        insert into Telefono (Cliente_cf, numero) values (var_cf, var_numero);
END

-- -----------------------------------------------------
-- procedure aggiungi_fax
```

```
-- ----------------------------------------------------

CREATE PROCEDURE `aggiungi_fax` (in var_cf varchar(16), in var_numero varchar(11))
BEGIN
        insert into Fax (Cliente_cf, numero) values (var_cf, var_numero);
END

-- ----------------------------------------------------
-- procedure visualizza_societa
-- ----------------------------------------------------
```

Prima di visualizzare il cliente controllo che il funzionario che ha richiesto i suoi dati sia lo stesso che lo ha in gestione e poi controllo se il cliente selezionato è un referente.

```
CREATE PROCEDURE `visualizza_societa` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
        declare var_num int;

   declare var_matricola varchar(6);
   declare exit handler for sqlexception
   begin
                rollback;
      resignal;
        end;

   set transaction read only;
   set transaction isolation level read committed; #evito letture sporche, controllo prima che il
```

funzionario che vuole visualizzare il cliente è quello che lo ha in gestione e poi controllo se il cliente selezionato è referente di una società.

```
   select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
   if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
   end if;

        select count(*) from Societa where Cliente_cf=var_cf into var_num;
        if var_num>0 then
                select ragione_sociale, partita_iva from Societa where Cliente_cf=var_cf;
   end if;
   commit;
END

-- ----------------------------------------------------
-- procedure visualizza_mail
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_mail` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
         declare var_matricola varchar(6);
   declare exit handler for sqlexception
   begin
```

```
                rollback;
        resignal;
            end;

        set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;

        select mail from Mail where Cliente_cf=var_cf;
END

-- ----------------------------------------------------
-- procedure visualizza_fax
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_fax` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
        declare var_num int;
        declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
                rollback;
        resignal;
            end;

    set transaction read only;
        set transaction isolation level read committed; #evito letture sporche

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;

        select count(*) from Fax where Cliente_cf=var_cf into var_num;
        if var_num>0 then
                select fax from Fax where Cliente_cf=var_cf;
    end if;
    commit;
END

-- ----------------------------------------------------
-- procedure visualizza_indirizzo
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_indirizzo` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
```

```
        declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
            rollback;
5       resignal;
        end;

        set transaction read only;
    set transaction isolation level read committed;
10
    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
            signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;
15      select via, civico, citta from Indirizzo where Cliente_cf=var_cf;
        commit;
END

-- ----------------------------------------------------
20 -- procedure visualizza_telefono
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_telefono` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
25       declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
            rollback;
       resignal;
30      end;

    set transaction isolation level read committed;

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
35  if matricola<>var_matricola then
            signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;

        set transaction read only;
40  set transaction isolation level read committed; #evito letture sporche
        select numero as telefono from Telefono where Cliente_cf=var_cf;
        commit;
END

45 -- ----------------------------------------------------
-- procedure visualizza_accettate
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_accettate` (in var_cf varchar(16), in matricola varchar(6))
50 BEGIN
```

```
        declare var_num int;
         declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
                rollback;
       resignal;
        end;

    set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;
        select count(*) from Accettata where cliente=var_cf into var_num;
    if var_num>0 then
                select codice, descrizione from Accettata join PropostaCommerciale on
proposta_commerciale=codice where cliente=var_cf;
        end if;
    commit;
END

-- ----------------------------------------------------
-- procedure visualizza_note
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_note` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
        declare var_num int;
    declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
                rollback;
       resignal;
        end;

        set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;
    if (select count(*) from Cliente where cf=var_cf)=0 then
                signal sqlstate '45001' set message_text="There is no customer with this tax code";
    end if;

        select count(*) from Interazione where Cliente_cf=var_cf into var_num;
    if var_num>0 then
```

```
                select DATE_FORMAT(`data`, '%d/%m/%Y %H:%i') as `data`, nota from Interazione
where Cliente_cf=var_cf;
        else
                signal sqlstate '45002' set message_text="There aren't note for this customer";
    end if;
    commit;
END

-- ----------------------------------------------------
-- procedure proposte_attive
-- ----------------------------------------------------

CREATE PROCEDURE `proposte_attive` ()
BEGIN
        set transaction read only;
    set transaction isolation level read committed;#evito letture sporche
        select codice, descrizione
    from PropostaCommerciale
    where terminata is null;
    commit;
END

-- ----------------------------------------------------
-- procedure visualizza_cliente
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_cliente` (in var_cf varchar(16), in matricola varchar(6))
BEGIN
         declare var_matricola varchar(6);
    declare exit handler for sqlexception
    begin
                rollback;
       resignal;
        end;


        set transaction read only;
    set transaction isolation level read committed; #evito letture sporche

    select Funzionario_matricola into var_matricola from Cliente where cf=var_cf;
    if matricola<>var_matricola then
                signal sqlstate '45001' set message_text="Selected customer isn't your";
    end if;
        select nome, cognome, DATE_FORMAT(nascita, '%d/%m/%Y') as nascita,
DATE_FORMAT(registrazione,'%d/%m/%Y') as registrazione from Cliente where cf=var_cf;
        commit;
END

-- ----------------------------------------------------
-- procedure visualizza_sale
```

```
-- ----------------------------------------------------

CREATE PROCEDURE `visualizza_sale` (in var_citta varchar(45))
BEGIN
        declare var_count int;

   declare exit handler for sqlexception
   begin
                rollback;
      resignal;
        end;
   set transaction read only;
   set transaction isolation level read committed; #evito letture sporche

   select count(*)
   from Appuntamento
   where SalaRiunione_sede_citta=var_citta and `data`>=NOW()
   into var_count;
   if var_count=0 then
                signal sqlstate '45002' set message_text="There are no busy meeting rooms in the
selected city";
   end if;
        select SalaRiunione_numero as numero_sala, SalaRiunione_sede_civico as civico_sede,
SalaRiunione_sede_via as via_sede, SalaRiunione_sede_citta as citta_sede, DATE_FORMAT(`data`,
'%d/%m/%Y') as `data`, DATE_FORMAT(ora_inizio, '%H:%i') as ora_inizio,
DATE_FORMAT(ora_fine, '%H:%i') as ora_fine
   from Appuntamento
   where SalaRiunione_sede_citta=var_citta and `data`>=NOW()
        order by DATE_FORMAT(`data`, '%Y/%m/%d'),`ora_inizio`, `SalaRiunione_numero`;
END

-- ----------------------------------------------------
-- procedure aggiungi_indirizzo
-- ----------------------------------------------------

CREATE PROCEDURE `aggiungi_indirizzo` (in var_cf varchar(16), in var_via varchar(45), in
var_civico varchar(5), in var_citta varchar(46))
BEGIN
        INSERT INTO `CRM-db`.`Indirizzo` (`Cliente_cf`, `via`, `civico`, `citta`) VALUES (var_cf,
var_via, var_civico, var_citta);
END

-- ----------------------------------------------------
-- procedure tutti_appuntamenti
-- ----------------------------------------------------

CREATE PROCEDURE `tutti_appuntamenti` (in var_matricola varchar(6))
BEGIN
        set transaction read only;
   set transaction isolation level read committed; #evito letture sporche
```

```
        select `a`.Cliente_cf, `a`.SalaRiunione_sede_citta as citta, `a`.SalaRiunione_sede_via as via,
`a`.SalaRiunione_sede_civico as civico,`a`.SalaRiunione_numero as numero_sala,
DATE_FORMAT(`data`, '%d/%m/%Y') as `data`, DATE_FORMAT(ora_inizio, '%H:%i') as
ora_inizio, DATE_FORMAT(ora_fine, '%H:%i') as ora_fine, `a`.PropostaCommerciale_codice as
codice
    from `Appuntamento` as `a` join `Cliente` on Cliente_cf=cf
    where Cliente_cf in (select cf from `Cliente` where Funzionario_matricola=var_matricola)
        order by DATE_FORMAT(`data`, '%Y/%m/%d'),`ora_inizio`, `SalaRiunione_numero`;
    commit;
END
```

Appendice: Implementazione

Codice SQL per instanziare il database

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema CRM-db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema CRM-db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `CRM-db` DEFAULT CHARACTER SET utf8 ;
USE `CRM-db` ;

-- -----------------------------------------------------
-- Table `CRM-db`.`Utente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Utente` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Utente` (
  `matricola` VARCHAR(6) NOT NULL,
  `nome` VARCHAR(20) NOT NULL,
  `cognome` VARCHAR(20) NOT NULL,
  `password` CHAR(32) NOT NULL,
  `ruolo` ENUM('manager', 'settorecommerciale', 'funzionario') NOT NULL,
  PRIMARY KEY (`matricola`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Sede`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Sede` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Sede` (
  `via` VARCHAR(45) NOT NULL,
  `civico` VARCHAR(5) NOT NULL,
  `citta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`via`, `civico`, `citta`))
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `CRM-db`.`PropostaCommerciale`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`PropostaCommerciale` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`PropostaCommerciale` (
  `codice` VARCHAR(10) NOT NULL,
  `descrizione` VARCHAR(45) NOT NULL,
  `terminata` DATE NULL,
  PRIMARY KEY (`codice`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`SalaRiunione`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`SalaRiunione` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`SalaRiunione` (
  `numero` INT NOT NULL,
  `Sede_via` VARCHAR(45) NOT NULL,
  `Sede_civico` VARCHAR(5) NOT NULL,
  `Sede_citta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`numero`, `Sede_via`, `Sede_civico`, `Sede_citta`),
  INDEX `SalaRiunione_FK_Sede_idx` (`Sede_via` ASC, `Sede_civico` ASC, `Sede_citta` ASC),
  CONSTRAINT `fk_SalaRiunione_Sede`
    FOREIGN KEY (`Sede_via` , `Sede_civico` , `Sede_citta`)
    REFERENCES `CRM-db`.`Sede` (`via` , `civico` , `citta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Cliente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Cliente` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Cliente` (
  `cf` VARCHAR(16) NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `cognome` VARCHAR(45) NOT NULL,
  `nascita` DATE NOT NULL,
  `registrazione` DATE NOT NULL,
  `Funzionario_matricola` VARCHAR(6) NOT NULL,
  PRIMARY KEY (`cf`),
  INDEX `Cliente_FK_Funzionario_idx` (`Funzionario_matricola` ASC),
  CONSTRAINT `fk_Cliente_Funzionario1`
    FOREIGN KEY (`Funzionario_matricola`)
    REFERENCES `CRM-db`.`Utente` (`matricola`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Accettata`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Accettata` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Accettata` (
  `proposta_commerciale` VARCHAR(10) NOT NULL,
  `cliente` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`proposta_commerciale`, `cliente`),
  INDEX `Accettata_FK_Cliente_idx` (`cliente` ASC),
  INDEX `Accettata_FK_Codice_idx` (`proposta_commerciale` ASC),
  CONSTRAINT `fk_PropostaCommerciale_has_Cliente_PropostaCommerciale1`
    FOREIGN KEY (`proposta_commerciale`)
    REFERENCES `CRM-db`.`PropostaCommerciale` (`codice`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PropostaCommerciale_has_Cliente_Cliente1`
    FOREIGN KEY (`cliente`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Telefono`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Telefono` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Telefono` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `numero` INT NOT NULL,
  PRIMARY KEY (`Cliente_cf`, `numero`),
  CONSTRAINT `fk_Telefono_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Mail`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Mail` ;

