|  |  |
| --- | --- |
|  |  |

Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

BACHECA ELETTRONICA

DI ANNUNCI

0267412

Anuar Elio Magliari

**Indice**

[1. Descrizione del Minimondo 3](#_Toc606296459)

[2. Analisi dei Requisiti 4](#_Toc1289394997)

[3. Progettazione concettuale 5](#_Toc2081466291)

[4. Progettazione logica 6](#_Toc2147004904)

[5. Progettazione fisica 8](#_Toc518560220)

[Appendice: Implementazione 9](#_Toc403811585)

L’assegnazione della tesina può essere effettuata online, visitando il sito <https://www.pellegrini.tk/progetti/> ed inserendo i propri dati. Per qualsiasi problema, contattare il docente via email all’indirizzo [a.pellegrini@ing.uniroma2.it](mailto:a.pellegrini@ing.uniroma2.it)

**NOTA:**

**In caso di necessità c’è un backup del progetto nel mio account github**

**https://github.com/Monello1299/Basi-di-Dati**

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  34  35  36  37 | Si vuole realizzare un sistema informativo per la gestione di una bacheca elettronica di annunci. Tale bacheca permette agli utenti del sistema di inserire annunci per la vendita di materiale usato, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell’oggetto, o di inserire domande (in maniera pubblica) sull’oggetto.  Un utente del sistema si registra scegliendo un username univoco, inserendo tutte le sue informazioni anagrafiche, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di recapiti (telefono, cellulare, email) indicandone uno come mezzo di comunicazione preferito, ed inserendo i dati relativi alla sua carta di credito. I dati della carta di credito non sono obbligatori.  I gestori del servizio possono creare una gerarchia di categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell’oggetto. Eventualmente, può decidere di caricare una foto dell’oggetto. Per creare un annuncio, un utente deve necessariamente aver inserito i dati della sua carta di credito. Quando un oggetto inserito in bacheca è stato venduto, l’utente lo indica come tale e questo non viene più visualizzato nella bacheca pubblica.  Un utente del sistema, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all’utente che ha inserito l’annuncio. Similmente, un utente può “seguire” uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserita una nuova nota).  In generale, un utente può:   * Inserire/rimuovere nuovi annunci * Modificare le sue informazioni anagrafiche * Seguire annunci * Mostrare gli annunci che sta seguendo, visualizzando un’indicazione legata al fatto se uno degli annunci che sta seguendo è stato modificato (un oggetto segnato come venduto o rimosso compare comunque nell’elenco degli annunci seguiti dagli utenti, portando l’indicazione del suo stato) * Inviare messaggi agli altri utenti e mostrare lo storico delle sue conversazioni, anche con la possibilità di rispondere ad una conversazione specifica * Inserire commenti agli annunci ancora attivi   I gestori del servizio prendono una percentuale su ciascun oggetto indicato come venduto. Per questo motivo, essi possono generare un report indicante per ciascun utente del sistema quanti annunci sono stati contrassegnati come venduti. Il sistema calcola un percentuale  pari al 3% della somma degli importi di tali oggetti, nel caso in cui la percentuale associata non sia già stata riscossa. Il report riporta anche le informazioni sulla carta di credito dell’utente, al fine di permettere la riscossione della percentuale. |
|  |  |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |
| --- | --- | --- | --- |
| **Linea** | **Termine** | **Nuovo termine** | **Motivo correzione** |
| 2 | Utente del sistema | Utente | Un utente che interagisce con il sistema è anche utente di sistema |
| 4 | Domanda | Commento | Una domanda è identificato come un commento |
| 10 | Gestore del sistema | Amministratore | Il termine “sistema” è ridondante |
| 10 | Gerarchia di categorie | Categoria | Il termine “gerarchia” è ridondante, in quanto una suddivisione degli annunci in categorie è implicitamente una gerarchia |
| 24 | Mostrare | Vedere | Un utente non può mostrare un annuncio, bensì è il sistema a farlo |
| 28 | Mostrare | Vedere | Un utente non può mostrare lo storico delle conversazioni, bensì è il sistema a farlo |

### Specifica disambiguata

|  |
| --- |
| Si vuole realizzare un sistema informativo per la gestione di una bacheca elettronica di annunci. Tale bacheca permette agli utenti di inserire annunci per la vendita di materiale usato, di scambiare messaggi tra di loro (in maniera privata) per accordarsi sulla vendita/consegna dell’oggetto, o di inserire commenti (in maniera pubblica) sull’oggetto.  Un utente si registra scegliendo un username univoco, inserendo tutte le sue informazioni anagrafiche, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di recapiti (telefono, cellulare, email) indicandone uno come mezzo di comunicazione preferito, ed inserendo i dati relativi alla sua carta di credito. I dati della carta di credito non sono obbligatori.  Gli amministratori possono creare delle categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell’oggetto. Eventualmente, può decidere di caricare una foto dell’oggetto. Per creare un annuncio, un utente deve necessariamente aver inserito i dati della sua carta di credito. Quando un oggetto inserito in bacheca è stato venduto, l’utente lo indica come tale e questo non viene più visualizzato nella bacheca pubblica.  Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all’utente che ha inserito l’annuncio. Similmente, un utente può seguire uno degli annunci, venendo così informato ogni volta che su questo viene effettuata una modifica (ad esempio, viene inserita una nuova nota).  In generale, un utente può:   * Inserire/rimuovere nuovi annunci * Modificare le sue informazioni anagrafiche * Seguire annunci * Vedere gli annunci che ha aggiunto tra i preferiti, visualizzando un’indicazione legata al fatto se uno degli annunci che sta seguendo è stato modificato (un oggetto segnato come venduto o rimosso compare comunque nell’elenco degli annunci aggiunti dagli utenti, portando l’indicazione del suo stato) * Inviare messaggi agli altri utenti e vedere lo storico delle sue conversazioni, anche con la possibilità di rispondere ad una conversazione specifica * Inserire commenti agli annunci ancora attivi.   Gli amministratori prendono una percentuale su ciascun oggetto indicato come venduto. Per questo motivo, essi possono generare un report indicante per ciascun utente quanti annunci sono stati contrassegnati come venduti. Il sistema calcola un percentuale pari al 3% della somma degli importi di tali oggetti, nel caso in cui la percentuale associata non sia già stata riscossa. Il report riporta anche le informazioni sulla carta di credito dell’utente, al fine di permettere la riscossione della percentuale. |

## Glossario dei Termini

Realizzare un dizionario dei termini, compilando la tabella qui sotto, a partire dalle specifiche precedentemente disambiguate

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Utente | Persona che si interfaccia con il database |  |  |
| Annuncio | Un inserzione nel sistema riguardante un oggetto | Oggetto | Utente, Categoria |
| Categoria | Permette di distingue la natura dell’annuncio da altri tipi di annunci |  | Annuncio |
| Messaggio | Tipo di comunicazione indiretta tra utenti |  | Conversazione |
| Report | Fattura riguardante gli annunci classificati come venduti per un utente |  | Utente |
| Conversazione | Interazione tramite messaggi tra utenti |  | Utente, Messaggio |
| Storico Conversazione | Traccia tutte le conversazioni dell’utente |  | Utente, Conversazione |
| Informazione anagrafica | Contiene tutti i dati identificativi necessari dell’utente |  | Utente |
| Carta di Credito | Contiene tutti i dati relativi alla carta di credito |  | Utente |

## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| **Annuncio** |
| Un utente, per creare un annuncio, seleziona una categoria e scrive una descrizione dell’oggetto. Eventualmente, può decidere di caricare una foto dell’oggetto. Un annuncio inserito in bacheca classificato come venduto non viene più visualizzato nella bacheca pubblica. Un utente può “seguire” gli annunci. |
| **Utente** |
| Un utente si registra scegliendo un username univoco, inserendo tutte le sue informazioni anagrafiche, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di recapiti (telefono, cellulare, email) indicandone uno come mezzo di comunicazione preferito, ed inserendo i dati relativi alla sua carta di credito. Per creare un annuncio, un utente deve necessariamente aver inserito i dati della sua carta di credito. Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all’utente che ha inserito l’annuncio. Similmente, un utente può seguire uno degli annunci. Il report riporta anche le informazioni sulla carta di credito dell’utente. |
| **Categoria** |
| Gli amministratori possono creare delle categorie per gli annunci. Un utente, per creare un annuncio, seleziona una categoria. |
| **Messaggio** |
| Un utente, una volta letto e scelto un annuncio, può decidere di inserire un commento pubblico o di inviare un messaggio privato all’utente che ha inserito l’annuncio. In generale, un utente può inviare messaggi agli altri utenti. |
| **Report** |
| Gli amministratori possono generare un report indicante per ciascun utente quanti annunci sono stati contrassegnati come venduti. Il report riporta anche le informazioni sulla carta di credito dell’utente. |
| **Conversazioni** |
| In generale, un utente può inviare messaggi agli altri utenti e mostrare lo storico delle sue conversazioni, anche con la possibilità di rispondere ad una conversazione specifica. |
| **Storico Conversazione** |
| In generale un utente può: Inviare messaggi agli altri utenti e vedere lo storico delle sue conversazioni, anche con la possibilità di rispondere ad una conversazione specifica. |
| **Informazione anagrafica** |
| Un utente si registra scegliendo un username univoco, inserendo tutte le sue informazioni anagrafiche, indicando un indirizzo di residenza ed eventualmente un indirizzo di fatturazione, un numero arbitrario di recapiti (telefono, cellulare, email) indicandone uno come mezzo di comunicazione preferito. |
| **Carta di credito** |
| Un utente si registra inserendo i dati relativi alla sua carta di credito. I dati della carta di credito non sono obbligatori.  Per creare un annuncio, un utente deve necessariamente aver inserito i dati della sua carta di credito. |

# Progettazione concettuale

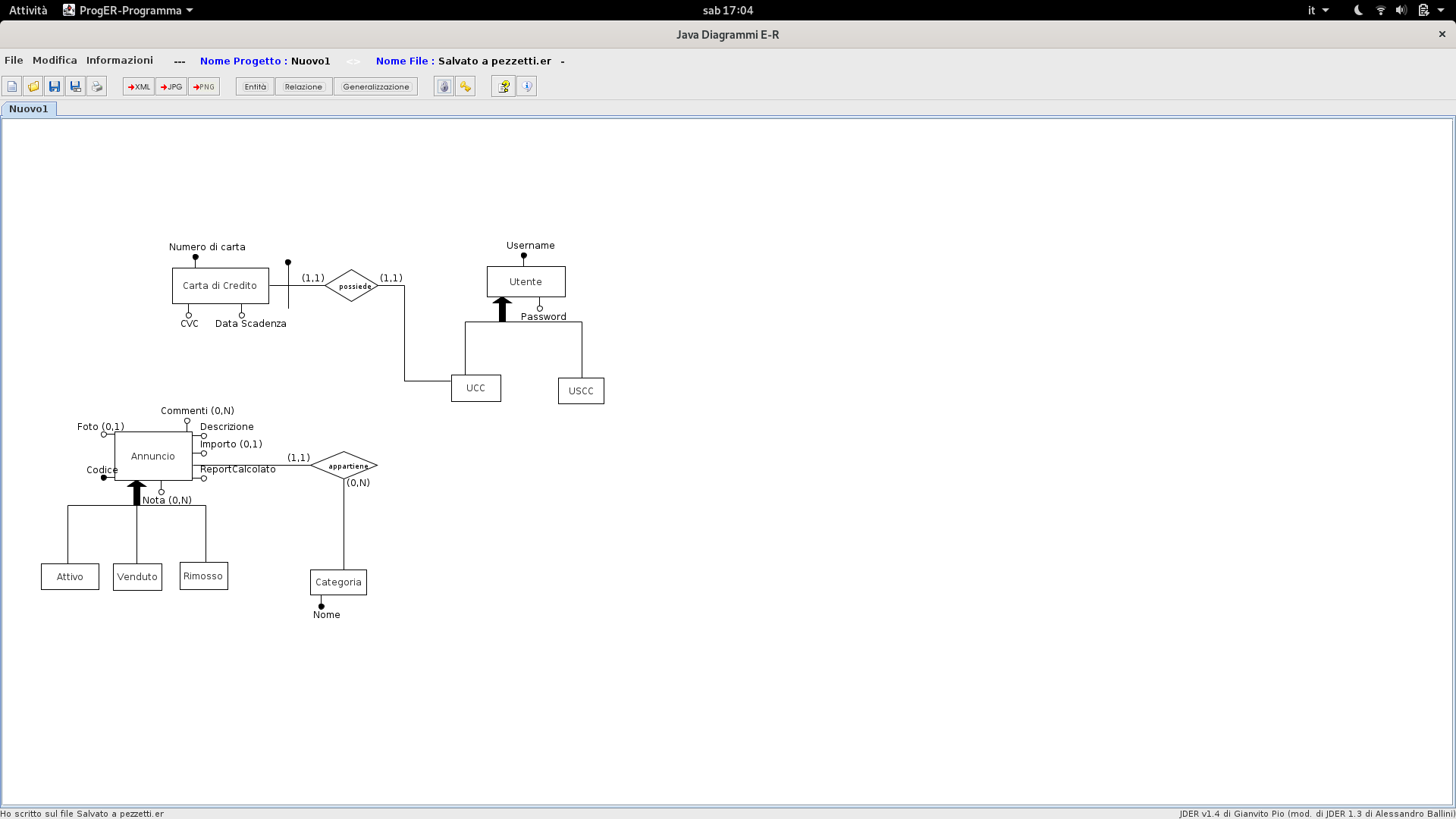
## Costruzione dello schema E-R

La progettazione del modello ER si è basata su un approccio bottom up, realizzando dapprima tutti i concetti chiave della specifica, approfondendoli con i suoi attributi, e in seguito collegare tutti i concetti per ottenere un unico modello. In seguito sono riportati i passi seguiti durante la progettazione.

* **Utente**

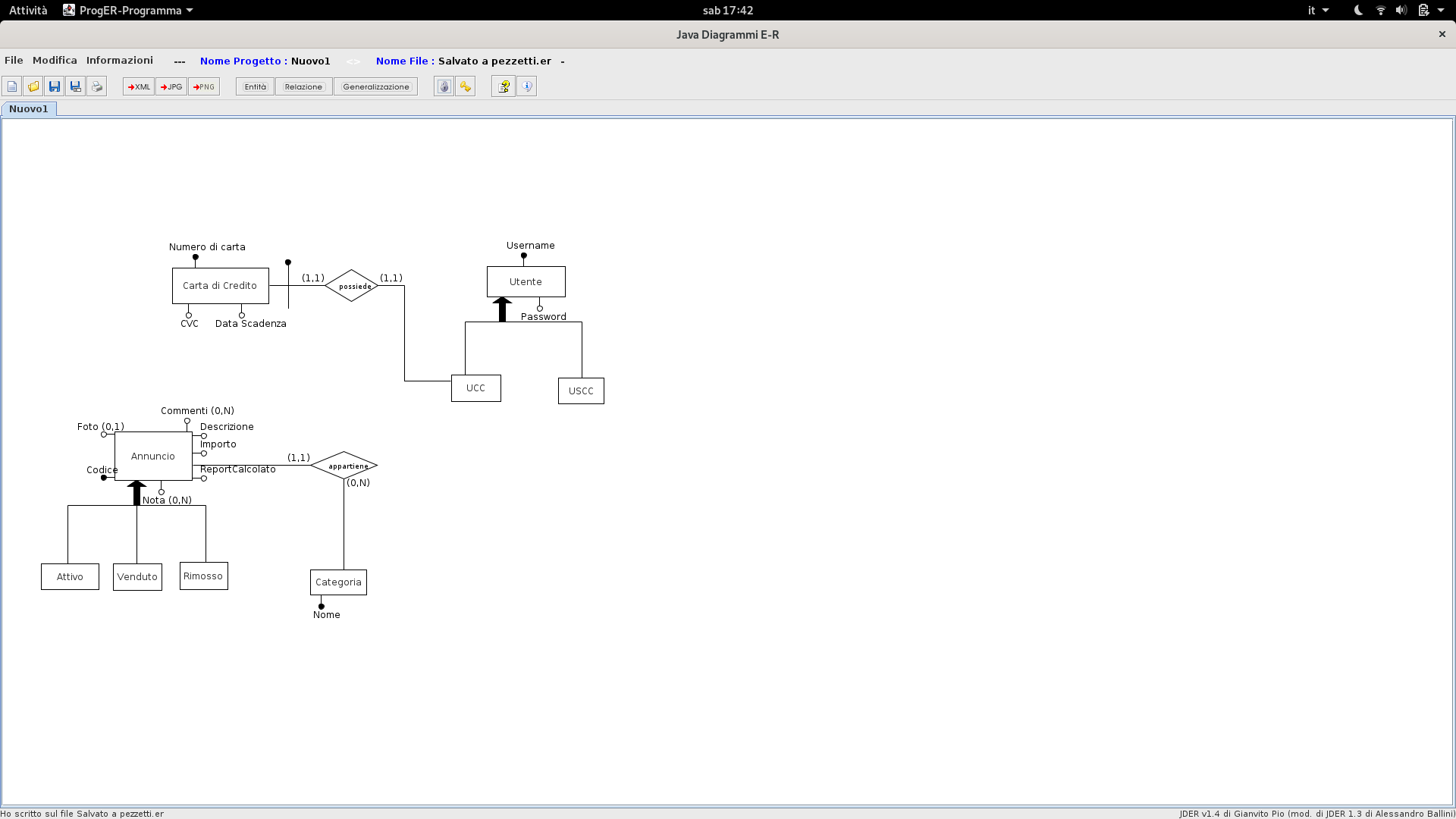
La specifica definisce chiaramente due tipi di utenti che utilizzano il sistema. Un utente UCC (Utente con Carta di Credito) ed un utente USCC (Utente Senza Carta di Credito). Entrambi condividono la caratteristica di possedere un username ed una password, ma l’utente UCC oltre a questo possiede anche una carta di credito, dove l’entità “Carta di Credito” riporta tutte le sue informazioni che dipendono unicamente dall’utente.

*Nota: Da qui in poi per semplicità entrambi gli utenti verranno indicati tramite i loro acronimi*



* **Annuncio**

Altro elemento fondamentale della specifica è l’entità “Annuncio”. Rappresenta il concetto chiave del sistema. In essa possiamo includere una foto rappresentativa dell’oggetto messo in vendita (come specificato, il suo inserimento è facoltativo), un importo (molto importante per l’entità Report che vedremo in seguito), un codice identificativo unico nel sistema, commenti pubblici inerenti all’oggetto in vendita e note pubbliche inerenti all’oggetto in vendita. Inoltre, nel momento dell’inserimento di un nuovo annuncio, l’utente UCC specificherà una e una sola categoria esistente nel database, inserita precedentemente da un amministratore.

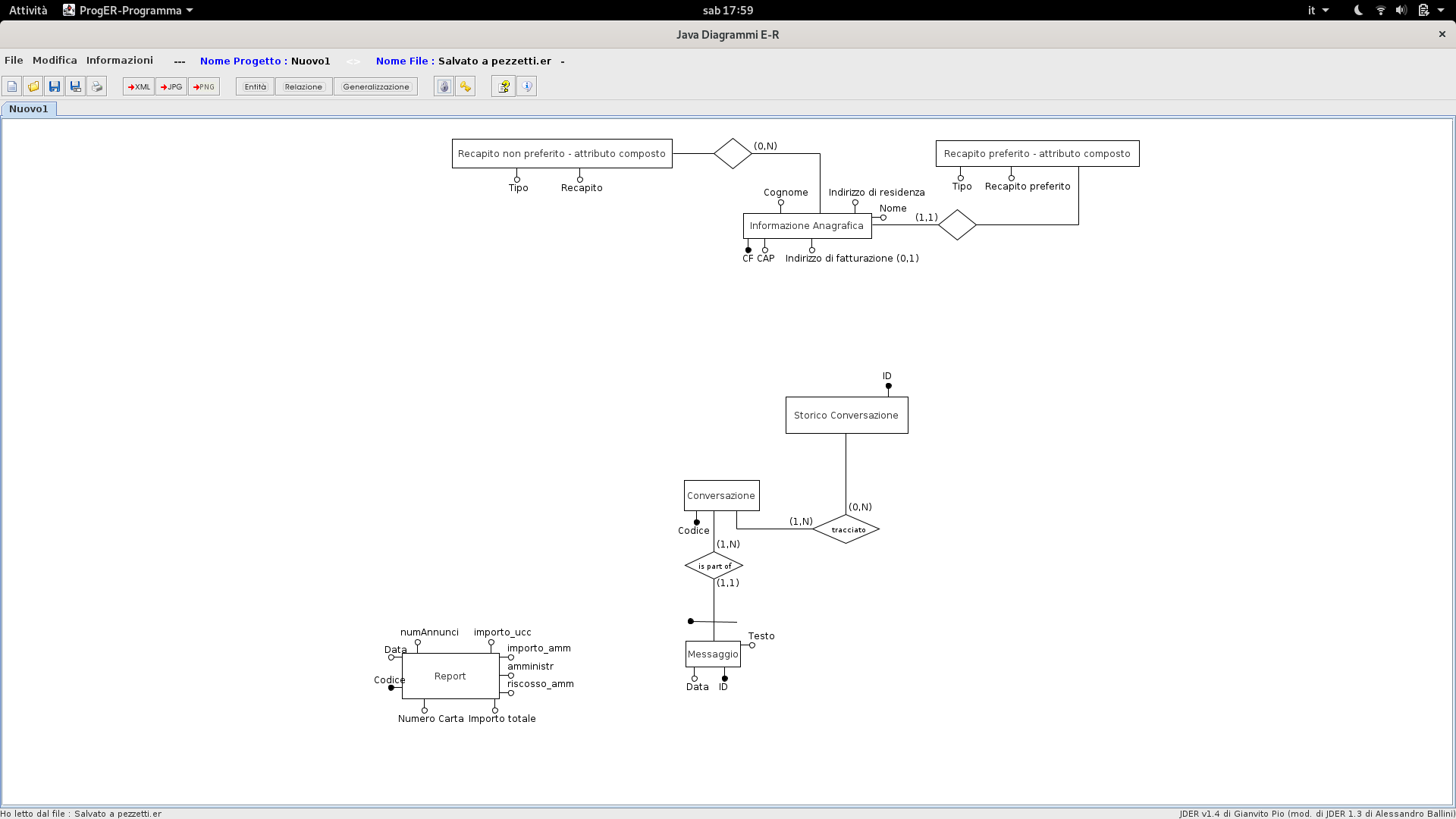


Inoltre si è deciso di differenziare l’annuncio in tre tipi per diversificare il proprio stato attuale e l’attributo “ReportCalcolato” servirà al sistema per sapere se quell’annuncio è già stato calcolato.

* **Informazione Anagrafica**

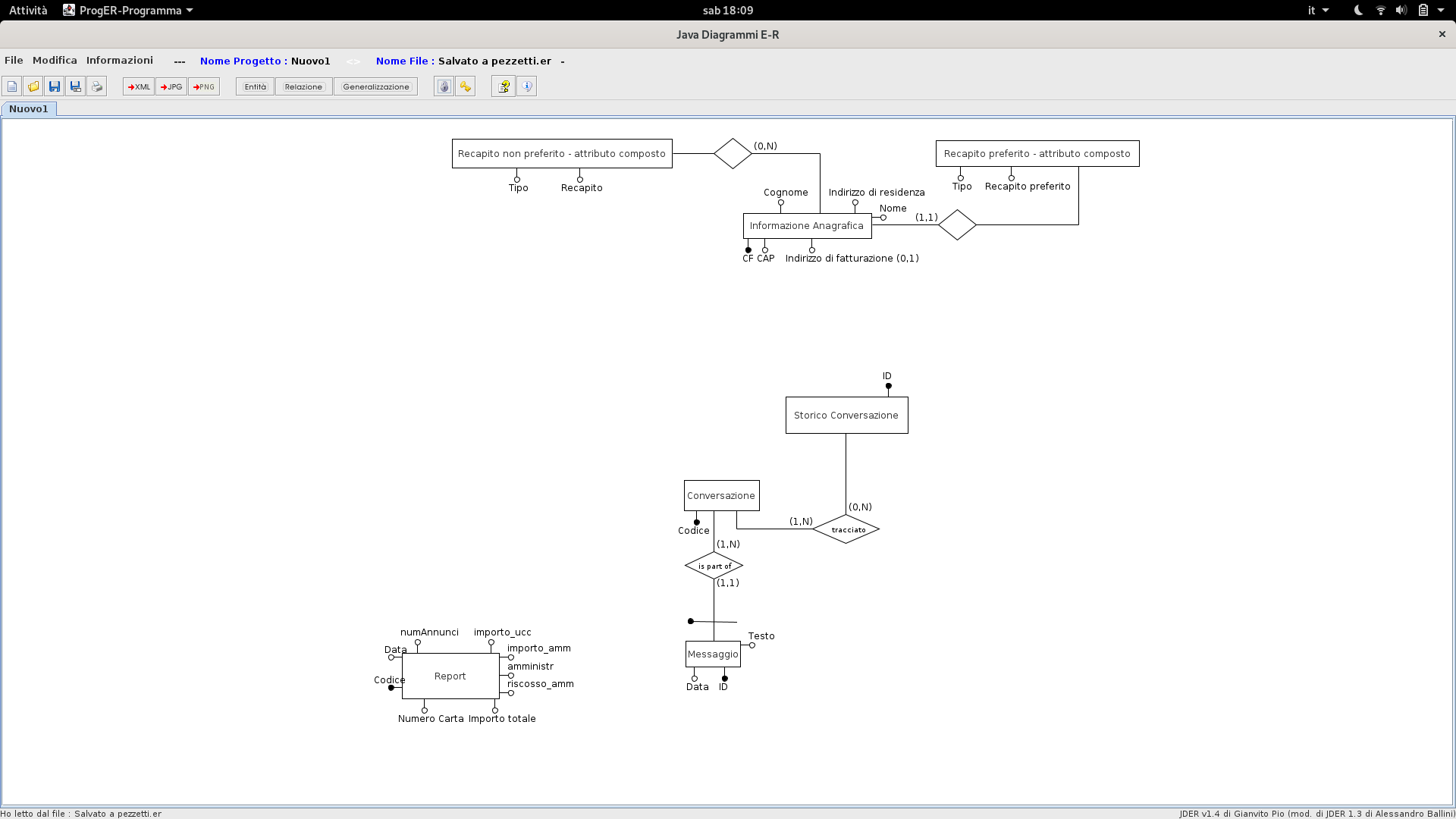
L’entità “Informazione Anagrafica” riporta tutte le informazioni richieste dell’utente (che sia UCC o USCC). I recapiti possono essere di due tipi: “Preferito” e “Non Preferito”.

Per un recapito non preferito è possibile inserire più di un contatto.

