|  |  |
| --- | --- |
| ***Definizione*** | ***Funzionamento*** |
| Persona/Account | |
| bool Login(string username, string password); | * TRUE: username e password corretti, cioè esiste un account con questi dati * FALSE: * Username e password sbagliati * Errore nella Query * Connessione non eseguita con il DB * Altra eccezione |
| string GetPrivilegi(string username); | Ritorna i privilegi dell’Account:   * “direttore” * “cliente” * “impiegato” * String.Empty : errore nell’esecuzione della query o mancata connessione con DB |
| List<Persona> GetListaPersone(string tipoAccount, string idFiliale); | Restituisce una lista di persone che hanno l’account del tipo specificato in tipoAccount (“client”, “impiegato”…) e che appartengono ad una determinata filiale con l’id uguale a idFiliale. Se si verifica un errore oppure non vengono trovati account/persone che rispettano le caratteristiche specificate, il programma ritorna una lista vuota. |
| bool EliminaAccount(string username); | Elimina l’Account selezionato con username.   * TRUE: se l’account viene eliminato correttamente * FALSE: * Username non trovato * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| Persona CheckUsername(string username); | Cerca i dati dell’Account e della Persona con l’username uguale a quello dato. Se viene trovato, restituisce l’oggetto Persona contente i dati anagrafici e dell’account. Altrimenti restituisce un’oggetto Persona con tutti gli attributi settati a Default. Questo avviene anche in caso di eccezioni. |
| Persona GetPersona(string codiceFiscale); | Cerca i dati dell’Account e della Persona con il codiceFiscale uguale a quello dato. Se viene trovato, restituisce l’oggetto Persona contente i dati anagrafici, mentre i dati dell’account sono settati a default, perché una persona individuata con il codice fiscale può avere più account. Altrimenti restituisce un’oggetto Persona con tutti gli attributi settati a Default. Questo avviene anche in caso di eccezioni. |
| bool AggiungiPersona(Persona persona, string password); | Aggiunge un Account, e aggiunge una Persona se la persona non ha già un altro account.   * TRUE: se l’account viene aggiunto correttamente * FALSE: * Account non aggiunto * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| bool ModificaPersona(string usernameOld, Persona persona); | INPUT:   * usernameOld: username dell’Account/Persona da modificare * Persona: contiene le informazioni da aggiornare nel DB. Se un attributo non è da aggiornare viene lasciato al suo valore di default in persona.   Es. persona.CAP = null, non verrà modificato nel DB.  persona.Nome = “Mario”, verrà aggiornato.  persona.Cogome = string.Empty, non verrà aggiornato.  OUTPUT:   * TRUE: profilo modificato correttamente. * FALSE: * Account non modificato * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| string GetIdFilialeByUsername(string username); | Ritorna l’id della filiale alla cui appartiene l’account, se l’account inserito non esiste o si verifica un errore nell’esecuzione del comando la funzione ritorna string.Empty |
| Conto Corrente | |
| ContoCorrente SelectContoCorrente(UInt64 idContoCorrente); | Cerca il conto corrente con l’idContoCorrente uguale a quello dato. Se viene trovato, restituisce l’oggetto ContoCorrente contente i dati del conto. Altrimenti restituisce un’oggetto ContoCorrente con tutti gli attributi settati a Default. Questo avviene anche in caso di eccezioni. |
| bool CheckIBAN(string IBAN); | Controlla se esiste un ContoCorrente con quell’IBAN, valore di ritorno:   * TRUE: * Esiste un ContoCorrente * FALSE: * Non è stato trovato un ContoCorrente con quell’IBAN * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| bool CheckIDConto(UInt64 idContoCorrente); | Controlla se esiste un ContoCorrente con quell’idContoCorrente, valore di ritorno:   * TRUE: * Esiste un ContoCorrente * FALSE: * Non è stato trovato un ContoCorrente con quell’idContoCorrente * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| List<ContoCorrente> GetListaContoCorrente(string username); | Restituisce una lista di Conti Correnti che appartengono all’account con l’username specificato. Se si verifica un errore, l’account non esiste o l’account non ha conti correnti collegati, il programma ritorna una lista vuota. |