```

```
CREATE TABLE IF NOT EXISTS `CRM-db`.`Mail` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `mail` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Cliente_cf`, `mail`),
  CONSTRAINT `fk_Mail_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Indirizzo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Indirizzo` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Indirizzo` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `via` VARCHAR(45) NOT NULL,
  `civico` VARCHAR(5) NOT NULL,
  `citta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Cliente_cf`, `via`, `civico`, `citta`),
  CONSTRAINT `fk_Indirizzo_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Fax`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Fax` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Fax` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `fax` INT NOT NULL,
  PRIMARY KEY (`Cliente_cf`, `fax`),
  CONSTRAINT `fk_Fax_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Interazione`
```

```
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Interazione` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Interazione` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `data` DATETIME NOT NULL,
  `nota` VARCHAR(128) NOT NULL,
  PRIMARY KEY (`Cliente_cf`, `data`),
  CONSTRAINT `fk_Interazione_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Appuntamento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Appuntamento` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Appuntamento` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `SalaRiunione_numero` INT NOT NULL,
  `SalaRiunione_sede_via` VARCHAR(45) NOT NULL,
  `SalaRiunione_sede_civico` VARCHAR(5) NOT NULL,
  `SalaRiunione_sede_citta` VARCHAR(45) NOT NULL,
  `data` DATE NOT NULL,
  `ora_inizio` TIME NOT NULL,
  `ora_fine` TIME NOT NULL,
  `PropostaCommerciale_codice` VARCHAR(10) NOT NULL,
  INDEX `Appuntamento_FK_Cliente_idx` (`Cliente_cf` ASC),
  PRIMARY KEY (`SalaRiunione_numero`, `SalaRiunione_sede_via`, `SalaRiunione_sede_civico`,
`SalaRiunione_sede_citta`, `data`, `ora_inizio`),
  INDEX `Appuntamento_FK_Codice_idx` (`PropostaCommerciale_codice` ASC),
  CONSTRAINT `fk_Appuntamento_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Appuntamento_SalaRiunione1`
    FOREIGN KEY (`SalaRiunione_numero` , `SalaRiunione_sede_via` ,
`SalaRiunione_sede_civico` , `SalaRiunione_sede_citta`)
    REFERENCES `CRM-db`.`SalaRiunione` (`numero` , `Sede_via` , `Sede_civico` , `Sede_citta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Appuntamento_PropostaCommerciale1`
    FOREIGN KEY (`PropostaCommerciale_codice`)
    REFERENCES `CRM-db`.`PropostaCommerciale` (`codice`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CRM-db`.`Societa`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `CRM-db`.`Societa` ;

CREATE TABLE IF NOT EXISTS `CRM-db`.`Societa` (
  `Cliente_cf` VARCHAR(16) NOT NULL,
  `ragione_sociale` VARCHAR(45) NOT NULL,
  `partita_iva` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`Cliente_cf`),
  CONSTRAINT `fk_Società_Cliente1`
    FOREIGN KEY (`Cliente_cf`)
    REFERENCES `CRM-db`.`Cliente` (`cf`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE = '';
GRANT USAGE ON *.* TO funzionario;
 DROP USER funzionario;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'funzionario' IDENTIFIED BY 'funzionario';

GRANT EXECUTE ON procedure `CRM-db`.`modifica_nota` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_lista_clienti` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`nuovi_appuntamenti` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`inserisci_proposta_accettata` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`elimina_nota` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_nota` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_appuntamento` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_note` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_fax` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_indirizzo` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_mail` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_accettate` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_societa` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_telefono` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`proposte_attive` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_cliente` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`visualizza_sale` TO 'funzionario';
GRANT SELECT, INSERT ON TABLE `CRM-db`.`Accettata` TO 'funzionario';
GRANT EXECUTE ON procedure `CRM-db`.`tutti_appuntamenti` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Cliente` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Fax` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Indirizzo` TO 'funzionario';
GRANT INSERT, SELECT, UPDATE ON TABLE `CRM-db`.`Interazione` TO 'funzionario';
```

```
GRANT SELECT ON TABLE `CRM-db`.`Mail` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`PropostaCommerciale` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`SalaRiunione` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Sede` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Societa` TO 'funzionario';
GRANT SELECT ON TABLE `CRM-db`.`Telefono` TO 'funzionario';
SET SQL_MODE = '';
GRANT USAGE ON *.* TO commerciale;
 DROP USER commerciale;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'commerciale' IDENTIFIED BY 'commerciale';

GRANT EXECUTE ON procedure `CRM-db`.`inserisci_cliente` TO 'commerciale';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_mail` TO 'commerciale';
GRANT EXECUTE ON procedure `CRM-db`.`inserisci_societa` TO 'commerciale';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_fax` TO 'commerciale';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_telefono` TO 'commerciale';
GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_indirizzo` TO 'commerciale';
GRANT INSERT, SELECT ON TABLE `CRM-db`.`Cliente` TO 'commerciale';
GRANT SELECT, INSERT ON TABLE `CRM-db`.`Fax` TO 'commerciale';
GRANT SELECT, INSERT ON TABLE `CRM-db`.`Indirizzo` TO 'commerciale';
GRANT INSERT ON TABLE `CRM-db`.`Mail` TO 'commerciale';
GRANT INSERT, SELECT ON TABLE `CRM-db`.`Societa` TO 'commerciale';
GRANT INSERT ON TABLE `CRM-db`.`Telefono` TO 'commerciale';
SET SQL_MODE = '';
GRANT USAGE ON *.* TO manager;
 DROP USER manager;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'manager' IDENTIFIED BY 'manager';

GRANT EXECUTE ON procedure `CRM-db`.`aggiungi_funzionario` TO 'manager';
GRANT EXECUTE ON procedure `CRM-db`.`termina_proposta` TO 'manager';
GRANT EXECUTE ON procedure `CRM-db`.`inserisci_proposta` TO 'manager';
GRANT EXECUTE ON procedure `CRM-db`.`proposte_attive` TO 'manager';
GRANT INSERT ON TABLE `CRM-db`.`Utente` TO 'manager';
GRANT SELECT, UPDATE, INSERT ON TABLE `CRM-db`.`PropostaCommerciale` TO 'manager';
SET SQL_MODE = '';
GRANT USAGE ON *.* TO login;
 DROP USER login;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'login' IDENTIFIED BY 'login';

GRANT EXECUTE ON procedure `CRM-db`.`login` TO 'login';
GRANT SELECT ON TABLE `CRM-db`.`Utente` TO 'login';

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Codice del Front-End

- Main

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>

#include "defines.h"


typedef enum {
	MANAGER=1,
	COMMERCIAL,
	OFFICER,
	FAILED_LOGIN
} role_t;


struct configuration conf;

static MYSQL *conn;


static role_t attempt_login(MYSQL *conn, char *username, char *password) {
	MYSQL_STMT *login_procedure;

	MYSQL_BIND param[3]; // Used both for input and output
	int role = 0;

	if(!setup_prepared_stmt(&login_procedure, "call login(?, ?, ?)", conn)) {
		print_stmt_error(login_procedure, "Unable to initialize login statement\n");
		goto err2;
	}
```

```
// Prepare parameters
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param[0].buffer = username;
param[0].buffer_length = strlen(username);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
param[1].buffer = password;
param[1].buffer_length = strlen(password);

param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
param[2].buffer = &role;
param[2].buffer_length = sizeof(role);

if (mysql_stmt_bind_param(login_procedure, param) != 0) { // Note _param
        print_stmt_error(login_procedure, "Could not bind parameters for login");
        goto err;
}

// Run procedure
if (mysql_stmt_execute(login_procedure) != 0) {
        print_stmt_error(login_procedure, "Could not execute login procedure");
        goto err;
}

// Prepare output parameters
memset(param, 0, sizeof(param));
param[0].buffer_type = MYSQL_TYPE_LONG; // OUT
param[0].buffer = &role;
param[0].buffer_length = sizeof(role);

if(mysql_stmt_bind_result(login_procedure, param)) {
```

```
                print_stmt_error(login_procedure, "Could not retrieve output parameter");
                goto err;
        }

        // Retrieve output parameter
        if(mysql_stmt_fetch(login_procedure)) {
                print_stmt_error(login_procedure, "Could not buffer results");
                goto err;
        }

        mysql_stmt_close(login_procedure);
        return role;

    err:
        mysql_stmt_close(login_procedure);
    err2:
        return FAILED_LOGIN;
}

int main(void) {
        role_t role;

        if(!parse_config("users/login.json", &conf)) {
                fprintf(stderr, "Unable to load login configuration\n");
                exit(EXIT_FAILURE);
        }

        conn = mysql_init (NULL);
        if (conn == NULL) {
                fprintf (stderr, "mysql_init() failed (probably out of memory)\n");
                exit(EXIT_FAILURE);
        }
```

```
        if (mysql_real_connect(conn, conf.host, conf.db_username, conf.db_password,
    conf.database, conf.port, NULL, CLIENT_MULTI_STATEMENTS |
    CLIENT_MULTI_RESULTS) == NULL) {
            fprintf (stderr, "mysql_real_connect() failed\n");
            mysql_close (conn);
            exit(EXIT_FAILURE);
        }

        printf("Username: ");
        getInput(128, conf.username, false);
        printf("Password: ");
        getInput(128, conf.password, true);

        role = attempt_login(conn, conf.username, conf.password);

        switch(role) {
            case MANAGER:
                run_as_manager(conn);
                break;

            case COMMERCIAL:
                run_as_commercial(conn);
                break;

            case OFFICER:
                run_as_officer(conn);
                break;

            case FAILED_LOGIN:
                fprintf(stderr, "Invalid credentials\n");
                exit(EXIT_FAILURE);
                break;

            default:
```

```
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                abort();
        }

        printf("Bye!\n");

        mysql_close (conn);
        return 0;
}
```