*Nota: A causa di alcune limitazioni del tool usato per costruire il modello, i due recapiti sono due attributi composti, come riportato nell’immagine.*

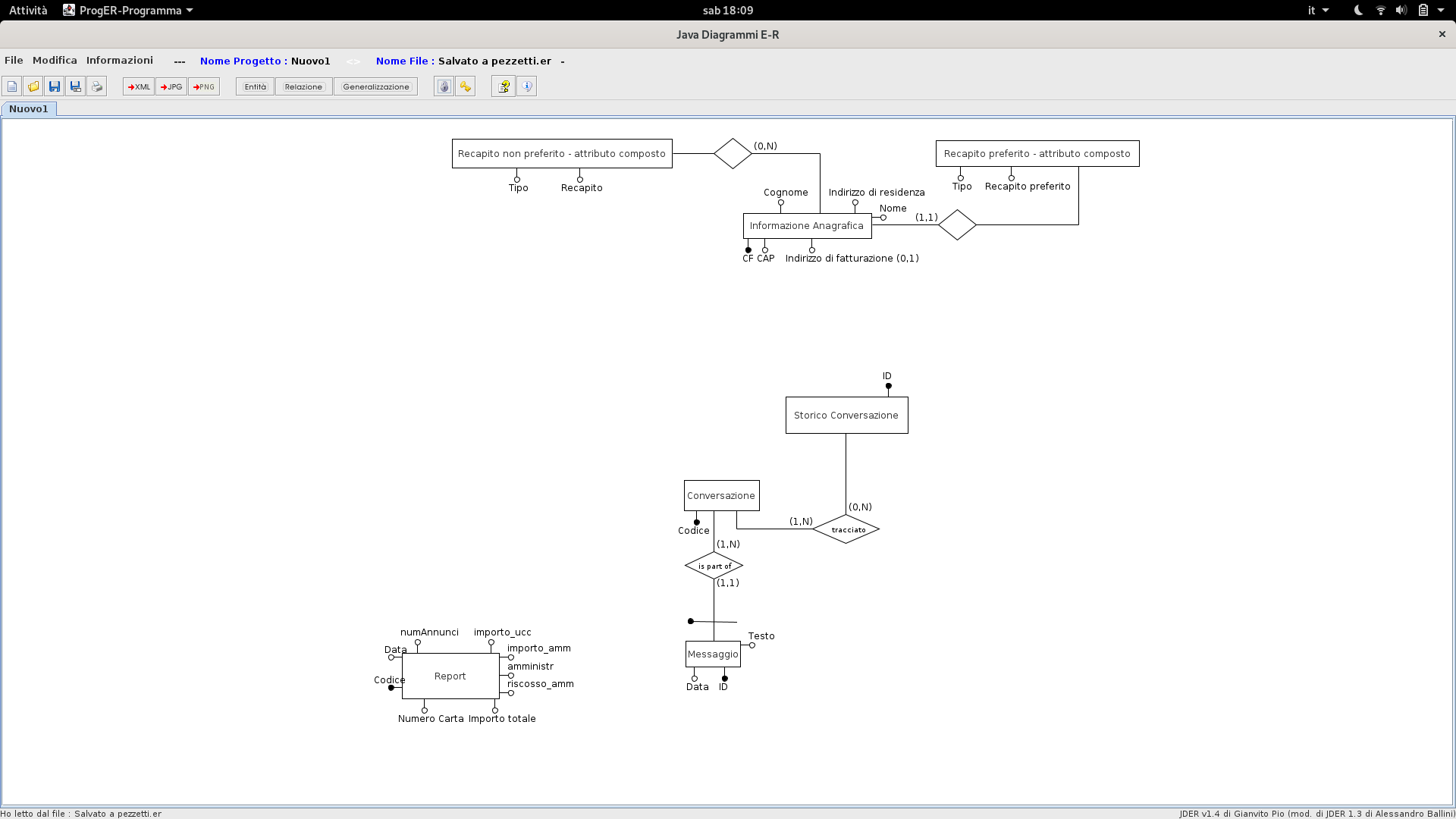
* **Report**

L’entità “Report” è una sorta di fattura riguardante un annuncio indicato come venduto. Riporta tutte le informazioni necessarie per il suo uso.



Gli attributi sono stati scelti con i seguenti criteri:

* **Codice**
  + Identificativo unico nel sistema.
* **Data**
  + Riporta la data di creazione del report.
* **Numero Carta**
  + Riporta il numero della carta di credito dell’utente UCC, necessario per la riscossione del report.
* **Importo totale**
  + Riporta l’importo totale senza l’applicazione della percentuale per gli amministratori.
* **numAnnunci**
  + Numero degli annunci a cui il report si riferisce.
* **importo\_ucc**
  + Importo che sarà destinato all’utente UCC, dopo l’applicazione della percentuale.
* **importo\_amm**
  + Importo che sarà destinato agli amministratori, dopo l’applicazione della percentuale.
* **amministr**
  + Username dell’amministratore che si è occupato della generazione del report
* **riscosso\_amm**
  + Si è pensato di aggiungere questo attributo per indicare nel sistema che la percentuale dell’amministratore è stato riscosso, essendo loro i gestori del sistema, questo dato può essere utile per mantenere un ordine nel database.
* **Conversazione**

Questo riporta un po’ tutto il mini modello riguardante la gestione della comunicazione tra utenti. Per sviluppare questa parte della specifica si è voluto creare un entità “Conversazione” contenente tutte le conversazioni tra utenti con ognuno un codice univoco per essere identificati nel sistema. Ovviamente una conversazione contiene messaggi, quindi si è deciso di usare il pattern “is part of” per indicare che una conversazione è costituita da messaggi. Ogni messaggio ha un suo codice identificativo e dipende unicamente dall’entità in relazione.

Inoltre nel momento della creazione di una conversazione essa verrà tracciata nello storico dei due utenti che stanno comunicando.

*Nota: A causa di alcune limitazioni del tool usato per costruire il modello, la cardinalità (1, N) con la relazione tracciato è (2, 2). In quanto quella conversazione è tracciata da due e solo due utenti.*

### Integrazione finale

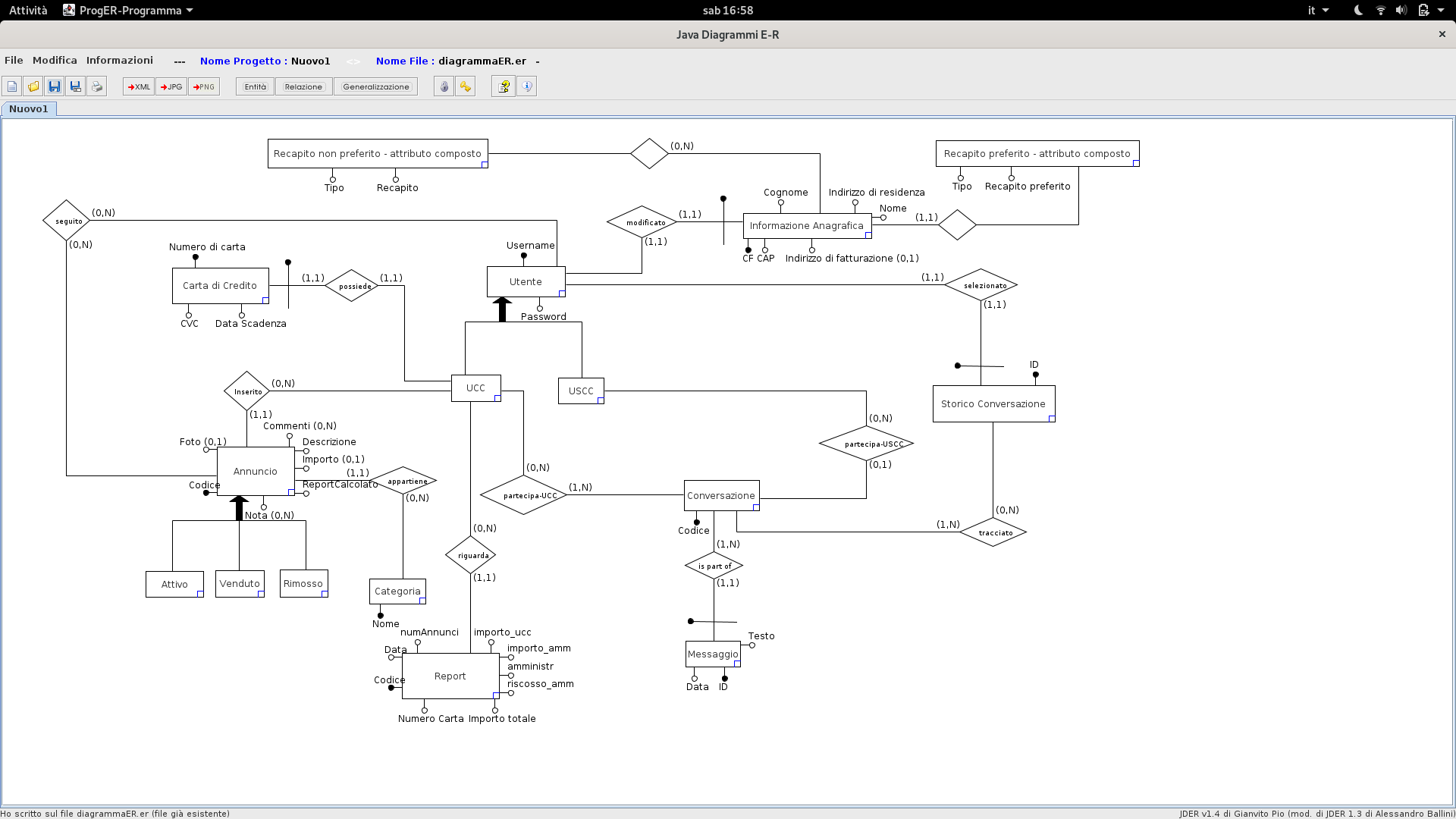
Infine si è unito ogni “mini” modello visti precedentemente con le seguenti relazioni.

1. **Inserito**
   * Tiene traccia di ogni utente UCC (l’unico che può pubblicare un annuncio) che ha inserito il suo annuncio
2. **Seguito**
   * Tiene traccia degli annunci seguiti dagli utenti
3. **Modificato**
   * Tiene traccia delle informazioni anagrafiche di ogni utente
4. **Riguarda**
   * Correla un report ad un utente
5. **Selezionato**
   * Tiene traccia del possesso di uno storico conversazione di ogni utente
6. **Partecipa-UCC**
   * Tiene traccia della partecipazione dell’utente UCC a determinate conversazioni
7. **Partecipa-USCC**
   * Tiene traccia della partecipazione, in questo caso, dell’utente USCC a determinate conversazioni

*Nota: A causa di alcune limitazioni del tool usato per costruire il modello, la cardinalità (1, N) tra “Conversazione” e “Partecipa-UCC” è (1, 2), in quanto le uniche conversazione previste sono:*

* *UCC - UCC*
* *UCC – USCC*

*Quindi non sono previste comunicazioni USCC – USCC vista la sua poca importanza a far comunicare due utenti che non hanno lo scopo di offrire un bene nel sistema. Quindi ciò avrebbe portato un aumento del traffico di informazioni poco “utili”, di conseguenza un degrado delle prestazioni.*



## Regole aziendali

## Regole di vincolo

## Un oggetto venduto, o rimosso, inserito in bacheca non deve essere visualizzato nella bacheca pubblica;

* Ogni utente deve essere informato delle modifiche svolte agli annunci da lui seguiti, compresi inserimenti, o eliminazioni, di note e commenti;
* Il sistema deve calcolare una percentuale pari al 3% sul report generato;
* Almeno un amministratore deve aver creato una categoria;
* Gli utenti possono vedere, seguire e commentare solo annunci attivi;

**Regole di derivazione**

* Il numero degli oggetti venduti si ottiene contando il numero degli annunci “venduti”;
* L’importo totale si ottiene sommando tutti gli importi degli annunci venduti;
* Le informazioni della carta di credito nel report si ottiene dall’utente UCC;

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatori** |
| Utente | Rappresenta una persona nel sistema | Username, Password | Username |
| UCC | Utente con Carta di Credito |  |  |
| USCC | Utente Senza Carta di Credito |  |  |
| Annuncio | Rappresenta un oggetto nel sistema | Codice, Nota, Commenti, Descrizione, Importo, Foto, ReportCalcolato | Codice |
| Attivo | Annuncio indicato come attivo, quindi ancora disponibile ad essere venduto |  |  |
| Venduto | Annuncio indicato come venduto |  |  |
| Rimosso | Annuncio indicato come rimosso |  |  |
| Categoria | Rappresenta il tipo di annuncio che si può avere nel sistema | Nome | Nome |
| Carta di Credito | Rappresenta la carta di credito posseduta da un utente UCC | Numero di carta, CVC, Data Scadenza | Numero di carta |
| Report | Rappresenta una sorta di fattura per l’utente riguardante i suoi annunci venduti | Codice, Data, numAnnunci, importo\_ucc, importo\_amm, amministr, riscosso\_amm, Numero Carta, Importo totale | Codice |
| Informazione Anagrafica | Contiene i dati personali degli utenti | CF, Nome, Cognome, Indirizzo di residenza, Indirizzo di fatturazione, CAP |  |
| Storico Conversazione | Rappresenta il registro delle conversazioni dell’utente | ID | ID |
| Messaggio | Rappresenta l’informazione che si inviano gli utenti | ID, Data, Testo | ID |
| Conversazione | Rappresenta la comunicazione tra gli utenti | Codice | Codice |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo**[[1]](#footnote-2) | **Volume atteso** |
| Utente | E | **40.000** |
| Utente con carta di credito (UCC) | E | **15.000** |
| Utente senza carta di credito (USCC) | E | **25.000** |
| Carta di credito | E | **15.000** *(ogni utente UCC ha una carta di credito)* |
| Annuncio | E | **30.000** *(ogni UCC ha almeno 2 annunci)* |
| Attivo | E | **10.000** |
| Venduto | E | **15.000** |
| Rimosso | E | **5.000** |
| Report | E | **15.000** *(almeno ogni utente UCC ha venduto qualcosa)* |
| Conversazione | E | **200.000** *(ogni utente ha avuto almeno circa 5 conversazioni)* |
| Storico Conversazione | E | **400.000** *(il doppio delle conversazioni)* |
| Messaggio | E | **2.000.000** *(ogni conversazione ha almeno 10 messaggi)* |
| Informazione Anagrafica | E | **40.000** |
| Categoria | E | **20** |
| Recapito Non Preferito | E | **40.000** *(ogni utente ha almeno un recapito non preferito)* |
| Possiede | R | **15.000** *(ogni utente UCC ha una carta di credito)* |
| Selezionato | R | **240.000** |
| Tracciato | R | **400.000** *(ogni volta che una conversazione viene creata, viene tracciata nello storico dei due utenti)* |
| Seguito | R | **120.000** *(in media ogni utente mette 3 annunci tra i preferiti)* |
| Inserito | R | **15.000** *(almeno 1 annuncio è stato pubblicato da un utente UCC)* |
| Appartiene | R | **30.000** *(ogni annuncio ha una e una sola relazione con categoria)* |
| Modificato | R | **40.000** |
| Partecipa-UCC | R | **178.000** |
| Riguarda | R | **15.000** *(ogni UCC ha venduto almeno una cosa)* |
| Is part of | R | **2.000.000** *(una relazione per ogni conversazione)* |
| Partecipa-USCC | R | **196.000** |

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Descrizione** | **Frequenza attesa** |
| 001 | Registrazione utente alla bacheca | **500 al giorno** |
| 002 | Modifica informazioni anagrafica dell’utente | **153 al mese** |
| 003 | Inserimento dei dati relativi alla carta di credito | **187 al giorno** *(circa il 38% dei nuovi utenti decide di essere anche venditore)* |
| 004 | Inserimento di una nuova categoria | **3 al mese** |
| 005 | Inserimento di un nuovo annuncio | **559 ogni 4 giorni** |
| 006 | Caricamento di una foto dell’oggetto | **102 ogni 4 giorni** *(operazione opzionale)* |
| 007 | Visualizzazione di un annuncio da parte di un utente | **25.340 al giorno** |
| 008 | Inserimento o Rimozione di un commento ad un annuncio | **1002 al giorno** *(circa il 40% degli utenti che visualizza un annuncio inserisce un nuovo commento)* |
| 009 | Invio di un nuovo messaggio da un utente USCC | **125.000 al giorno** |
| 010 | Invio di un nuovo messaggio da un UCC | **220.000 al giorno** |
| 011 | Visualizzazione di messaggio da parte di utente USCC | **140.000 al giorno** |
| 012 | Visualizzazione messaggio da parte di utente UCC | **240.000 al giorno** |
| 013 | Seguire un annuncio | **7020 al giorno** |
| 014 | Inserimento o Rimozione nota ad un annuncio da parte di un UCC *(come scritto nelle specifiche, il sistema notifica l’evento a tutti gli utenti che stanno seguendo l’annuncio)* | **178 al giorno** |
| 015 | Rimozione di un annuncio | **94 ogni 4 giorni** |
| 016 | Vendita di un oggetto | **270 ogni 4 giorni** |
| 017 | Visualizzazione di tutti gli annunci seguiti | **784 al giorno** |
| 018 | Visualizzazione dello storico delle proprie conversazioni da parte di un USCC | **98.000 al giorno** |
| 019 | Visualizzazione dello storico delle proprie conversazioni da parte di un UCC | **172.000 al giorno** |
| 020 | Generazione di un nuovo report | **143 ogni 4 giorni** *(circa il 53% degli oggetti inseriti vengono venduti)* |
| 021 | Selezione di una conversazione dallo storico delle conversazioni da parte di un UCC | **205.000 al giorno** |
| 022 | Selezione di una conversazione dallo storico delle conversazioni da parte di un USCC | **116.000 al giorno** |
| 023 | Visualizzazione commento di un annuncio | **14.000 al giorno** |
| 024 | Visualizzazione di una nota di un annuncio | **10.000 al giorno** |
| 025 | Visualizzazione di una categoria | **559 ogni 4 giorni** *(ogni volta che si inserisce un annuncio si consultano le categorie esistenti)* |
| 026 | Inserimento o Rimozione di un recapito non preferito | **1000 al giorno** |
| 027 | Visualizzazione delle informazioni anagrafiche | **2500 al giorno** |
| 028 | Riscossione report di un Amministratore | **572 al giorno** |
| 029 | Visualizzazione Report | **1200 al giorno** |
| 030 | Visualizzazione delle proprie informazioni utente UCC | **2500 al giorno** |
| 031 | Visualizzazione delle proprie informazioni utente USCC | **1500 al giorno** |

## *Nota:*

* *Si è scelto di dare la possibilità all’amministratore di poter segnalare nel database la riscossione della sua percentuale nel report generato, solo per scopi gestionali.*
* *Molte operazioni sono state inventate per rendere più ampio il sistema, ma comunque dedotte e pertinenti alla specifica assegnata*

## Costo delle operazioni

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione 001 con costo 7** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | S |
| USCC | E | 1 | S |
| Informazioni Anagrafica | E | 1 | S |
| Modificato | R | 1 | S |
| **Operazione 002 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Informazioni Anagrafica | E | 1 | S |
| Modificato | R | 1 | S |
| **Operazione 003 con costo 6** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| UCC | E | 1 | L |
| Carta di Credito | E | 1 | S |
| Possiede | R | 1 | S |
| **Operazione 004 con costo 2** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Categoria | E | 1 | S |
| **Operazione 005 con costo 8** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Annuncio | E | 1 | S |
| Categoria | E | 1 | L |
| Inserito | R | 1 | S |
| Appartiene | R | 1 | S |
| **Operazione 006 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Annuncio | E | 1 | S |
| Inserito | R | 1 | S |
| **Operazione 007 con costo 6** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Attivo | E | 1 | L |
| Annuncio | E | 1 | L |
| UCC | E | 1 | L |
| Seguito | R | 1 | L |
| Inserito | R | 1 | L |
| **Operazione 008 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Annuncio | E | 1 | S |
| Seguito | R | 1 | S |
| **Operazione 009 con costo 7** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| USCC | E | 1 | L |
| Partecipa-USCC | R | 1 | L |
| Conversazione | E | 1 | L |
| Messaggio | E | 1 | S |
| Is part of | R | 1 | S |
| **Operazione 010 con costo 7** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Partecipa-UCC | R | 1 | L |
| Conversazione | E | 1 | L |
| Is part of | R | 1 | S |
| Messaggio | E | 1 | S |
| **Operazione 011 con costo 6** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| USCC | E | 1 | L |
| Partecipa-USCC | R | 1 | L |
| Conversazione | E | 1 | L |
| Messaggio | E | 1 | L |
| Is part of | R | 1 | L |
| **Operazione 012 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Partecipa-UCC | R | 1 | L |
| Conversazione | E | 1 | L |
| Is part of | R | 1 | L |
| Messaggio | E | 1 | L |
| **Operazione 013 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Seguito | R | 1 | S |
| Attivo | E | 1 | L |
| Annuncio | E | 1 | L |
| **Operazione 014 con costo 32** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Annuncio | E | 1 | S |
| Inserito | R | 1 | S |
| Attivo | E | 1 | L |
| UCC | E | 1 | L |
| Seguito | R | 14 | L |
| Utente | E | 14 | L |
| **Operazione 015 con costo 7** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Annuncio | E | 1 | S |
| UCC | E | 1 | L |
| Rimosso | E | 1 | S |
| Inserito | R | 1 | S |
| **Operazione 016 con costo 7** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Annuncio | E | 1 | S |
| Venduto | E | 1 | S |
| Inserito | R | 1 | S |
| **Operazione 017 con costo 43** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Annuncio | E | 14 | L |
| Attivo | E | 5 | L |
| Venduto | E | 7 | L |
| Rimosso | E | 2 | L |
| Seguito | R | 14 | L |
| **Operazione 018 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| USCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |
| **Operazione 019 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |
| **Operazione 020 con costo 8** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Report | E | 1 | S |
| Riguarda | R | 1 | S |
| UCC | E | 1 | L |
| Inserito | R | 1 | L |
| Annuncio | E | 1 | L |
| Venduto | E | 1 | L |
| **Operazione 021 con costo 14** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |
| Tracciato | E | 5 | L |
| Conversazione | E | 5 | L |
| **Operazione 022 con costo 14** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| USCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |
| Tracciato | E | 5 | L |
| Conversazione | E | 5 | L |
| **Operazione 023 con costo 3** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Annuncio | E | 1 | L |
| Seguito | R | 1 | L |
| **Operazione 024 con costo 3** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Annuncio | E | 1 | L |
| Seguito | R | 1 | L |
| **Operazione 025 con costo 20** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Categoria | E | 20 | L |
| **Operazione 026 con costo 5** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Informazioni Anagrafica | E | 1 | S |
| Modificato | R | 1 | S |
| **Operazione 027 con costo 3** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Informazioni Anagrafica | E | 1 | L |
| Modificato | R | 1 | L |
| **Operazione 028 con costo 2** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Report | E | 1 | S |
| **Operazione 029 con costo 1** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Report | E | 1 | L |
| **Operazione 030 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| UCC | E | 1 | L |
| Carta di Credito | E | 1 | L |
| Possiede | R | 1 | L |
| **Operazione 031 con costo 2** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| USCC | E | 1 | L |

* 001 costo 7, Al giorno 500 Costi al giorno 3500
* 002 costo 5, Al mese 153 Costi al mese 765
* 003 costo 6, Al giorno 187 Costi al giorno 1122
* 004 costo 2, Al mese 3 Costi al mese 6
* 005 costo 8, Ogni 4 giorni 559 Costi ogni 4 giorni 4472
* 006 costo 5, Ogni 4 giorni 102 Costi ogni 4 giorni 510
* 007 costo 6, Al giorno 25.340 Costi al giorno 152.040
* 008 costo 5, Al giorno 1002 Costi al giorno 5010
* 009 costo 7, Al giorno 125.000 Costi al giorno 875.000
* 010 costo 7, Al giorno 220.000 Costi al giorno 1.540.000
* 011 costo 6, Al giorno 140.000 Costi al giorno 840.000
* 012 costo 5, Al giorno 240.000 Costi al giorno 1.200.000
* 013 costo 5, Al giorno 7020 Costi al giorno 35.100
* 014 costo 32, Al giorno 178 Costi al giorno 5696
* 015 costo 7, Ogni 4 giorni 94 Costi ogni 4 giorni 658
* 016 costo 7, Ogni 4 giorni 270 Costi ogni 4 giorni 1890
* 017 costo 43, Al giorno 784 Costi al giorno 33.712
* 018 costo 4, Al giorno 98.000 Costi al giorno 392.000
* 019 costo 4, Al giorno 172.000 Costi al giorno 688.000
* 020 costo 8, Ogni 4 giorni 143 Costi ogni 4 giorni 1144
* 021 costo 14, Ogni giorno 205.000 Costi ogni giorno 2.870.000
* 022 costo 14, Ogni giorno 116.000 Costi ogni giorno 1.624.000
* 023 costo 3, Ogni giorno 14.000 Costi ogni giorno 42.000
* 024 costo 3, Ogni giorno 10.000 Costi ogni giorno 30.000
* 025 costo 20, Ogni 4 giorni 559 Costi ogni 4 giorni 11.180
* 026 costo 5, Ogni 4 giorni 559 Costi ogni giorno 5.000
* 027 costo 3, Ogni 4 giorni 559 Costi ogni giorno 7.500
* 028 costo 2, Ogni 4 giorni 559 Costi ogni giorno 1144
* 029 costo 1, Ogni 4 giorni 559 Costi ogni giorno 1200
* 030 costo 4, Ogni 4 giorni 559 Costi ogni giorno 10.000
* 031 costo 2, Ogni 4 giorni 559 Costi ogni giorno 3.000

## Ristrutturazione dello schema E-R

**Analisi delle ridondanze**

* Visualizzazione di un annuncio da parte di un utente senza dato ridondante (operazione 007)

152.040 accessi

* + Visualizzazione di un annuncio da parte di un utente con dato ridondante

101.360 accessi (attributo: foreign key to UCC)

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione 007 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| Attivo | E | 1 | L |
| Annuncio | E | 1 | L |
| Seguito | R | 1 | L |

Si è scelto di inserire un attributo ridondante “UCC\_Username” che sarà una foreign key ad UCC, ovvero si attuerà una traduzione non ottima in “Annuncio”.

* Inserimento dei dati relativi alla carta di credito senza dato ridondante (operazione 003)

1122 accessi

* + Inserimento dei dati relativi alla carta di credito con dati ridondanti

561 accessi (attributi: Codice carta, Data di scadenza e CVC)

Essendo l’operazione 001 molto frequente, solo circa il 40% degli utenti diventano venditori, ma se si riflettesse in spazio di memoria e in accessi risparmiati in un lungo periodo questo porterebbe un notevole vantaggio di prestazioni. Considerando un risparmio del circa 50% in termini di prestazioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione 003 con costo 3** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Utente | E | 1 | L |
| UCC | E | 1 | S |

* Selezione di una conversazione dallo storico delle conversazioni senza dato ridondante (operazione 021 ed operazione 022)

2.870.000 accessi, 1.624.000 accessi

* + Selezione di una conversazione dallo storico delle conversazioni con dato ridondante

820.000 accessi, 464.000 accessi (attributo multivalore: ConversazioniCodice)

L’aggiunta dell’attributo comporta un grande vantaggio in termini di tempo

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione 021 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| UCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |
| **Operazione 022 con costo 4** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| USCC | E | 1 | L |
| Utente | E | 1 | L |
| Selezionato | R | 1 | L |
| Storico Conversazione | E | 1 | L |

**Eliminazione delle generalizzazioni**

La seconda ristrutturazione consiste nel diversificare i concetti di “Annuncio” e “Utente” andando a eliminare le generalizzazioni. Si è deciso così di:

1. Eliminare le entità figlie di “Annuncio” in quanto le operazioni più frequenti sono eseguite sull’entità padre;
2. Accorpare l’entità “Utente” nelle figlie in quanto i concetti chiavi e le operazioni più importanti vengono effettuate sulle entità figlie;

**Eliminazione degli attributi multivalori**

* Annuncio:
  + Commento;
  + Nota;

Sono stati eliminati questi due attributi multivalori e si è deciso quindi di creare due entità con due relazioni di dipendenza verso “Annuncio”.

* Storico Conversazione
  + ConversazioniCodice

È stato eliminato l’attributo multivalore e si è deciso quindi di creare un entità con una relazione di dipendenza verso “Storico Conversazione”.