| bool AggiungiContoCorrente(string username, string idFiliale, decimal? saldo); | Aggiunge un conto corrente appartenente all’username e alla filiale inseriti e con un saldo inziale specificato. L’idContoCorrente e IBAN sono generati automaticamente dal DB. Se si verifica un errore durante l’operazione il metodo ritorna False. |
| string GetIBANByIdContoCorrente(UInt64 idContoCorrente); | Ritorna l’IBAN del conto corrente con l’id inserito, se non esiste o si verifica un errore nell’esecuzione del comando la funzione ritorna string.Empty |
| Filiale | |
| Filiale GetFiliale(string username); | Cerca i dati della Filiale alla quale appartiene l’account con l’username uguale a quello dato. Se viene trovato, restituisce l’oggetto di tipo Filiale contente i dati come nome, indirizzo, idFiliale, etc.. . Altrimenti restituisce un’oggetto con tutti gli attributi settati a Default. Questo avviene anche in caso di eccezioni. |
| bool ModificaDatiFiliale(string idFiliale, Filiale nuovaFiliale); | INPUT:   * Identificativo: idFiliale della Filiale da modificare * filiale: contiene le informazioni da aggiornare nel DB. Se un attributo non è da aggiornare viene lasciato al suo valore di default.   Es. filiale.CAP = null, non verrà modificato nel DB.  filiale.nome = “UniPr Bank Parma”, verrà aggiornato.  filiale.citta = string.Empty, non verrà aggiornato.  OUTPUT:   * TRUE: filiale modificata correttamente. * FALSE: * filiale non modificata * Errore nella query * Connessione non eseguita con il DB * Altra eccezione |
| string GetNameFiliale(string idFiliale); | Ritorna il nome della filiale con l’id specificato, se l’id inserito non esiste o si verifica un errore nell’esecuzione del comando la funzione ritorna string.Empty |
| Movimenti | |
| List<Movimento> GetListaMovimenti(UInt64 idContoCorrente); | Restituisce una lista di Conti Correnti che appartengono all’account con l’username specificato. Se si verifica un errore, l’account non esiste o l’account non ha conti correnti collegati, il programma ritorna una lista vuota. |
| bool CheckImporto(decimal importo, string IBANCommittente); | Questa funzione controlla se il saldo del conto corrente avente l’IBAN inserito è sufficiente a coprire l’operazione che richiede l’importo inserito. Se è sufficiente ritorna True, altrimenti ritorna False. Ritorna False anche in caso di errori durante l’esecuzione o conto corrente non trovato. |
| bool EseguiBonifico(string IBANCommittente, string IBANBeneficiario, decimal Importo); | Questa funzione aggiunge un bonifico, per farlo prima detrae l’importo al committente, poi aggiunge l’importo al beneficiario e solo alla fine aggiunge il movimento nel database. Genera in automatico l’id movimento e salva la data e l’ora in cui viene eseguito. Se tutto dovesse andare a buon fine ritorna un True, altrimenti ritorna False. |
| bool EseguiPrelievoDenaro(UInt64 idContoCorrente, decimal importo); | Questa funzione aggiunge un prelievo, per farlo prima detrae l’importo dal conto corrente, e poi aggiunge il movimento nel database. Genera in automatico l’id movimento e salva la data e l’ora in cui viene eseguito. Se tutto dovesse andare a buon fine ritorna un True, altrimenti ritorna False. |
| bool EseguiDeposito(UInt64 idContoCorrente, decimal importo); | Questa funzione aggiunge un deposito, per farlo prima aggiunge l’importo al conto corrente, e poi aggiunge il movimento nel database. Genera in automatico l’id movimento e salva la data e l’ora in cui viene eseguito. Se tutto dovesse andare a buon fine ritorna un True, altrimenti ritorna False. |