- Funzionario

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

static void view_customer(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[2];

        char cf[17];
        printf("\nTax code: ");
        getInput(17, cf, false);



        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_cliente(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize customer statement\n", false);
        }

        memset(param, 0, sizeof(param));
```

```
        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = conf.username;
        param[1].buffer_length = strlen(conf.username);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for customer\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve customer\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nCustomer personal data");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);


        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_societa(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize company list statement\n", false);
        }



        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for company list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve company list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nCompany of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_indirizzo(?, ?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize address list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for address list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve address list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of address of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);
```

```
        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_telefono(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize telephone list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for telephone list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve telephone list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of telephone of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_fax(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize fax list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for fax list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve fax list\n");
        }
```

```
        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of fax of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_mail(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize mail list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for mail list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve mail list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of mail of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_note(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize note list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for note list\n", true);
```

```
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve note list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of note of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_accettate(?, ?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize accepted proposal list statement\n", false);
        }

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for accepted proposal list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve accepted proposal list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of accepted proposal of selected customer");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);
    }
```

```
static void add_appointment(MYSQL *conn){
    MYSQL_STMT *prepared_stmt;
    MYSQL_BIND param[9];

    char cf[17];
    char num_sala[3];
    int num;
    char via_sede[46];
    char civico_sede[6];
    char citta_sede[46];
    char codice_proposta[10];
    char anno[5];
    char mese[3];
    char giorno[3];
    char ora[3];
    char minuti[3];
    MYSQL_TIME data, inzio, fine;
    memset(&data,0, sizeof(data));
    memset(&inzio,0, sizeof(inzio));
    memset(&fine,0, sizeof(fine));

    printf("\nTax code: ");
    getInput(17, cf, false);
    printf("Room number: ");
    getInput(3, num_sala, false);
    num=atoi(num_sala);
    printf("Address: ");
    printf("\nStreet: ");
    getInput(46, via_sede, false);
    printf("Street number: ");
    getInput(6, civico_sede, false);
    printf("City: ");
    getInput(46, citta_sede, false);
    printf("Proposal ID: ");
```

```
        getInput(10, codice_proposta, false);
        printf("Date and hour: ");
        printf("\nDay: ");
        getInput(3, giorno, false);
        printf("Month: ");
        getInput(3, mese, false);
        printf("Year: ");
        getInput(5, anno, false);
        printf("Start hour: ");
        getInput(3, ora, false);
        printf("Start minute: ");
        getInput(3, minuti, false);

        data.day=atoi(giorno);
        data.month=atoi(mese);
        data.year=atoi(anno);
        inzio.hour=atoi(ora);
        inzio.minute=atoi(minuti);

        printf("End hour: ");
        getInput(3, ora, false);
        printf("End minute: ");
        getInput(3, minuti, false);

        fine.hour=atoi(ora);
        fine.minute=atoi(minuti);

        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_appuntamento(?, ?, ?, ?, ?, ?, ?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize customers list statement\n", false);
        }

        memset(param, 0, sizeof(param));
```

```
param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = cf;
param[0].buffer_length = strlen(cf);

param[1].buffer_type = MYSQL_TYPE_LONG;
param[1].buffer = &num;
param[1].buffer_length = sizeof(num);

param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
param[2].buffer = via_sede;
param[2].buffer_length = strlen(via_sede);

param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
param[3].buffer = civico_sede;
param[3].buffer_length = strlen(civico_sede);

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param[4].buffer = citta_sede;
param[4].buffer_length = strlen(citta_sede);

param[5].buffer_type = MYSQL_TYPE_DATE;
param[5].buffer = (char *)&data;
param[5].buffer_length = sizeof(data);

param[6].buffer_type = MYSQL_TYPE_TIME;
param[6].buffer = (char *)&inzio;
param[6].buffer_length = sizeof(inzio);

param[7].buffer_type = MYSQL_TYPE_TIME;
param[7].buffer = (char *)&fine;
param[7].buffer_length = sizeof(fine);

param[8].buffer_type = MYSQL_TYPE_VAR_STRING;
```

```
        param[8].buffer = codice_proposta;
        param[8].buffer_length = strlen(codice_proposta);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for add appointment\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve add appointment\n");
        }else{
            printf("Appointment correctly added...\n");
        }

        mysql_stmt_close(prepared_stmt);


    }

    static void view_new_appointments(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];

        if(!setup_prepared_stmt(&prepared_stmt, "call nuovi_appuntamenti(?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize appointments list statement\n", false);
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = conf.username;
        param[0].buffer_length = strlen(conf.username);
```

```
        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for appointments list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve appointments list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of appointments assigned to you");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);


}

static void view_all_appointments(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];

        if(!setup_prepared_stmt(&prepared_stmt, "call tutti_appuntamenti(?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize appointments list statement\n", false);
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = conf.username;
        param[0].buffer_length = strlen(conf.username);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
```

```
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        appointments list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve appointments list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of appointments assigned to you");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);


}

static void delete_note(MYSQL *conn, char cf[17]){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[3];

//      char cf[17];
        char anno[5];
        char mese[3];
        char giorno[3];
        char ora[3];
        char minuti[3];
        MYSQL_TIME data;
        memset(&data,0, sizeof(data));

//      printf("\nCodice fiscale: ");
//      getInput(17, cf, false);
        printf("Data and hour: ");
        printf("\nDay: ");
```

```
        getInput(3, giorno, false);
        printf("Month: ");
        getInput(3, mese, false);
        printf("Year: ");
        getInput(5, anno, false);
        printf("Hour: ");
        getInput(3, ora, false);
        printf("Minutes: ");
        getInput(3, minuti, false);

        data.day=atoi(giorno);
        data.month=atoi(mese);
        data.year=atoi(anno);
        data.hour=atoi(ora);
        data.minute=atoi(minuti);



        if(!setup_prepared_stmt(&prepared_stmt, "call elimina_nota(?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize delete note statement\n", false);
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_DATETIME;
        param[1].buffer = (char*)&data;
        param[1].buffer_length = sizeof(data);


        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
```

```
        param[2].buffer = conf.username;
        param[2].buffer_length = strlen(conf.username);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for delete note\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not delete note\n");
        }else{
            printf("Note delete correctly...\n");
        }

        mysql_stmt_close(prepared_stmt);


    }

    static void view_active_proposals(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;

        if(!setup_prepared_stmt(&prepared_stmt, "call proposte_attive()", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize active proposals list statement\n", false);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve active proposals list\n");
        }

        // Dump the result set
```

```
        dump_result_set(conn, prepared_stmt, "\nActive proposals list");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

5
    }

    static void edit_note(MYSQL *conn, char cf[17]){
        MYSQL_STMT *prepared_stmt;
10      MYSQL_BIND param[4];

    //  char cf[17];
        char nota[129];
        char anno[5];
15      char mese[3];
        char giorno[3];
        char ora[3];
        char minuti[3];
        MYSQL_TIME data;
20      memset(&data,0, sizeof(data));



    //  printf("\nCodice fiscale: ");
25  //  getInput(17, cf, false);
        printf("Data and hour: ");
        printf("\nDay: ");
        getInput(3, giorno, false);
        printf("Month: ");
30      getInput(3, mese, false);
        printf("Year: ");
        getInput(5, anno, false);
        printf("Hour: ");
        getInput(3, ora, false);
```

```
            printf("Minutes: ");
            getInput(3, minuti, false);
            printf("Note: ");
            getInput(129, nota, false);

            data.day=atoi(giorno);
            data.month=atoi(mese);
            data.year=atoi(anno);
            data.hour=atoi(ora);
            data.minute=atoi(minuti);


            if(!setup_prepared_stmt(&prepared_stmt, "call modifica_nota(?, ?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize edit note statement\n", false);
            }
            memset(param, 0, sizeof(param));

            param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
            param[0].buffer = cf;
            param[0].buffer_length = strlen(cf);

            param[1].buffer_type = MYSQL_TYPE_DATETIME;
            param[1].buffer = (char*)&data;
            param[1].buffer_length = sizeof(data);

            param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
            param[2].buffer = nota;
            param[2].buffer_length = strlen(nota);

            param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
            param[3].buffer = conf.username;
            param[3].buffer_length = strlen(conf.username);
```