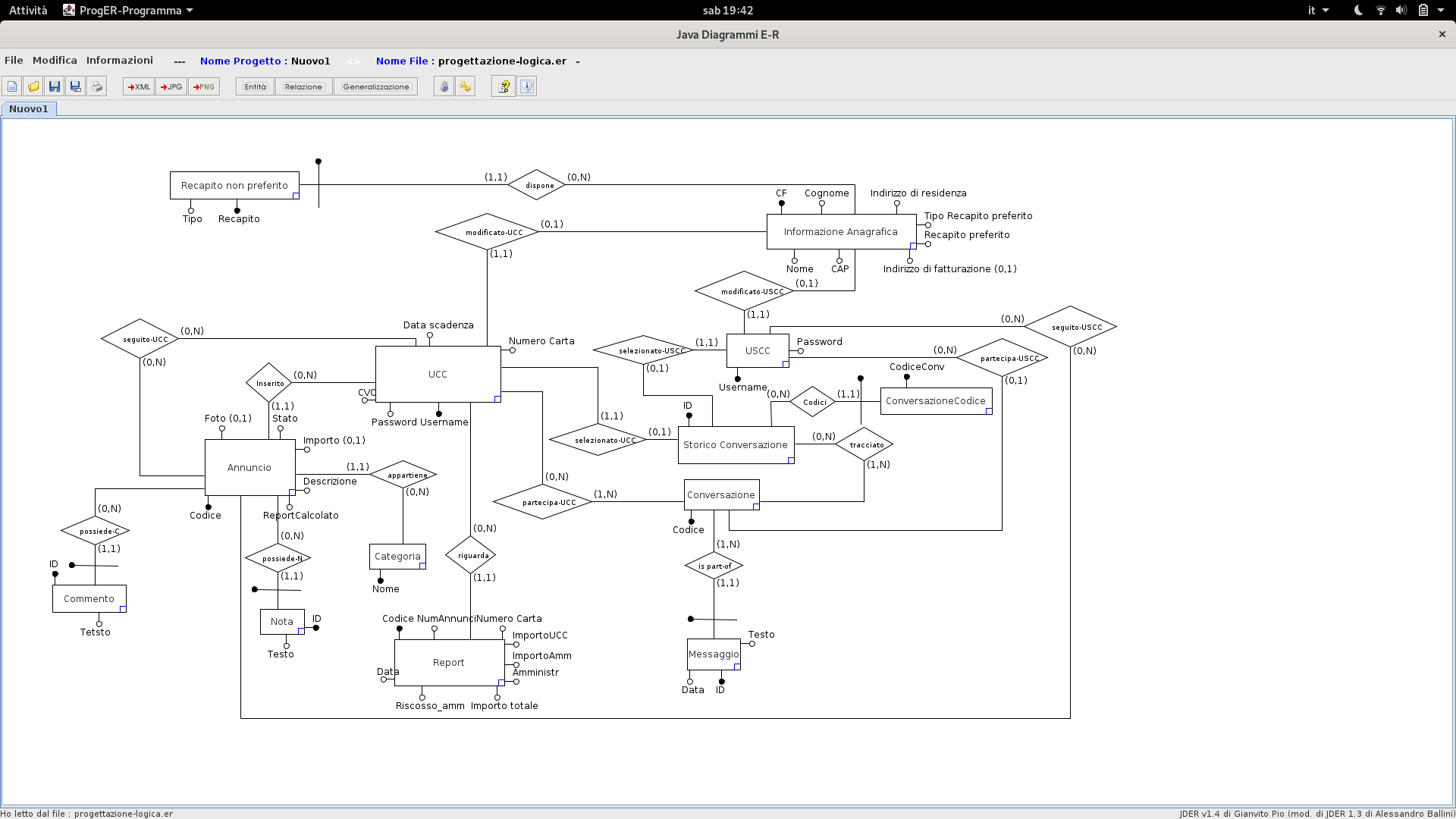
**Scelta identificatori primari**

|  |  |
| --- | --- |
| **Entità** | **Identificatore primario** |
| UCC | Username |
| USCC | Username |
| Annuncio | Codice |
| Informazione Anagrafica | CF |
| Conversazione | Codice |
| Messaggio | ID |
| Categoria | Nome |
| Recapito non preferito | Recapito |
| Nota | ID |
| Commento | ID |
| Storico Conversazione | ID |
| ConversazioneCodice | CodiceConv |
| Report | Codice |

## Trasformazione di attributi e identificatori

* Gli attributi Numero Carta, Data Scadenza, CVC in UCC sono riferiti alla carta di credito.

**Modello Ristrutturato**



*Nota: A causa di alcune limitazioni del tool usato per costruire il modello, la cardinalità (1, N) tra “Conversazione” e “Partecipa-UCC” è (1, 2), in quanto le uniche conversazione previste sono:*

* *UCC - UCC*
* *UCC – USCC*

*Quindi non sono previste comunicazioni USCC – USCC vista la sua poca importanza a far comunicare due utenti che non hanno lo scopo di offrire un bene nel sistema. Quindi ciò avrebbe portato un aumento del traffico di informazioni poco “utili”, di conseguenza un degrado delle prestazioni.*

## Traduzione di entità e associazioni

* Annuncio(Codice, Descrizione, Importo\*, Stato, Foto\*, ReportCalcolato, *Categoria\_Nome, UCC\_Username*);
* Nota(ID, *Annuncio\_Codice*, Testo);
* Commento(*ID*, *Annuncio\_Codice*, Testo);
* Categoria(Nome);
* Seguito-UCC(*Annuncio\_Codice*, *UCC\_Username*);
* Seguito-USCC(*Annuncio\_Codice*, *USCC\_Username*);
* UCC(Username, Password, Numero Carta, Data scadenza, CVC, *ID\_StoricoConversazione*, *CF\_Anagrafico*);
* Report(*Codice*, ImportoTotale, NumAnnunci, Numero Carta, Data, Riscosso\_Amm, Importo\_UCC, Importo\_Amm, Amministr, *UCC\_Username*);
* USCC(Username, Password, *ID\_StoricoConversazione, CF\_Anagrafico*);
* Tracciato(*Codice\_Conversazione*, *ID\_StoricoConversazione*);
* ConversazioneCodice(CodiceConv, *ID\_StoricoConversazione*);
* StoricoCoversazione(ID);
* Conversazione(Codice, *UCC\_Username\_1, UCC\_Username\_2\*, USCC\_Username\**);
* Messaggio(ID, *Codice\_Conversazione*, Data, Testo);
* Informazione Anagrafica(CF, Cognome, Nome, Indirizzo di residenza, CAP, Indirizzo di fatturazione\*, Tipo Recapito preferito, Recapito preferito);
* Recapito non preferito(Recapito, *Informazione Anagrafica\_CF*, Tipo);

**Vincoli referenziali**

ConversazioneCodice(ID\_StoricoConversazione) ⊆ StoricoConversazione(ID);

Tracciato(ID\_StoricoConversazione) ⊆ StoricoConversazione(ID);

Tracciato(Codice\_Conversazione) ⊆ Conversazione (Codice);

Conversazione(*UCC\_Username\_1*) ⊆ UCC(Username);

Conversazione(*UCC\_Username\_2*) ⊆ UCC(Username);

Conversazione(*USCC\_Username*) ⊆ USCC(Username);

USCC(ID\_StoricoConversazione) ⊆ StoricoConversazione(ID);

USCC(*CF\_Anagrafico*) ⊆ Informazione Anagrafica(CF);

UCC(ID\_StoricoConversazione) ⊆ StoricoConversazione(ID);

UCC(*CF\_Anagrafico*) ⊆ Informazione Anagrafica(CF);

Annuncio(Categoria\_Nome) ⊆ Categoria(Nome);

Annuncio(*UCC\_Username*) ⊆ UCC(Username);

Nota(Annuncio\_Codice) ⊆ Annuncio(Codice);

Commento(Annuncio\_Codice) ⊆ Annuncio(Codice);

Seguito-UCC(Annuncio\_Codice) ⊆ Annuncio(Codice);

Seguito-UCC(UCC\_Username) ⊆ UCC(Username);

Seguito-USCC(Annuncio\_Codice) ⊆ Annuncio(Codice);

Seguito-USCC(USCC\_Username) ⊆ USCC(Username);

Report(UCC\_Username) ⊆ UCC(Username);

Messaggio(Codice\_Conversazione) ⊆ Conversazione(Codice);

Recapito non preferito(Informazione Anagrafica\_CF) ⊆ Informazione Anagrafica(CF);

*Nota: i valori scritti in corsivo sono vincoli referenziali*

## Normalizzazione del modello relazionale

Tutte le tabelle sono in 3NF.

# Progettazione fisica

## Utenti

Descrivere, all’interno dell’applicazione, quali utenti sono stati previsti con quali privilegi di accesso su quali tabelle, giustificando le scelte progettuali.

## Utente con carta di credito (UCC);

* Utente senza carta di credito (USCC);
* Amministratore;
* Login

**Privilegi**

* UCC (dispone anche di tutti i privilegi di USCC):
* Inserire un nuovo Annuncio
* Rimuovere il proprio Annuncio
* Indicare come Venduto il suo annuncio
* Modificare le proprie informazioni Anagrafiche
* Aggiungere o Rimuovere una foto ad un suo Annuncio
* Aggiungere o Rimuovere una nota ad un suo Annuncio
* Inviare un messaggio
* Visualizzare gli Annunci seguiti
* Visualizzare i suoi Report
* USCC:
* Visualizzare un Annuncio
* Visualizzare una Categoria
* Visualizzare le proprie informazioni
* Modificare le proprie informazioni Anagrafiche
* Aggiungere o Rimuovere un commento in un annuncio
* Visualizzare un commento
* Visualizzare una nota
* Aggiungere o Rimuovere un recapito non preferito
* Visualizzare tutti i suoi recapiti
* Inviare un messaggio
* Visualizzare un messaggio
* Visualizzare il suo Storico
* Seguire un Annuncio
* Visualizzare gli Annunci seguiti
* Visualizzare le sue notifiche
* Amministratore:
* Inserimento nuova Categoria
* Visualizzare una categoria
* Visualizzare un Annuncio
* Generare un report
* Riscuotere un report
* Visualizzare un report
* Login:
* Login
* Registrare un nuovo utente UCC
* Registrare un nuovo utente USCC

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| **Tabella Categoria** | | |
| **Attributo** | **Tipo di dato** | **Attributi**[[2]](#footnote-3) |
| Nome | VARCHAR(20) | PK, NN, UQ |

|  |  |  |
| --- | --- | --- |
| **Tabella UCC** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Username | VARCHAR(45) | PK, NN, UQ |
| Password | VARCHAR(45) | NN |
| NumeroCarta | VARCHAR(16) | NN |
| DataScadenza | DATE | NN |
| CVC | INT | NN |
| StoricoConversazione\_ID | INT | NN |
| CF\_Anagrafico | VARCHAR(16) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Annuncio** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Codice | INT | PK, NN, UQ, AI |
| Stato | ENUM('Attivo', 'Venduto', 'Rimosso') | NN |
| Descrizione | VARCHAR(100) | NN |
| Importo | INT | NN |
| Foto | VARCHAR(9) |  |
| Report\_calcolato | TINYINT | NN |
| UCC\_Username | VARCHAR(45) | NN |
| Categoria\_Nome | VARCHAR(20) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Amministratore** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Username | VARCHAR(45) | PK, NN, UQ |
| Password | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Nota** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ID | INT | PK, NN, AI |
| Testo | VARCHAR(45) | NN |
| Annuncio\_Codice | INT | PK. NN. UQ |

|  |  |  |
| --- | --- | --- |
| **Tabella Seguito-USCC** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Annuncio\_Codice | INT | PK, NN |
| USCC\_Username | VARCHAR(45) | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Conversazione** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Codice | INT | PK, NN, UQ, AI |
| UCC\_Username\_1 | VARCHAR(45) | NN |
| UCC\_Username\_2 | VARCHAR(45) |  |
| USCC\_Username | VARCHAR(45) |  |

|  |  |  |
| --- | --- | --- |
| **Tabella StoricoConversazione** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ID | INT | PK, NN, UQ, AI |

|  |  |  |
| --- | --- | --- |
| **Tabella Seguito-UCC** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Annuncio\_Codice | INT | PK, NN |
| UCC\_Username | VARCHAR(45) | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella USCC** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Username | VARCHAR(45) | PK, NN, UQ |
| Password | VARCHAR(45) | NN |
| StoricoConversazione\_ID | INT | NN |
| CF\_Anagrafico | VARCHAR(16) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Messaggio** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ID | INT | PK, NN, AI |
| Data | DATETIME | NN |
| Conversazione\_Codice | INT | PK, NN |
| Testo | VARCHAR(100) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella InformazioneAnagrafica** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| CF | VARCHAR(16) | PK, NN, UQ |
| Cognome | VARCHAR(20) | NN |
| Nome | VARCHAR(20) | NN |
| IndirizzoDiResidenza | VARCHAR(20) | NN |
| CAP | INT | NN |
| IndirizzoDiFatturazione | VARCHAR(20) |  |
| TipoRecapitoPreferito | ENUM('email', 'cellulare', 'social', 'sms') | NN |
| RecapitoPreferito | VARCHAR(40) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Notifica** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Codice\_Notifica | INT | PK, NN, UQ, AI |
| Codice | INT | NN |
| Tipo\_Notifica | VARCHAR(45) | NN |
| Username\_Utente | VARCHAR(45) | NN |
| Data | DATETIME | NN |
| Messaggio | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella RecapitoNonPreferito** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Recapito | VARCHAR(45) | PK, NN |
| Tipo | ENUM(‘email’, ‘cellulare’, ‘social’, ‘sms’) | NN |
| InformazioneAnagrafica\_CF | VARCHAR(16) | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Commento** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ID | INT | PK, NN, AI |
| Testo | VARCHAR(45) | NN |
| Annuncio\_Codice | INT | PK, NN, UQ |

|  |  |  |
| --- | --- | --- |
| **Tabella Report** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Codice | INT | PK, NN, UQ, AI |
| UCC\_Username | VARCHAR(45) | NN |
| ImportoTotale | INT | NN |
| NumeroAnnunci | INT | NN |
| NumeroCarta | VARCHAR(16) | NN |
| Data | DATE | NN |
| Amministratore\_Username | VARCHAR(45) | NN |
| Riscosso\_Amministratore | TINYINT | NN |
| Importo\_Amministratore | INT | NN |
| Importo\_UCC | INT | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella ConversazioneCodice** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| CodiceConv | INT | PK, NN |
| StoricoConversazione\_ID | INT | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella Tracciato** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| Conversazione\_Codice | INT | PK, NN |
| StoricoConversazione\_ID | INT | PK, NN |

## Indici

|  |  |
| --- | --- |
| **Tabella UCC** | |
| **Indice PRIMARY** | **Tipo**[[3]](#footnote-4)**:** |
| Username | PRIMARY |
| **Indice Username\_UNIQUE** | **Tipo:** |
| Username | UNIQUE |
| **Indice fk\_UCC\_StoricoConversazioni1\_idx** | **Tipo:** |
| StoricoConversazione\_ID | INDEX |
| **Indice fk\_UCC\_InformazioneAnagrafica1\_idx** | **Tipo:** |
| CF\_Anagrafico | INDEX |

|  |  |
| --- | --- |
| **Tabella Nota** | |
| **Indice PRIMARY** | **Tipo:** |
| ID | PRIMARY |
| **Indice fk\_Annuncio\_Codice \_UNIQUE** | **Tipo:** |
| Annuncio\_Codice | UNIQUE |
| **Indice fk\_Nota\_Annuncio1\_idx** | **Tipo:** |
| Annuncio\_Codice | INDEX |

|  |  |
| --- | --- |
| **Tabella Annuncio** | |
| **Indice PRIMARY** | **Tipo:** |
| Codice | PRIMARY |
| **Indice** **Codice\_UNIQUE** | **Tipo:** |
| Codice | UNIQUE |
| **Indice fk\_Annuncio\_UCC\_idx** | **Tipo:** |
| UCC\_Username | INDEX |
| **Indice fk\_Annuncio\_Categoria 1\_idx** | **Tipo:** |
| Categoria\_Nome | INDEX |

|  |  |
| --- | --- |
| **Tabella Seguito-USCC** | |
| **Indice PRIMARY** | **Tipo:** |
| Annuncio\_Codice | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| USCC\_Username | PRIMARY |
| **Indice fk\_Seguito-USCC\_Annuncio1\_idx** | **Tipo:** |
| Annuncio\_Codice | INDEX |
| **Indice fk\_Seguito-USCC\_USCC1\_idx** | **Tipo:** |
| USCC\_Username | INDEX |

|  |  |
| --- | --- |
| **Tabella Seguito-UCC** | |
| **Indice PRIMARY** | **Tipo:** |
| Annuncio\_Codice | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| UCC\_Username | PRIMARY |
| **Indice fk\_Seguito-UCC\_Annuncio1\_idx** | **Tipo:** |
| Annuncio\_Codice | INDEX |
| **Indice fk\_Seguito-UCC\_UCC1\_idx** | **Tipo:** |
| UCC\_Username | INDEX |

|  |  |
| --- | --- |
| **Tabella Conversazione** | |
| **Indice PRIMARY** | **Tipo:** |
| Codice | PRIMARY |
| **Indice Codice\_UNIQUE** | **Tipo:** |
| Codice | UNIQUE |
| **Indice fk\_Conversazione\_UCC1\_idx** | **Tipo:** |
| UCC\_Username\_1 | INDEX |
| **Indice fk\_Conversazione\_USCC1\_idx** | **Tipo:** |
| USCC\_Username | INDEX |
| **Indice fk\_Conversazione\_UCC2\_idx** | **Tipo:** |
| UCC\_Username\_2 | INDEX |

|  |  |
| --- | --- |
| **Tabella USCC** | |
| **Indice PRIMARY** | **Tipo:** |
| Username | PRIMARY |
| **Indice Username\_UNIQUE** | **Tipo:** |
| Username | UNIQUE |
| **Indice fk\_USCC\_StoricoConversazione1\_idx** | **Tipo:** |
| StoricoConversazione­\_ID | INDEX |
| **Indice fk\_USCC\_InformazioneAnagrafica1\_idx** | **Tipo:** |
| CF\_Anagrafico | INDEX |

|  |  |
| --- | --- |
| **Tabella Messaggio** | |
| **Indice PRIMARY** | **Tipo:** |
| ID | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| Conversazione\_Codice | PRIMARY |
| **Indice fk\_Messaggio\_Conversazione1\_idx** | **Tipo:** |
| Conversazione\_Codice | INDEX |

|  |  |
| --- | --- |
| **Tabella InformazioneAnagrafica** | |
| **Indice PRIMARY** | **Tipo:** |
| CF | PRIMARY |
| **Indice CF\_UNIQUE** | **Tipo:** |
| CF | UNIQUE |

|  |  |
| --- | --- |
| **Tabella Notifica** | |
| **Indice PRIMARY** | **Tipo:** |
| Codice\_Notifica | PRIMARY |
| **Indice CF\_UNIQUE** | **Tipo:** |
| Codice\_Notifica | UNIQUE |

|  |  |
| --- | --- |
| **Tabella RecapitoNonPreferito** | |
| **Indice PRIMARY** | **Tipo:** |
| Recapito | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| InformazioneAnagrafica\_CF | PRIMARY |
| **Indice fk\_RecapitoNonPreferito\_InformazioneAnagrafica1\_idx** | **Tipo:** |
| InformazioneAnagrafica\_CF | INDEX |

|  |  |
| --- | --- |
| **Tabella Amministratore** | |
| **Indice PRIMARY** | **Tipo:** |
| Username | PRIMARY |
| **Indice Username\_UNIQUE** | **Tipo:** |
| Username | UNIQUE |

|  |  |
| --- | --- |
| **Tabella Commento** | |
| **Indice PRIMARY** | **Tipo:** |
| ID | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| Annuncio\_Codice | PRIMARY |
| **Indice Annuncio**\_**Codice\_UNIQUE** | **Tipo:** |
| Annuncio\_Codice | UNIQUE |
| **Indice fk\_Commento\_Annuncio1\_idx** | **Tipo:** |
| Annuncio\_Codice | INDEX |

|  |  |
| --- | --- |
| **Tabella Report** | |
| **Indice PRIMARY** | **Tipo:** |
| Codice | PRIMARY |
| **Indice Codice \_UNIQUE** | **Tipo:** |
| Codice | UNIQUE |
| **Indice fk\_Report\_UCC1\_idx** | **Tipo:** |
| UCC\_Username | INDEX |

|  |  |
| --- | --- |
| **Tabella ConversazioneCodice** | |
| **Indice PRIMARY** | **Tipo:** |
| CodiceConv | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| StoricoConversazione\_ID | PRIMARY |
| **Indice fk\_ConversazioneCodice\_StoricoConversazione1\_idx** | **Tipo:** |
| StoricoConversazione\_ID | INDEX |

|  |  |
| --- | --- |
| **Tabella Tracciato** | |
| **Indice PRIMARY** | **Tipo:** |
| Conversazione\_Codice | PRIMARY |
| **Indice PRIMARY** | **Tipo:** |
| StoricoConversazione\_ID | PRIMARY |
| **Indice fk\_Tracciato\_StoricoConversazione1\_idx** | **Tipo:** |
| StoricoConversazione\_ID | INDEX |
| **Indice fk\_Tracciato\_Conversazione1\_idx** | **Tipo:** |
| Conversazione\_Codice | INDEX |

|  |  |
| --- | --- |
| **Tabella Categoria** | |
| **Indice PRIMARY** | **Tipo:** |
| Nome | PRIMARY |
| **Indice Nome\_UNIQUE** | **Tipo:** |
| Nome | UNIQUE |

|  |  |
| --- | --- |
| **Tabella StoricoConversazione** | |
| **Indice PRIMARY** | **Tipo:** |
| ID | PRIMARY |
| **Indice ID\_UNIQUE** | **Tipo:** |
| ID | UNIQUE |

**NOTA:**

1. **In caso di necessità c’è un backup del progetto nel mio account github**
2. [**https://github.com/Monello1299/Basi-di-Dati**](https://github.com/Monello1299/Basi-di-Dati)
3. **Per una lettura migliore del codice utilizzare un editor di testo**

## Trigger

-- **Avviso Nota Inserita** ----------------------------------------------

L'idea del trigger è di soddisfare la regola aziendale di avvisare ogni utente che segue l'annuncio di una nuova nota. Viene eseguito un ciclo in modo da permettere di inserire per ogni utente il suo messaggio e sarà poi alla stored procedure la responsabilità di gestire i dati. Ho applicato una sorta di pattern GRASP in modo da avere che ogni utente ha il suo messaggio e non un unico messaggio dove è condiviso per tutti, quindi da avere un low coupling tra gli utenti (ogni row contentente il messaggio dipende solo dall'utente indirizzato)

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.warn\_new\_note ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.warn\_new\_note AFTER INSERT ON BachecaElettronicadb.Nota FOR EACH ROW

BEGIN

declare username VARCHAR(45);

declare loop\_username INT DEFAULT 0;

declare cur\_username\_ucc\_user CURSOR FOR SELECT seguiti.UCC\_Username FROM BachecaElettronicadb.Nota AS nota JOIN BachecaElettronicadb.`Seguito-UCC` AS seguiti ON nota.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare cur\_username\_uscc\_user CURSOR FOR SELECT seguiti.USCC\_Username FROM BachecaElettronicadb.Nota AS nota JOIN BachecaElettronicadb.`Seguito-USCC` AS seguiti ON nota.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_username = 1;

-- Seleziona tutti gli utenti che seguono l'annuncio e inserisce una nuova entry in modo da notificare l'evento

OPEN cur\_username\_ucc\_user;

insert\_new\_ucc\_warning: LOOP

FETCH cur\_username\_ucc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_ucc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.ID, 'Nota Annuncio', username, now(), 'Nuovo Inserimento');

end LOOP insert\_new\_ucc\_warning;

CLOSE cur\_username\_ucc\_user;

SET loop\_username = 0;

SET username = NULL;

OPEN cur\_username\_uscc\_user;

insert\_new\_uscc\_warning: LOOP

FETCH cur\_username\_uscc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_uscc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.ID, 'Nota Annuncio', username, now(), 'Nuovo Inserimento');

end LOOP insert\_new\_uscc\_warning;

CLOSE cur\_username\_uscc\_user;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Avviso Nota Eliminata** ----------------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.warn\_note\_deleted ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.warn\_note\_deleted AFTER DELETE ON BachecaElettronicadb.Nota FOR EACH ROW

BEGIN

declare username VARCHAR(45);

declare loop\_username INT DEFAULT 0;

declare cur\_username\_ucc\_user CURSOR FOR SELECT seguiti.UCC\_Username FROM BachecaElettronicadb.Nota AS nota JOIN BachecaElettronicadb.`Seguito-UCC` AS seguiti ON nota.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare cur\_username\_uscc\_user CURSOR FOR SELECT seguiti.USCC\_Username FROM BachecaElettronicadb.Nota AS nota JOIN BachecaElettronicadb.`Seguito-USCC` AS seguiti ON nota.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_username = 1;

/\*NOTA

\* L'idea del trigger è di soddisfare la regola aziendale di avvisare ogni utente

\* che segue l'annuncio di una nota eliminata. Viene eseguito un ciclo in modo da permettere

\* di inserire per ogni utente il suo messaggio e sarà poi alla stored procedure la responsabilità

\* di gestire i dati. Ho applicato una sorta di pattern GRASP in modo da avere che ogni utente

\* ha il suo messaggio e non un unico messaggio dove è condiviso per tutti, quindi da avere un low

\* coupling tra gli utenti (ogni row contentente il messaggio dipende solo dall'utente indirizzato)

\*/

-- Seleziona tutti gli utenti che seguono l'annuncio e inserisce una nuova entry in modo da notificare l'evento

OPEN cur\_username\_ucc\_user;

insert\_new\_ucc\_warning: LOOP

FETCH cur\_username\_ucc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_ucc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (OLD.ID, 'Nota Annuncio', username, now(), 'Nota Eliminata');

end LOOP insert\_new\_ucc\_warning;

CLOSE cur\_username\_ucc\_user;

SET loop\_username = 0;

SET username = NULL;

OPEN cur\_username\_uscc\_user;

insert\_new\_uscc\_warning: LOOP

FETCH cur\_username\_uscc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_uscc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (OLD.ID, 'Nota Annuncio', username, now(), 'Nota Eliminata');

end LOOP insert\_new\_uscc\_warning;

CLOSE cur\_username\_uscc\_user;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Avviso Commento Inserito** ------------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.warn\_new\_comment ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.warn\_new\_comment AFTER INSERT ON BachecaElettronicadb.Commento FOR EACH ROW

BEGIN

declare username VARCHAR(45);

declare loop\_username INT DEFAULT 0;

declare cur\_username\_ucc\_user CURSOR FOR SELECT seguiti.UCC\_Username FROM BachecaElettronicadb.Commento AS commento JOIN BachecaElettronicadb.`Seguito-UCC` AS seguiti ON commento.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare cur\_username\_uscc\_user CURSOR FOR SELECT seguiti.USCC\_Username FROM BachecaElettronicadb.Commento AS commento JOIN BachecaElettronicadb.`Seguito-USCC` AS seguiti ON commento.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_username = 1;

/\*NOTA

\* L'idea del trigger è di soddisfare la regola aziendale di avvisare ogni utente

\* che segue l'annuncio di un nuovo commento. Viene eseguito un ciclo in modo da permettere

\* di inserire per ogni utente il suo messaggio e sarà poi alla stored procedure la responsabilità

\* di gestire i dati. Ho applicato una sorta di pattern GRASP in modo da avere che ogni utente

\* ha il suo messaggio e non un unico messaggio dove è condiviso per tutti, quindi da avere un low

\* coupling tra gli utenti (ogni row contentente il messaggio dipende solo dall'utente indirizzato)

\*/

-- Seleziona tutti gli utenti che seguono l'annuncio e inserisce una nuova entry in modo da notificare l'evento

OPEN cur\_username\_ucc\_user;

insert\_new\_ucc\_warning: LOOP

FETCH cur\_username\_ucc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_ucc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.ID, 'Commento Annuncio', username, now(), 'Nuovo Commento');

end LOOP insert\_new\_ucc\_warning;

CLOSE cur\_username\_ucc\_user;

SET loop\_username = 0;

SET username = NULL;

OPEN cur\_username\_uscc\_user;

insert\_new\_uscc\_warning: LOOP

FETCH cur\_username\_uscc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_uscc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.ID, 'Commento Annuncio', username, now(), 'Nuovo Commento');

end LOOP insert\_new\_uscc\_warning;

CLOSE cur\_username\_uscc\_user;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Avviso Commento Inserito** ------------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.warn\_comment\_deleted ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.warn\_comment\_deleted AFTER DELETE ON BachecaElettronicadb.Commento FOR EACH ROW

BEGIN

declare username VARCHAR(45);

declare loop\_username INT DEFAULT 0;

declare cur\_username\_ucc\_user CURSOR FOR SELECT seguiti.UCC\_Username FROM BachecaElettronicadb.Commento AS commento JOIN BachecaElettronicadb.`Seguito-UCC` AS seguiti ON commento.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare cur\_username\_uscc\_user CURSOR FOR SELECT seguiti.USCC\_Username FROM BachecaElettronicadb.Commento AS commento JOIN BachecaElettronicadb.`Seguito-USCC` AS seguiti ON commento.Annuncio\_Codice = seguiti.Annuncio\_Codice;

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_username = 1;

/\*NOTA

\* L'idea del trigger è di soddisfare la regola aziendale di avvisare ogni utente

\* che segue l'annuncio che è stato eliminato il commento. Viene eseguito un ciclo in modo da permettere

\* di inserire per ogni utente il suo messaggio e sarà poi alla stored procedure la responsabilità

\* di gestire i dati. Ho applicato una sorta di pattern GRASP in modo da avere che ogni utente

\* ha il suo messaggio e non un unico messaggio dove è condiviso per tutti, quindi da avere un low

\* coupling tra gli utenti (ogni row contentente il messaggio dipende solo dall'utente indirizzato)

\*/

-- Seleziona tutti gli utenti che seguono l'annuncio e inserisce una nuova entry in modo da notificare l'evento

OPEN cur\_username\_ucc\_user;

insert\_new\_ucc\_warning: LOOP

FETCH cur\_username\_ucc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_ucc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (OLD.ID, 'Commento Annuncio', username, now(), 'Commento Eliminato');

end LOOP insert\_new\_ucc\_warning;