```
        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for edit note\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not edit note\n");
        }else{
                printf("Change made...\n");
        }

        mysql_stmt_close(prepared_stmt);



}

static void add_note(MYSQL *conn, char cf[17]){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[3];

//      char cf[17];
        char nota[129];

//      printf("\nCodice fiscale: ");
//      getInput(17, cf, false);
        printf("Note: ");
        getInput(129, nota, false);

        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_nota(?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add note statement\n", false);
```

```
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = nota;
        param[1].buffer_length = strlen(nota);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = conf.username;
        param[2].buffer_length = strlen(conf.username);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for add note\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not add note\n");
        }else{
                printf("Note correctly added..\n");
        }

        mysql_stmt_close(prepared_stmt);

}

static void view_meeting_room(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];
```

```
        char citta[45];

        printf("\nCity: ");
        getInput(45, citta, false);

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_sale(?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize meeting rooms list statement\n", false);
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = citta;
        param[0].buffer_length = strlen(citta);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for meeting rooms list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve meeting rooms list\n");
        }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of busy meeting rooms in selected city");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);

    }
```

```
static void new_accepted_proposal(MYSQL *conn){
    MYSQL_STMT *prepared_stmt;
    MYSQL_BIND param[3];

    char codice[11];
    char cf[17];

    printf("\nProposal ID: ");
    getInput(11, codice, false);
    printf("\nTax code: ");
    getInput(17, cf, false);


    if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_proposta_accettata(?, ?, ?)",
conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert
proposal statement\n", false);
    }
    memset(param, 0, sizeof(param));

    param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[0].buffer = codice;
    param[0].buffer_length = strlen(codice);

    param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[1].buffer = cf;
    param[1].buffer_length = strlen(cf);

    param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[2].buffer = conf.username;
    param[2].buffer_length = strlen(conf.username);

    if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for insert proposal\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not insert new proposal\n");
        }else{
            printf("New proposal added correctly...\n");
        }

        mysql_stmt_close(prepared_stmt);

    }

    static void view_customers_list(MYSQL *conn) {
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];

        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_lista_clienti(?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize customers list statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = conf.username;
        param[0].buffer_length = strlen(conf.username);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for customers list\n", true);
```

```
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve customers list\n");
        }


        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of customers assigned to you");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);
    }

    static void notes_managment(MYSQL *conn){

        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[2];

        char options[4] = {'1','2', '3', '4'};
        char op;

        char cf[17];



        printf("\033[2J\033[H");
        printf("*** What should I do for you? ***\n\n");

        printf("\nCustomer tax code: ");
        getInput(17, cf, false);
        if(!setup_prepared_stmt(&prepared_stmt, "call visualizza_note(?, ?)", conn)) {
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize meeting room list statement\n", false);
        }
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = conf.username;
        param[1].buffer_length = strlen(conf.username);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for customers list\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "Could not retrieve customers list\n");
            if(strcmp(mysql_stmt_sqlstate(prepared_stmt), "45002")==0){
                goto next;
            }
            getchar();
            return;
        }
    next:
        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of notes");
        mysql_stmt_next_result(prepared_stmt);
        mysql_stmt_close(prepared_stmt);
```

```
        printf("\n\n1) Insert a note\n");
        printf("2) Edit a note\n");
        printf("3) Delete a note\n");
        printf("4) Return to main menu\n");

        op = multiChoice("Select an option", options, 4);

        switch(op) {
            case '1':
                add_note(conn, cf);
                break;

            case '2':
                edit_note(conn, cf);
                break;

            case '3':
                delete_note(conn, cf);
                break;

            case '4':
                return;

            default:
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                abort();
        }
        getchar();
        return;
    }

    static void view_appointments(MYSQL *conn){
        char options[3] = {'1','2', '3'};
        char op;
```

```
		printf("\033[2J\033[H");
		printf("*** What should I do for you? ***\n\n");
		printf("\n\n1) View new appointments\n");
		printf("2) View all appointments\n");
		printf("3) Return to main menu\n");

		op = multiChoice("Select an option", options, 3);

		switch(op) {
			case '1':
				view_new_appointments(conn);
				break;

			case '2':
				view_all_appointments(conn);
				break;

			case '3':
				break;

			default:
				fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
				abort();
		}
		return;
	}

	void run_as_officer(MYSQL *conn)
	{
		char options[8] = {'1','2', '3', '4', '5', '6', '7', '8'};
		char op;

		printf("Switching to officer role...\n");
```

```
if(!parse_config("users/funzionario.json", &conf)) {
        fprintf(stderr, "Unable to load officer configuration\n");
        exit(EXIT_FAILURE);
}

if(mysql_change_user(conn, conf.db_username, conf.db_password, conf.database)) {
        fprintf(stderr, "mysql_change_user() failed\n");
        exit(EXIT_FAILURE);
}

while(true) {
        printf("\033[2J\033[H");
        printf("*** What should I do for you? ***\n\n");
        printf("1) View customers list\n");
        printf("2) View selected customer\n");
        printf("3) View active proposals\n");
        printf("4) Insert new accepted proposal\n");
        printf("5) Manage notes (Insert/Edit/Delete)\n");
        printf("6) Add appointment\n");
        printf("7) View appointments\n");
        printf("8) Quit\n");

        op = multiChoice("Select an option", options, 8);

        switch(op) {
                case '1':
                        view_customers_list(conn);
                        break;

                case '2':
                        view_customer(conn);
                        break;
```

```
                case '3':
                    view_active_proposals(conn);
                    break;

                case '4':
                    new_accepted_proposal(conn);
                    break;

                case '5':
                    notes_managment(conn);
                    continue;

                case '6':
                    view_meeting_room(conn);
                    add_appointment(conn);
                    break;

                case '7':
                    view_appointments(conn);
                    break;

                case '8':
                    return;

                default:
                    fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__,
        __LINE__);
                    abort();
            }

            getchar();
        }
    }
```

- Membro del settore commerciale

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

static void register_company(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[15];

        // Input for the registration routine
        char cf[17];
        char nome[46];
        char cognome[46];
        char matricola[7];
        char giorno[3];
        char mese[3];
        char anno[5];
        char mail[46];
        char telefono[11];
        char fax[12];
        char via[46];
        char civico[6];
        char citta[46];
        char ragione_sociale[46];
        char partita_iva[12];
        MYSQL_TIME nascita;
        memset(&nascita,0, sizeof(nascita));
        int numerot, numerof;

        // Get the required information
        printf("\nTax code: ");
        getInput(17, cf, false);
```

```
printf("Name: ");
getInput(46, nome, false);
printf("Surname: ");
getInput(46, cognome, false);
printf("Officer ID: ");
getInput(7, matricola, false);
printf("Date of birth:\n");
printf("Day: ");
getInput(3, giorno, false);
printf("Month: ");
getInput(3, mese, false);
printf("Year: ");
getInput(5, anno, false);
printf("Mail: ");
getInput(46, mail, false);
printf("Telephone: ");
getInput(11, telefono, false);
printf("Fax: ");
getInput(12, fax, false);
printf("Address:\n");
printf("Street: ");
getInput(46, via, false);
printf("Street number: ");
getInput(6, civico, false);
printf("City: ");
getInput(46, citta, false);
printf("Legal company mail: ");
getInput(46, ragione_sociale, false);
printf("VAT number: ");
getInput(46, partita_iva, false);

nascita.day=atoi(giorno);
nascita.month=atoi(mese);
nascita.year=atoi(anno);
```

```
        numerot=atoi(telefono);
        numerof=atoi(fax);

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_societa(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize register company statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = nome;
        param[1].buffer_length = strlen(nome);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = cognome;
        param[2].buffer_length = strlen(cognome);

        param[3].buffer_type = MYSQL_TYPE_DATE;
        param[3].buffer = &nascita;
        param[3].buffer_length = sizeof(nascita);

        param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[4].buffer = matricola;
        param[4].buffer_length = strlen(matricola);

        param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
```

```
        param[5].buffer = mail;
        param[5].buffer_length = strlen(mail);

        param[6].buffer_type = MYSQL_TYPE_LONG;
        param[6].buffer = &numerot;
        param[6].buffer_length = sizeof(numerot);

        param[7].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[7].buffer = via;
        param[7].buffer_length = strlen(via);

        param[8].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[8].buffer = civico;
        param[8].buffer_length = strlen(civico);

        param[9].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[9].buffer = citta;
        param[9].buffer_length = strlen(citta);

        param[10].buffer_type = MYSQL_TYPE_LONG;
        param[10].buffer = &numerof;
        param[10].buffer_length = sizeof(numerof);

        param[11].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[11].buffer = ragione_sociale;
        param[11].buffer_length = strlen(ragione_sociale);

        param[12].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[12].buffer = partita_iva;
        param[12].buffer_length = strlen(partita_iva);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
register company\n", true);
```

```
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error (prepared_stmt, "An error occurred while registering company.");
        } else{
            printf("Registration correctly completed...\n");
        }

        mysql_stmt_close(prepared_stmt);

    }

    static void register_customer(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[12];

        // Input for the registration routine
        char cf[17];
        char nome[46];
        char cognome[46];
        char matricola[7];
        char giorno[3];
        char mese[3];
        char anno[5];
        char mail[46];
        char telefono[11];
        char fax[12];
        char via[46];
        char civico[6];
        char citta[46];
        MYSQL_TIME nascita;
        memset(&nascita,0, sizeof(nascita));
```

```
        int numerot, numerof;

        // Get the required information
        printf("\nTax code: ");
 5      getInput(17, cf, false);
        printf("Name: ");
        getInput(46, nome, false);
        printf("Surname: ");
        getInput(46, cognome, false);
10      printf("Officer ID: ");
        getInput(7, matricola, false);
        printf("Date of birth:\n");
        printf("Day: ");
        getInput(3, giorno, false);
15      printf("Month: ");
        getInput(3, mese, false);
        printf("Year: ");
        getInput(5, anno, false);
        printf("Mail: ");
20      getInput(46, mail, false);
        printf("Telephone: ");
        getInput(11, telefono, false);
        printf("Fax: ");
        getInput(12, fax, false);
25      printf("Address:\n");
        printf("Street: ");
        getInput(46, via, false);
        printf("Street number: ");
        getInput(6, civico, false);
30      printf("City: ");
        getInput(46, citta, false);

        nascita.day=atoi(giorno);
        nascita.month=atoi(mese);
```

```
        nascita.year=atoi(anno);
        nascita.day=atoi(giorno);
        nascita.month=atoi(mese);
        nascita.year=atoi(anno);
        numerot=atoi(telefono);
        numerof=atoi(fax);

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_cliente(?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize register
customer statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = nome;
        param[1].buffer_length = strlen(nome);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = cognome;
        param[2].buffer_length = strlen(cognome);

        param[3].buffer_type = MYSQL_TYPE_DATE;
        param[3].buffer = &nascita;
        param[3].buffer_length = sizeof(nascita);

        param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
```

```
        param[4].buffer = matricola;
        param[4].buffer_length = strlen(matricola);

        param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[5].buffer = mail;
        param[5].buffer_length = strlen(mail);

        param[6].buffer_type = MYSQL_TYPE_LONG;
        param[6].buffer = &numerot;
        param[6].buffer_length = sizeof(numerot);

        param[7].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[7].buffer = via;
        param[7].buffer_length = strlen(via);

        param[8].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[8].buffer = civico;
        param[8].buffer_length = strlen(civico);

        param[9].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[9].buffer = citta;
        param[9].buffer_length = strlen(citta);

        param[10].buffer_type = MYSQL_TYPE_LONG;
        param[10].buffer = &numerof;
        param[10].buffer_length = sizeof(numerof);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
register customer\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
```

```
            print_stmt_error (prepared_stmt, "An error occurred while registering the customer.");
        } else{
            printf("Registration correctly completed...\n");
        }

        mysql_stmt_close(prepared_stmt);
    }

    static void add_another_mail(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[2];

        // Input for the registration routine
        char cf[17];
        char mail[46];

        printf("\nTax code: ");
        getInput(17, cf, false);
        printf("Mail: ");
        getInput(46, mail, false);


        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_mail(?, ?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add mail to customer statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
```

```
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = mail;
        param[1].buffer_length = strlen(mail);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for add mail\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while adding mail.");
        }else{
                printf("Operation correctly completed...\n");
        }

        mysql_stmt_close(prepared_stmt);
}

static void add_another_telephone(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[2];

        // Input for the registration routine
        char cf[17];
        char telefono[11];
        int numero;

        printf("\nTax code: ");
        getInput(17, cf, false);
        printf("Telephone number: ");
        getInput(11, telefono, false);
```

```
        numero=atoi(telefono);


        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_telefono(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add telephone to customer statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_LONG;
        param[1].buffer = &numero;
        param[1].buffer_length = sizeof(numero);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for add telephone\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while adding telephone.");
        }else{
                printf("Operation correctly completed...\n");
        }
```

```
        mysql_stmt_close(prepared_stmt);
    }

    static void add_another_fax(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[2];

        // Input for the registration routine
        char cf[17];
        char fax[12];
        int numero;

        printf("\nTax code: ");
        getInput(17, cf, false);
        printf("Fax: ");
        getInput(12, fax, false);
        numero=atoi(fax);

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_fax(?, ?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add fax to customer statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = &numero;
        param[1].buffer_length = sizeof(numero);
```

```
        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for add fax\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while adding fax.");
        }else{
                printf("Operation correctly completed...\n");
        }

        mysql_stmt_close(prepared_stmt);
}

static void add_another_address(MYSQL *conn){
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[4];

        // Input for the registration routine
        char cf[17];
        char via[46];
        char civico[6];
        char citta[46];

        printf("\nTax code: ");
        getInput(17, cf, false);
        printf("Address:\n");
        printf("Street: ");
        getInput(46, via, false);
        printf("Street number: ");
        getInput(6, civico, false);
        printf("City: ");
```

```
        getInput(46, citta, false);


        // Prepare stored procedure call
5       if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_indirizzo(?, ?, ?, ?)", conn))
{
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize add address
to customer statement\n\n", false);
        }
10
        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
15      param[0].buffer = cf;
        param[0].buffer_length = strlen(cf);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = via;
20      param[1].buffer_length = strlen(via);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = civico;
        param[2].buffer_length = strlen(civico);
25
        param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[3].buffer = citta;
        param[3].buffer_length = strlen(citta);

30      if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
add address\n", true);
        }
```

```
        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while adding address.");
        }else{
                printf("Operation correctly completed...\n");
        }

        mysql_stmt_close(prepared_stmt);
}

void run_as_commercial(MYSQL *conn)
{
        char options[7] = {'1','2','3','4','5','6','7'};
        char op;

        printf("Switching to commercial role...\n");

        if(!parse_config("users/commerciale.json", &conf)) {
                fprintf(stderr, "Unable to load commercial configuration\n");
                exit(EXIT_FAILURE);
        }

        if(mysql_change_user(conn, conf.db_username, conf.db_password, conf.database)) {
                fprintf(stderr, "mysql_change_user() failed\n");
                exit(EXIT_FAILURE);
        }

        while(true) {
                printf("\033[2J\033[H");
                printf("*** What should I do for you? ***\n\n");
                printf("1) Register customer\n");
                printf("2) Register company\n");
                printf("3) Add another mail to a customer\n");
                printf("4) Add another number telephone to a customer\n");
```

```
            printf("5) Add another fax to a customer\n");
            printf("6) Add another address to a customer\n");
            printf("7) Quit\n");

            op = multiChoice("Select an option", options, 7);

            switch(op) {
                case '1':
                    register_customer(conn);
                    break;

                case '2':
                    register_company(conn);
                    break;

                case '3':
                    add_another_mail(conn);
                    break;

                case '4':
                    add_another_telephone(conn);
                    break;

                case '5':
                    add_another_fax(conn);
                    break;

                case '6':
                    add_another_address(conn);
                    break;

                case '7':
                    return;
```

```
                default:
                    fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                    abort();
            }

            getchar();
        }
    }
```

- Manager

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"


static void create_proposal(MYSQL *conn) {
    MYSQL_STMT *prepared_stmt;
    MYSQL_BIND param[2];

    // Input for the registration routine
    char codice[11];
    char descrizione[46];

    printf("\nProposal ID: ");
    getInput(10, codice, false);
    printf("Description: ");
    getInput(45, descrizione, false);


    if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_proposta(?, ?)", conn)) {
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize create proposal statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = codice;
        param[0].buffer_length = strlen(codice);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = descrizione;
        param[1].buffer_length = strlen(descrizione);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for create proposal\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error(prepared_stmt, "An error occured while creating new proposal\n");
        }else{
            printf("Proposal created successfully...\n");
        }

        mysql_stmt_close(prepared_stmt);

    }

    static void terminate_proposal(MYSQL *conn) {
        MYSQL_STMT *prepared_stmt;
```

```
        MYSQL_BIND param[2];

        // Input for the registration routine
        char codice[11];
5


        if(!setup_prepared_stmt(&prepared_stmt, "call proposte_attive()", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize active proposals list statement\n", false);
10      }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error(prepared_stmt, "Could not retrieve active proposals list\n");
15      }

        // Dump the result set
        dump_result_set(conn, prepared_stmt, "\nList of active proposals");
        mysql_stmt_next_result(prepared_stmt);
20      mysql_stmt_close(prepared_stmt);

        printf("\nProposal ID: ");
        getInput(10, codice, false);

25      if(!setup_prepared_stmt(&prepared_stmt, "call termina_proposta(?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize marks proposal finished statement\n", false);
        }

30      // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = codice;
```

```
        param[0].buffer_length = strlen(codice);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
5   marks proposal finished\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
10              print_stmt_error(prepared_stmt, "An error occured while marking proposal
    finished\n");
        }else{
                printf("Request completed successfully...\n");
        }
15
        mysql_stmt_close(prepared_stmt);

}

20  static void add_officer(MYSQL *conn)
    {
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[5];

25      // Input for the registration routine
        char matricola[46];
        char nome[46];
        char cognome[46];
        char password[46];
30
        // Get the required information
        printf("\nUser ID: ");
        getInput(46, matricola, false);
        printf("Name: ");
```

```
        getInput(46, nome, false);
        printf("Surname: ");
        getInput(46, cognome, false);
        printf("Password: ");
        getInput(46, password, true);

        // Prepare stored procedure call
        if(!setup_prepared_stmt(&prepared_stmt, "call aggiungi_funzionario(?, ?, ?, ?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize user insertion statement\n", false);
        }

        // Prepare parameters
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = matricola;
        param[0].buffer_length = strlen(matricola);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = nome;
        param[1].buffer_length = strlen(nome);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = cognome;
        param[2].buffer_length = strlen(cognome);

        param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[3].buffer = password;
        param[3].buffer_length = strlen(password);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for user insertion\n", true);
        }

        // Run procedure
        if (mysql_stmt_execute(prepared_stmt) != 0) {
            print_stmt_error (prepared_stmt, "An error occurred while adding the user.");
        } else {
            printf("User correctly added...\n");
        }

        mysql_stmt_close(prepared_stmt);
    }


    void run_as_manager(MYSQL *conn)
    {
        char options[4] = {'1','2', '3', '4'};
        char op;

        printf("Switching to manager role...\n");

        if(!parse_config("users/manager.json", &conf)) {
            fprintf(stderr, "Unable to load manager configuration\n");
            exit(EXIT_FAILURE);
        }

        if(mysql_change_user(conn, conf.db_username, conf.db_password, conf.database)) {
            fprintf(stderr, "mysql_change_user() failed\n");
            exit(EXIT_FAILURE);
        }

        while(true) {
            printf("\033[2J\033[H");
```

```
                printf("*** What should I do for you? ***\n\n");
                printf("1) Create new proposal\n");
                printf("2) Terminate proposal\n");
                printf("3) Add new officer\n");
                printf("4) Quit\n");

                op = multiChoice("Select an option", options, 4);

                switch(op) {
                        case '1':
                                create_proposal(conn);
                                break;