CLOSE cur\_username\_ucc\_user;

SET loop\_username = 0;

SET username = NULL;

OPEN cur\_username\_uscc\_user;

insert\_new\_uscc\_warning: LOOP

FETCH cur\_username\_uscc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_uscc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (OLD.ID, 'Commento Annuncio', username, now(), 'Commento Eliminato');

end LOOP insert\_new\_uscc\_warning;

CLOSE cur\_username\_uscc\_user;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- Avviso Annuncio Modificato ----------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.warn\_ad\_updated ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.warn\_ad\_updated AFTER UPDATE ON BachecaElettronicadb.Annuncio FOR EACH ROW

BEGIN

declare username VARCHAR(45) DEFAULT NULL;

declare loop\_username INT DEFAULT 0;

declare cur\_username\_ucc\_user CURSOR FOR SELECT seguiti.UCC\_Username FROM BachecaElettronicadb.Annuncio AS annuncio JOIN BachecaElettronicadb.`Seguito-UCC` AS seguiti ON annuncio.Codice = seguiti.Annuncio\_Codice;

declare cur\_username\_uscc\_user CURSOR FOR SELECT seguiti.USCC\_Username FROM BachecaElettronicadb.Annuncio AS annuncio JOIN BachecaElettronicadb.`Seguito-USCC` AS seguiti ON annuncio.Codice = seguiti.Annuncio\_Codice;

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_username = 1;

/\*NOTA

\* L'idea del trigger è di soddisfare la regola aziendale di avvisare ogni utente

\* che segue l'annuncio che è stato modificato. Viene eseguito un ciclo in modo da permettere

\* di inserire per ogni utente il suo messaggio e sarà poi alla stored procedure la responsabilità

\* di gestire i dati. Ho applicato una sorta di pattern GRASP in modo da avere che ogni utente

\* ha il suo messaggio e non un unico messaggio dove è condiviso per tutti, quindi da avere un low

\* coupling tra gli utenti (ogni row contentente il messaggio dipende solo dall'utente indirizzato)

\*/

-- Seleziona tutti gli utenti che seguono l'annuncio e inserisce una nuova entry in modo da notificare l'evento

OPEN cur\_username\_ucc\_user;

insert\_new\_ucc\_warning: LOOP

FETCH cur\_username\_ucc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_ucc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.Codice, 'Annuncio', username, now(), 'Annuncio Modificato');

end LOOP insert\_new\_ucc\_warning;

CLOSE cur\_username\_ucc\_user;

SET loop\_username = 0;

SET username = NULL;

OPEN cur\_username\_uscc\_user;

insert\_new\_uscc\_warning: LOOP

FETCH cur\_username\_uscc\_user INTO username;

IF loop\_username = 1 THEN

LEAVE insert\_new\_uscc\_warning;

end IF;

INSERT INTO BachecaElettronicadb.Notifica(Codice, Tipo\_Notifica, Username\_Utente, Data, Messaggio)

VALUES (NEW.Codice, 'Annuncio', username, now(), 'Annuncio Modificato');

end LOOP insert\_new\_uscc\_warning;

CLOSE cur\_username\_uscc\_user;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Controlla il Codice Fiscale** ---------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.check\_tax\_code ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.check\_tax\_code BEFORE INSERT ON BachecaElettronicadb.InformazioneAnagrafica FOR EACH ROW

BEGIN

-- Controlla che il formato del codice fiscale sia valido

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 '45019' set message\_text = 'Incorrect Tax Code';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Controlla Commento Inserimento** ------------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.check\_commento\_insert ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.check\_commento\_insert BEFORE INSERT ON BachecaElettronicadb.Commento FOR EACH ROW

BEGIN

-- Controlla che l'annuncio sia attivo

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = NEW.Annuncio\_Codice AND BachecaElettronicadb.Annuncio.Stato='Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Controlla Commento Cancellazione** ----------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.check\_commento\_delete ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.check\_commento\_delete BEFORE DELETE ON BachecaElettronicadb.Commento FOR EACH ROW

BEGIN

-- Controlla che l'annuncio sia attivo

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = OLD.Annuncio\_Codice AND BachecaElettronicadb.Annuncio.Stato='Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Controlla Annuncio in Seguito UCC** ---------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.check\_annuncio\_seguito\_ucc ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.check\_annuncio\_seguito\_ucc AFTER INSERT ON BachecaElettronicadb.`Seguito-UCC` FOR EACH ROW

BEGIN

-- Controlla che l'annuncio esista e sia attivo

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = NEW.Annuncio\_Codice)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = NEW.Annuncio\_Codice and BachecaElettronicadb.Annuncio.Stato = 'Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Controlla Annuncio in Seguito USCC** --------------------------------

DELIMITER //

DROP TRIGGER IF EXISTS BachecaElettronicadb.check\_annuncio\_seguito\_uscc ;

CREATE DEFINER = CURRENT\_USER TRIGGER BachecaElettronicadb.check\_annuncio\_seguito\_uscc AFTER INSERT ON BachecaElettronicadb.`Seguito-USCC` FOR EACH ROW

BEGIN

-- Controlla che l'annuncio esista e sia attivo

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = NEW.Annuncio\_Codice)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = NEW.Annuncio\_Codice and BachecaElettronicadb.Annuncio.Stato = 'Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

## Eventi

Nessun evento previsto.

## Viste

Nessuna vista prevista.

## Stored Procedures e transazioni

* Inserisce un nuovo utente UCC nel database;

-- **Registrazione di un utente UCC** --------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.registra\_utente\_UCC ;

CREATE PROCEDURE BachecaElettronicadb.registra\_utente\_UCC (IN username VARCHAR(45), IN password VARCHAR(45), IN numeroCarta VARCHAR(16), IN dataScadenza DATE, IN cvc INT, IN cf\_anagrafico VARCHAR(16), IN cognome VARCHAR(20), IN nome VARCHAR(20), IN indirizzoDiResidenza VARCHAR(20), IN cap INT, IN indirizzoDiFatturazione VARCHAR(20), IN tipoRecapitoPreferito VARCHAR(20), IN recapitoPreferito VARCHAR(40), IN tipoRecapitoNonPreferito VARCHAR(20), IN recapitoNonPreferito VARCHAR(40))

BEGIN

declare idStorico INT;

call BachecaElettronicadb.inserisci\_Storico(idStorico);

call BachecaElettronicadb.inserimentoInfoAnagrafiche(cf\_anagrafico, cognome, nome, indirizzoDiResidenza, cap, indirizzoDiFatturazione, tipoRecapitoPreferito, recapitoPreferito);

call BachecaElettronicadb.modifica\_RecapitoNonPreferito(tipoRecapitoNonPreferito, recapitoNonPreferito, cf\_anagrafico, null);

INSERT INTO BachecaElettronicadb.UCC (Username, Password, NumeroCarta, DataScadenza, CVC, StoricoConversazione\_ID, CF\_Anagrafico)

VALUES(username, md5(password), numeroCarta, dataScadenza, cvc, idStorico, cf\_anagrafico);

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Questa procedure ha la responsabilità di inserire un nuovo Storico nel momento della registrazione di un utente. È stato scelto un livello di isolamento per evitare meno errori possibili con la funzione LAST\_INSERT\_ID() che potrebbe creare letture errate in caso di un grande traffico nel sistema.

-- **Inserimento di uno Storico Conversazioni** --------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.inserisci\_Storico ;

CREATE PROCEDURE BachecaElettronicadb.inserisci\_Storico (OUT id INT)

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello perchè voglio che non ci siano letture

\* sporche sul valore dell'ID in caso concorrenza su questa procedura.

\* Il trucco è quello di usare il lock preso dalla write, che verrà restituito al commit.

\*/

SET transaction isolation level read committed;

start transaction;

INSERT INTO BachecaElettronicadb.StoricoConversazione (ID) VALUES(NULL);

SET id = LAST\_INSERT\_ID();

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Inserisce o Rimuove un recapito non preferito a seconda del parametro “rimuovi”

-- **Inserimento o Rimozione Recapito non preferito** --------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.modifica\_RecapitoNonPreferito ;

CREATE PROCEDURE BachecaElettronicadb.modifica\_RecapitoNonPreferito (IN tipo VARCHAR(20), IN recapito VARCHAR(40), IN cf\_anagrafico VARCHAR(16), IN rimuovi INT)

BEGIN

if (not exists (SELECT CF FROM BachecaElettronicadb.InformazioneAnagrafica WHERE BachecaElettronicadb.InformazioneAnagrafica.CF = cf\_anagrafico)) then

signal sqlstate '45004' set message\_text = "Tax code not found";

end if;

if(rimuovi is null) then

INSERT INTO BachecaElettronicadb.RecapitoNonPreferito (Recapito, Tipo, InformazioneAnagrafica\_CF) VALUES(recapito, tipo, cf\_anagrafico);

else

DELETE FROM BachecaElettronicadb.RecapitoNonPreferito

WHERE BachecaElettronicadb.RecapitoNonPreferito.InformazioneAnagrafica\_CF = cf\_anagrafico AND BachecaElettronicadb.RecapitoNonPreferito.Recapito = recapito AND BachecaElettronicadb.RecapitoNonPreferito.Tipo = tipo;

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Visualizza tutti i contatti dell’utente

-- **Visualizzazione contatti** ------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizza\_contatti ;

CREATE PROCEDURE BachecaElettronicadb.visualizza\_contatti (IN cf\_anagrafico VARCHAR(16), IN preferito INT)

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture inconsistenti sugli stessi dati

\*/

SET transaction read only;

SET transaction isolation level repeatable read;

start transaction;

if (not exists (SELECT CF FROM BachecaElettronicadb.InformazioneAnagrafica WHERE BachecaElettronicadb.InformazioneAnagrafica.CF = cf\_anagrafico)) then

signal sqlstate '45004' set message\_text = 'Tax code not found';

end if;

if (preferito is not null) then

SELECT RecapitoPreferito AS 'Recapito', tipoRecapitoPreferito AS 'Tipo', CF AS 'Codice Fiscale'

FROM BachecaElettronicadb.InformazioneAnagrafica

WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf\_anagrafico;

else

SELECT Recapito, Tipo, InformazioneAnagrafica\_CF AS 'Codice Fiscale'

FROM BachecaElettronicadb.RecapitoNonPreferito

WHERE BachecaElettronicadb.RecapitoNonPreferito.InformazioneAnagrafica\_CF=cf\_anagrafico;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Inserisce un nuovo utente USCC nel database

-- **Registrazione di un utente USCC** -----------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.registra\_utente\_USCC ;

CREATE PROCEDURE BachecaElettronicadb.registra\_utente\_USCC (IN username VARCHAR(45), IN password VARCHAR(45), IN cf\_anagrafico VARCHAR(16), IN cognome VARCHAR(20), IN nome VARCHAR(20), IN indirizzoDiResidenza VARCHAR(20), IN cap INT, IN indirizzoDiFatturazione VARCHAR(20), IN tipoRecapitoPreferito VARCHAR(20), IN recapitoPreferito VARCHAR(40), IN tipoRecapitoNonPreferito VARCHAR(20), IN recapitoNonPreferito VARCHAR(40))

BEGIN

declare idStorico INT;

call BachecaElettronicadb.inserisci\_Storico(idStorico);

call BachecaElettronicadb.inserimentoInfoAnagrafiche(cf\_anagrafico, cognome, nome, indirizzoDiResidenza, cap, indirizzoDiFatturazione, tipoRecapitoPreferito, recapitoPreferito);

call BachecaElettronicadb.modifica\_RecapitoNonPreferito(tipoRecapitoNonPreferito, recapitoNonPreferito, cf\_anagrafico, null);

INSERT INTO BachecaElettronicadb.USCC (Username, Password, StoricoConversazione\_ID, CF\_Anagrafico)

VALUES(username, md5(password), idStorico, cf\_anagrafico);

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Inserisce le informazioni Anagrafiche dell’utente. Viene usata nel momento della registrazione e nessun utente ha i privilegi di eseguirla direttamente

-- **Inserimento delle Informazioni Anagrafiche** ------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.inserimentoInfoAnagrafiche ;

CREATE PROCEDURE BachecaElettronicadb.inserimentoInfoAnagrafiche (IN cf VARCHAR(16), IN cognome VARCHAR(20), IN nome VARCHAR(20), IN indirizzoDiResidenza VARCHAR(20), IN cap INT, IN indirizzoDiFatturazione VARCHAR(20), IN tipoRecapitoPreferito VARCHAR(20), IN recapitoPreferito VARCHAR(40))

BEGIN

INSERT INTO BachecaElettronicadb.InformazioneAnagrafica (CF, Cognome, Nome, IndirizzoDiResidenza, CAP, IndirizzoDiFatturazione, tipoRecapitoPreferito, RecapitoPreferito)

VALUES(cf, cognome, nome, indirizzoDiResidenza, cap, indirizzoDiFatturazione, tipoRecapitoPreferito, recapitoPreferito);

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Implementata per soddisfare alla regola aziendale in cui tutti gli utenti, che seguono un annuncio, devono essere notificati di ogni loro nuova modifica

-- **Visualizzazione nuove notifiche** -----------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizza\_notifiche ;

CREATE PROCEDURE BachecaElettronicadb.visualizza\_notifiche (IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture sporche

\*/

SET transaction isolation level read committed;

start transaction;

/\*NOTA

\* Questa stored procedure è utilizzata per mostrare all'utente, che lo ha chiamato,

\* le sue nuove notifiche. Applica un concetto simile ad una coda di messaggi

\* dove è possibile fare un'estrazione dei messaggi richiesti ed infine eliminarli per \* non essere visualizzati di nuovo. È correlata ai trigger precedentemente descritti.

\*/

-- Mostra le nuove notifiche rimaste in coda

SELECT Codice, Tipo\_Notifica, Data, Messaggio

FROM BachecaElettronicadb.Notifica

WHERE BachecaElettronicadb.Notifica.Username\_Utente = username

ORDER BY Data ASC;

-- Elimina le notifiche visualizzate

DELETE FROM BachecaElettronicadb.Notifica

WHERE BachecaElettronicadb.Notifica.Username\_Utente = username;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Modifica le informazioni anagrafiche a seconda dei parametri NULL e NOT NULL

-- **Modifica delle Informazioni Anagrafiche** ---------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.modificaInfoAnagrafiche ;

CREATE PROCEDURE BachecaElettronicadb.modificaInfoAnagrafiche (IN cf VARCHAR(16), IN cognome VARCHAR(20), IN nome VARCHAR(20), IN indirizzoDiResidenza VARCHAR(20), IN cap INT, IN indirizzoDiFatturazione VARCHAR(20), IN tipoRecapitoPreferito VARCHAR(20), IN recapitoPreferito VARCHAR(40))

BEGIN

IF cognome IS NOT NULL THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.Cognome=cognome WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf;

END IF;

IF nome IS NOT NULL THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.Nome=nome WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf;

END IF;

IF indirizzoDiResidenza IS NOT NULL THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.IndirizzoDiResidenza=indirizzoDiResidenza WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf;

END IF;

IF indirizzoDiFatturazione IS NOT NULL THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.IndirizzoDiFatturazione=indirizzoDiFatturazione WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf;

END IF;

IF cap IS NOT NULL THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.CAP=cap, IndirizzoDiFatturazione=indirizzoDiFatturazione WHERE BachecaElettronicadb.InformazioneAnagrafica.CF=cf;

END IF;

IF ((tipoRecapitoPreferito IS NOT NULL) AND (recapitoPreferito IS NOT NULL)) THEN

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.TipoRecapitoPreferito = tipoRecapitoPreferito WHERE BachecaElettronicadb.InformazioneAnagrafica.CF = cf;

UPDATE BachecaElettronicadb.InformazioneAnagrafica SET BachecaElettronicadb.InformazioneAnagrafica.RecapitoPreferito = recapitoPreferito WHERE BachecaElettronicadb.InformazioneAnagrafica.CF = cf;

END IF;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione delle Informazioni Anagrafiche** --------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaInfoAnagrafiche ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaInfoAnagrafiche (IN cf VARCHAR(16), IN check\_owner INT, IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

if ((check\_owner IS NOT NULL) AND (username IS NOT NULL)) then

IF (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.CF\_Anagrafico = cf AND BachecaElettronicadb.UCC.Username = username)) then

if (not exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.CF\_Anagrafico = cf AND BachecaElettronicadb.USCC.Username = username)) then

signal sqlstate '45009' set message\_text = 'The Tax code is not yours';

end if;

end IF;

end if;

SELECT CF AS 'Codice Fiscale', Cognome, Nome, IndirizzoDiResidenza AS 'Indirizzo di Residenza', CAP, IndirizzoDiFatturazione AS 'Indirizzo di Fatturazione', tipoRecapitoPreferito AS 'Tipo Recapito', RecapitoPreferito AS 'Recapito Preferito'

FROM BachecaElettronicadb.InformazioneAnagrafica

WHERE BachecaElettronicadb.InformazioneAnagrafica.CF = cf;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento di una nuova categoria** --------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.inserimentoNuovaCategoria ;

CREATE PROCEDURE BachecaElettronicadb.inserimentoNuovaCategoria (IN nome VARCHAR(20))

BEGIN

INSERT INTO BachecaElettronicadb.Categoria (Nome) VALUES(nome);

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione categoria** -----------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaCategoria ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaCategoria ()

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture sporche

\* in caso di inserimento di una nuova categoria ma non andato a buon fine

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

SELECT Nome

FROM BachecaElettronicadb.Categoria;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento di un nuovo annuncio** ----------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.inserimentoNuovoAnnuncio ;

CREATE PROCEDURE BachecaElettronicadb.inserimentoNuovoAnnuncio (IN descrizione VARCHAR(100), IN importo INT, IN foto VARCHAR(9), IN ucc\_username VARCHAR(45), IN categoria\_nome VARCHAR(20))

BEGIN

if (foto IS NULL) then

INSERT INTO BachecaElettronicadb.Annuncio (Codice, Stato, Descrizione, Importo, UCC\_Username, Categoria\_Nome) VALUES(NULL, 'Attivo', descrizione, importo, ucc\_username, categoria\_nome);

else

INSERT INTO BachecaElettronicadb.Annuncio (Codice, Stato, Descrizione, Importo, Foto, UCC\_Username, Categoria\_Nome) VALUES(NULL, 'Attivo', descrizione, importo, 'Presente', ucc\_username, categoria\_nome);

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento o Rimozione di una foto in annuncio** -------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.modificaFotoAnnuncio ;

CREATE PROCEDURE BachecaElettronicadb.modificaFotoAnnuncio (IN codice INT, IN username VARCHAR(45), IN foto VARCHAR(9))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture inconsistenti

\* e per restituire il lock al commit

\*/

SET transaction isolation level repeatable read;

start transaction;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = codice)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = codice AND BachecaElettronicadb.Annuncio.UCC\_Username = username)) then

signal sqlstate '45008' set message\_text = 'You are not the owner of the ad';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = codice AND BachecaElettronicadb.Annuncio.Stato = 'Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

if (foto is not NULL) then

UPDATE BachecaElettronicadb.Annuncio

SET BachecaElettronicadb.Annuncio.Foto = 'Presente'

WHERE BachecaElettronicadb.Annuncio.Codice = codice;

else

UPDATE BachecaElettronicadb.Annuncio

SET BachecaElettronicadb.Annuncio.Foto = NULL

WHERE BachecaElettronicadb.Annuncio.Codice = codice;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizza annuncio** -----------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaAnnuncio ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaAnnuncio (IN annuncio\_codice INT, IN username VARCHAR(45), IN check\_owner INT)

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture inconsistenti sullo stesso dato

\*/

SET transaction read only;

SET transaction isolation level repeatable read;

start transaction;

if ((username IS NOT NULL) AND not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username=username)) then

signal sqlstate '45000' set message\_text = 'User not found';

end if;

if ((annuncio\_codice IS NOT NULL) AND not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice=annuncio\_codice)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

if ((check\_owner = 1) AND (username IS NOT NULL) AND (annuncio\_codice IS NOT NULL)) then

IF (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice=annuncio\_codice AND BachecaElettronicadb.Annuncio.UCC\_Username=username)) then

signal sqlstate '45008' set message\_text = 'You are not the owner of the ad';

end IF;

end if;

IF ((annuncio\_codice IS NOT NULL) AND (username IS NOT NULL)) THEN

SELECT Codice, Stato, Descrizione, Importo, Foto, UCC\_Username AS 'Proprietario', Categoria\_Nome AS 'Categoria'

FROM BachecaElettronicadb.Annuncio

WHERE BachecaElettronicadb.Annuncio.UCC\_Username=username AND BachecaElettronicadb.Annuncio.Codice=annuncio\_codice AND BachecaElettronicadb.Annuncio.Stato='Attivo';

ELSEIF ((annuncio\_codice IS NULL) AND (username IS NOT NULL)) THEN

SELECT Codice, Stato, Descrizione, Importo, Foto, UCC\_Username AS 'Proprietario', Categoria\_Nome AS 'Categoria'

FROM BachecaElettronicadb.Annuncio

WHERE BachecaElettronicadb.Annuncio.UCC\_Username=username AND BachecaElettronicadb.Annuncio.Stato='Attivo';

ELSEIF ((annuncio\_codice IS NOT NULL) AND (username IS NULL)) THEN

SELECT Codice, Stato, Descrizione, Importo, Foto, UCC\_Username AS 'Proprietario', Categoria\_Nome AS 'Categoria'

FROM BachecaElettronicadb.Annuncio

WHERE BachecaElettronicadb.Annuncio.Codice=annuncio\_codice AND BachecaElettronicadb.Annuncio.Stato='Attivo';

ELSE

SELECT Codice, Stato, Descrizione, Importo, Foto, UCC\_Username AS 'Proprietario', Categoria\_Nome AS 'Categoria'

FROM BachecaElettronicadb.Annuncio

WHERE BachecaElettronicadb.Annuncio.Stato='Attivo';

end IF;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento o Rimozione commento** ----------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.modificaCommento ;

CREATE PROCEDURE BachecaElettronicadb.modificaCommento (IN testo VARCHAR(45), IN annuncio\_codice INT, IN ID\_rimozione\_commento INT)

BEGIN

if (ID\_rimozione\_commento is null) then

INSERT INTO BachecaElettronicadb.Commento (Testo, Annuncio\_Codice) VALUES(testo, annuncio\_codice);

else

DELETE FROM BachecaElettronicadb.Commento

WHERE BachecaElettronicadb.Commento.ID = ID\_rimozione\_commento;

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione commento** ------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaCommento ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaCommento (IN annuncio\_codice INT)

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare le letture sporche

\* in caso di inserimento di un nuovo commento ma ancora non committato

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice=annuncio\_codice AND BachecaElettronicadb.Annuncio.Stato='Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

SELECT ID, Testo, Annuncio\_codice AS "Codice dell'annuncio"

FROM BachecaElettronicadb.Commento

WHERE BachecaElettronicadb.Commento.Annuncio\_Codice = annuncio\_codice;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento Conversazione** -----------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.inserimentoConversazione ;

CREATE PROCEDURE BachecaElettronicadb.inserimentoConversazione (OUT id\_conversazione INT, IN ucc\_username\_1 VARCHAR(45), ucc\_username\_2 VARCHAR(45), IN uscc\_username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare concorrenza sull'ID, usando il lock

\* ottenuto dal write che sarà poi rilasciato al commit.

\*/

SET transaction isolation level read committed;

start transaction;

INSERT INTO BachecaElettronicadb.Conversazione (Codice, UCC\_Username\_1, UCC\_Username\_2, USCC\_Username) VALUES(NULL, ucc\_username\_1, ucc\_username\_2, uscc\_username);

SET id\_conversazione = LAST\_INSERT\_ID();

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Seleziona lo storico di un utente UCC che viene utilizzata per tracciare le conversazioni

-- **Seleziona Storico UCC per la stored procedure** ---------------------

DELIMITER //

DROP FUNCTION IF EXISTS BachecaElettronicadb.seleziona\_Storico\_UCC ;

CREATE FUNCTION BachecaElettronicadb.seleziona\_Storico\_UCC (username VARCHAR(45)) RETURNS INT DETERMINISTIC

BEGIN

declare storico\_id\_ucc INT;

-- Prende il codice dello storico dell'utente

SELECT StoricoConversazione\_ID INTO storico\_id\_ucc

FROM BachecaElettronicadb.UCC

WHERE BachecaElettronicadb.UCC.Username = username;

RETURN storico\_id\_ucc;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Seleziona lo storico di un utente USCC che viene utilizzata per tracciare le conversazioni

-- **Seleziona Storico USCC per la stored procedure** --------------------

DELIMITER //

DROP FUNCTION IF EXISTS BachecaElettronicadb.seleziona\_Storico\_USCC ;

CREATE FUNCTION BachecaElettronicadb.seleziona\_Storico\_USCC (username VARCHAR(45)) RETURNS INT DETERMINISTIC

BEGIN

declare storico\_id\_uscc INT;

-- Prende il codice dello storico dell'utente

SELECT StoricoConversazione\_ID INTO storico\_id\_uscc

FROM BachecaElettronicadb.USCC

WHERE BachecaElettronicadb.USCC.Username = username;

RETURN storico\_id\_uscc;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Traccia le nuove conversazioni per gli utenti. Essa non viene usata direttamente dall’utente ma indirettamente tramite la stored procedure “invioMessaggio”

-- **Tracciamento Conversazioni** ----------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.traccia\_Conversazione ;

CREATE PROCEDURE BachecaElettronicadb.traccia\_Conversazione (IN conversazione\_codice INT, IN sender\_ucc\_username VARCHAR(45), IN receiver\_ucc\_username VARCHAR(45), IN receiver\_uscc\_username VARCHAR(45))

BEGIN

declare storico\_id\_ucc, storico\_id\_uscc INT;

if ((sender\_ucc\_username is not null) and (receiver\_uscc\_username is not null)) then

SET storico\_id\_ucc = seleziona\_Storico\_UCC(sender\_ucc\_username);

SET storico\_id\_uscc = seleziona\_Storico\_USCC(receiver\_uscc\_username);

INSERT INTO BachecaElettronicadb.Tracciato (Conversazione\_Codice, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_ucc); -- traccia lo storico di UCC

INSERT INTO BachecaElettronicadb.Tracciato (Conversazione\_Codice, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_uscc); -- traccia lo storico di USCC

INSERT INTO BachecaElettronicadb.ConversazioneCodice (CodiceConv, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_ucc); -- inserisce il nuovo codice della conversazione nello storico di UCC

INSERT INTO BachecaElettronicadb.ConversazioneCodice (CodiceConv, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_uscc); -- inserisce il nuovo codice della conversazione nello storico di USCC

elseif ((sender\_ucc\_username is not null) and (receiver\_ucc\_username is not null)) then

SET storico\_id\_ucc = seleziona\_Storico\_UCC(sender\_ucc\_username);

SET storico\_id\_uscc = seleziona\_Storico\_UCC(receiver\_ucc\_username);

INSERT INTO BachecaElettronicadb.Tracciato (Conversazione\_Codice, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_ucc); -- traccia lo storico di UCC

INSERT INTO BachecaElettronicadb.Tracciato (Conversazione\_Codice, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_uscc); -- traccia lo storico di UCC

INSERT INTO BachecaElettronicadb.ConversazioneCodice (CodiceConv, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_ucc); -- inserisce il nuovo codice della conversazione nello storico di UCC

INSERT INTO BachecaElettronicadb.ConversazioneCodice (CodiceConv, StoricoConversazione\_ID) VALUES(conversazione\_codice, storico\_id\_uscc); -- inserisce il nuovo codice della conversazione nello storico di UCC

else

signal sqlstate '45010' set message\_text = 'Error Parameters';

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Utilizzata per controllare di quale tipo è l’utente (UCC oppure USCC)

-- **Controlla il tipo di utente** ---------------------------------------

DELIMITER //

DROP FUNCTION IF EXISTS BachecaElettronicadb.controlla\_utente ;

CREATE FUNCTION BachecaElettronicadb.controlla\_utente (username VARCHAR(45)) RETURNS VARCHAR(4) DETERMINISTIC

BEGIN

if (exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = username)) then

RETURN 'UCC';

elseif (exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username = username)) then

RETURN 'USCC';

else

signal sqlstate '45000' set message\_text = 'User not found'; -- utente non trovato nel database

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Questa è una delle stored procedure più complicate. Il suo compito è di far comunicare due utenti e creare nuove conversazioni se questi non hanno avuto mai un contatto. È stata progettata in modo da non dare tutta la responsabilità solo ad essa ma separare i compiti con altre stored procedures o functions di appoggio. Il livello di isolamento scelto è stato repeatable read visto che si eseguono molte letture sullo stesso dato e quindi si cerca di non avere inconsistenze sui dat