                        case '2':
                                terminate_proposal(conn);
                                break;

                        case '3':
                                add_officer(conn);
                                break;

                        case '4':
                                return;

                        default:
                                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__,
        __LINE__);
                                abort();
                }

                getchar();
        }
}
```

- Inout

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <pthread.h>
#include <signal.h>
#include <stdbool.h>

#include "defines.h"

// Per la gestione dei segnali
static volatile sig_atomic_t signo;
typedef struct sigaction sigaction_t;
static void handler(int s);

char *getInput(unsigned int lung, char *stringa, bool hide)
{
        char c;
        unsigned int i;

        // Dichiara le variabili necessarie ad un possibile mascheramento dell'input
        sigaction_t sa, savealrm, saveint, savehup, savequit, saveterm;
        sigaction_t savetstp, savettin, savettou;
        struct termios term, oterm;

        if(hide) {
                // Svuota il buffer
                (void) fflush(stdout);
```

```
                // Cattura i segnali che altrimenti potrebbero far terminare il programma, lasciando l'utente senza output sulla shell
                sigemptyset(&sa.sa_mask);
                sa.sa_flags = SA_INTERRUPT; // Per non resettare le system call
                sa.sa_handler = handler;
                (void) sigaction(SIGALRM, &sa, &savealrm);
                (void) sigaction(SIGINT, &sa, &saveint);
                (void) sigaction(SIGHUP, &sa, &savehup);
                (void) sigaction(SIGQUIT, &sa, &savequit);
                (void) sigaction(SIGTERM, &sa, &saveterm);
                (void) sigaction(SIGTSTP, &sa, &savetstp);
                (void) sigaction(SIGTTIN, &sa, &savettin);
                (void) sigaction(SIGTTOU, &sa, &savettou);

                // Disattiva l'output su schermo
                if (tcgetattr(fileno(stdin), &oterm) == 0) {
                        (void) memcpy(&term, &oterm, sizeof(struct termios));
                        term.c_lflag &= ~(ECHO|ECHONL);
                        (void) tcsetattr(fileno(stdin), TCSAFLUSH, &term);
                } else {
                        (void) memset(&term, 0, sizeof(struct termios));
                        (void) memset(&oterm, 0, sizeof(struct termios));
                }
        }

        // Acquisisce da tastiera al più lung - 1 caratteri
        for(i = 0; i < lung; i++) {
                (void) fread(&c, sizeof(char), 1, stdin);
                if(c == '\n') {
                        stringa[i] = '\0';
                        break;
                } else
                        stringa[i] = c;
```

```
                // Gestisce gli asterischi
                if(hide) {
                        if(c == '\b') // Backspace
                                (void) write(fileno(stdout), &c, sizeof(char));
                        else
                                (void) write(fileno(stdout), "*", sizeof(char));
                }
        }

        // Controlla che il terminatore di stringa sia stato inserito
        if(i == lung - 1)
                stringa[i] = '\0';

        // Se sono stati digitati più caratteri, svuota il buffer della tastiera
        if(strlen(stringa) >= lung) {
                // Svuota il buffer della tastiera
                do {
                        c = getchar();
                } while (c != '\n');
        }

        if(hide) {
                //L'a capo dopo l'input
                (void) write(fileno(stdout), "\n", 1);

                // Ripristina le impostazioni precedenti dello schermo
                (void) tcsetattr(fileno(stdin), TCSAFLUSH, &oterm);

                // Ripristina la gestione dei segnali
                (void) sigaction(SIGALRM, &savealrm, NULL);
                (void) sigaction(SIGINT, &saveint, NULL);
                (void) sigaction(SIGHUP, &savehup, NULL);
                (void) sigaction(SIGQUIT, &savequit, NULL);
                (void) sigaction(SIGTERM, &saveterm, NULL);
```

```
                (void) sigaction(SIGTSTP, &savetstp, NULL);
                (void) sigaction(SIGTTIN, &savettin, NULL);
                (void) sigaction(SIGTTOU, &savettou, NULL);

                // Se era stato ricevuto un segnale viene rilanciato al processo stesso
                if(signo)
                        (void) raise(signo);
        }

        return stringa;
}

// Per la gestione dei segnali
static void handler(int s) {
        signo = s;
}


bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive)
{

        // I caratteri 'yes' e 'no' devono essere minuscoli
        yes = tolower(yes);
        no = tolower(no);

        // Decide quale delle due lettere mostrare come predefinite
        char s, n;
        if(predef) {
                s = toupper(yes);
                n = no;
        } else {
                s = yes;
                n = toupper(no);
        }
```

```
        // Richiesta della risposta
        while(true) {
            // Mostra la domanda
            printf("%s [%c/%c]: ", domanda, s, n);

            char c;
            getInput(1, &c, false);

            // Controlla quale risposta è stata data
            if(c == '\0') { // getInput() non può restituire '\n'!
                return predef;
            } else if(c == yes) {
                return true;
            } else if(c == no) {
                return false;
            } else if(c == toupper(yes)) {
                if(predef || insensitive) return true;
            } else if(c == toupper(yes)) {
                if(!predef || insensitive) return false;
            }
        }
    }

    char multiChoice(char *domanda, char choices[], int num)
    {

        // Genera la stringa delle possibilità
        char *possib = malloc(2 * num * sizeof(char));
        int i, j = 0;
        for(i = 0; i < num; i++) {
            possib[j++] = choices[i];
            possib[j++] = '/';
        }
```

```
        possib[j-1] = '\0'; // Per eliminare l'ultima '/'

        // Chiede la risposta
        while(true) {
            // Mostra la domanda
            printf("%s [%s]: ", domanda, possib);

            char c;
            getInput(1, &c, false);

            // Controlla se è un carattere valido
            for(i = 0; i < num; i++) {
                if(c == choices[i])
                    return c;
            }
        }
    }
```

- Parse

```
    #include <stddef.h>
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>

    #include "defines.h"

    #define BUFF_SIZE 4096

    // The final config struct will point into this
    static char config[BUFF_SIZE];

    /**
     * JSON type identifier. Basic types are:
     *      o Object
```

```
 *      o Array
 *      o String
 *      o Other primitive: number, boolean (true/false) or null
 */
typedef enum {
        JSMN_UNDEFINED = 0,
        JSMN_OBJECT = 1,
        JSMN_ARRAY = 2,
        JSMN_STRING = 3,
        JSMN_PRIMITIVE = 4
} jsmntype_t;

enum jsmnerr {
        /* Not enough tokens were provided */
        JSMN_ERROR_NOMEM = -1,
        /* Invalid character inside JSON string */
        JSMN_ERROR_INVAL = -2,
        /* The string is not a full JSON packet, more bytes expected */
        JSMN_ERROR_PART = -3
};

/**
 * JSON token description.
 * type         type (object, array, string etc.)
 * start start position in JSON data string
 * end          end position in JSON data string
 */
typedef struct {
        jsmntype_t type;
        int start;
        int end;
        int size;
#ifdef JSMN_PARENT_LINKS
        int parent;
```

```
#endif
} jsmntok_t;

/**
 * JSON parser. Contains an array of token blocks available. Also stores
 * the string being parsed now and current position in that string
 */
typedef struct {
	unsigned int pos; /* offset in the JSON string */
	unsigned int toknext; /* next token to allocate */
	int toksuper; /* superior token node, e.g parent object or array */
} jsmn_parser;

/**
 * Allocates a fresh unused token from the token pool.
 */
static jsmntok_t *jsmn_alloc_token(jsmn_parser *parser, jsmntok_t *tokens, size_t num_tokens) {
	jsmntok_t *tok;
	if (parser->toknext >= num_tokens) {
		return NULL;
	}
	tok = &tokens[parser->toknext++];
	tok->start = tok->end = -1;
	tok->size = 0;
#ifdef JSMN_PARENT_LINKS
	tok->parent = -1;
#endif
	return tok;
}

/**
 * Fills token type and boundaries.
 */
```

```
static void jsmn_fill_token(jsmntok_t *token, jsmntype_t type,
                            int start, int end) {
        token->type = type;
        token->start = start;
        token->end = end;
        token->size = 0;
}

/**
 * Fills next available token with JSON primitive.
 */
static int jsmn_parse_primitive(jsmn_parser *parser, const char *js,
                size_t len, jsmntok_t *tokens, size_t num_tokens) {
        jsmntok_t *token;
        int start;

        start = parser->pos;

        for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
                switch (js[parser->pos]) {
#ifndef JSMN_STRICT
                        /* In strict mode primitive must be followed by "," or "}" or "]" */
                        case ':':
#endif
                        case '\t' : case '\r' : case '\n' : case ' ' :
                        case ','  : case ']'  : case '}' :
                                goto found;
                }
                if (js[parser->pos] < 32 || js[parser->pos] >= 127) {
                        parser->pos = start;
                        return JSMN_ERROR_INVAL;
                }
        }
#ifdef JSMN_STRICT
```

```
		/* In strict mode primitive must be followed by a comma/object/array */
		parser->pos = start;
		return JSMN_ERROR_PART;
	#endif

	found:
		if (tokens == NULL) {
			parser->pos--;
			return 0;
		}
		token = jsmn_alloc_token(parser, tokens, num_tokens);
		if (token == NULL) {
			parser->pos = start;
			return JSMN_ERROR_NOMEM;
		}
		jsmn_fill_token(token, JSMN_PRIMITIVE, start, parser->pos);
	#ifdef JSMN_PARENT_LINKS
		token->parent = parser->toksuper;
	#endif
		parser->pos--;
		return 0;
	}

	/**
	 * Fills next token with JSON string.
	 */
	static int jsmn_parse_string(jsmn_parser *parser, const char *js,
			size_t len, jsmntok_t *tokens, size_t num_tokens) {
		jsmntok_t *token;

		int start = parser->pos;

		parser->pos++;
```

```
        /* Skip starting quote */
        for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
                char c = js[parser->pos];

 5              /* Quote: end of string */
                if (c == '\"') {
                        if (tokens == NULL) {
                                return 0;
                        }
10                      token = jsmn_alloc_token(parser, tokens, num_tokens);
                        if (token == NULL) {
                                parser->pos = start;
                                return JSMN_ERROR_NOMEM;
                        }
15                      jsmn_fill_token(token, JSMN_STRING, start+1, parser->pos);
#ifdef JSMN_PARENT_LINKS
                        token->parent = parser->toksuper;
#endif
                        return 0;
20              }

                /* Backslash: Quoted symbol expected */
                if (c == '\\' && parser->pos + 1 < len) {
                        int i;
25                      parser->pos++;
                        switch (js[parser->pos]) {
                                /* Allowed escaped symbols */
                                case '\"': case '/' : case '\\' : case 'b' :
                                case 'f' : case 'r' : case 'n'  : case 't' :
30                                      break;
                                /* Allows escaped symbol \uXXXX */
                                case 'u':
                                        parser->pos++;
```

```
                                        for(i = 0; i < 4 && parser->pos < len && js[parser->pos] != '\0'; i++) {
                                                /* If it isn't a hex character we have an error */
                                                if(!((js[parser->pos] >= 48 && js[parser->pos] <= 57) || /* 0-9 */
                                                                (js[parser->pos] >= 65 && js[parser->pos] <= 70) || /* A-F */
                                                                (js[parser->pos] >= 97 && js[parser->pos] <= 102))) { /* a-f */
                                                        parser->pos = start;
                                                        return JSMN_ERROR_INVAL;
                                                }
                                                parser->pos++;
                                        }
                                        parser->pos--;
                                        break;
                                /* Unexpected symbol */
                                default:
                                        parser->pos = start;
                                        return JSMN_ERROR_INVAL;
                        }
                }
        }
        parser->pos = start;
        return JSMN_ERROR_PART;
}

/**
 * Parse JSON string and fill tokens.
 */
static int jsmn_parse(jsmn_parser *parser, const char *js, size_t len, jsmntok_t *tokens, unsigned int num_tokens) {
        int r;
        int i;
```

```
	jsmntok_t *token;
	int count = parser->toknext;

	for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
		char c;
		jsmntype_t type;

		c = js[parser->pos];
		switch (c) {
			case '{': case '[':
				count++;
				if (tokens == NULL) {
					break;
				}
				token = jsmn_alloc_token(parser, tokens, num_tokens);
				if (token == NULL)
					return JSMN_ERROR_NOMEM;
				if (parser->toksuper != -1) {
					tokens[parser->toksuper].size++;
#ifdef JSMN_PARENT_LINKS
					token->parent = parser->toksuper;
#endif
				}
				token->type = (c == '{' ? JSMN_OBJECT : JSMN_ARRAY);
				token->start = parser->pos;
				parser->toksuper = parser->toknext - 1;
				break;
			case '}': case ']':
				if (tokens == NULL)
					break;
				type = (c == '}' ? JSMN_OBJECT : JSMN_ARRAY);
#ifdef JSMN_PARENT_LINKS
				if (parser->toknext < 1) {
					return JSMN_ERROR_INVAL;
```

```
                }
                token = &tokens[parser->toknext - 1];
                for (;;) {
                        if (token->start != -1 && token->end == -1) {
                                if (token->type != type) {
                                        return JSMN_ERROR_INVAL;
                                }
                                token->end = parser->pos + 1;
                                parser->toksuper = token->parent;
                                break;
                        }
                        if (token->parent == -1) {
                                if(token->type != type || parser->toksuper == -1)
        {
                                        return JSMN_ERROR_INVAL;
                                }
                                break;
                        }
                        token = &tokens[token->parent];
                }
        #else
                for (i = parser->toknext - 1; i >= 0; i--) {
                        token = &tokens[i];
                        if (token->start != -1 && token->end == -1) {
                                if (token->type != type) {
                                        return JSMN_ERROR_INVAL;
                                }
                                parser->toksuper = -1;
                                token->end = parser->pos + 1;
                                break;
                        }
                }
                /* Error if unmatched closing bracket */
                if (i == -1) return JSMN_ERROR_INVAL;
```

```
                        for (; i >= 0; i--) {
                            token = &tokens[i];
                            if (token->start != -1 && token->end == -1) {
                                parser->toksuper = i;
                                break;
                            }
                        }
#endif
                        break;
                    case '\"':
                        r = jsmn_parse_string(parser, js, len, tokens, num_tokens);
                        if (r < 0) return r;
                        count++;
                        if (parser->toksuper != -1 && tokens != NULL)
                            tokens[parser->toksuper].size++;
                        break;
                    case '\t' : case '\r' : case '\n' : case ' ':
                        break;
                    case ':':
                        parser->toksuper = parser->toknext - 1;
                        break;
                    case ',':
                        if (tokens != NULL && parser->toksuper != -1 &&
                                tokens[parser->toksuper].type != JSMN_ARRAY &&
                                tokens[parser->toksuper].type != JSMN_OBJECT) {
#ifdef JSMN_PARENT_LINKS
                            parser->toksuper = tokens[parser->toksuper].parent;
#else
                            for (i = parser->toknext - 1; i >= 0; i--) {
                                if (tokens[i].type == JSMN_ARRAY || tokens[i].type == JSMN_OBJECT) {
```

```
                                                    if (tokens[i].start != -1 && tokens[i].end
        == -1) {
                                                        parser->toksuper = i;
                                                        break;
5                                                   }
                                                }
                                            }
        #endif
                                    }
10                                  break;
        #ifdef JSMN_STRICT
                                /* In strict mode primitives are: numbers and booleans */
                                case '-': case '0': case '1' : case '2': case '3' : case '4':
                                case '5': case '6': case '7' : case '8': case '9':
15                              case 't': case 'f': case 'n' :
                                    /* And they must not be keys of the object */
                                    if (tokens != NULL && parser->toksuper != -1) {
                                        jsmntok_t *t = &tokens[parser->toksuper];
                                        if (t->type == JSMN_OBJECT ||
20                                                  (t->type == JSMN_STRING && t->size
        != 0)) {
                                            return JSMN_ERROR_INVAL;
                                        }
                                    }
25      #else
                                /* In non-strict mode every unquoted value is a primitive */
                                default:
        #endif
                                    r = jsmn_parse_primitive(parser, js, len, tokens, num_tokens);
30                                  if (r < 0) return r;
                                    count++;
                                    if (parser->toksuper != -1 && tokens != NULL)
                                        tokens[parser->toksuper].size++;
                                    break;
```

```
#ifdef JSMN_STRICT
                        /* Unexpected char in strict mode */
                        default:
                                return JSMN_ERROR_INVAL;
#endif
                }
        }

        if (tokens != NULL) {
                for (i = parser->toknext - 1; i >= 0; i--) {
                        /* Unmatched opened object or array */
                        if (tokens[i].start != -1 && tokens[i].end == -1) {
                                return JSMN_ERROR_PART;
                        }
                }
        }

        return count;
}

/**
 * Creates a new parser based over a given  buffer with an array of tokens
 * available.
 */
static void jsmn_init(jsmn_parser *parser) {
        parser->pos = 0;
        parser->toknext = 0;
        parser->toksuper = -1;
}

static int jsoneq(const char *json, jsmntok_t *tok, const char *s)
{
        if (tok->type == JSMN_STRING
```

```
            && (int) strlen(s) == tok->end - tok->start
            && strncmp(json + tok->start, s, tok->end - tok->start) == 0) {
                return 0;
        }
        return -1;
}

static size_t load_file(char *filename)
{
        FILE *f = fopen(filename, "rb");
        if(f == NULL) {
                fprintf(stderr, "Unable to open file %s\n", filename);
                exit(1);
        }

        fseek(f, 0, SEEK_END);
        size_t fsize = ftell(f);
        fseek(f, 0, SEEK_SET);  //same as rewind(f);

        if(fsize >= BUFF_SIZE) {
                fprintf(stderr, "Configuration file too large\n");
                abort();
        }

        fread(config, fsize, 1, f);
        fclose(f);

        config[fsize] = 0;
        return fsize;
}

int parse_config(char *path, struct configuration *conf)
{
        int i;
```

```
int r;
jsmn_parser p;
jsmntok_t t[128]; /* We expect no more than 128 tokens */

load_file(path);

jsmn_init(&p);
r = jsmn_parse(&p, config, strlen(config), t, sizeof(t)/sizeof(t[0]));
if (r < 0) {
        printf("Failed to parse JSON: %d\n", r);
        return 0;
}

/* Assume the top-level element is an object */
if (r < 1 || t[0].type != JSMN_OBJECT) {
        printf("Object expected\n");
        return 0;
}

/* Loop over all keys of the root object */
for (i = 1; i < r; i++) {
        if (jsoneq(config, &t[i], "host") == 0) {
                /* We may use strndup() to fetch string value */
                conf->host = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
                i++;
        } else if (jsoneq(config, &t[i], "username") == 0) {
                conf->db_username = strndup(config + t[i+1].start, t[i+1].end-
t[i+1].start);
                i++;
        } else if (jsoneq(config, &t[i], "password") == 0) {
                conf->db_password = strndup(config + t[i+1].start, t[i+1].end-
t[i+1].start);
                i++;
        } else if (jsoneq(config, &t[i], "port") == 0) {
```

```
                conf->port = strtol(config + t[i+1].start, NULL, 10);
                i++;
            } else if (jsoneq(config, &t[i], "database") == 0) {
                conf->database = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
                i++;
            } else {
                printf("Unexpected key: %.*s\n", t[i].end-t[i].start, config + t[i].start);
            }
        }
        return 1;
    }
```