-- **Invio messaggio da parte utente** -----------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.invioMessaggio ;

CREATE PROCEDURE BachecaElettronicadb.invioMessaggio (IN sender\_username VARCHAR(45), IN receiver\_username VARCHAR(45), IN testo VARCHAR(100))

BEGIN

declare conversazione\_codice INT DEFAULT NULL;

declare type\_user\_sender VARCHAR(4);

declare type\_user\_receiver VARCHAR(4);

declare sender\_is\_ucc, sender\_is\_uscc, receiver\_is\_ucc, receiver\_is\_uscc INT;

declare exit handler for sqlexception

BEGIN

rollback; -- rollback any changes made in the transaction

resignal; -- raise again the sql exceptionto the caller

END;

/\*NOTA

\* Ho scelto questo livello per evitare letture

\* inconsistenti sull'utente nel database, vista la necessita di fare

\* più di una lettura sullo stesso dato

\*/

SET transaction isolation level repeatable read;

start transaction;

if (sender\_username = receiver\_username) then

signal sqlstate '45010' set message\_text = 'Error parameters';

end if;

SET sender\_is\_ucc = 0;

SET sender\_is\_uscc = 0;

SET receiver\_is\_ucc = 0;

SET receiver\_is\_uscc = 0;

SET type\_user\_sender = controlla\_utente(sender\_username); -- Restituisce la natura dell'utente che invia il messaggio

SET type\_user\_receiver = controlla\_utente(receiver\_username); -- Restituisce la natura dell'utente che riceve il messaggio

-- Seleziona gli username dei due utenti

if (type\_user\_sender = 'UCC') then

SET sender\_is\_ucc = 1;

SELECT Username INTO sender\_username

FROM BachecaElettronicadb.UCC

WHERE BachecaElettronicadb.UCC.Username = sender\_username;

else

SET sender\_is\_uscc = 1;

SELECT Username INTO sender\_username

FROM BachecaElettronicadb.USCC

WHERE BachecaElettronicadb.USCC.Username = sender\_username;

end if;

if (type\_user\_receiver = 'UCC') then

SET receiver\_is\_ucc = 1;

SELECT Username INTO receiver\_username

FROM BachecaElettronicadb.UCC

WHERE BachecaElettronicadb.UCC.Username = receiver\_username;

else

SET receiver\_is\_uscc = 1;

SELECT Username INTO receiver\_username

FROM BachecaElettronicadb.USCC

WHERE BachecaElettronicadb.USCC.Username = receiver\_username;

end if;

if ((sender\_is\_uscc = 1) AND (receiver\_is\_uscc = 1)) then

signal sqlstate '45013' set message\_text = 'You cannot write to another USCC user';

end if;

commit; -- Inserisco un commit onde evitare un dead lock con le altre query e per non ottenere l'errore: Error 1568 (25001)

SET conversazione\_codice = controllo\_conversazione(NULL, sender\_username, receiver\_username); -- cerca il codice della conversazione

if(conversazione\_codice = -1) then -- conversazione non trovata quindi inesistente, si inserisce una nuova

-- traccia le conversazioni

if ((sender\_is\_ucc = 1) and (receiver\_is\_ucc = 1)) then

call BachecaElettronicadb.inserimentoConversazione(conversazione\_codice, sender\_username, receiver\_username, NULL); -- nuova conversazione

call traccia\_Conversazione(conversazione\_codice, sender\_username, receiver\_username, NULL); -- inserisce tutti i vincoli

elseif ((sender\_is\_ucc = 1) and (receiver\_is\_uscc = 1)) then

call BachecaElettronicadb.inserimentoConversazione(conversazione\_codice, sender\_username, NULL, receiver\_username);

call traccia\_Conversazione(conversazione\_codice, sender\_username, NULL, receiver\_username);

else -- (sender\_is\_uscc = 1) and (receiver\_is\_ucc = 1)

call BachecaElettronicadb.inserimentoConversazione(conversazione\_codice, receiver\_username, NULL, sender\_username);

call traccia\_Conversazione(conversazione\_codice, receiver\_username, NULL, sender\_username);

end if;

end if;

/\* NOTA

\* Non è presente la condizione (sender\_is\_uscc = 1) and (receiver\_is\_uscc = 1))

\* perchè non è possibile che due utenti USCC si scrivano

\*/

if (conversazione\_codice is not null) then

INSERT INTO BachecaElettronicadb.Messaggio (Data, Conversazione\_Codice, Testo) -- conversazione esistente, allora inserisco il messaggio

VALUES (now(), conversazione\_codice, testo);

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Funzione di appoggio per altre stored procedure, utilizzata per controllare se gli utenti (passati nei parametri) abbiano l’autorizzazione ad accedere a quella conversazione oppure utilizzata per cercare una conversazione tra i due utenti (facendo delle combinazioni sugli username)

-- **Controllo Privacy e Ricerca conversazione** -------------------------

DELIMITER //

DROP FUNCTION IF EXISTS BachecaElettronicadb.controllo\_conversazione ;

CREATE FUNCTION BachecaElettronicadb.controllo\_conversazione (id\_conversation\_check INT, sender\_username VARCHAR(45), receiver\_username VARCHAR(45)) RETURNS INT DETERMINISTIC

BEGIN

declare conversazione\_codice\_controllo INT DEFAULT NULL;

-- Ricerca del codice della conversazione, basando la ricerca sulla combinazione degli username

if (id\_conversation\_check is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.UCC\_Username\_1 = sender\_username and BachecaElettronicadb.Conversazione.USCC\_Username = receiver\_username;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.UCC\_Username\_1 = sender\_username and BachecaElettronicadb.Conversazione.UCC\_Username\_2 = receiver\_username;

end if;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.UCC\_Username\_1 = receiver\_username and BachecaElettronicadb.Conversazione.UCC\_Username\_2 = sender\_username;

end if;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.UCC\_Username\_1 = receiver\_username and BachecaElettronicadb.Conversazione.USCC\_Username = sender\_username;

end if;

if(conversazione\_codice\_controllo is null) then -- errore, conversazione non trovata quindi inesistente

SET conversazione\_codice\_controllo = -1;

end if;

end if;

-- Controlla che gli utenti, con gli username passati, siano autorizzati ad accedere al codice di conversazione passato

if (id\_conversation\_check is not null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice = id\_conversation\_check and BachecaElettronicadb.Conversazione.UCC\_Username\_1 = sender\_username and BachecaElettronicadb.Conversazione.USCC\_Username = receiver\_username;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice = id\_conversation\_check and BachecaElettronicadb.Conversazione.UCC\_Username\_1 = sender\_username and BachecaElettronicadb.Conversazione.UCC\_Username\_2 = receiver\_username;

end if;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice = id\_conversation\_check and BachecaElettronicadb.Conversazione.UCC\_Username\_1 = receiver\_username and BachecaElettronicadb.Conversazione.UCC\_Username\_2 = sender\_username;

end if;

if(conversazione\_codice\_controllo is null) then

SELECT Codice into conversazione\_codice\_controllo

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice = id\_conversation\_check and BachecaElettronicadb.Conversazione.UCC\_Username\_1 = receiver\_username and BachecaElettronicadb.Conversazione.USCC\_Username = sender\_username;

end if;

if(conversazione\_codice\_controllo is null) then -- errore, conversazione non trovata quindi inesistente

SET conversazione\_codice\_controllo = -1;

end if;

end if;

RETURN conversazione\_codice\_controllo;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizza messaggio** ----------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaMessaggio ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaMessaggio (IN id\_conversation INT, sender\_username VARCHAR(45), IN receiver\_username VARCHAR(45))

BEGIN

declare conversazione\_codice INT DEFAULT NULL;

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

-- sender\_username è l'username dell'utente che sta chiamando la stored procedure

-- receiver\_username è l'username dell'utente con cui sender\_username ha avuto una conversazione

-- l'utente ha passato un codice di conversazione, probabilmente visualizzando lo storico

if(id\_conversation is not null) then

SET conversazione\_codice = controllo\_conversazione(id\_conversation, sender\_username, receiver\_username);

IF (conversazione\_codice = -1) then

signal sqlstate '45012' set message\_text = 'You are not authorized to access this conversation';

end IF;

end if;

if(id\_conversation is null) then

SET conversazione\_codice = controllo\_conversazione(NULL, sender\_username, receiver\_username);

end if;

if(conversazione\_codice = -1) then -- errore, conversazione non trovata quindi inesistente

signal sqlstate '45011' set message\_text = 'Conversation not found';

else

SELECT Testo, Data

FROM BachecaElettronicadb.Messaggio

WHERE BachecaElettronicadb.Messaggio.Conversazione\_Codice = conversazione\_codice

ORDER BY Data ASC;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione Storico UCC** ---------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaStorico\_UCC ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaStorico\_UCC (IN ucc\_username VARCHAR(45))

BEGIN

declare idStorico INT;

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

-- Trova il codice dello storico creato nel momento della registrazione

SELECT StoricoConversazione\_ID INTO idStorico FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = ucc\_username;

-- Stampa tutte le conversazioni dell'utente con il relativo partecipante

SELECT Codice, UCC\_Username\_2 AS 'Username utente UCC', USCC\_Username AS 'Username utente USCC'

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice in (SELECT CodiceConv

FROM BachecaElettronicadb.ConversazioneCodice

WHERE BachecaElettronicadb.ConversazioneCodice.StoricoConversazione\_ID = idStorico);

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione Storico USCC** --------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaStorico\_USCC ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaStorico\_USCC (IN uscc\_username VARCHAR(45))

BEGIN

declare idStorico INT;

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

-- Trova il codice dello storico creato nel momento della registrazione

SELECT StoricoConversazione\_ID INTO idStorico FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username = uscc\_username;

-- Stampa tutte le conversazioni dell'utente con il relativo partecipante

SELECT Codice, UCC\_Username\_1 AS 'Username utente UCC', UCC\_Username\_2 AS 'Username utente UCC'

FROM BachecaElettronicadb.Conversazione

WHERE BachecaElettronicadb.Conversazione.Codice in (SELECT CodiceConv

FROM BachecaElettronicadb.ConversazioneCodice

WHERE BachecaElettronicadb.ConversazioneCodice.StoricoConversazione\_ID = idStorico);

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Seguire un annuncio** -----------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.segui\_Annuncio ;

CREATE PROCEDURE BachecaElettronicadb.segui\_Annuncio (IN codice\_annuncio INT, IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction isolation level read committed;

start transaction;

/\* NOTA

\* L'idea è: Verifica che l'utente sia un UCC, altrimenti controlla se è un USCC.

\* Se non fosse nè UCC e nè USCC allora l'utente non esiste

\*/

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = username)) then

IF (not exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username = username)) THEN

signal sqlstate '45000' set message\_text = 'User not found';

ELSE

INSERT INTO BachecaElettronicadb.`Seguito-USCC`(USCC\_Username, Annuncio\_Codice) VALUES(username, codice\_annuncio);

end IF;

else

INSERT INTO BachecaElettronicadb.`Seguito-UCC`(UCC\_Username, Annuncio\_Codice) VALUES(username, codice\_annuncio);

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione degli Annunci seguiti** -----------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizza\_Annunci\_Seguiti ;

CREATE PROCEDURE BachecaElettronicadb.visualizza\_Annunci\_Seguiti (IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

/\* NOTA

\* L'idea è: Verifica che l'utente sia un UCC, altrimenti controlla se è un USCC.

\* Se non fosse nè UCC e nè USCC allora l'utente non esiste

\*/

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = username)) then

IF (not exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username = username)) THEN

signal sqlstate '45000' set message\_text = 'User not found';

ELSE

SELECT annuncio.Codice, annuncio.Stato, annuncio.UCC\_Username AS 'Proprietario'

FROM BachecaElettronicadb.`Seguito-USCC` AS seguiti JOIN BachecaElettronicadb.Annuncio AS annuncio ON seguiti.Annuncio\_Codice = annuncio.Codice

WHERE seguiti.USCC\_Username = username;

end IF;

else

SELECT annuncio.Codice, annuncio.Stato, annuncio.UCC\_Username AS 'Proprietario'

FROM BachecaElettronicadb.`Seguito-UCC` AS seguiti JOIN BachecaElettronicadb.Annuncio AS annuncio ON seguiti.Annuncio\_Codice = annuncio.Codice

WHERE seguiti.UCC\_Username = username;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Inserimento o Rimozione nota in un Annuncio** -----------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.modificaNota ;

CREATE PROCEDURE BachecaElettronicadb.modificaNota (IN testo VARCHAR(45), IN annuncio\_codice INT, IN ID\_rimozione\_nota INT, IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction isolation level read committed;

start transaction;

-- possibile implementazione trigger che controlla se una nuova nota inserita sia riferita ad un annuncio attivo

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = annuncio\_codice and BachecaElettronicadb.Annuncio.UCC\_Username = username)) then

signal sqlstate '45008' set message\_text = 'You are not the owner of the ad';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = annuncio\_codice and BachecaElettronicadb.Annuncio.Stato = 'Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

if (ID\_rimozione\_nota is null) then

INSERT INTO BachecaElettronicadb.Nota (ID, Testo, Annuncio\_Codice) VALUES(NULL, testo, annuncio\_codice);

else

DELETE FROM BachecaElettronicadb.Nota

WHERE BachecaElettronicadb.Nota.ID = ID\_rimozione\_nota;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizza nota** ---------------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizzaNota ;

CREATE PROCEDURE BachecaElettronicadb.visualizzaNota (IN annuncio\_Codice INT)

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice = annuncio\_Codice and BachecaElettronicadb.Annuncio.Stato = 'Attivo')) then

signal sqlstate '45007' set message\_text = 'Ad not active now';

end if;

SELECT ID, Testo, Annuncio\_Codice AS "Codice dell'annuncio"

FROM BachecaElettronicadb.Nota

WHERE BachecaElettronicadb.Nota.Annuncio\_Codice = annuncio\_Codice;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Rimozione di un Annuncio** ------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.rimuoviAnnuncio ;

CREATE PROCEDURE BachecaElettronicadb.rimuoviAnnuncio (IN codice\_annuncio INT, IN ucc\_username VARCHAR(45))

BEGIN

/\*NOTA

\* Non ho impostato nessun livello di isolamento perchè

\* non c'è concorrenza sulla singolo annuncio, ovvero

\* solo un solo utente può modificare il suo annuncio.

\*/

-- Controlla che il paramentro sia corretto con la regola aziendale impostata

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice=codice\_annuncio and BachecaElettronicadb.Annuncio.Stato = 'Attivo' and BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

UPDATE BachecaElettronicadb.Annuncio

SET BachecaElettronicadb.Annuncio.Stato = 'Rimosso'

WHERE BachecaElettronicadb.Annuncio.Codice = codice\_annuncio AND BachecaElettronicadb.Annuncio.Stato = 'Attivo' and BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Annuncio Venduto** --------------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.vendutoAnnuncio ;

CREATE PROCEDURE BachecaElettronicadb.vendutoAnnuncio (IN codice\_annuncio INT, IN ucc\_username VARCHAR(45))

BEGIN

/\*NOTA

\* Non ho impostato nessun livello di isolamento perchè

\* non c'è concorrenza sulla singolo annuncio, ovvero

\* solo un solo utente può modificare il suo annuncio.

\*/

-- Controlla che il paramentro sia corretto con la regola aziendale impostata

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.Codice=codice\_annuncio and BachecaElettronicadb.Annuncio.Stato = 'Attivo' and BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username)) then

signal sqlstate '45006' set message\_text = 'Ad not found';

end if;

UPDATE BachecaElettronicadb.Annuncio

SET BachecaElettronicadb.Annuncio.Stato = 'Venduto'

WHERE BachecaElettronicadb.Annuncio.Codice = codice\_annuncio AND BachecaElettronicadb.Annuncio.Stato = 'Attivo' and BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Questa è un’altra stored procedure una delle più “complicate”. Ha il compito di generare un nuovo report e i privilegi sono solamente dell’amministratore. Crea un cursore sugli annunci venduti e ancora non calcolati dell’utente, che con un loop calcola la somma totale degli annunci, numero degli annunci ed altre informazioni da mandare in persistenza. Inoltre durante la progettazione si è deciso di lasciare la possibilità di riscuotere l’annuncio solamente all’amministratore che si è occupato di generare il report. La generazione di un nuovo report poteva essere implementato anche attraverso un trigger che esegue un AFTER UPDATE ON Annunci, dove lo stato era in “venduto” ma si è scelto di implementare una stored procedure, rispetto ad un trigger, in quanto essi potevano garantire un uso delle risorse migliore, perché il calcolo viene effettuato a discrezione dell'amministratore. Ovvero un trigger si sarebbe attivato ogni qualvolta che un annuncio venisse indicato come venduto, quindi per un elevato traffico nel sistema, questo portava una degradazione delle prestazioni e anche un inutile calcolo da fare immediatamente dal punto di vista delle specifiche.

*NOTA: La riscossione del report è solamente per scopi gestionali degli amministratori, in modo da tracciare i report da loro riscossi.*

-- **Generazione di un nuovo report** ------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.genera\_report ;

CREATE PROCEDURE BachecaElettronicadb.genera\_report (IN ucc\_username VARCHAR(45), IN amministratore\_username VARCHAR(45))

BEGIN

declare numeroCarta VARCHAR(16);

declare codice\_annuncio INT;

declare numero\_annunci INT DEFAULT 0;

declare importo\_totale INT DEFAULT 0;

declare loop\_amount INT DEFAULT 0;

declare cur\_id\_annuncio CURSOR FOR SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username and BachecaElettronicadb.Annuncio.Stato = 'Venduto';

declare CONTINUE HANDLER FOR NOT FOUND SET loop\_amount = 1;

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture inconsistenti

\* leggendo l'utente più di una volta

\*/

SET transaction isolation level repeatable read;

start transaction;

/\*NOTA

\* Ho scelto di implementare una stored procedure, rispetto ad un trigger, in quanto essi potevano garantire un uso delle risorse migliore, perché

\* il calcolo viene effettuato a discrezione dell'amministratore.

\* Ovvero un trigger si sarebbe attivato ogni qualvolta che un annuncio venisse indicato come venduto, quindi per un elevato traffico nel

\* sistema, questo portava una degradazione delle prestazioni e anche un inutile calcolo da fare immediatamente dal punto di vista delle specifiche.

\*/

/\*NOTA

\* L'idea della stored procedure è:

\* Calcolare la somma degli importi di tutti gli annunci venduti non calcolati e la percentuale per l'amministratore che ha chiamanto la stored procedure

\*/

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = ucc\_username)) then

signal sqlstate '45000' set message\_text = 'User not found';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Annuncio WHERE BachecaElettronicadb.Annuncio.UCC\_Username = ucc\_username and BachecaElettronicadb.Annuncio.Stato = 'Venduto')) then

signal sqlstate '45014' set message\_text = 'This user has not sold any ads';

end if;

-- Cerca il numero di carta dell'utente

SELECT BachecaElettronicadb.UCC.NumeroCarta INTO numeroCarta

FROM BachecaElettronicadb.UCC

WHERE BachecaElettronicadb.UCC.Username = ucc\_username;

-- Calcola l'importo totale degli annunci venduti ancora non riscossi

OPEN cur\_id\_annuncio;

calculate\_amount: LOOP

FETCH cur\_id\_annuncio INTO codice\_annuncio;

IF loop\_amount = 1 THEN

LEAVE calculate\_amount;

end IF;

-- calcola gli importi dei singoli annunci ancora non calcolati

SET numero\_annunci = numero\_annunci + 1;

SET importo\_totale = importo\_totale + (SELECT Importo

FROM BachecaElettronicadb.Annuncio

WHERE BachecaElettronicadb.Annuncio.Codice = codice\_annuncio AND BachecaElettronicadb.Annuncio.Report\_calcolato = false);

-- Imposta a true l'attributo in modo tale da non essere ricalcolato

UPDATE BachecaElettronicadb.Annuncio

SET BachecaElettronicadb.Annuncio.Report\_calcolato = true

WHERE BachecaElettronicadb.Annuncio.Codice = codice\_annuncio;

end LOOP calculate\_amount;

CLOSE cur\_id\_annuncio;

INSERT INTO BachecaElettronicadb.Report (Codice, UCC\_Username, ImportoTotale, NumeroAnnunci, NumeroCarta, Data, Amministratore\_Username, Riscosso\_Amministratore, Importo\_Amministratore, Importo\_UCC)

VALUES(NULL, ucc\_username, importo\_totale, numero\_annunci, numeroCarta, now(), amministratore\_username, false, importo\_totale\*0.3, importo\_totale\*0.7);

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Riscossione di un report Amministratore** ---------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.riscossione\_report ;

CREATE PROCEDURE BachecaElettronicadb.riscossione\_report (IN codice\_report INT, IN ucc\_username VARCHAR(45), IN amministratore\_username VARCHAR(45))

BEGIN

declare importo INT;

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture inconsistenti con il report

\*/

SET transaction isolation level repeatable read;

start transaction;

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = ucc\_username)) then

signal sqlstate '45000' set message\_text = 'User not found';

end if;

if (not exists (SELECT Codice FROM BachecaElettronicadb.Report WHERE BachecaElettronicadb.Report.Codice = codice\_report and BachecaElettronicadb.Report.UCC\_Username = ucc\_username)) then

signal sqlstate '45015' set message\_text = 'Report not found';

end if;

if (exists (SELECT Codice FROM BachecaElettronicadb.Report WHERE BachecaElettronicadb.Report.Codice = codice\_report and BachecaElettronicadb.Report.Riscosso\_Amministratore = true)) then

signal sqlstate '45016' set message\_text = 'Report already collected';

end if;

-- Solo l'amministratore che ha generato il report può riscuotere

if (not exists (SELECT Codice FROM BachecaElettronicadb.Report WHERE BachecaElettronicadb.Report.Codice = codice\_report and BachecaElettronicadb.Report.Amministratore\_Username = amministratore\_username)) then

signal sqlstate '45017' set message\_text = 'You are not the owner of the report';

end if;

SELECT ImportoTotale INTO importo

FROM BachecaElettronicadb.Report

WHERE BachecaElettronicadb.Report.Codice = codice\_report;

UPDATE BachecaElettronicadb.Report

SET BachecaElettronicadb.Report.Riscosso\_Amministratore = true

WHERE BachecaElettronicadb.Report.Codice = codice\_report;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

-- **Visualizzazione report** --------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizza\_report ;

CREATE PROCEDURE BachecaElettronicadb.visualizza\_report (IN codice\_report INT, IN ucc\_username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = ucc\_username)) then

signal sqlstate '45000' set message\_text = 'User not found';

end if;

if (codice\_report is null) then

SELECT Codice, UCC\_Username, ImportoTotale, NumeroAnnunci, NumeroCarta, Importo\_UCC, Amministratore\_Username, Riscosso\_Amministratore, Importo\_Amministratore

FROM BachecaElettronicadb.Report

WHERE BachecaElettronicadb.Report.UCC\_Username = ucc\_username;

else

SELECT Codice, UCC\_Username, ImportoTotale, NumeroAnnunci, NumeroCarta, Importo\_UCC, Amministratore\_Username, Riscosso\_Amministratore, Importo\_Amministratore

FROM BachecaElettronicadb.Report

WHERE BachecaElettronicadb.Report.Codice = codice\_report and BachecaElettronicadb.Report.UCC\_Username = ucc\_username;

end if;

commit;

END//

DELIMITER ;

-- -------------------------------------------------------------------

* Visualizza i dati di base dell’utente. È stata risparmiata una stored procedure grazie alla doppia condizione

-- **Visualizzazione informazioni utente** -------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.visualizza\_Info\_Utente ;

CREATE PROCEDURE BachecaElettronicadb.visualizza\_Info\_Utente (IN username VARCHAR(45))

BEGIN

declare exit handler for sqlexception

begin

ROLLBACK; -- rollback any changes made in the transaction

RESIGNAL; -- raise again the sql exceptionto the caller

end;

/\*NOTA

\* Ho scelto questo livello per evitare letture sporche

\*/

SET transaction read only;

SET transaction isolation level read committed;

start transaction;

/\* Nota

\* L'idea è: Verifica che l'utente sia un UCC, altrimenti controlla se è un USCC.

\* Se non sarà nè UCC e nè USCC allora l'utente non esiste.

\*/

if (not exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username=username)) then

IF (not exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username=username)) THEN

signal sqlstate '45000' set message\_text = 'User not found';

ELSE

SELECT Username, Password, CF\_Anagrafico AS 'Codice Fiscale'

FROM BachecaElettronicadb.USCC

WHERE BachecaElettronicadb.USCC.Username = username;

end IF;

else

SELECT Username, Password, NumeroCarta AS 'Numero Carta', DataScadenza AS 'Scadenza', CVC, CF\_Anagrafico AS 'Codice Fiscale'

FROM BachecaElettronicadb.UCC

WHERE BachecaElettronicadb.UCC.Username = username;

end if;

commit;

END//

DELIMITER ;

-- **Login** -------------------------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS BachecaElettronicadb.login ;

CREATE PROCEDURE BachecaElettronicadb.login (IN username VARCHAR(45), IN password VARCHAR(45), OUT role INT)

BEGIN

SET role = -1;

if exists (SELECT Username FROM BachecaElettronicadb.Amministratore WHERE BachecaElettronicadb.Amministratore.Username = username and BachecaElettronicadb.Amministratore.Password = md5(password)) then

SET role = 1;

end if;

if exists (SELECT Username FROM BachecaElettronicadb.UCC WHERE BachecaElettronicadb.UCC.Username = username and BachecaElettronicadb.UCC.Password = md5(password)) and role = -1 then

SET role = 2;

end if;

if exists (SELECT Username FROM BachecaElettronicadb.USCC WHERE BachecaElettronicadb.USCC.Username = username and BachecaElettronicadb.USCC.Password = md5(password)) and role = -1 then

SET role = 3;

end if;

if (role = -1) then

SET role = 4;

end if;

END//

DELIMITER ;

-- -------------------------------------------------------------------

# Appendice: Implementazione

## Codice SQL per instanziare il database

## -- MySQL Workbench Forward Engineering

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='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema BachecaElettronicadb

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `BachecaElettronicadb` ;

-- -----------------------------------------------------

-- Schema BachecaElettronicadb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `BachecaElettronicadb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `BachecaElettronicadb` ;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Amministratore`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Amministratore` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Amministratore` (

`Username` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Username\_UNIQUE` ON `BachecaElettronicadb`.`Amministratore` (`Username` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Categoria`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Categoria` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Categoria` (

`Nome` VARCHAR(20) NOT NULL,

PRIMARY KEY (`Nome`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Nome\_UNIQUE` ON `BachecaElettronicadb`.`Categoria` (`Nome` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`InformazioneAnagrafica`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`InformazioneAnagrafica` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`InformazioneAnagrafica` (

`CF` VARCHAR(16) NOT NULL,

`Cognome` VARCHAR(20) NOT NULL,

`Nome` VARCHAR(20) NOT NULL,

`IndirizzoDiResidenza` VARCHAR(20) NOT NULL,

`CAP` INT NOT NULL,

`IndirizzoDiFatturazione` VARCHAR(20) NULL DEFAULT NULL,

`TipoRecapitoPreferito` ENUM('email', 'cellulare', 'social', 'sms') NOT NULL,

`RecapitoPreferito` VARCHAR(40) NOT NULL,

PRIMARY KEY (`CF`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `CF\_UNIQUE` ON `BachecaElettronicadb`.`InformazioneAnagrafica` (`CF` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`StoricoConversazione`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`StoricoConversazione` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`StoricoConversazione` (

`ID` INT NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `ID\_UNIQUE` ON `BachecaElettronicadb`.`StoricoConversazione` (`ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`UCC`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`UCC` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`UCC` (

`Username` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

`NumeroCarta` VARCHAR(16) NOT NULL,

`DataScadenza` DATE NOT NULL,

`CVC` INT NOT NULL,

`StoricoConversazione\_ID` INT NOT NULL,

`CF\_Anagrafico` VARCHAR(16) NOT NULL,

PRIMARY KEY (`Username`),

CONSTRAINT `fk\_UCC\_InformazioneAnagrafica1`

FOREIGN KEY (`CF\_Anagrafico`)

REFERENCES `BachecaElettronicadb`.`InformazioneAnagrafica` (`CF`),

CONSTRAINT `fk\_UCC\_StoricoConversazione1`

FOREIGN KEY (`StoricoConversazione\_ID`)

REFERENCES `BachecaElettronicadb`.`StoricoConversazione` (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Username\_UNIQUE` ON `BachecaElettronicadb`.`UCC` (`Username` ASC) VISIBLE;

CREATE INDEX `fk\_UCC\_StoricoConversazione1\_idx` ON `BachecaElettronicadb`.`UCC` (`StoricoConversazione\_ID` ASC) VISIBLE;

CREATE INDEX `fk\_UCC\_InformazioneAnagrafica1\_idx` ON `BachecaElettronicadb`.`UCC` (`CF\_Anagrafico` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Annuncio`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Annuncio` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Annuncio` (

`Codice` INT NOT NULL AUTO\_INCREMENT,

`Stato` ENUM('Attivo', 'Venduto', 'Rimosso') NOT NULL,

`Descrizione` VARCHAR(100) NOT NULL,

`Importo` INT NOT NULL,

`Foto` VARCHAR(9) NULL DEFAULT NULL,

`Report\_calcolato` TINYINT NOT NULL DEFAULT '0',

`UCC\_Username` VARCHAR(45) NOT NULL,

`Categoria\_Nome` VARCHAR(20) NOT NULL,

PRIMARY KEY (`Codice`),

CONSTRAINT `fk\_Annuncio\_Categoria1`

FOREIGN KEY (`Categoria\_Nome`)

REFERENCES `BachecaElettronicadb`.`Categoria` (`Nome`),

CONSTRAINT `fk\_Annuncio\_UCC`

FOREIGN KEY (`UCC\_Username`)

REFERENCES `BachecaElettronicadb`.`UCC` (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Codice\_UNIQUE` ON `BachecaElettronicadb`.`Annuncio` (`Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Annuncio\_UCC\_idx` ON `BachecaElettronicadb`.`Annuncio` (`UCC\_Username` ASC) VISIBLE;

CREATE INDEX `fk\_Annuncio\_Categoria1\_idx` ON `BachecaElettronicadb`.`Annuncio` (`Categoria\_Nome` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Commento`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Commento` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Commento` (

`ID` INT NOT NULL AUTO\_INCREMENT,

`Testo` VARCHAR(45) NOT NULL,

`Annuncio\_Codice` INT NOT NULL,

PRIMARY KEY (`ID`, `Annuncio\_Codice`),

CONSTRAINT `fk\_Commento\_Annuncio1`

FOREIGN KEY (`Annuncio\_Codice`)

REFERENCES `BachecaElettronicadb`.`Annuncio` (`Codice`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Annuncio\_Codice\_UNIQUE` ON `BachecaElettronicadb`.`Commento` (`Annuncio\_Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Commento\_Annuncio1\_idx` ON `BachecaElettronicadb`.`Commento` (`Annuncio\_Codice` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`USCC`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`USCC` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`USCC` (

`Username` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

`StoricoConversazione\_ID` INT NOT NULL,

`CF\_Anagrafico` VARCHAR(16) NOT NULL,

PRIMARY KEY (`Username`),

CONSTRAINT `fk\_USCC\_InformazioneAnagrafica1`

FOREIGN KEY (`CF\_Anagrafico`)

REFERENCES `BachecaElettronicadb`.`InformazioneAnagrafica` (`CF`),

CONSTRAINT `fk\_USCC\_StoricoConversazione1`

FOREIGN KEY (`StoricoConversazione\_ID`)

REFERENCES `BachecaElettronicadb`.`StoricoConversazione` (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Username\_UNIQUE` ON `BachecaElettronicadb`.`USCC` (`Username` ASC) VISIBLE;

CREATE INDEX `fk\_USCC\_StoricoConversazione1\_idx` ON `BachecaElettronicadb`.`USCC` (`StoricoConversazione\_ID` ASC) VISIBLE;

CREATE INDEX `fk\_USCC\_InformazioneAnagrafica1\_idx` ON `BachecaElettronicadb`.`USCC` (`CF\_Anagrafico` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Conversazione`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Conversazione` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Conversazione` (

`Codice` INT NOT NULL AUTO\_INCREMENT,

`UCC\_Username\_1` VARCHAR(45) NOT NULL,

`UCC\_Username\_2` VARCHAR(45) NULL DEFAULT NULL,

`USCC\_Username` VARCHAR(45) NULL DEFAULT NULL,

PRIMARY KEY (`Codice`),

CONSTRAINT `fk\_Conversazione\_UCC1`

FOREIGN KEY (`UCC\_Username\_1`)

REFERENCES `BachecaElettronicadb`.`UCC` (`Username`),

CONSTRAINT `fk\_Conversazione\_UCC2`

FOREIGN KEY (`UCC\_Username\_2`)

REFERENCES `BachecaElettronicadb`.`UCC` (`Username`),

CONSTRAINT `fk\_Conversazione\_USCC1`

FOREIGN KEY (`USCC\_Username`)

REFERENCES `BachecaElettronicadb`.`USCC` (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Codice\_UNIQUE` ON `BachecaElettronicadb`.`Conversazione` (`Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Conversazione\_UCC1\_idx` ON `BachecaElettronicadb`.`Conversazione` (`UCC\_Username\_1` ASC) VISIBLE;

CREATE INDEX `fk\_Conversazione\_USCC1\_idx` ON `BachecaElettronicadb`.`Conversazione` (`USCC\_Username` ASC) VISIBLE;

CREATE INDEX `fk\_Conversazione\_UCC2\_idx` ON `BachecaElettronicadb`.`Conversazione` (`UCC\_Username\_2` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`ConversazioneCodice`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`ConversazioneCodice` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`ConversazioneCodice` (

`CodiceConv` INT NOT NULL,

`StoricoConversazione\_ID` INT NOT NULL,

PRIMARY KEY (`StoricoConversazione\_ID`, `CodiceConv`),

CONSTRAINT `fk\_ConversazioneCodice\_StoricoConversazione1`

FOREIGN KEY (`StoricoConversazione\_ID`)

REFERENCES `BachecaElettronicadb`.`StoricoConversazione` (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_ConversazioneCodice\_StoricoConversazione1\_idx` ON `BachecaElettronicadb`.`ConversazioneCodice` (`StoricoConversazione\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Messaggio`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Messaggio` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Messaggio` (

`ID` INT NOT NULL AUTO\_INCREMENT,

`Data` DATETIME NOT NULL,

`Conversazione\_Codice` INT NOT NULL,

`Testo` VARCHAR(100) NOT NULL,

PRIMARY KEY (`ID`, `Conversazione\_Codice`),

CONSTRAINT `fk\_Messaggio\_Conversazione1`

FOREIGN KEY (`Conversazione\_Codice`)

REFERENCES `BachecaElettronicadb`.`Conversazione` (`Codice`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_Messaggio\_Conversazione1\_idx` ON `BachecaElettronicadb`.`Messaggio` (`Conversazione\_Codice` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Nota`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Nota` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Nota` (

`ID` INT NOT NULL AUTO\_INCREMENT,

`Testo` VARCHAR(45) NOT NULL,

`Annuncio\_Codice` INT NOT NULL,

PRIMARY KEY (`ID`, `Annuncio\_Codice`),

CONSTRAINT `fk\_Nota\_Annuncio1`

FOREIGN KEY (`Annuncio\_Codice`)

REFERENCES `BachecaElettronicadb`.`Annuncio` (`Codice`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Annuncio\_Codice\_UNIQUE` ON `BachecaElettronicadb`.`Nota` (`Annuncio\_Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Nota\_Annuncio1\_idx` ON `BachecaElettronicadb`.`Nota` (`Annuncio\_Codice` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Notifica`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Notifica` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Notifica` (

`Codice\_Notifica` INT NOT NULL AUTO\_INCREMENT,

`Codice` INT NOT NULL,

`Tipo\_Notifica` VARCHAR(45) NOT NULL,

`Username\_Utente` VARCHAR(45) NOT NULL,

`Data` DATETIME NOT NULL,

`Messaggio` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Codice\_Notifica`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Codice\_Notifica\_UNIQUE` ON `BachecaElettronicadb`.`Notifica` (`Codice\_Notifica` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`RecapitoNonPreferito`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`RecapitoNonPreferito` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`RecapitoNonPreferito` (

`Recapito` VARCHAR(40) NOT NULL,

`Tipo` ENUM('email', 'cellulare', 'social', 'sms') NOT NULL,

`InformazioneAnagrafica\_CF` VARCHAR(16) NOT NULL,

PRIMARY KEY (`InformazioneAnagrafica\_CF`, `Recapito`),

CONSTRAINT `fk\_RecapitoNonPreferito\_InformazioneAnagrafica1`

FOREIGN KEY (`InformazioneAnagrafica\_CF`)

REFERENCES `BachecaElettronicadb`.`InformazioneAnagrafica` (`CF`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_RecapitoNonPreferito\_InformazioneAnagrafica1\_idx` ON `BachecaElettronicadb`.`RecapitoNonPreferito` (`InformazioneAnagrafica\_CF` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Report`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Report` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Report` (

`Codice` INT NOT NULL AUTO\_INCREMENT,

`UCC\_Username` VARCHAR(45) NOT NULL,

`ImportoTotale` INT NOT NULL,

`NumeroAnnunci` INT NOT NULL,

`NumeroCarta` VARCHAR(16) NOT NULL,

`Data` DATE NOT NULL,

`Amministratore\_Username` VARCHAR(45) NOT NULL,

`Riscosso\_Amministratore` TINYINT NOT NULL DEFAULT '0',

`Importo\_Amministratore` INT NOT NULL DEFAULT '0',

`Importo\_UCC` INT NOT NULL DEFAULT '0',

PRIMARY KEY (`Codice`),

CONSTRAINT `fk\_Report\_UCC1`

FOREIGN KEY (`UCC\_Username`)

REFERENCES `BachecaElettronicadb`.`UCC` (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE UNIQUE INDEX `Codice\_UNIQUE` ON `BachecaElettronicadb`.`Report` (`Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Report\_UCC1\_idx` ON `BachecaElettronicadb`.`Report` (`UCC\_Username` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Seguito-UCC`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Seguito-UCC` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Seguito-UCC` (

`Annuncio\_Codice` INT NOT NULL,

`UCC\_Username` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Annuncio\_Codice`, `UCC\_Username`),

CONSTRAINT `fk\_Seguito-UCC\_Annuncio1`

FOREIGN KEY (`Annuncio\_Codice`)

REFERENCES `BachecaElettronicadb`.`Annuncio` (`Codice`),

CONSTRAINT `fk\_Seguito-UCC\_UCC1`

FOREIGN KEY (`UCC\_Username`)

REFERENCES `BachecaElettronicadb`.`UCC` (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_Seguito-UCC\_UCC1\_idx` ON `BachecaElettronicadb`.`Seguito-UCC` (`UCC\_Username` ASC) VISIBLE;

CREATE INDEX `fk\_Seguito-UCC\_Annuncio1\_idx` ON `BachecaElettronicadb`.`Seguito-UCC` (`Annuncio\_Codice` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Seguito-USCC`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Seguito-USCC` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Seguito-USCC` (

`Annuncio\_Codice` INT NOT NULL,

`USCC\_Username` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Annuncio\_Codice`, `USCC\_Username`),

CONSTRAINT `fk\_Seguito-USCC\_Annuncio1`

FOREIGN KEY (`Annuncio\_Codice`)

REFERENCES `BachecaElettronicadb`.`Annuncio` (`Codice`),

CONSTRAINT `fk\_Seguito-USCC\_USCC1`

FOREIGN KEY (`USCC\_Username`)

REFERENCES `BachecaElettronicadb`.`USCC` (`Username`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_Seguito-USCC\_Annuncio1\_idx` ON `BachecaElettronicadb`.`Seguito-USCC` (`Annuncio\_Codice` ASC) VISIBLE;

CREATE INDEX `fk\_Seguito-USCC\_USCC1\_idx` ON `BachecaElettronicadb`.`Seguito-USCC` (`USCC\_Username` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `BachecaElettronicadb`.`Tracciato`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `BachecaElettronicadb`.`Tracciato` ;

CREATE TABLE IF NOT EXISTS `BachecaElettronicadb`.`Tracciato` (

`Conversazione\_Codice` INT NOT NULL,

`StoricoConversazione\_ID` INT NOT NULL,

PRIMARY KEY (`Conversazione\_Codice`, `StoricoConversazione\_ID`),

CONSTRAINT `fk\_Tracciato\_Conversazione1`

FOREIGN KEY (`Conversazione\_Codice`)

REFERENCES `BachecaElettronicadb`.`Conversazione` (`Codice`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_Tracciato\_StoricoConversazione1`

FOREIGN KEY (`StoricoConversazione\_ID`)

REFERENCES `BachecaElettronicadb`.`StoricoConversazione` (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE INDEX `fk\_Tracciato\_StoricoConversazione1\_idx` ON `BachecaElettronicadb`.`Tracciato` (`StoricoConversazione\_ID` ASC) VISIBLE;

CREATE INDEX `fk\_Tracciato\_Conversazione1\_idx` ON `BachecaElettronicadb`.`Tracciato` (`Conversazione\_Codice` ASC) VISIBLE;

## Codice del Front-End

* **main.c**

#include "defines.h"

struct configuration conf;

static MYSQL \*conn;

typedef enum {

ADMINISTRATOR = 1,

UCC,

USCC,

FAILED\_LOGIN

} role\_t;

static role\_t attempt\_login(MYSQL \*login\_conn, char \*username, char \*password){

MYSQL\_STMT \*login\_stored\_procedure;

MYSQL\_BIND param[3];

int role = 0;

if(!setup\_prepared\_stmt(&login\_stored\_procedure, "call login(?, ?, ?)", login\_conn)) {

finish\_with\_stmt\_error(login\_conn, login\_stored\_procedure, "Unable to initialize ad statement", true);

goto err2;

}

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

param[0].buffer\_type = MYSQL\_TYPE\_STRING;

param[0].buffer = username;

param[0].buffer\_length = strlen(username);

param[1].buffer\_type = MYSQL\_TYPE\_STRING;

param[1].buffer = password;

param[1].buffer\_length = strlen(password);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &role;

param[2].buffer\_length = sizeof(role);

if (mysql\_stmt\_bind\_param(login\_stored\_procedure, param) != 0) {

print\_stmt\_error(login\_stored\_procedure, "Could not bind parameters for login");

goto err;

}

if (mysql\_stmt\_execute(login\_stored\_procedure) != 0) {

print\_stmt\_error(login\_stored\_procedure, NULL);

goto err;

}

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &role;

param[0].buffer\_length = sizeof(role);

if (mysql\_stmt\_bind\_result(login\_stored\_procedure, param)) {

print\_stmt\_error(login\_stored\_procedure, "Could not bind result");

goto err;

}

if (mysql\_stmt\_fetch(login\_stored\_procedure)) {

print\_stmt\_error(login\_stored\_procedure, "Could not fetch result");

goto err;

}

mysql\_stmt\_close(login\_stored\_procedure);

return role;

err:

mysql\_stmt\_close(login\_stored\_procedure);

err2:

return FAILED\_LOGIN;

}

void sign\_up(MYSQL \*account\_conn) {

MYSQL\_STMT \*prepared\_stmt;

char username[MAX\_LENGHT\_USERNAME], password[MAX\_LENGHT\_USERNAME], cf[MAX\_CF\_LENGHT], surname[20], name[20], residential\_address[20], billing\_address[20], type\_favorite\_contact[20], favorite\_contact[40], type\_not\_favorite\_contact[20], not\_favorite\_contact[40];

char cap\_string[5];

int cap;

char \*list\_choice\_type\_it[] = {"email", "cellulare", "social", "sms"};

char \*list\_choice\_type\_en[] = {"email", "mobile phone", "social", "sms"};

int lenght\_choice\_type = 4;

char favorite\_choice[20], not\_favorite\_choice[20];

bool request, request\_billing\_address, is\_null;

is\_null = 1;

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, username, false);

print\_color("Password: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, password, true);

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

cf[16] = '\0';

print\_color("Surname: ", "yellow", ' ', false, false, false, false);

getInput(20, surname, false);

print\_color("Name: ", "yellow", ' ', false, false, false, false);

getInput(20, name, false);

print\_color("Residential address: ", "yellow", ' ', false, false, false, false);

getInput(20, residential\_address, false);

print\_color("CAP: ", "light cyan", ' ', false, false, false, false);

getInput(5, cap\_string, false);

cap = atoi(cap\_string);

request\_billing\_address = yesOrNo("Do you want to edit your billing address?", 'y', 'n');

if(request\_billing\_address) {

print\_color("Billing address: ", "yellow", ' ', false, false, false, false);

getInput(20, billing\_address, false);

}

while(1) {

print\_color(" Which do you choose to enter as your favorite contact?", "orange", ' ', true, true, false, false);

for(int i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, favorite\_choice, false);

for(int i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], favorite\_choice) == 0) {

sprintf(type\_favorite\_contact, "%s", list\_choice\_type\_it[i]);

type\_favorite\_contact[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, favorite\_contact, false);

goto execution\_favorite\_contact;

}

}

}

execution\_favorite\_contact:

while(1) {

print\_color(" Which do you choose to enter as your not favorite contact?", "orange", ' ', true, true, false, false);

for(int i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, not\_favorite\_choice, false);

for(int i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], not\_favorite\_choice) == 0) {

sprintf(type\_not\_favorite\_contact, "%s", list\_choice\_type\_it[i]);

type\_not\_favorite\_contact[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, not\_favorite\_contact, false);

goto execution\_not\_favorite\_contact;

}

}

}

execution\_not\_favorite\_contact:

request = yesOrNo("Do you want to add your credit card number?", 'y', 'n');

if(request) {

char card\_number[17], cvc\_string[3], day\_string[3], month\_string[3], year\_string[5];

MYSQL\_TIME expiration\_date;

int cvc;

MYSQL\_BIND param[15];

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

restart\_card\_number:

print\_color("Card Number: ", "yellow", ' ', false, false, false, false);

getInput(17, card\_number, false);

card\_number[16] = '\0';

if (strlen(card\_number)<16) {

print\_color(" Error Card Number", "light red", ' ', false, true, false, true);

goto restart\_card\_number;

}

restart\_cvc:

print\_color("CVC: ", "yellow", ' ', false, false, false, false);

getInput(3, cvc\_string, false);

cvc\_string[3] = '\0';

cvc = atoi(cvc\_string);

if (strlen(cvc\_string)<3) {

print\_color(" Error CVC", "light red", ' ', false, true, false, true);

goto restart\_cvc;

}

print\_color("Enter the expiration date in numeric format", "orange", ' ', true, true, false, false);

restart\_day:

print\_color("Day: ", "light cyan", ' ', false, false, false, false);

getInput(3, day\_string, false);

day\_string[2] = '\0';

if(atoi(day\_string)>0 && atoi(day\_string)<32) {

print\_color(" Error Day", "light red", ' ', false, true, false, true);

goto restart\_day;

}

expiration\_date.day = atoi(day\_string);

restart\_month:

print\_color("Month: ", "light cyan", ' ', false, false, false, false);

getInput(3, month\_string, false);

month\_string[2] = '\0';

if(atoi(month\_string)>0 && atoi(month\_string)<13) {

print\_color(" Error Month", "light red", ' ', false, true, false, true);

goto restart\_month;

}

expiration\_date.month= atoi(month\_string);

restart\_year:

print\_color("Year: ", "light cyan", ' ', false, false, false, false);

getInput(5, year\_string, false);

year\_string[4] = '\0';

if(atoi(year\_string)>=2021 && atoi(year\_string)<2051) {

print\_color(" Error Year", "light red", ' ', false, true, false, true);

goto restart\_year;

}

expiration\_date.year= atoi(year\_string);

if(!request\_billing\_address)

param[10].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = username;

param[0].buffer\_length = strlen(username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = password;

param[1].buffer\_length = strlen(surname);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = card\_number;

param[2].buffer\_length = strlen(card\_number);

param[3].buffer\_type = MYSQL\_TYPE\_DATE;

param[3].buffer = (char \*) &expiration\_date;

param[3].buffer\_length = sizeof(expiration\_date);

param[4].buffer\_type = MYSQL\_TYPE\_LONG;

param[4].buffer = &cvc;

param[4].buffer\_length = sizeof(cvc);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = cf;

param[5].buffer\_length = strlen(cf);

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = surname;

param[6].buffer\_length = strlen(surname);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = name;

param[7].buffer\_length = strlen(name);

param[8].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[8].buffer = residential\_address;

param[8].buffer\_length = strlen(residential\_address);

param[9].buffer\_type = MYSQL\_TYPE\_LONG;

param[9].buffer = &cap;

param[9].buffer\_length = sizeof(favorite\_contact);

param[10].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[10].buffer = billing\_address;

param[10].buffer\_length = strlen(billing\_address);

param[11].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[11].buffer = type\_favorite\_contact;

param[11].buffer\_length = strlen(type\_favorite\_contact);

param[12].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[12].buffer = favorite\_contact;

param[12].buffer\_length = strlen(favorite\_contact);

param[13].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[13].buffer = type\_not\_favorite\_contact;

param[13].buffer\_length = strlen(type\_not\_favorite\_contact);

param[14].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[14].buffer = not\_favorite\_contact;

param[14].buffer\_length = strlen(not\_favorite\_contact);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call registra\_utente\_UCC (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", account\_conn))

finish\_with\_stmt\_error(account\_conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(account\_conn, prepared\_stmt, "Unable to bind parameters to create user\n", true);

} else {

MYSQL\_BIND param[12];

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

if(!request\_billing\_address)

param[7].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = username;

param[0].buffer\_length = strlen(username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = password;

param[1].buffer\_length = strlen(surname);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = cf;

param[2].buffer\_length = strlen(cf);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = surname;

param[3].buffer\_length = strlen(surname);

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[4].buffer = name;

param[4].buffer\_length = strlen(name);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = residential\_address;

param[5].buffer\_length = strlen(residential\_address);

param[6].buffer\_type = MYSQL\_TYPE\_LONG;

param[6].buffer = &cap;

param[6].buffer\_length = sizeof(favorite\_contact);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = billing\_address;

param[7].buffer\_length = strlen(billing\_address);

param[8].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[8].buffer = type\_favorite\_contact;

param[8].buffer\_length = strlen(type\_favorite\_contact);

param[9].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[9].buffer = favorite\_contact;

param[9].buffer\_length = strlen(favorite\_contact);

param[10].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[10].buffer = type\_not\_favorite\_contact;

param[10].buffer\_length = strlen(type\_not\_favorite\_contact);

param[11].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[11].buffer = not\_favorite\_contact;

param[11].buffer\_length = strlen(not\_favorite\_contact);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call registra\_utente\_USCC (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", account\_conn))

finish\_with\_stmt\_error(account\_conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(account\_conn, prepared\_stmt, "Unable to bind parameters to create user\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return;

}

else

print\_color(" Successfully created!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

// Sign in account

strcat(conf.username, username);

strcat(conf.password, password);

if(request)

run\_as\_ucc(account\_conn, conf);

else

run\_as\_uscc(account\_conn, conf);

return;

}

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); //connection initialized

if (conn == NULL) {

fprintf(stderr, "%s\n", mysql\_error(conn));

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, "%s\n", mysql\_error(conn));

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

printf("\n \t\t\033[40m\033[1;31m@@@@@@@ @@@@@@@@ @@@@@@@ @@ @@ @@@@@@@ @@@@@@@ @@@@@@@\033[0m\n");

printf(" \t\t\033[40m\033[1;31m@@ @ @@ @@ @@ @@ @@ @@ @@ @@ @@\033[0m\n");

printf(" \t\t\033[40m\033[1;31m@@@@@@@ @@@@@@@@ @@ @@@@@@@ @@@@@@@ @@ @@@@@@@\033[0m\n");

printf(" \t\t\033[40m\033[1;31m@@ @ @@ @@ @@ @@ @@ @@ @@ @@ @@\033[0m\n");

printf(" \t\t\033[40m\033[1;31m@@@@@@@ @@ @@ @@@@@@@ @@ @@ @@@@@@@ @@@@@@@ @@ @@\033[0m\n");

printf("\n\t\e[34m\e[1m@@@@@@@ @@ @@@@@@@ @@@@@@@ @@@@@@@ @@@@@@@ @@@@@@@ @@@@ @@ @@ @@@@@@@ @@@@@@@\e[22m\e[39m\n");

printf("\t\e[34m\e[1m@@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@\e[22m\e[39m\n");

printf("\t\e[34m\e[1m@@@@@@@ @@ @@@@@@@ @@ @@ @@@@@@@ @@ @@ @@ @@ @@ @@ @@ @@@@@@@\e[22m\e[39m\n");

printf("\t\e[34m\e[1m@@ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@@@ @@ @@ @@ @@\e[22m\e[39m\n");

printf("\t\e[34m\e[1m@@@@@@@ @@@@@@@@ @@@@@@@ @@ @@ @@ @@ @@@@@@@ @@ @@@ @@ @@@@@@@ @@ @@\e[22m\e[39m\n\n");

if (yesOrNo("\nDo you want to sign up?", 'y', 'n')) {

sign\_up(conn);

goto restart;

}

goto start;

restart: if (yesOrNo("Do you want continue?", 'y', 'n') == false) goto exit;

start:

print\_color(" ENTER YOUR USERNAME AND PASSWORD", "pink", ' ', true, true, false, true);

print\_color("Username: ", "yellow", ' ', false, false, true, false);

getInput(45, conf.username, false);

print\_color("Password: ", "light cyan", ' ', false, false, false, false);

getInput(45, conf.password, true);

role = attempt\_login(conn, conf.username, conf.password);

switch(role){

case ADMINISTRATOR:

run\_as\_administrator(conn, conf);

break;

case UCC:

run\_as\_ucc(conn, conf);

break;

case USCC:

run\_as\_uscc(conn, conf);

break;

case FAILED\_LOGIN:

print\_color("Incorrect Username or Password!", "red", ' ', false, true, false, true);

goto restart;

break;

default:

print\_color("Error to login", "red", ' ', false, true, false, true);

abort(); //it may not delete temporary files and may not flush stream buffer

}

goto restart;

exit:

print\_color(" Bye!\n", "light blue", ' ', true, true, true, true);

mysql\_close(conn);

return 0;

}

* **ucc.c**

#include "defines.h"

struct configuration conf;

MYSQL \*conn;

int ad\_code;

void new\_ad() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[5];

char description[100], photo[9], category[20], amount\_string[6];

int amount;

bool request, is\_null;

is\_null = 1;

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

print\_color("Ad description: ", "yellow", ' ', false, false, false, false);

strcpy(description, getInput(100, description, false));

request = yesOrNo("Do you want to add a photo?", 'y', 'n');

if (request)

strcpy(photo, "Presente");

else

param[2].is\_null = &is\_null;

print\_color("Category: ", "yellow", ' ', false, false, false, false);

strcpy(category, getInput(20, category, false));

while(1) {

print\_color("Amount: ", "light cyan", ' ', false, false, false, false);

getInput(6, amount\_string, false);

amount = atoi(amount\_string);

if(amount < 0)

print\_error(NULL, "The amount must be positive");

else

break;

}

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = description;

param[0].buffer\_length = strlen(description);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &amount;

param[1].buffer\_length = sizeof(amount);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = photo;

param[2].buffer\_length = strlen(photo);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = conf.username;

param[3].buffer\_length = strlen(conf.username);

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[4].buffer = category;

param[4].buffer\_length = strlen(category);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call inserimentoNuovoAnnuncio(?, ?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0) {

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters for a new ad\n", true);

}

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully generated!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

bool view\_ad(char \*username, bool check\_owner\_value) {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

int check\_owner; // Parameters are used to check

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT]; // of the owner of the ads for other functions

if(check\_owner\_value) {

check\_owner = 1;

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code = atoi(ad\_code\_string);

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = username;

param[1].buffer\_length = strlen(username);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &check\_owner;

param[2].buffer\_length = sizeof(check\_owner);

goto execution;

}

check\_owner = 0;

char ucc\_username[45];

bool request, is\_null;

is\_null = 1;

check\_owner = 0;

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

request = yesOrNo("Do you have any preference on ad?", 'y', 'n');

if(request == true) {

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code = atoi(ad\_code\_string);

param[1].is\_null = &is\_null;

goto execution\_ucc\_views\_ad;

}

if(request == false){

param[0].is\_null = &is\_null;

}

request = yesOrNo("Do you have any preference on user?", 'y', 'n');

if(request == true) {

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(45, ucc\_username, false);

}

if(request == false){

param[1].is\_null = &is\_null;

}

execution\_ucc\_views\_ad:

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = ucc\_username;

param[1].buffer\_length = strlen(ucc\_username);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &check\_owner;

param[2].buffer\_length = sizeof(check\_owner);

execution:

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaAnnuncio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ads\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return false;