- Utils

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>

    #include "defines.h"

    void print_stmt_error (MYSQL_STMT *stmt, char *message)
    {
        fprintf (stderr, "%s\n", message);
        if (stmt != NULL) {
            fprintf (stderr, "Error %u (%s): %s\n",
                mysql_stmt_errno (stmt),
                mysql_stmt_sqlstate(stmt),
                mysql_stmt_error (stmt));
        }
    }


    void print_error(MYSQL *conn, char *message)
    {
        fprintf (stderr, "%s\n", message);
```

```
        if (conn != NULL) {
            #if MYSQL_VERSION_ID >= 40101
            fprintf (stderr, "Error %u (%s): %s\n",
            mysql_errno (conn), mysql_sqlstate(conn), mysql_error (conn));
            #else
            fprintf (stderr, "Error %u: %s\n",
            mysql_errno (conn), mysql_error (conn));
            #endif
        }
    }

    bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn)
    {
        my_bool update_length = true;

        *stmt = mysql_stmt_init(conn);
        if (*stmt == NULL)
        {
            print_error(conn, "Could not initialize statement handler");
            return false;
        }

        if (mysql_stmt_prepare (*stmt, statement, strlen(statement)) != 0) {
            print_stmt_error(*stmt, "Could not prepare statement");
            return false;
        }

        mysql_stmt_attr_set(*stmt,          STMT_ATTR_UPDATE_MAX_LENGTH,
    &update_length);

        return true;
    }

    void finish_with_error(MYSQL *conn, char *message)
```

```
{
        print_error(conn, message);
        mysql_close(conn);
        exit(EXIT_FAILURE);
}

void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message, bool close_stmt)
{
        print_stmt_error(stmt, message);
        if(close_stmt)  mysql_stmt_close(stmt);
        mysql_close(conn);
        exit(EXIT_FAILURE);
}

static void print_dashes(MYSQL_RES *res_set)
{
        MYSQL_FIELD *field;
        unsigned int i, j;

        mysql_field_seek(res_set, 0);
        putchar('+');
        for (i = 0; i < mysql_num_fields(res_set); i++) {
                field = mysql_fetch_field(res_set);
                for (j = 0; j < field->max_length + 2; j++)
                        putchar('-');
                putchar('+');
        }
        putchar('\n');
}

static void dump_result_set_header(MYSQL_RES *res_set)
{
        MYSQL_FIELD *field;
```

```
        unsigned long col_len;
        unsigned int i;

        /* determine column display widths -- requires result set to be */
        /* generated with mysql_store_result(), not mysql_use_result() */

        mysql_field_seek (res_set, 0);

        for (i = 0; i < mysql_num_fields (res_set); i++) {
                field = mysql_fetch_field (res_set);
                col_len = strlen(field->name);

                if (col_len < field->max_length)
                        col_len = field->max_length;
                if (col_len < 4 && !IS_NOT_NULL(field->flags))
                        col_len = 4; /* 4 = length of the word "NULL" */
                field->max_length = col_len; /* reset column info */
        }

        print_dashes(res_set);
        putchar('|');
        mysql_field_seek (res_set, 0);
        for (i = 0; i < mysql_num_fields(res_set); i++) {
                field = mysql_fetch_field(res_set);
                printf(" %-*s |", (int)field->max_length, field->name);
        }
        putchar('\n');

        print_dashes(res_set);
}

void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title)
{
        int i;
```

```
int status;
int num_fields;       /* number of columns in result */
MYSQL_FIELD *fields;  /* for result set metadata */
MYSQL_BIND *rs_bind;  /* for output buffers */
MYSQL_RES *rs_metadata;
MYSQL_TIME *date;
size_t attr_size;

/* Prefetch the whole result set. This in conjunction with
 * STMT_ATTR_UPDATE_MAX_LENGTH set in `setup_prepared_stmt`
 * updates the result set metadata which are fetched in this
 * function, to allow to compute the actual max length of
 * the columns.
 */
if (mysql_stmt_store_result(stmt)) {
        fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
        fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
        exit(0);
}

/* the column count is > 0 if there is a result set */
/* 0 if the result is only the final status packet */
num_fields = mysql_stmt_field_count(stmt);

if (num_fields > 0) {
        /* there is a result set to fetch */
        printf("%s\n", title);

        if((rs_metadata = mysql_stmt_result_metadata(stmt)) == NULL) {
                finish_with_stmt_error(conn, stmt, "Unable to retrieve result metadata\n", true);
        }

        dump_result_set_header(rs_metadata);
```

```
fields = mysql_fetch_fields(rs_metadata);

rs_bind = (MYSQL_BIND *)malloc(sizeof (MYSQL_BIND) * num_fields);
if (!rs_bind) {
    finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n",
true);
}
memset(rs_bind, 0, sizeof (MYSQL_BIND) * num_fields);

/* set up and bind result set output buffers */
for (i = 0; i < num_fields; ++i) {

    // Properly size the parameter buffer
    switch(fields[i].type) {
        case MYSQL_TYPE_DATE:
        case MYSQL_TYPE_TIMESTAMP:
        case MYSQL_TYPE_DATETIME:
        case MYSQL_TYPE_TIME:
            attr_size = sizeof(MYSQL_TIME);
            break;
        case MYSQL_TYPE_FLOAT:
            attr_size = sizeof(float);
            break;
        case MYSQL_TYPE_DOUBLE:
            attr_size = sizeof(double);
            break;
        case MYSQL_TYPE_TINY:
            attr_size = sizeof(signed char);
            break;
        case MYSQL_TYPE_SHORT:
        case MYSQL_TYPE_YEAR:
            attr_size = sizeof(short int);
            break;
```

```
                        case MYSQL_TYPE_LONG:
                        case MYSQL_TYPE_INT24:
                                attr_size = sizeof(int);
                                break;
                        case MYSQL_TYPE_LONGLONG:
                                attr_size = sizeof(int);
                                break;
                        default:
                                attr_size = fields[i].max_length;
                                break;
                }

                // Setup the binding for the current parameter
                rs_bind[i].buffer_type = fields[i].type;
                rs_bind[i].buffer = malloc(attr_size + 1);
                rs_bind[i].buffer_length = attr_size + 1;

                if(rs_bind[i].buffer == NULL) {
                        finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n", true);
                }
        }

        if(mysql_stmt_bind_result(stmt, rs_bind)) {
                finish_with_stmt_error(conn, stmt, "Unable to bind output parameters\n", true);
        }

        /* fetch and display result set rows */
        while (true) {
                status = mysql_stmt_fetch(stmt);

                if (status == 1 || status == MYSQL_NO_DATA)
                        break;
```

```
                putchar('|');

                for (i = 0; i < num_fields; i++) {
5
                        if (rs_bind[i].is_null_value) {
                                printf (" %-*s |", (int)fields[i].max_length, "NULL");
                                continue;
                        }
10
                        switch (rs_bind[i].buffer_type) {

                                case MYSQL_TYPE_VAR_STRING:
                                case MYSQL_TYPE_DATETIME:
15                                      printf(" %-*s |", (int)fields[i].max_length,
        (char*)rs_bind[i].buffer);
                                        break;

                                case MYSQL_TYPE_DATE:
20                              case MYSQL_TYPE_TIMESTAMP:
                                        date = (MYSQL_TIME *)rs_bind[i].buffer;
                                        printf(" %d-%02d-%02d |", date->year, date-
        >month, date->day);
                                        break;
25
                                case MYSQL_TYPE_STRING:
                                        printf(" %-*s |", (int)fields[i].max_length, (char
        *)rs_bind[i].buffer);
                                        break;
30
                                case MYSQL_TYPE_FLOAT:
                                case MYSQL_TYPE_DOUBLE:
                                        printf(" %.02f |", *(float *)rs_bind[i].buffer);
                                        break;
```

```
                                        case MYSQL_TYPE_LONG:
                                        case MYSQL_TYPE_SHORT:
                                        case MYSQL_TYPE_TINY:
5                                               printf(" %-*d |", (int)fields[i].max_length, *(int
    *)rs_bind[i].buffer);
                                                break;

                                        case MYSQL_TYPE_NEWDECIMAL:
10                                              printf(" %-*.02lf |", (int)fields[i].max_length,
    *(float*) rs_bind[i].buffer);
                                                break;

                                        default:
15                                        printf("ERROR: Unhandled type (%d)\n",
    rs_bind[i].buffer_type);
                                          abort();
                                }
                        }
20                      putchar('\n');
                        print_dashes(rs_metadata);
                }

                mysql_free_result(rs_metadata); /* free metadata */
25
                /* free output buffers */
                for (i = 0; i < num_fields; i++) {
                        free(rs_bind[i].buffer);
                }
30              free(rs_bind);
        }
}
```

- Defines.h

```
#pragma once

#include <stdbool.h>
#include <mysql.h>

struct configuration {
	char *host;
	char *db_username;
	char *db_password;
	unsigned int port;
	char *database;

	char username[128];
	char password[128];
};

extern struct configuration conf;

extern int parse_config(char *path, struct configuration *conf);
extern char *getInput(unsigned int lung, char *stringa, bool hide);
extern bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive);
extern char multiChoice(char *domanda, char choices[], int num);
extern void print_error (MYSQL *conn, char *message);
extern void print_stmt_error (MYSQL_STMT *stmt, char *message);
extern void finish_with_error(MYSQL *conn, char *message);
extern void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message,
bool close_stmt);
extern bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn);
extern void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title);
extern void run_as_commercial(MYSQL *conn);
extern void run_as_officer(MYSQL *conn);
extern void run_as_manager(MYSQL *conn);
```