}

if (!check\_owner\_value)

dump\_result\_set(conn, prepared\_stmt, "Ads list\n"); // dump the result set

mysql\_stmt\_close(prepared\_stmt);

return true;

}

void view\_category\_online() {

MYSQL\_STMT \*prepared\_stmt;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaCategoria()", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize view category statement\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error (prepared\_stmt, "An error occurred while viewing categories.");

dump\_result\_set(conn, prepared\_stmt, "Online Category list\n"); // dump the result set

mysql\_stmt\_next\_result(prepared\_stmt);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void remove\_ad() {

// Check if the user is the owner

if(view\_ad(conf.username, true) == false)

return;

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call rimuoviAnnuncio (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ads\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully removed!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

ad\_code = -1;

return;

}

void ad\_sold() {

// Check if the user is the owner

if(view\_ad(conf.username, true) == false)

return;

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call vendutoAnnuncio (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ads\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully sold!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

ad\_code = -1;

return;

}

bool view\_personal\_information(char cf\_owner[], char username\_owner[], int check\_owner) {

MYSQL\_STMT \*prepared\_stmt;

// Check information for other functions

if(check\_owner == 1){

MYSQL\_BIND param[3];

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf\_owner;

param[0].buffer\_length = strlen(cf\_owner);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &check\_owner;

param[1].buffer\_length = sizeof(check\_owner);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = username\_owner;

param[2].buffer\_length = strlen(username\_owner);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaInfoAnagrafiche (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

goto execution;

}

// View UCC information

if(yesOrNo("Do you want to see your account information?", 'y', 'n')) {

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_Info\_Utente (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

goto execution;

}

// View personal information

MYSQL\_BIND param[3];

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

char cf[MAX\_CF\_LENGHT];

bool is\_null;

is\_null = 1;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

cf[16] = '\0';

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf;

param[0].buffer\_length = strlen(cf);

param[1].is\_null = &is\_null; // check\_owner

param[2].is\_null = &is\_null; // username

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaInfoAnagrafiche (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

execution:

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return false;

}

if(check\_owner != 1)

dump\_result\_set(conn, prepared\_stmt, "Personal Information\n"); // dump the result set

mysql\_stmt\_close(prepared\_stmt);

return true;

}

void edit\_personal\_information(char cf\_set\_favorite[], char type\_set\_favorite[], char contact\_set\_favorites[]) {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[8];

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

bool is\_null = 1;

// Set contact as favorite

if(cf\_set\_favorite != NULL && type\_set\_favorite != NULL && contact\_set\_favorites != NULL) {

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf\_set\_favorite;

param[0].buffer\_length = strlen(cf\_set\_favorite);

param[1].is\_null = &is\_null;

param[2].is\_null = &is\_null;

param[3].is\_null = &is\_null;

param[4].is\_null = &is\_null;

param[5].is\_null = &is\_null;

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = type\_set\_favorite;

param[6].buffer\_length = strlen(type\_set\_favorite);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = contact\_set\_favorites;

param[7].buffer\_length = strlen(contact\_set\_favorites);

goto execution;

}

char cf[MAX\_CF\_LENGHT], surname[20], name[20], residential\_address[20], cap\_string[6], billing\_address[20], type\_favorite\_contact[20], favorite\_contact[40];

int cap;

char \*list\_choice\_type\_it[] = {"email", "cellulare", "social", "sms"};

char \*list\_choice\_type\_en[] = {"email", "mobile phone", "social", "sms"};

int lenght\_choice\_type = 4;

char choice[20];

bool request;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

cf[16] = '\0';

// Check if the user has this tax code

if(!view\_personal\_information(cf, conf.username, 1))

return;

request = yesOrNo("Do you want to edit your surname?", 'y', 'n');

if(request == true) {

print\_color("Surname: ", "yellow", ' ', false, false, false, false);

getInput(20, surname, false);

}

if(request == false){

param[1].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your name?", 'y', 'n');

if(request == true) {

print\_color("Name: ", "yellow", ' ', false, false, false, false);

getInput(20, name, false);

}

if(request == false){

param[2].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your residential address?", 'y', 'n');

if(request == true) {

print\_color("Residential address: ", "yellow", ' ', false, false, false, false);

getInput(20, residential\_address, false);

}

if(request == false){

param[3].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your CAP?", 'y', 'n');

if(request == true) {

print\_color("CAP: ", "light cyan", ' ', false, false, false, false);

getInput(6, cap\_string, false);

cap\_string[5] = '\0';

cap = atoi(cap\_string);

}

if(request == false){

param[4].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your billing address?", 'y', 'n');

if(request == true) {

print\_color("Billing address: ", "yellow", ' ', false, false, false, false);

getInput(20, billing\_address, false);

}

if(request == false){

param[5].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your favorite contact?", 'y', 'n');

while(1) {

if (request)

print\_color(" Which do you choose to edit?", "orange", ' ', true, true, false, false);

else {

param[6].is\_null = &is\_null;

param[7].is\_null = &is\_null;

break;

}

for(int i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, choice, false);

for(int i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], choice) == 0) {

sprintf(type\_favorite\_contact, "%s", list\_choice\_type\_it[i]);

type\_favorite\_contact[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, favorite\_contact, false);

goto execution\_editing\_information;

}

}

}

execution\_editing\_information:

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 = surname;

param[1].buffer\_length = strlen(surname);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = name;

param[2].buffer\_length = strlen(name);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = residential\_address;

param[3].buffer\_length = strlen(residential\_address);

param[4].buffer\_type = MYSQL\_TYPE\_LONG;

param[4].buffer = &cap;

param[4].buffer\_length = sizeof(cap);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = billing\_address;

param[5].buffer\_length = strlen(billing\_address);

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = type\_favorite\_contact;

param[6].buffer\_length = strlen(type\_favorite\_contact);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = favorite\_contact;

param[7].buffer\_length = strlen(favorite\_contact);

execution:

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaInfoAnagrafiche (?, ?, ?, ?, ?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to edit information\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully edited!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void edit\_photo\_ad() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], photo[9];

int ad\_code\_edit;

bool request, is\_null;

is\_null = 1;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_edit = atoi(ad\_code\_string);

request = yesOrNo("Do you want to add a photo to an ad?", 'y', 'n');

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_edit;

param[0].buffer\_length = sizeof(ad\_code\_edit);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if(request) {

strcpy(photo, "Presente");

}

else

param[2].is\_null = &is\_null;

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = photo;

param[2].buffer\_length = strlen(photo);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaFotoAnnuncio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully edited!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void insert\_remove\_comment() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], id\_comment\_string[MAX\_COMMENT\_ID\_LENGHT], text[45];

int ad\_code\_comment, id\_comment;

bool request, is\_null;

is\_null = 1;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_comment = atoi(ad\_code\_string);

request = yesOrNo("Do you want to add a comment to an ad?", 'y', 'n');

if(request) {

print\_color("Text of the comment: ", "yellow", ' ', false, false, false, false);

getInput(45, text, false);

param[2].is\_null = &is\_null;

} else {

print\_color("Comment ID to remove: ", "light cyan", ' ', false, false, false, false);

strcpy(id\_comment\_string, getInput(MAX\_COMMENT\_ID\_LENGHT, id\_comment\_string, false));

id\_comment = atoi(id\_comment\_string);

param[0].is\_null = &is\_null;

}

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = text;

param[0].buffer\_length = strlen(text);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &ad\_code\_comment;

param[1].buffer\_length = sizeof(ad\_code\_comment);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &id\_comment;

param[2].buffer\_length = sizeof(id\_comment);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaCommento (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to add or remove comment\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return;

}

if(request)

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

else

print\_color(" Successfully removed!", "light red", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_comment() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_comment;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_comment = atoi(ad\_code\_string);

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_comment;

param[0].buffer\_length = sizeof(ad\_code\_comment);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaCommento (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view comments\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Comments\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void insert\_remove\_note() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], id\_note\_string[MAX\_NOTE\_ID\_LENGHT], text[45];

int ad\_code\_note, id\_note;

bool request, is\_null;

is\_null = 1;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_note = atoi(ad\_code\_string);

request = yesOrNo("Do you want to add a note to the ad?", 'y', 'n');

if(request) {

print\_color("Text of the note: ", "yellow", ' ', false, false, false, false);

getInput(45, text, false);

param[2].is\_null = &is\_null;

} else {

print\_color("ID of the note to remove: ", "light cyan", ' ', false, false, false, false);

strcpy(id\_note\_string, getInput(MAX\_NOTE\_ID\_LENGHT, id\_note\_string, false));

id\_note = atoi(id\_note\_string);

param[0].is\_null = &is\_null;

}

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = text;

param[0].buffer\_length = strlen(text);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &ad\_code\_note;

param[1].buffer\_length = sizeof(ad\_code\_note);

param[2].buffer\_type = MYSQL\_TYPE\_LONG;

param[2].buffer = &id\_note;

param[2].buffer\_length = sizeof(id\_note);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = conf.username;

param[3].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaNota (?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to add or remove note\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return;

}

if(request)

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

else

print\_color(" Successfully removed!", "light red", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_note() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_note;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_note = atoi(ad\_code\_string);

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_note;

param[0].buffer\_length = sizeof(ad\_code\_note);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaNota (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view note\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Note\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void insert\_remove\_contact() {

char type[20], contact[40], cf[MAX\_CF\_LENGHT];

int remove = 1;

char \*list\_choice\_type\_it[] = {"email", "cellulare", "social", "sms"}, choice[20];

char \*list\_choice\_type\_en[] = {"email", "mobile phone", "social", "sms"};

int lenght\_choice\_type = 4, i;

bool request, request\_2, is\_null;

is\_null = 1;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

// Check if the user has this tax code

if(!view\_personal\_information(cf, conf.username, 1))

return;

request = yesOrNo("Do you want to add a contact?", 'y', 'n');

while(1) {

if (request)

print\_color(" Which do you choose to add?", "orange", ' ', false, true, false, false);

else

print\_color(" Which do you choose to remove?", "orange", ' ', false, true, false, false);

for(i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, choice, false);

for(i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], choice) == 0) {

sprintf(type, "%s", list\_choice\_type\_it[i]);

type[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, contact, false);

goto execution;

}

}

}

execution:

if(request) {

request\_2 = yesOrNo("Do you want to set it as a favorite?", 'y', 'n');

// If the user sends yes then it set as favorite

if (request\_2) {

edit\_personal\_information(cf, type, contact);

return;

}

}

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

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

// If the user wants to remove a contact

if(request)

param[3].is\_null= &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = type;

param[0].buffer\_length = strlen(type);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = contact;

param[1].buffer\_length = strlen(contact);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = cf;

param[2].buffer\_length = strlen(cf);

param[3].buffer\_type = MYSQL\_TYPE\_LONG;

param[3].buffer = &remove;

param[3].buffer\_length = sizeof(remove);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modifica\_RecapitoNonPreferito (?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to add or remove note\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return;

}

if(request)

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

else

print\_color(" Successfully removed!", "light red", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_contact() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char cf[MAX\_CF\_LENGHT];

int favorite = 1;

bool request, is\_null;

is\_null = 1;

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

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

request = yesOrNo("Do you want to see favorite contact?", 'y', 'n');

// Set the var to null

if(!request)

param[1].is\_null = &is\_null;

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 = &favorite;

param[1].buffer\_length = sizeof(favorite);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_contatti (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view contacts\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Contacts\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void send\_message() {

char receiver\_username[45], text[100];

print\_color(" Username of the user that you want to send the message", "white", ' ', true, true, false, false);

print\_color("Username: ", "yellow", ' ', true, false, false, false);

getInput(45, receiver\_username, false);

print\_color("Message text: ", "yellow", ' ', false, false, false, false);

getInput(100, text, false);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

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);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = receiver\_username;

param[1].buffer\_length = strlen(receiver\_username);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = text;

param[2].buffer\_length = strlen(text);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call invioMessaggio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to send message\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully sent", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_message() {

char receiver\_username[45], code\_conversation\_string[MAX\_CODE\_CONVERSATION];

int code\_conversation;

bool request, is\_null;

is\_null = 1;

print\_color(" Username of the user that you want to view the message", "white", ' ', true, true, false, false);

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(45, receiver\_username, false);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

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

request = yesOrNo("Do you know the conversation code", 'y', 'n');

if(request) {

print\_color("Code: ", "ligh cyan", ' ', false, false, false, false);

getInput(MAX\_CODE\_CONVERSATION, code\_conversation\_string, false);

code\_conversation = atoi(code\_conversation\_string);

} else

param[0].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &code\_conversation;

param[0].buffer\_length = sizeof(code\_conversation);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = receiver\_username;

param[2].buffer\_length = strlen(receiver\_username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaMessaggio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view messages\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Messages\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_conversation\_history() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaStorico\_UCC (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view conversation history\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Conversation history\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void follow\_ad() {

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_follow;

print\_color(" Enter the ad code that you want to follow", "white", ' ', true, true, false, false);

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code\_follow = atoi(ad\_code\_string);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_follow;

param[0].buffer\_length = sizeof(ad\_code\_follow);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call segui\_Annuncio (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to follow the ad\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully followed", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_ad\_followed() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_Annunci\_Seguiti (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ad followed\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Ad followed\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_report\_ucc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char report\_code\_string[10];

int report\_code;

bool request, is\_null;

is\_null = 1;

request = yesOrNo("Do you know the report code?", 'y', 'n');

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

if(request) {

print\_color("Report code: ", "light cyan", ' ', false, false, false, false);

getInput(10, report\_code\_string, false);

report\_code = atoi(report\_code\_string);

} else

param[0].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &report\_code;

param[0].buffer\_length = sizeof(report\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_report(?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize report statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to collect the report", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Reports\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_new\_notifications\_ucc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_notifiche (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view new notifications\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "New notifications\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

int run\_as\_ucc(MYSQL \*main\_conn, struct configuration main\_conf){

conn = main\_conn;

conf = main\_conf;

ad\_code = -1;

int num\_list = 22, chosen\_num; // length of list

char \*list[] = {"1","2","3","4","5","6","7","8","9","10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22"}; // list of choice

char option;

print\_color("Welcome ", "cyan", ' ', true, false, true, false);

print\_color(conf.username, "cyan", ' ', false, true, true, false);

if(!parse\_config("Users/UCC.json", &conf)) {

print\_color(" Unable to load ucc configuration", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

print\_color(" mysql\_change\_user() failed", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

while(1){

option = ' ';

chosen\_num = -1;

// Sleep process for 3 seconds so that the new instructions can be read by the user

//poll(0, 0, 290);

print\_color(" What would do you want to do? ", "white", ' ', true, true, false, false);

print\_color(list[0], "light blue", ' ', true, false, false, false); print\_color(") Insert a new Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[1], "light blue", ' ', true, false, false, false); print\_color(") View Ad", "orange", ' ', false, false, false, false);

print\_color(list[2], "light blue", ' ', true, false, false, false); print\_color(") View categories", "light cyan", ' ', false, false, false, false);

print\_color(list[3], "light blue", ' ', true, false, false, false); print\_color(") Remove Ad", "orange", ' ', false, false, false, false);

print\_color(list[4], "light blue", ' ', true, false, false, false); print\_color(") Sell Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[5], "light blue", ' ', true, false, false, false); print\_color(") View Personal Information", "orange", ' ', false, false, false, false);

print\_color(list[6], "light blue", ' ', true, false, false, false); print\_color(") Edit Personal Information", "light cyan", ' ', false, false, false, false);

print\_color(list[7], "light blue", ' ', true, false, false, false); print\_color(") View New Notifications", "orange", ' ', false, false, false, false);

print\_color(list[8], "light blue", ' ', true, false, false, false); print\_color(") Add or Remove a picture of your Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[9], "light blue", ' ', true, false, false, false); print\_color(") View comments of an Ad", "orange", ' ', false, false, false, false);

print\_color(list[10], "light blue", ' ', true, false, false, false); print\_color(") Add or Remove a comment to an Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[11], "light blue", ' ', true, false, false, false); print\_color(") View note of an Ad", "orange", ' ', false, false, false, false);

print\_color(list[12], "light blue", ' ', true, false, false, false); print\_color(") Add or Remove a note to an Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[13], "light blue", ' ', true, false, false, false); print\_color(") View your contacts", "orange", ' ', false, false, false, false);

print\_color(list[14], "light blue", ' ', true, false, false, false); print\_color(") Add o remove a contact", "light cyan", ' ', false, false, false, false);

print\_color(list[15], "light blue", ' ', true, false, false, false); print\_color(") Send a message", "orange", ' ', false, false, false, false);

print\_color(list[16], "light blue", ' ', true, false, false, false); print\_color(") View your messages", "light cyan", ' ', false, false, false, false);

print\_color(list[17], "light blue", ' ', true, false, false, false); print\_color(") View your conversation history", "orange", ' ', false, false, false, false);

print\_color(list[18], "light blue", ' ', true, false, false, false); print\_color(") Follow an Ad", "light cyan", ' ', false, false, false, false);

print\_color(list[19], "light blue", ' ', true, false, false, false); print\_color(") View Ad followed", "orange", ' ', false, false, false, false);

print\_color(list[20], "light blue", ' ', true, false, false, false); print\_color(") View own report", "light cyan", ' ', false, false, false, false);

print\_color(list[21], "light red", ' ', true, false, false, false); print\_color(") QUIT", "light red", ' ', false, true, false, false);

multiChoice("Which do you choose?", list, num\_list, &chosen\_num, &option);

if(option == '#') {

print\_color(" Number doesn't exists", "red", ' ', false, true, false, true);

continue;

}

if(chosen\_num == num\_list-1) {

print\_color(" Goodbye ", "orange", ' ', true, false, true, true);

print\_color(conf.username, "orange", ' ', false, true, true, true);

printf("\n");

break;

}

switch(chosen\_num) {

case 0:

new\_ad();

break;

case 1:

view\_ad("NULL", false);

break;

case 2:

view\_category\_online();

break;

case 3:

remove\_ad();

break;

case 4:

ad\_sold();

break;

case 5:

view\_personal\_information("", "", 0);

break;

case 6:

edit\_personal\_information(NULL, NULL, NULL);

break;

case 7:

view\_new\_notifications\_ucc();

break;

case 8:

edit\_photo\_ad();

break;

case 9:

view\_comment();

break;

case 10:

insert\_remove\_comment();

break;

case 11:

view\_note();

break;

case 12:

insert\_remove\_note();

break;

case 13:

view\_contact();

break;

case 14:

insert\_remove\_contact();

break;

case 15:

send\_message();

break;

case 16:

view\_message();

break;

case 17:

view\_conversation\_history();

break;

case 18:

follow\_ad();

break;

case 19:

view\_ad\_followed();

break;

case 20:

view\_report\_ucc();

break;

default:

print\_color(" Error to choose a number", "red", ' ', false, true, false, true);

}

}

return 0;

}

* **uscc.c**

#include "defines.h"

struct configuration conf;

MYSQL \*conn;

void view\_ad\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], ucc\_username[45];

int ad\_code;

bool request, is\_null;

is\_null = 1;

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

request = yesOrNo("Do you have any preference on ad?", 'y', 'n');

if(request == true) {

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code = atoi(ad\_code\_string);

param[1].is\_null = &is\_null;

goto execution;

}

if(request == false){

param[0].is\_null = &is\_null;

}

request = yesOrNo("Do you have any preference on user?", 'y', 'n');

if(request == true) {

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(45, ucc\_username, false);

}

if(request == false){

param[1].is\_null = &is\_null;

}

execution:

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = ucc\_username;

param[1].buffer\_length = strlen(ucc\_username);

param[2].is\_null = &is\_null;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaAnnuncio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ads\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Ads list\n"); // dump the result set

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_category\_online\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaCategoria()", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize view category statement\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error (prepared\_stmt, "An error occurred while viewing categories.");

dump\_result\_set(conn, prepared\_stmt, "Online Category list\n"); // dump the result set

mysql\_stmt\_next\_result(prepared\_stmt);

mysql\_stmt\_close(prepared\_stmt);

}

bool view\_personal\_information\_uscc(char cf\_owner[], char username\_owner[], int check\_owner) {

MYSQL\_STMT \*prepared\_stmt;

// Check information for other functions

if(cf\_owner != NULL && username\_owner != NULL && check\_owner == 1){

MYSQL\_BIND param[3];

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf\_owner;

param[0].buffer\_length = strlen(cf\_owner);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &check\_owner;

param[1].buffer\_length = sizeof(check\_owner);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = username\_owner;

param[2].buffer\_length = strlen(username\_owner);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaInfoAnagrafiche (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

goto execution;

}

// View USCC information

if(yesOrNo("Do you want to see your account information?", 'y', 'n')) {

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_Info\_Utente (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

goto execution;

}

// View personal information

MYSQL\_BIND param[3];

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

char cf[MAX\_CF\_LENGHT];

bool is\_null;

is\_null = 1;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

cf[16] = '\0';

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf;

param[0].buffer\_length = strlen(cf);

param[1].is\_null = &is\_null; // check\_owner

param[2].is\_null = &is\_null; // username

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaInfoAnagrafiche (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view information\n", true);

execution:

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return false;

}

if(check\_owner != 1)

dump\_result\_set(conn, prepared\_stmt, "Personal Information\n"); // dump the result set

mysql\_stmt\_close(prepared\_stmt);

return true;

}

void edit\_personal\_information\_uscc(char cf\_set\_favorite[], char type\_set\_favorite[], char contact\_set\_favorites[]) {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[8];

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

bool is\_null = 1;

// Set contact as favorite

if(cf\_set\_favorite != NULL && type\_set\_favorite != NULL && contact\_set\_favorites != NULL) {

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = cf\_set\_favorite;

param[0].buffer\_length = strlen(cf\_set\_favorite);

param[1].is\_null = &is\_null;

param[2].is\_null = &is\_null;

param[3].is\_null = &is\_null;

param[4].is\_null = &is\_null;

param[5].is\_null = &is\_null;

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = type\_set\_favorite;

param[6].buffer\_length = strlen(type\_set\_favorite);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = contact\_set\_favorites;

param[7].buffer\_length = strlen(contact\_set\_favorites);

goto execution;

}

char cf[MAX\_CF\_LENGHT], surname[20], name[20], residential\_address[20], cap\_string[5], billing\_address[20], type\_favorite\_contact[20], favorite\_contact[40];

char \*list\_choice\_type\_it[] = {"email", "cellulare", "social", "sms"};

char \*list\_choice\_type\_en[] = {"email", "mobile phone", "social", "sms"};

int lenght\_choice\_type = 4;

char choice[20];

int cap;

bool request;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

cf[16] = '\0';

// Check if the user has this tax code

if(!view\_personal\_information\_uscc(cf, conf.username, 1))

return;

request = yesOrNo("Do you want to edit your surname?", 'y', 'n');

if(request == true) {

print\_color("Surname: ", "yellow", ' ', false, false, false, false);

getInput(20, surname, false);

}

if(request == false){

param[1].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your name?", 'y', 'n');

if(request == true) {

print\_color("Name: ", "yellow", ' ', false, false, false, false);

getInput(20, name, false);

}

if(request == false){

param[2].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your residential address?", 'y', 'n');

if(request == true) {

print\_color("Residential address: ", "yellow", ' ', false, false, false, false);

getInput(20, residential\_address, false);

}

if(request == false){

param[3].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your CAP?", 'y', 'n');

if(request == true) {

print\_color("CAP: ", "light cyan", ' ', false, false, false, false);

getInput(5, cap\_string, false);

cap = atoi(cap\_string);

}

if(request == false){

param[4].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your billing address?", 'y', 'n');

if(request == true) {

print\_color("Billing address: ", "yellow", ' ', false, false, false, false);

getInput(20, billing\_address, false);

}

if(request == false){

param[5].is\_null = &is\_null;

}

request = yesOrNo("Do you want to edit your favorite contact?", 'y', 'n');

while(1) {

if (request)

print\_color(" Which do you choose to edit?", "orange", ' ', true, true, false, false);

else {

param[6].is\_null = &is\_null;

param[7].is\_null = &is\_null;

break;

}

for(int i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, choice, false);

for(int i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], choice) == 0) {

sprintf(type\_favorite\_contact, "%s", list\_choice\_type\_it[i]);

type\_favorite\_contact[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, favorite\_contact, false);

goto execution\_editing\_information;

}

}

}

execution\_editing\_information:

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 = surname;

param[1].buffer\_length = strlen(surname);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = name;

param[2].buffer\_length = strlen(name);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[3].buffer = residential\_address;

param[3].buffer\_length = strlen(residential\_address);

param[4].buffer\_type = MYSQL\_TYPE\_LONG;

param[4].buffer = &cap;

param[4].buffer\_length = sizeof(cap);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[5].buffer = billing\_address;

param[5].buffer\_length = strlen(billing\_address);

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[6].buffer = type\_favorite\_contact;

param[6].buffer\_length = strlen(type\_favorite\_contact);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[7].buffer = favorite\_contact;

param[7].buffer\_length = strlen(favorite\_contact);

execution:

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaInfoAnagrafiche (?, ?, ?, ?, ?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to edit information\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully edited!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void insert\_comment\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], text[45];

int ad\_code\_comment;

bool is\_null;

is\_null = 1;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_comment = atoi(ad\_code\_string);

print\_color("Text of the comment: ", "yellow", ' ', false, false, false, false);

getInput(45, text, false);

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = text;

param[0].buffer\_length = strlen(text);

param[1].buffer\_type = MYSQL\_TYPE\_LONG;

param[1].buffer = &ad\_code\_comment;

param[1].buffer\_length = sizeof(ad\_code\_comment);

param[2].is\_null = &is\_null;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modificaCommento (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to add or remove comment\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_comment\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_comment;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_comment = atoi(ad\_code\_string);

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_comment;

param[0].buffer\_length = sizeof(ad\_code\_comment);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaCommento (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view comments\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Comments\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_note\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_note;

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

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

strcpy(ad\_code\_string, getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false));

ad\_code\_note = atoi(ad\_code\_string);

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_note;

param[0].buffer\_length = sizeof(ad\_code\_note);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaNota (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view note\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Note\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void insert\_remove\_contact\_uscc() {

char type[20], contact[40], cf[MAX\_CF\_LENGHT];

int remove = 1;

char \*list\_choice\_type\_it[] = {"email", "cellulare", "social", "sms"}, choice[20];

char \*list\_choice\_type\_en[] = {"email", "mobile phone", "social", "sms"};

int lenght\_choice\_type = 4, i;

bool request, request\_2, is\_null;

is\_null = 1;

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

// Check if the user has this tax code

if(!view\_personal\_information\_uscc(cf, conf.username, 1))

return;

request = yesOrNo("Do you want to add a contact?", 'y', 'n');

while(1) {

if (request)

print\_color(" Which do you choose to add?\n", "white", ' ', true, true, false, false);

else

print\_color( "Which do you choose to remove?\n", "white", ' ', true, true, false, false);

for(i=0; i<lenght\_choice\_type; i++) {

print\_color(" - ", "cyan", ' ', false, false, false, false);

print\_color(list\_choice\_type\_en[i], "light blue", ' ', false, true, false, false);

}

// Take input

getInput(20, choice, false);

for(i=0; i<lenght\_choice\_type; i++) {

if(strcmp(list\_choice\_type\_en[i], choice) == 0) {

sprintf(type, "%s", list\_choice\_type\_it[i]);

type[strlen(list\_choice\_type\_it[i])] = '\0';

print\_color("Contact: ", "yellow", ' ', false, false, false, false);

getInput(40, contact, false);

goto execution;

}

}

}

execution:

if(request) {

request\_2 = yesOrNo("Do you want to set it as a favorite?", 'y', 'n');

// If the user sends yes then it set as favorite

if (request\_2) {

edit\_personal\_information\_uscc(cf, type, contact);

return;

}

}

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[4];

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = type;

param[0].buffer\_length = strlen(type);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = contact;

param[1].buffer\_length = strlen(contact);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = cf;

param[2].buffer\_length = strlen(cf);

param[3].buffer\_type = MYSQL\_TYPE\_LONG;

param[3].buffer = &remove;

param[3].buffer\_length = sizeof(remove);

// If the user wants to remove a contact

if(request)

param[3].is\_null= &is\_null;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call modifica\_RecapitoNonPreferito (?, ?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to add or remove note\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0) {

print\_stmt\_error(prepared\_stmt, NULL);

mysql\_stmt\_close(prepared\_stmt);

return;

}

if(request)

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

else

print\_color(" Successfully removed!", "light red", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_contact\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char cf[MAX\_CF\_LENGHT];

int favorite = 1;

bool request, is\_null;

is\_null = 1;

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

print\_color("Tax code: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_CF\_LENGHT, cf, false);

request = yesOrNo("Do you want to see favorite contact?", 'y', 'n');

// Set the var to null

if(!request)

param[1].is\_null = &is\_null;

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 = &favorite;

param[1].buffer\_length = sizeof(favorite);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_contatti (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view contacts\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Contacts\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void send\_message\_uscc() {

char receiver\_username[45], text[100];

print\_color(" Username of the user that you want to send the message", "white", ' ', true, true, false, false);

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(45, receiver\_username, false);

print\_color("Message text: ", "yellow", ' ', false, false, false, false);

getInput(100, text, false);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

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);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = receiver\_username;

param[1].buffer\_length = strlen(receiver\_username);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = text;

param[2].buffer\_length = strlen(text);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call invioMessaggio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to send message\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully sent", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_message\_uscc() {

char receiver\_username[45], code\_conversation\_string[MAX\_CODE\_CONVERSATION];

int code\_conversation;

bool request, is\_null;

is\_null = 1;

print\_color(" Username of the user that you want to view the message", "white", ' ', true, true, false, false);

print\_color("Username: ", "yellow", ' ', true, false, false, false);

getInput(45, receiver\_username, false);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

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

request = yesOrNo("Do you know the conversation code", 'y', 'n');

if(request) {

print\_color("Code: ", "ligh cyan", ' ', false, false, false, false);

getInput(MAX\_CODE\_CONVERSATION, code\_conversation\_string, false);

code\_conversation = atoi(code\_conversation\_string);

} else

param[0].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &code\_conversation;

param[0].buffer\_length = sizeof(code\_conversation);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = receiver\_username;

param[2].buffer\_length = strlen(receiver\_username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaMessaggio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view messages\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Messages\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_conversation\_history\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaStorico\_USCC (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view conversation history\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Conversation history\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void follow\_ad\_uscc() {

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT];

int ad\_code\_follow;

print\_color(" Enter the ad code that you want to follow", "white", ' ', true, true, false, false);

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code\_follow = atoi(ad\_code\_string);

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code\_follow;

param[0].buffer\_length = sizeof(ad\_code\_follow);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call segui\_Annuncio (?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to follow the ad\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully followed", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_ad\_followed\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_Annunci\_Seguiti (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ad followed\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Ad followed\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_new\_notifications\_uscc() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

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 (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_notifiche (?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view new notifications\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "New notifications\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

int run\_as\_uscc(MYSQL \*main\_conn, struct configuration main\_conf){

conn = main\_conn;

conf = main\_conf;

int num\_list = 16, chosen\_num; // length of list

char \*list[] = {"1","2","3","4","5","6","7","8","9","10", "11", "12", "13", "14", "15", "16"}; // list of choice

char option;

print\_color("Welcome ", "cyan", ' ', true, false, true, false);

print\_color(conf.username, "cyan", ' ', false, true, true, false);

if(!parse\_config("Users/USCC.json", &conf)) {

print\_color(" Unable to load uscc configuration", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

print\_color(" mysql\_change\_user() failed", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

while(1){

option = ' ';

chosen\_num = -1;

// Sleep process for 3 seconds so that the new instructions can be read by the user

//poll(0, 0, 290);

print\_color(" What would do you want to do? ", "white", ' ', true, true, false, false);

print\_color(list[0], "light blue", ' ', true, false, false, false); print\_color(") View ad", "orange", ' ', false, false, false, false);

print\_color(list[1], "light blue", ' ', true, false, false, false); print\_color(") View categories", "light cyan", ' ', false, false, false, false);

print\_color(list[2], "light blue", ' ', true, false, false, false); print\_color(") View Personal Information", "orange", ' ', false, false, false, false);

print\_color(list[3], "light blue", ' ', true, false, false, false); print\_color(") Edit Personal Information", "light cyan", ' ', false, false, false, false);

print\_color(list[4], "light blue", ' ', true, false, false, false); print\_color(") View comments of an ad", "orange", ' ', false, false, false, false);

print\_color(list[5], "light blue", ' ', true, false, false, false); print\_color(") Add a comment to an ad", "light cyan", ' ', false, false, false, false);

print\_color(list[6], "light blue", ' ', true, false, false, false); print\_color(") View note of an ad", "orange", ' ', false, false, false, false);

print\_color(list[7], "light blue", ' ', true, false, false, false); print\_color(") View your contacts", "light cyan", ' ', false, false, false, false);

print\_color(list[8], "light blue", ' ', true, false, false, false); print\_color(") Add o remove a contact", "orange", ' ', false, false, false, false);

print\_color(list[9], "light blue", ' ', true, false, false, false); print\_color(") Send a message", "light cyan", ' ', false, false, false, false);

print\_color(list[10], "light blue", ' ', true, false, false, false); print\_color(") View your messages", "orange", ' ', false, false, false, false);

print\_color(list[11], "light blue", ' ', true, false, false, false); print\_color(") View your conversation history", "light cyan", ' ', false, false, false, false);

print\_color(list[12], "light blue", ' ', true, false, false, false); print\_color(") View new Notifications", "orange", ' ', false, false, false, false);

print\_color(list[13], "light blue", ' ', true, false, false, false); print\_color(") Follow an ad", "light cyan", ' ', false, false, false, false);

print\_color(list[14], "light blue", ' ', true, false, false, false); print\_color(") View ad followed", "orange", ' ', false, false, false, false);

print\_color(list[15], "light red", ' ', true, false, false, false); print\_color(") QUIT", "light red", ' ', false, true, false, false);

multiChoice("Which do you choose?", list, num\_list, &chosen\_num, &option);

if(option == '#') {

print\_color(" Number doesn't exists", "red", ' ', false, true, false, true);

continue;

}

if(chosen\_num == num\_list-1) {

print\_color(" Goodbye ", "orange", ' ', true, false, true, true);

print\_color(conf.username, "orange", ' ', false, true, true, true);

printf("\n");

break;

}

switch(chosen\_num) {

case 0:

view\_ad\_uscc();

break;

case 1:

view\_category\_online\_uscc();

break;

case 2:

view\_personal\_information\_uscc("", "", 0);

break;

case 3:

edit\_personal\_information\_uscc(NULL, NULL, NULL);

break;

case 4:

view\_comment\_uscc();

break;

case 5:

insert\_comment\_uscc();

break;

case 6:

view\_note\_uscc();

break;

case 7:

view\_contact\_uscc();

break;

case 8:

insert\_remove\_contact\_uscc();

break;

case 9:

send\_message\_uscc();

break;

case 10:

view\_message\_uscc();

break;

case 11:

view\_conversation\_history\_uscc();

break;

case 12:

view\_new\_notifications\_uscc();

break;

case 13:

follow\_ad\_uscc();

break;

case 14:

view\_ad\_followed\_uscc();

break;

default:

print\_color(" Error to choose a number", "red", ' ', false, true, false, true);

}

}

return 0;

}

* **administrator.c**

#include "defines.h"

struct configuration conf;

MYSQL \*conn;

void new\_category() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[1];

char name[20]; // name of category

print\_color("Name of new category: ", "yellow", ' ', false, false, false, false);

getInput(20, name, false);

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = name;

param[0].buffer\_length = strlen(name);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call inserimentoNuovaCategoria(?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize new category statement\n", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters for a new category\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully added!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_category() {

MYSQL\_STMT \*prepared\_stmt;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaCategoria()", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize view category statement\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error (prepared\_stmt, "An error occurred while viewing categories.");

dump\_result\_set(conn, prepared\_stmt, "Online Category list\n"); // dump the result set

mysql\_stmt\_next\_result(prepared\_stmt);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_ad\_administrator() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ad\_code\_string[MAX\_AD\_CODE\_LENGHT], ucc\_username[MAX\_LENGHT\_USERNAME];

int ad\_code;

bool request, is\_null;

is\_null = 1;

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

request = yesOrNo("Do you have any preference on ad?", 'y', 'n');

if(request == true) {

print\_color("Ad code: ", "light cyan", ' ', false, false, false, false);

getInput(MAX\_AD\_CODE\_LENGHT, ad\_code\_string, false);

ad\_code = atoi(ad\_code\_string);

param[1].is\_null = &is\_null;

goto execution;

}

if(request == false){

param[0].is\_null = &is\_null;

}

request = yesOrNo("Do you have any preference on user?", 'y', 'n');

if(request == true) {

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, ucc\_username, false);

}

if(request == false){

param[1].is\_null = &is\_null;

}

execution:

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &ad\_code;

param[0].buffer\_length = sizeof(ad\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = ucc\_username;

param[1].buffer\_length = strlen(ucc\_username);

param[2].is\_null = &is\_null;

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizzaAnnuncio (?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize ad statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to view ads\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Ads list\n"); // dump the result set

mysql\_stmt\_close(prepared\_stmt);

return;

}

void generate\_report() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char ucc\_username[MAX\_LENGHT\_USERNAME];

print\_color("User's username: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, ucc\_username, false);

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

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[0].buffer = ucc\_username;

param[0].buffer\_length = strlen(ucc\_username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = conf.username;

param[1].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call genera\_report(?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize report statement\n", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters for a new report\n", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully generated!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

}

void collect\_report() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[3];

char ucc\_username[MAX\_LENGHT\_USERNAME], report\_code\_string[10];

int report\_code;

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, ucc\_username, false);

print\_color("Report code: ", "light cyan", ' ', false, false, false, false);

getInput(10, report\_code\_string, false);

report\_code = atoi(report\_code\_string);

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

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &report\_code;

param[0].buffer\_length = sizeof(report\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = ucc\_username;

param[1].buffer\_length = strlen(ucc\_username);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[2].buffer = conf.username;

param[2].buffer\_length = strlen(conf.username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call riscossione\_report(?, ?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize report statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to collect the report", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

print\_color(" Successfully collected!", "light blue", ' ', false, true, false, true);

mysql\_stmt\_close(prepared\_stmt);

return;

}

void view\_report() {

MYSQL\_STMT \*prepared\_stmt;

MYSQL\_BIND param[2];

char ucc\_username[MAX\_LENGHT\_USERNAME], report\_code\_string[10];

int report\_code;

bool request, is\_null;

is\_null = 1;

print\_color("Username: ", "yellow", ' ', false, false, false, false);

getInput(MAX\_LENGHT\_USERNAME, ucc\_username, false);

request = yesOrNo("Do you know the report code?", 'y', 'n');

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

if(request) {

print\_color("Report code: ", "light cyan", ' ', false, false, false, false);

getInput(10, report\_code\_string, false);

report\_code = atoi(report\_code\_string);

} else

param[0].is\_null = &is\_null;

param[0].buffer\_type = MYSQL\_TYPE\_LONG;

param[0].buffer = &report\_code;

param[0].buffer\_length = sizeof(report\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING;

param[1].buffer = ucc\_username;

param[1].buffer\_length = strlen(ucc\_username);

if (!setup\_prepared\_stmt(&prepared\_stmt, "call visualizza\_report(?, ?)", conn))

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to initialize report statement", true);

if (mysql\_stmt\_bind\_param(prepared\_stmt, param) != 0)

finish\_with\_stmt\_error(conn, prepared\_stmt, "Unable to bind parameters to collect the report", true);

if (mysql\_stmt\_execute(prepared\_stmt) != 0)

print\_stmt\_error(prepared\_stmt, NULL);

else

dump\_result\_set(conn, prepared\_stmt, "Reports\n");

mysql\_stmt\_close(prepared\_stmt);

return;

}

int run\_as\_administrator(MYSQL \*main\_conn, struct configuration main\_conf){

conn = main\_conn;

conf = main\_conf;

int num\_list = 7, chosen\_num; // length of list

char \*list[] = {"1","2","3","4","5","6","7"}; // list of choice

char option;

print\_color("Welcome ", "cyan", ' ', true, false, true, false);

print\_color(conf.username, "cyan", ' ', false, true, true, false);

if(!parse\_config("Users/Administrator.json", &conf)) {

print\_color(" Unable to load administrator configuration", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, conf.db\_username, conf.db\_password, conf.database)) {

print\_color(" mysql\_change\_user() failed", "red", ' ', false, true, true, true);

exit(EXIT\_FAILURE);

}

while(1){

option = ' ';

chosen\_num = -1;

// Sleep process for 3 seconds so that the new instructions can be read by the user

//poll(0, 0, 290);

print\_color(" What would do you want to do? ", "white", ' ', true, true, false, false);

print\_color(list[0], "light blue", ' ', true, false, false, false); print\_color(") Insert a new category", "orange", ' ', false, false, false, false);

print\_color(list[1], "light blue", ' ', true, false, false, false); print\_color(") View category names online", "light cyan", ' ', false, false, false, false);

print\_color(list[2], "light blue", ' ', true, false, false, false); print\_color(") View ad", "orange", ' ', false, false, false, false);

print\_color(list[3], "light blue", ' ', true, false, false, false); print\_color(") Generate a report", "light cyan", ' ', false, false, false, false);

print\_color(list[4], "light blue", ' ', true, false, false, false); print\_color(") Collect reports", "orange", ' ', false, false, false, false);

print\_color(list[5], "light blue", ' ', true, false, false, false); print\_color(") View reports", "light cyan", ' ', false, false, false, false);

print\_color(list[6], "light red", ' ', true, false, false, false); print\_color(") QUIT", "light red", ' ', false, true, false, false);

multiChoice("Which do you choose?", list, num\_list, &chosen\_num, &option);

if(option == '#') {

print\_color(" Number doesn't exists", "red", ' ', false, true, false, true);

continue;

}

if(chosen\_num == num\_list-1) {

print\_color(" Goodbye ", "orange", ' ', true, false, true, true);

print\_color(conf.username, "orange", ' ', false, true, true, true);

printf("\n");

break;

}

switch(chosen\_num) {

case 0:

new\_category();

break;

case 1:

view\_category();

break;

case 2:

view\_ad\_administrator();

break;

case 3:

generate\_report();

break;

case 4:

collect\_report();

break;

case 5:

view\_report();

break;

default:

print\_color(" Error to choose a number", "red", ' ', false, true, false, true);

}

}

return 0;

}

* **defines.h**

#pragma once

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#define MAX\_AD\_CODE\_LENGHT 10

#define MAX\_COMMENT\_ID\_LENGHT 10

#define MAX\_NOTE\_ID\_LENGHT 10

#define MAX\_CF\_LENGHT 17

#define MAX\_CODE\_CONVERSATION 10

#define MAX\_LENGHT\_USERNAME 45

struct configuration {

char \*host;

char \*db\_username;

char \*db\_password;

unsigned int port;

char \*database;

char username[45];

char password[45];

};

extern struct configuration conf;

extern int parse\_config(char \*path, struct configuration \*conf);

extern char \*getInput(unsigned int lung, char \*stringa, bool hide);

extern char \*getInputScanf(char \*domanda, char \*stringa, int length\_max);

extern bool yesOrNo(char \*domanda, char yes, char no);

extern void print\_color(char \*stringa, char \*colore\_scelto, char c, bool first\_space, bool last\_space, bool bold, bool blink);

extern void multiChoice(char \*domanda, char \*choices[], int num, int \*chosen\_num, char \*option);

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 int run\_as\_administrator(MYSQL \*conn, struct configuration conf);

extern int run\_as\_ucc(MYSQL \*conn, struct configuration conf);

extern int run\_as\_uscc(MYSQL \*conn, struct configuration conf);

* **inout.c**

#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);

void handler\_ign(){

signal(SIGALRM, handler);

signal(SIGINT, handler);

signal(SIGHUP, handler);

signal(SIGQUIT, handler);

signal(SIGTERM, handler);

signal(SIGTSTP, handler);

signal(SIGTTIN, handler);

signal(SIGTTOU, handler);

}

void handler\_dfl(){

signal(SIGALRM, SIG\_DFL);

signal(SIGINT, SIG\_DFL);

signal(SIGHUP, SIG\_DFL);

signal(SIGQUIT, SIG\_DFL);

signal(SIGTERM, SIG\_DFL);

signal(SIGTSTP, SIG\_DFL);

signal(SIGTTIN, SIG\_DFL);

signal(SIGTTOU, SIG\_DFL);

}

char \*getInput(unsigned int lung, char \*stringa, bool hide)

{

char c;

unsigned int i;

struct termios term, oterm;

handler\_ign(); // Ignora eventuali segnali

if(hide) {

// Svuota il buffer

(void) fflush(stdout);

// 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));

}

}

// 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');

}

// Controlla che il terminatore di stringa sia stato inserito

if(i == lung - 1)

stringa[i] = '\0';

if(hide) {

//Va a capo dopo l'input

(void) write(fileno(stdout), "\n", 1);

// Ripristina le impostazioni precedenti dello schermo

(void) tcsetattr(fileno(stdin), TCSAFLUSH, &oterm);

handler\_dfl(); // Ripristina la gestione dei segnali

// Se era stato ricevuto un segnale viene rilanciato al processo stesso

if(signo)

(void) raise(signo);

}

return stringa;

}

// Scanf alternativo

char \*getInputScanf(char \*domanda, char \*stringa, int length\_max) {

int length, i=0;

char c;

printf("%s", domanda);

handler\_ign(); // Ignora eventuali segnali

length = 0;

stringa = (char \*) malloc(length\_max \* sizeof(char \*));

while(1) {

c = getchar();

stringa[i] = c;

if(c == '\n') break;

length++;

i++;

}

stringa[length] = '\0';

if(length > length\_max)

stringa[length\_max-1] = '\0';

handler\_dfl(); // Reset di default dei segnali

// Rilancio dei segnali pendenti

if(signo)

(void) raise(signo);

return stringa;

}

// Per la gestione dei segnali

static void handler(int s) {

signo = s;

}

void print\_color(char \*colorless, char \*colore\_scelto, char c, bool first\_space, bool last\_space, bool bold, bool blink) {

char \*stringa;

char \*list\_color[] = {"orange", "blue", "light blue", "red", "light red", "white", "purple", "cyan", "light cyan", "yellow", "pink"};

int i, length\_list\_color;

// Controllo dei parametri

if(strlen(colorless) == 0 && c == ' ') {

print\_error(NULL, "Error Parameters");

return;

}

// Il colore deve essere minuscolo

char \*colore = (char \*) malloc(strlen(colore\_scelto) \* sizeof(char \*));

for(i=0; colore[i]; i++){

colore[i] = tolower(colore\_scelto[i]);

}

colore[i] = '\0';

// Controllo che la variabile passata sia una stringa o un char

if(strcmp(colorless, "") == 0) {

stringa = (char \*) malloc(2);

stringa[0] = c;

stringa[1] = '\0';

} else {

stringa = (char \*) malloc(strlen(colorless)\*sizeof(char \*));

strcpy(stringa, colorless);

stringa[strlen(stringa)] = '\0';

}

// Cerco il colore

length\_list\_color = 11; i = 0;

while(1) {

if (i == length\_list\_color) {break;}

if (strcmp(colore\_scelto, list\_color[i]) == 0) {break;}

i++;

}

free(colore);

// Stampa spazio

if(first\_space)

printf("\n");

// Stampa blink

if(blink)

printf("\e[1m\e[5m");

// Stampa in grassetto

if(bold)

printf("\e[1m");

// Stampa della stringa

switch(i) {

case 0:

printf("\033[40m\033[1;32m");

printf("%s", stringa);

break;

case 1:

printf("\033[40m\033[34m");

printf("%s", stringa);

break;

case 2:

printf("\033[40m\033[1;34m");

printf("%s", stringa);

break;

case 3:

printf("\033[40m\033[31m");

printf("%s", stringa);

break;

case 4:

printf("\033[40m\033[1;31m");

printf("%s", stringa);

break;

case 5:

printf("\033[40m\033[1;37m");

printf("%s", stringa);

break;

case 6:

printf("\033[40m\033[35m");

printf("%s", stringa);

break;

case 7:

printf("\033[40m\033[36m");

printf("%s", stringa);

break;

case 8:

printf("\033[40m\033[1;36m");

printf("%s", stringa);

break;

case 9:

printf("\033[40m\033[1;33m");

printf("%s", stringa);

break;

case 10:

printf("\033[40m\033[1;35m");

printf("%s", stringa);

break;

default:

printf("%s", stringa);

break;

}

// Delimiter color

if(i<length\_list\_color)

printf("\033[0m");

// Fine stampa in grassetto

if(bold)

printf("\e[22m");

// Fine stampa blink

if(blink)

printf("\e[25m\e[22m");

// Stampa spazio

if(last\_space)

printf("\n");

free(stringa); // Deallocazione spazio nell'heap

return;

}

bool yesOrNo(char \*domanda, char yes, char no) {

// I caratteri 'yes' e 'no' devono essere minuscoli

yes = tolower(yes);

no = tolower(no);

// Richiesta della risposta

while(true) {

print\_color(domanda, "white", ' ', false, false, true, false); // Mostra la domanda

printf(" [");

print\_color("", "light blue", yes, false, false, false, false);

printf("/");

print\_color("", "light red", no, false, false, false, false);

printf("]: ");

//printf("\e[1m%s\e[22m [\033[40m\033[1;34m%c\033[0m/\033[40m\033[1;31m%c\033[0m]: ", domanda, yes, no); // Mostra la domanda

char c[2];

getInput(2, c, false);

// Controlla quale risposta è stata data

if(c[0] == yes || c[0] == toupper(yes))

return true;

else if(c[0] == no || c[0] == toupper(no))

return false;

}

}

void multiChoice(char \*domanda, char \*choices[], int num, int \*chosen\_num, char \*option)

{

int i = 0, j = 0, lenght = 0;

while(i<num)

lenght += (int)strlen(choices[i++]);

char \*possib = (char \*) malloc(lenght \* num \* sizeof(char \*)); // Genera la stringa delle possibilità

// Copia tutti i valori in un unica stringa

for(i = 0; i < num; i++) {

for(int z=0; z<(int)strlen(choices[i]); z++) {

possib[j++] = choices[i][z];

}

possib[j++] = '/';

}

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

// Chiede la risposta

print\_color(domanda, "white", ' ', true, false, true, false); // Mostra la domanda

printf(" [");

print\_color(possib, "light blue", ' ', false, false, false, false);

printf("]: ");

char c[3];

sprintf(c, "%s", getInput(3, c, false));

// Controlla se è un carattere valido

for(i = 0; i < num; i++) {

if(strcmp(c, choices[i]) == 0) {

\*chosen\_num = i; // Imposta l'indice all'interno della lista tramite dereferenziamento di puntatore

return;

}

}

\*option = '#'; // numero non trovato

return;

}

* **parse.c**

#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];

/\* Quote: end of string \*/

if (c == '\"') {

if (tokens == NULL) {

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\_STRING, start+1, parser->pos);

#ifdef JSMN\_PARENT\_LINKS

token->parent = parser->toksuper;

#endif

return 0;

}

/\* Backslash: Quoted symbol expected \*/

if (c == '\\' && parser->pos + 1 < len) {

int i;

parser->pos++;

switch (js[parser->pos]) {

/\* Allowed escaped symbols \*/

case '\"': case '/' : case '\\' : case 'b' :

case 'f' : case 'r' : case 'n' : case 't' :

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;

}

}

}

#endif

}

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':

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 ||

(t->type == JSMN\_STRING && t->size != 0)) {

return JSMN\_ERROR\_INVAL;

}

}

#else

/\* In non-strict mode every unquoted value is a primitive \*/

default:

#endif

r = jsmn\_parse\_primitive(parser, js, len, tokens, num\_tokens);

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;

}

* **utilis.h**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

void print\_stmt\_error (MYSQL\_STMT \*stmt, char \*message)

{

if (message)

fprintf (stderr, " \e[1m\e[5m\033[40m\033[31m%s\033[0m\e[25m\e[22m\n", message);

if (stmt != NULL) {

fprintf (stderr, " \e[1m\e[5m\033[40m\033[31mError %u (%s): %s\033[0m\e[25m\e[22m\n",

mysql\_stmt\_errno (stmt), // returns SQLSTATE value

mysql\_stmt\_sqlstate(stmt), // returns MYSQL\_ERRNO value

mysql\_stmt\_error (stmt)); // returns MESSAGE\_TEXT value

}

}

void print\_error(MYSQL \*conn, char \*message)

{

fprintf (stderr, " \e[1m\e[5m\033[40m\033[31m%s\033[0m\e[25m\e[22m\n", message);

if (conn != NULL) {

#if MYSQL\_VERSION\_ID >= 40101

fprintf (stderr, " \e[1m\e[5m\033[40m\033[31mError %u (%s): %s\033[0m\e[25m\e[22m\n",

mysql\_errno (conn), mysql\_sqlstate(conn), mysql\_error (conn));

#else

fprintf (stderr, " \e[1m\e[5m\033[40m\033[31mError %u (%s): %s\033[0m\e[25m\e[22m\n",

mysql\_errno (conn), mysql\_error (conn));

#endif

}

}

bool setup\_prepared\_stmt(MYSQL\_STMT \*\*stmt, char \*statement, MYSQL \*conn)

{

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);

bool is\_null[num\_fields];

if (num\_fields > 0) {

/\* there is a result set to fetch \*/

printf("\n ");

print\_color(title, "light cyan", ' ', false, true, true, true);

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;

rs\_bind[i].is\_null = &is\_null[i];

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++) {

if (rs\_bind[i].is\_null\_value) {

printf (" %-\*s |", (int)fields[i].max\_length, "NULL");

continue;

}

if(\*rs\_bind[i].is\_null){

printf(" %-\*s |", (int)fields[i].max\_length, "NULL");

continue;

}

if(rs\_bind[i].buffer\_type == MYSQL\_TYPE\_TINY) {

if (\*(char \*)rs\_bind[i].buffer == 0)

printf(" %-\*s |", (int)fields[i].max\_length+((int)strlen("FALSE")), "FALSE");

if (\*(char \*)rs\_bind[i].buffer == 1)

printf(" %-\*s |", (int)fields[i].max\_length+((int)strlen("TRUE")), "TRUE");

} else {

switch (rs\_bind[i].buffer\_type) {

case MYSQL\_TYPE\_VAR\_STRING:

printf(" %-\*s |", (int)fields[i].max\_length, (char \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_DATETIME:

date = (MYSQL\_TIME \*)rs\_bind[i].buffer;

printf("%02d/%02d/%4d %02d:%02d:%02d %-\*s |", date->day, date->month, date->year,date->hour,date->minute,date->second, ((int)fields[i].max\_length)-19, " ");

break;

case MYSQL\_TYPE\_DATE:

date = (MYSQL\_TIME \*)rs\_bind[i].buffer;

printf(" %d-%02d-%02d %-\*s |", date->year, date->month, date->day, 1, "");

break;

case MYSQL\_TYPE\_TIMESTAMP:

date = (MYSQL\_TIME \*)rs\_bind[i].buffer;

printf(" %d-%02d-%02d %-\*s |", date->year, date->month, date->day, 1, "");

break;

case MYSQL\_TYPE\_STRING:

printf(" %-\*s |", (int)fields[i].max\_length, (char \*)rs\_bind[i].buffer);

break;

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:

printf(" %-\*d |", (int)fields[i].max\_length, \*(int \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_NEWDECIMAL:

printf(" %-\*.02lf |", (int)fields[i].max\_length, \*(float\*) rs\_bind[i].buffer);

break;

default:

printf(" \e[1m\e[5m\033[40m\033[31mERROR: Unhandled type (%d)\033[0m\e[25m\e[22m\n", rs\_bind[i].buffer\_type);

abort();

}

}

}

putchar('\n');

print\_dashes(rs\_metadata);

}

mysql\_stmt\_next\_result(stmt);

mysql\_free\_result(rs\_metadata); /\* free metadata \*/

/\* free output buffers \*/

for (i = 0; i < num\_fields; i++) {

free(rs\_bind[i].buffer);

}

free(rs\_bind);

}

}

* **Makefile**

CC = gcc

OTPS = `mysql\_config --cflags --include --libs`

all:

$(CC) -Wall -Wextra \*.c -o client $(OTPS)

clean:

rm \*.o client

* **UCC.json**

{

"host": "localhost",

"username" : "USCC",

"password" : "usccpassword",

"port" : 3306,

"database" : "BachecaElettronicadb"

}

* **USCC.json**

{

"host": "localhost",

"username" : "USCC",

"password" : "usccpassword",

"port" : 3306,

"database" : "BachecaElettronicadb"

}

* **Administrator.json**

{

"host": "localhost",

"username" : "USCC",

"password" : "usccpassword",

"port" : 3306,

"database" : "BachecaElettronicadb"

}

* **Login.json**

{

"host": "localhost",

"username" : "USCC",

"password" : "usccpassword",

"port" : 3306,

"database" : "BachecaElettronicadb"

}

1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-2)
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. [↑](#footnote-ref-3)
3. [↑](#footnote-ref-4)